inmos

occam 2 toolset handbook

INMOS Limited

72 TDS 199 00

INMOS document number: 72 TDS 199 00

Contents

Toolset summary

Tool	Description
`iboot`	The bootstrap tool. Produces bootable code for single transputers.
`icheck`	The occam 2 syntax checker. Produces no object code.
`iconf`	The configurer tool. Produces bootable code for multitransputer networks.
`idebug`	The toolset debugger. Provides symbolic and low level debugging of transputer programs.
`idump`	The memory dump tool. Used when debugging programs on the root transputer.
`ilibr`	The librarian. Creates library files.
`ilink`	The linker. Links object code into a single file.
`ilist`	The binary lister tool. Decodes and displays data from object and bootable files.
`imakef`	The Makefile generator. Builds Makefiles for object and bootable files.
`iserver`	The host file server and program loader. Loads programs onto transputer boards and provides runtime access to the host.
`isim`	The T414 simulator tool. Used to test and debug programs without transputer hardware.
`iskip`	The skip loader tool. Allows programs to be loaded onto external networks over the root transputer.
`occam`	The occam 2 compiler. Compiles occam source.

File extensions

Extension	Description
`.btl`	Boot from link code for multitransputer programs.
`.btr`	Dynamically loadable code for multitransputer programs.
`.b`*xx*	Boot from link code for single transputer programs.
`.c`*xx*	Linked object file.
`.dmp`	Memory dump file produced by **`idump`**. Used by **`idebug`**.
`.dsc`	Configuration code description generated by **`iconf`**. ASCII format.
`.d`*xx*	Code description generated by **`iboot`**. ASCII format.
`.inc`	Source file containing constants and declarations.
`.lbb`	Library indirect file. Command input file for **`ilibr`**.
`.lib`	Library code created by **`ilibr`**.
`.liu`	Library usage file.
`.l`*xx*	Linker indirect file. Command input file for **`ilink`**.
`.map`	Link map file generated by **`ilink`**.
`.m`*xx*	Module map file generated by **`ilink`**. Binary format.
`.occ`	occam source.
`.pgm`	Configuration description source.
`.r`*xx*	Dynamically loadable code for single transputer programs.
`.s`*xx*	Symbol table generated by **`ilink`**.
`.t`*xx*	Compiled code produced by `occam`.
xx varies according to transputer type (`2`, `4`, `5`, `8`, `a`, `b`, `c`) and error mode (`h`, `s`, `u`, `x`).	

Tools

`iboot` – bootstrap tool

Generates bootable code files for single transputers or creates new bootstrap loader programs. Can also be used to create non-bootable single transputer programs for dynamic loading or booting from ROM.

Syntax: `iboot` *filename* {*options*}

where: *filename* is a compiled (`.t`*xx*) or linked (`.c`*xx*) file.

Options:

`B` *filename*	Specifies external bootstrap loader program.
`C`	Generates code for C, FORTRAN, and Pascal configurer programs.
`E`	Reverses error behaviour when the halt-on-error flag becomes set.
`I`	Displays brief progress information.
`M`	Disables binary map file.
`O` *filename*	Specifies output file.
`P`	Creates new bootstrap loader program.
`R`	Prevents addition of bootstrap code.
`S` *size*	Defines run time stack size for C, FORTRAN and Pascal programs.
`V`	Displays detailed (verbose) progress information.

`icheck` – syntax checker

Checks occam source for correct syntax. Generates no object code.

Syntax: `icheck` *filename* {*options*}

where: *filename* is an occam source (`.occ`) file.

Options:

`A`	Disables alias checking.
`B`	Displays brief error messages.
`G`	Permits use of transputer instructions (restricted set).
`H`	Checks for HALT mode.
`I`	Displays progress information.
`N`	Disables usage checking.
`R` *filename*	Redirects error messages to a file.
`S`	Checks for STOP mode.
`T2/T212` `T222/M212`	Checks for IMS 16-bit processors.
`T4/T414`	Checks for IMS T414 processor. Default.
`T5/T425`	Checks for IMS T425 processor.
`T8/T800`	Checks for IMS T800 processor.
`TA`	Checks for transputer class TA (T414/T425/T800).
`TB`	Checks for transputer class TB (T414/T425).
`TC`	Checks for transputer class TC (T425/T800).
`U`	Checks for UNDEFINED mode.
`W`	Permits use of transputer instructions (full set).
`X`	Checks for UNIVERSAL mode.
Plus: All compiler options.	

iconf – configurer

Generates bootable code for multitransputer programs.

Syntax: `iconf` *filename* {*options*}

where: *filename* is a configuration description (`.pgm`) file.

Options:

`A`	Disables alias/usage checking in occam source.
`C`	Performs syntax check only.
`G`	Permits use of transputer instructions (restricted set).
`H`	Compiles occam source in HALT mode. Default.
`I`	Displays progress information.
`K`	Disables range checking in occam source.
`L`	Loads the tool and terminates.
`M`	Generates configuration map file.
`N`	Disables usage checking.
`O` *filename*	Specifies output file.
`R`	Generates boot from ROM code.
`S`	Compiles occam source in STOP mode.
`U`	Compiles occam source in UNDEFINED mode.
`V`	Disables separate vector space.
`W`	Permits use of transputer instructions (full set).
`X`	Compiles occam source in UNIVERSAL mode.
Some options are mutually exclusive (e.g. `H` and `S`).	

idebug – debugger

Provides post-mortem debugging of occam programs at source code level (Symbolic functions) and at assembly code level (Monitor page functions).

Syntax: `idebug` *filename* {*options*}

where: *filename* is a `.b`*xx* or `.btl` bootable file.

Options:

`A`	Analyse subsystem. Directs the debugger to assert **Analyse** on the network.
`D`	Debugs a dummy network.
`N` *networkdumpfile*	Debugs a program from the file *networkdumpfile*.
`R` *memdumpfile*	Debugs a program that uses the root processor. *memdumpfile* contains the memory contents of the root transputer.
`T` *linknumber*	Debugs a program that does not use the root processor, down the link identified by *linknumber*.

idump – memory dumper

Writes the contents of the root transputer's memory to disk. Used to debug programs that use the root transputer.

Syntax: **idump** *filename* *memsize* {*offset length*}

where: *filename* is the bootable program file; *memsize* is the amount of memory measured in bytes to be dumped to the file; *offset* is a byte offset from the start of memory; *length* is the number of bytes of memory starting at *offset* to be written to the file in addition to *memsize*.

ilibr – librarian

Builds libraries of code from separate files. Libraries are referenced in source code by the **#USE** directive.

Syntax: **ilibr** {*filenames*} {*options*}

where: *filenames* is a list of `.t`*xx*, `.c`*xx*, or `.lib` files.

Options:

`D`	Includes backtrace debugging data only.
`F` *filename*	Specifies library indirect file.
`I`	Displays progress information.
`O` *filename*	Specifies output file.
`X`	Explodes library into constituent files. Library is not deleted. Also writes a library indirect file.

`ilink` – linker

Links object files together, resolving external references, and produces a single linked object file. Accepts compiled code, linked code or library code, and permits prelinking of program modules.

Syntax: `ilink` {*filenames*} {*options*}

where: *filenames* is a list of `.t`*xx*, `.c`*xx*, or `.lib` files.

Options:

`A`	Displays buffer sizes.
`B`	Sets buffer sizes.
`E`	Extends linker capacity. Reduces linker speed.
`F` *filename*	Specifies linker indirect file.
`I`	Displays progress information.
`L`	Loads the tool and terminates.
`M`	Disables the file map.
`O` *filename*	Specifies output file.
`Q (`*symbol* `, ...)`	Optimizes selected library functions.
`S`	Writes a symbol table.
`U`	Allows unresolved references.
`V`	Displays verbose progress information.

`ilist` – binary lister

Decodes and displays information from object files and bootable files.

Syntax: `ilist` *filename* {*options*}

where: *filename* is an object or bootable file.

Options:

`C`	Displays code in hexadecimal.
`D`	Displays debugging data.
`E`	Displays entry point table.
`I`	Displays progress information.
`M`	Displays module data.
`O` *filename*	Specifies output file.
`P`	Displays procedural interface data. Default.
`S (`*m1,m2 ...*`)`	Selects modules for display.
`T`	Displays file tokens.
`V`	Displays global data.
`X`	Displays external reference table.

imakef – Makefile generator

Creates Makefiles for object files and bootable files created by the toolset. Also creates library usage files.

Syntax: `imakef` {*filenames*} {*options*}

where: *filenames* is a list of target object or bootable files. Target files can be any of the following: `.btl`; `.btr`; `.b`*xx*; `.c`*xx*; `.r`*xx*; `.t`*xx*; `.lib`; `.liu`.

Options:

`I`	Displays progress information.
`O` *filename*	Specifies output file.

iserver – program loader

Loads programs onto transputers and transputer boards and provides the run-time environment for communication with the host.

Syntax: `iserver` {*options*}

Options:

`SA`	Analyses root transputer and peeks 8K of memory.
`SB` *filename*	Loads the program in the specified file.
`SC` *filename*	Loads the specified file to the network via the root transputer link.
`SE`	Terminates the server if the error flag is set.
`SI`	Displays progress information.
`SL` *name*	Specifies link address or device name.
`SR`	Resets the root transputer.
`SS`	Provides access to host services.
Option `SB` is equivalent to invoking the following combination of options: `SR SS SI SC` *filename*.	

`isim` – T414 simulator

Simulates the operation of the T414 processor in HALT mode. For debugging programs without hardware. Requires a bootable file for a single transputer.

Syntax: `isim` *filename programparameters*

where: *filename* is a `.b4h` file.

`iskip` – skip loader

Allows programs to be loaded onto transputer networks beyond the root transputer. Used for loading programs onto external networks that are connected to the host via a root transputer, and used when debugging programs that use the root transputer.

Syntax: `iskip` *linknumber* {*options*}

where: *linknumber* is the root transputer link to which the target network is connected.

Options:

`E`	Monitors the subsystem error flag and terminates the server when the flag is set by the program.
`I`	Displays progress information as the tool runs.
`R`	Reset subsystem. Resets the network of transputers connected to *linknumber*. Does not reset the root transputer.

occam– occam 2 compiler

The occam 2 compiler. Compiles occam 2 source into object files that are used as input to `iboot`, `iconf`, `ilink`, and `ilibr`.

Syntax: `occam` *filename* {*options*}

where: *filename* is an occam source (`.occ`) file.

Options:

`A`	Disables alias checking.
`C`	Runs syntax checking only.
`D`	Disables debugging data.
`E`	Disables compiler libraries.
`G`	Permits use of transputer instructions (restricted set).
`H`	Generates HALT code.
`I`	Displays progress information.
`L`	Loads the compiler and terminates.
`N`	Disables usage checking.
`O` *outputfile*	Specifies output filename.
`S`	Generates STOP code.
`T2/T212` `T222/M212`	Checks for IMS 16-bit processors.
`T4/T414`	Checks for IMS T414 processor. Default.
`T5/T425`	Checks for IMS T425 processor.
`T8/T800`	Checks for IMS T800 processor.
`TA`	Checks for transputer class TA (T414/T425/T800).
`TB`	Checks for transputer class TB (T414/T425).
`TC`	Checks for processors in transputer class TC (T425/T800).
`U`	Generates UNDEFINED code.
`V`	Disables separate vector space.
`W`	Permits use of transputer instructions (full set).
`X`	Generates UNIVERSAL code.

Debugger commands

Symbolic functions

Function	Description
$\boxed{\text{INSPECT}}$	Inspect symbol.
$\boxed{\text{CHANNEL}}$	Locate to process waiting on channel.
$\boxed{\text{TOP}}$	Locate to last error or location.
$\boxed{\text{RETRACE}}$	Retrace last operation.
$\boxed{\text{RELOCATE}}$	Locate back to last location line.
$\boxed{\text{INFO}}$	Display extra information.
$\boxed{\text{SEARCH}}$	Search for string.
$\boxed{\text{LINKS}}$	Display link connections.
$\boxed{\text{MONITOR}}$	Change to Monitor page.
$\boxed{\text{BACKTRACE}}$	Locate to procedure or function call.
$\boxed{\text{HELP}}$	Display function keys.
$\boxed{\text{GET ADDRESS}}$	Display address of source line.
$\boxed{\text{CHANGE FILE}}$	Display different source file.
$\boxed{\text{ENTER FILE}}$	Display included file.
$\boxed{\text{EXIT FILE}}$	Display enclosing file.
$\boxed{\text{GOTO LINE}}$	Go to specific line.
$\boxed{\text{TOP OF FILE}}$	Go to first line.
$\boxed{\text{BOTTOM OF FILE}}$	Go to last line.
$\boxed{\text{FINISH}}$	Quit.

Monitor page functions

Key	Meaning	Description
A	ASCII	View memory in ASCII.
C	Compare	Compare real with expected code.
D	Disassemble	Display transputer instructions.
E	Next Error	Go to next processor with error flag set.
G	Goto process	Enter source level debugging for a process.
H	Hex	View memory in hexadecimal.
I	Inspect	View memory in any occam type.
L	Links	Display processes waiting on links or the **Event** pin.
M	Memory map	Display transputer's memory map.
N	Network dump	Dump network memory.
O	occam	Resume symbolic debugging.
P	Processor	Change processor.
Q	Quit	Quit debugger.
R	Run queue	Display active process queues.
T	Timer queue	Display timer queues.
X	Exit	Change to symbolic mode.
?	Help	Display help information.

Debugger Monitor page functions (contd)

Function	Description
RETRACE	As symbolic mode.
RELOCATE	As symbolic mode.
CURSOR UP	Move cursor up.
CURSOR DOWN	Move cursor down.
LINE UP	Next line.
LINE DOWN	Previous line.
PAGE UP	Previous page.
PAGE DOWN	Next page.
CURSOR	Scroll the currently displayed processor.
CURSOR RIGHT	Move cursor right.
CODE INFO	Display help information.
REFRESH	Re-draw the screen.
TOP	Find last instruction executed.

Simulator commands

Symbolic functions

Function	Description
BACKTRACE	Locate to procedure or function call.
BOTTOM OF FILE	Go to last line.
CHANGE FILE	Display different source file.
CHANNEL	Locate to process waiting on channel.
CODE INFORMATION	Display function keys.
ENTER FILE	Display included file.
EXIT FILE	Display enclosing file.
GET ADDRESS	Display address of source line.
GOTO LINE	Go to specific line.
INFO	Display extra information.
INSPECT	Display information about symbol.
LINKS	Display link connections.
MONITOR	Change to Monitor page.
RELOCATE	Locate back to last location line.
RETRACE	Retrace last operation.
SEARCH	Search for string.
SET BREAK	Set or remove break point.
SINGLE STEP	Scroll source line by line.
TOP OF FILE	Go to first line.
WALK	As SINGLE STEP but prevents the process descheduling.

Monitor page functions

Key	Meaning	Description
A	ASCII	View memory in ASCII.
B	Break points	Break point menu.
C	Load debug	Loads debugging data for a specific module.
D	Disassemble	Display transputer instructions.
G	Go	Run/resume program.
H	Hex	View memory in hexadecimal.
I	Inspect	Display a portion of memory in any occam type.
L	Links	Display processes waiting on links or the **Event** pin.
M	Memory map	Display transputer's memory map.
O	occam	Resume symbolic debugging.
P	Procedures	Display occam procedures and addresses.
Q	Quit	Quit simulator.
R	Run queue	Display active process queues.
S	Single step	Single step one transputer instruction.
T	Timer queue	Display timer queues.
U	Assign register	Assign value to register.
X	Boot	Load program and run bootstrap.
?	Help	Display help information.
CURSOR UP		Scroll display.
CURSOR DOWN		Scroll display.
CODE INFO		Display help information.
REFRESH		Redraw the screen.
FINISH		Quit simulator.

Libraries

User libraries

Library	Description
`hostio.lib`	Host file server library.
`streamio.lib`	Stream i/o library.
`snglmath.lib`	Single length maths library.
`dblmath.lib`	Double length maths library.
`tbmaths.lib`	T414/T425 optimised maths functions.
`string.lib`	String handling library.
`convert.lib`	Type conversion library.
`xlink.lib`	Extraordinary link handling library.
`crc.lib`	Block CRC library.
`process.lib`	Process library.

Include files

File	Contents
`hostio.inc`	Host file server constants.
`streamio.inc`	Stream i/o constants.
`mathvals.inc`	Maths and trigonometric constants.
`linkaddr.inc`	Transputer link addresses.

Compiler libraries

File	Processor type/error mode
`occam2h.lib`	T212/T222/M212 halt.
`occam2s.lib`	T212/T222/M212 stop.
`occam2u.lib`	T212/T222/M212 undefined.
`occambh.lib`	T414/T425 halt.
`occambs.lib`	T414/T425 stop.
`occambu.lib`	T414/T425 undefined.
`occam8h.lib`	T800 halt.
`occam8s.lib`	T800 stop.
`occam8u.lib`	T800 undefined.

Hostio library

#USE "hostio.lib"

Procedure	Parameter Specifiers
af.to.sp	CHAN OF SP fs, ts, CHAN OF ANY from.user, to.user, VAL BOOL pass.terminate
so.ask	CHAN OF SP fs, ts, VAL []BYTE prompt, replies, VAL BOOL display.possible.replies, VAL BOOL echo.reply, INT reply.number
so.buffer	CHAN OF SP fs, ts, from.user, to.user, CHAN OF BOOL stopper
so.close	CHAN OF SP fs, ts, VAL INT32 streamid, BYTE result
so.commandline	CHAN OF SP fs, ts, VAL BYTE all, INT length, []BYTE string, BYTE result
so.core	CHAN OF SP fs, ts, VAL INT32 offset, INT length, []BYTE data, BYTE result
so.date.to.ascii	VAL [so.date.len]INT date, VAL BOOL long.years, VAL BOOL days.first, [so.time.string.len]BYTE string
so.eof	CHAN OF SP fs, ts, VAL INT32 streamid, BYTE result
so.exit	CHAN OF SP fs, ts, VAL INT32 status
so.ferror	CHAN OF SP fs, ts, VAL INT32 streamid, INT32 error.no, INT length, []BYTE message, BYTE result

Procedure	Parameter Specifiers
so.flush	CHAN OF SP fs, ts, VAL INT32 streamid, BYTE result
so.fwrite.char	CHAN OF SP fs, ts, VAL INT32 streamid, VAL BYTE char, BYTE result
so.fwrite.hex.int	CHAN OF SP fs, ts, VAL INT32 streamid, VAL INT n, width, BYTE result
so.fwrite.hex.int64	CHAN OF SP fs, ts, VAL INT32 streamid, VAL INT64 n, VAL INT width, BYTE result
so.fwrite.int	CHAN OF SP fs, ts, VAL INT32 streamid, VAL INT n, field, BYTE result
so.fwrite.int64	CHAN OF SP fs, ts, VAL INT32 streamid, VAL INT64 n, VAL INT field, BYTE result
so.fwrite.nl	CHAN OF SP fs, ts, VAL INT32 streamid, BYTE result
so.fwrite.real32	CHAN OF SP fs, ts, VAL INT32 streamid, VAL REAL32 r, VAL INT Ip, Dp, BYTE result
so.fwrite.real64	CHAN OF SP fs, ts, VAL INT32 streamid, VAL REAL64 r, VAL INT Ip, Dp, BYTE result
so.fwrite.string	CHAN OF SP fs, ts, VAL INT32 streamid, VAL []BYTE string, BYTE result
so.fwrite.string.nl	CHAN OF SP fs, ts, VAL INT32 streamid, VAL []BYTE string, BYTE result

Procedure	Parameter Specifiers
so.getenv	CHAN OF SP fs, ts, VAL []BYTE name, INT length, []BYTE value, BYTE result
so.getkey	CHAN OF SP fs, ts, BYTE key, result
so.gets	CHAN OF SP fs, ts, VAL INT32 streamid, INT length, []BYTE data, BYTE result
so.multiplexor	CHAN OF SP fs, ts, []CHAN OF SP from.user, to.user, CHAN OF BOOL stopper
so.open	CHAN OF SP fs, ts, VAL []BYTE name, VAL BYTE type, mode, INT32 streamid, BYTE result
so.open.temp	CHAN OF SP fs, ts, VAL BYTE type, [so.temp.filename.length] BYTE filename, INT32 streamid, BYTE result
so.overlapped.buffer	CHAN OF SP fs, ts, from.user, to.user, CHAN OF BOOL stopper
so.overlapped.multiplexor	CHAN OF SP fs, ts, []CHAN OF SP from.user, to.user, CHAN OF BOOL stopper, []INT queue

Procedure	Parameter Specifiers
so.parse.command.line	CHAN OF SP fs, ts, VAL [][]BYTE option.strings, VAL []INT option.parameters.required, []BOOL option.exists, [][2]INT option.parameters, INT error.len, []BYTE line
so.pollkey	CHAN OF SP fs, ts, BYTE key, result
so.popen.read	CHAN OF SP fs, ts, VAL []BYTE filename, VAL []BYTE path.variable.name, VAL BYTE open.type, INT full.len, []BYTE full.name, INT32 streamid, BYTE result
so.puts	CHAN OF SP fs, ts, VAL INT32 streamid, VAL []BYTE data, BYTE result
so.read	CHAN OF SP fs, ts, VAL INT32 streamid, INT length, []BYTE data
so.read.echo.any.int	CHAN OF SP fs, ts, INT n, BOOL error
so.read.echo.hex.int	CHAN OF SP fs, ts, INT n, BOOL error
so.read.echo.hex.int64	CHAN OF SP fs, ts, INT64 n, BOOL error
so.read.echo.int	CHAN OF SP fs, ts, INT n, BOOL error
so.read.echo.int64	CHAN OF SP fs, ts, INT64 n, BOOL error
so.read.echo.line	CHAN OF SP fs, ts, INT len, []BYTE line, BYTE result

Procedure	Parameter Specifiers
`so.read.echo.real32`	`CHAN OF SP fs, ts, REAL32 n, BOOL error`
`so.read.echo.real64`	`CHAN OF SP fs, ts, REAL64 n, BOOL error`
`so.read.line`	`CHAN OF SP fs, ts, INT len, []BYTE line, BYTE result`
`so.remove`	`CHAN OF SP fs, ts, VAL []BYTE name, BYTE result`
`so.rename`	`CHAN OF SP fs, ts, VAL []BYTE oldname, newname, BYTE result`
`so.seek`	`CHAN OF SP fs, ts, VAL INT32 streamid, VAL INT32 offset, origin, BYTE result`
`so.system`	`CHAN OF SP fs, ts, VAL []BYTE command, INT32 status, BYTE result`
`so.tell`	`CHAN OF SP fs, ts, VAL INT32 streamid, INT32 position, BYTE result`
`so.test.exists`	`CHAN OF SP fs, ts, VAL []BYTE filename, BOOL exists`
`so.time`	`CHAN OF SP fs, ts, INT32 localtime, UTCtime`
`so.time.to.ascii`	`VAL INT32 time, VAL BOOL long.years, VAL BOOL days.first [so.time.string.len]BYTE string`
`so.time.to.date`	`VAL INT32 input.time, [so.date.len]INT date`
`so.today.ascii`	`CHAN OF SP fs, ts, VAL BOOL long.years, days.first, [so.time.string.len]BYTE string`

Procedure	Parameter Specifiers
`so.today.date`	`CHAN OF SP fs, ts, [so.date.len]INT date`
`so.version`	`CHAN OF SP fs, BYTE version, host, os, board`
`so.write`	`CHAN OF SP fs, ts, VAL INT32 streamid, VAL []BYTE data, INT length`
`so.write.char`	`CHAN OF SP fs, ts, VAL BYTE char`
`so.write.hex.int`	`CHAN OF SP fs, ts, VAL INT n, width`
`so.write.hex.int64`	`CHAN OF SP fs, ts, VAL INT64 n, VAL INT width`
`so.write.int`	`CHAN OF SP fs, ts, VAL INT n, field`
`so.write.int64`	`CHAN OF SP fs, ts, VAL INT64 n, VAL INT field`
`so.write.nl`	`CHAN OF SP fs, ts`
`so.write.real32`	`CHAN OF SP fs, ts, VAL REAL32 r, VAL INT Ip, Dp`
`so.write.real64`	`CHAN OF SP fs, ts, VAL REAL64 r, VAL INT Ip, Dp`
`so.write.string`	`CHAN OF SP fs, ts, VAL []BYTE string`
`so.write.string.nl`	`CHAN OF SP fs, ts, VAL []BYTE string`

Streamio library

#USE "streamio.lib"

Procedure	Parameter Specifiers
ks.keystream.sink	CHAN OF KS keys
ks.keystream.to.scrstream	CHAN OF KS keyboard, CHAN OF SS scrn
ks.read.char	CHAN OF KS source, INT char
ks.read.int	CHAN OF KS source, INT number, char
ks.read.int64	CHAN OF KS source, INT64 number, INT char
ks.read.line	CHAN OF KS source, INT len, []BYTE line, INT char
ks.read.real32	CHAN OF KS source, REAL32 number, INT char
ks.read.real64	CHAN OF KS source, REAL64 number, INT char
so.keystream.from.file	CHAN OF SP fs, ts, CHAN OF KS keys.out, VAL []BYTE filename, BYTE result
so.keystream.from.kbd	CHAN OF SP fs, ts, CHAN OF KS keys.out, CHAN OF BOOL stopper, VAL INT ticks.per.poll
so.keystream.from.stdin	CHAN OF SP fs, ts, CHAN OF KS keys.out, BYTE result
so.scrstream.to.ANSI	CHAN OF SP fs, ts, CHAN OF SS scrn

Procedure	Parameter Specifiers
so.scrstream.to.file	CHAN OF SP fs, ts, CHAN OF SS scrn, VAL []BYTE filename, BYTE result
so.scrstream.to.stdout	CHAN OF SP fs, ts, CHAN OF SS scrn, BYTE result
so.scrstream.to.TVI920	CHAN OF SP fs, ts, CHAN OF SS scrn
ss.beep	CHAN OF SS scrn
ss.clear.eol	CHAN OF SS scrn
ss.clear.eos	CHAN OF SS scrn
ss.del.line	CHAN OF SS scrn
ss.delete.chr	CHAN OF SS scrn
ss.delete.chl	CHAN OF SS scrn
ss.down	CHAN OF SS scrn
ss.goto.xy	CHAN OF SS scrn, VAL INT x, y
ss.ins.line	CHAN OF SS scrn
ss.insert.char	CHAN OF SS scrn, VAL BYTE ch
ss.left	CHAN OF SS scrn
ss.right	CHAN OF SS scrn
ss.scrstream.copy	CHAN OF SS scrn, scrn.out
ss.scrstream.fan.out	CHAN OF SS scrn, CHAN OF SS screen.out1, screen.out2
ss.scrstream.from.array	CHAN OF SS scrn, VAL []BYTE buffer
ss.scrstream.sink	CHAN OF SS scrn
ss.scrstream.to.array	CHAN OF SS scrn, []BYTE buffer
ss.up	CHAN OF SS scrn

Procedure	Parameter Specifiers
ss.write.char	CHAN OF SS scrn, VAL BYTE char
ss.write.endstream	CHAN OF SS scrn
ss.write.hex.int	CHAN OF SS scrn, VAL INT number, field
ss.write.hex.int64	CHAN OF SS scrn, VAL INT64 number, VAL INT field
ss.write.int	CHAN OF SS scrn, VAL INT number, field
ss.write.int64	CHAN OF SS scrn, VAL INT64 number, VAL INT field
ss.write.nl	CHAN OF SS scrn
ss.write.string	CHAN OF SS scrn, VAL []BYTE str
ss.write.real32	CHAN OF SS scrn, VAL REAL32 number, VAL INT Ip, Dp
ss.write.real64	CHAN OF SS scrn, VAL REAL64 number, VAL INT Ip, Dp
ss.write.text.line	CHAN OF SS scrn, VAL []BYTE str

Single length maths library #USE "snglmath.lib"

Result(s)	Function	Parameter specifiers
REAL32	ACOS	VAL REAL32 X
REAL32	ALOG	VAL REAL32 X
REAL32	ALOG10	VAL REAL32 X
REAL32	ASIN	VAL REAL32 X
REAL32	ATAN	VAL REAL32 X
REAL32	ATAN2	VAL REAL32 X, VAL REAL32 Y
REAL32	COS	VAL REAL32 X
REAL32	COSH	VAL REAL32 X
REAL32	EXP	VAL REAL32 X
REAL32	POWER	VAL REAL32 X, VAL REAL32 Y
REAL32,INT32	RAN	VAL INT32 X
REAL32	SIN	VAL REAL32 X
REAL32	SINH	VAL REAL32 X
REAL32	TAN	VAL REAL32 X
REAL32	TANH	VAL REAL32 X

Double length maths library #USE "dblmath.lib"

Result(s)	Function	Parameter specifiers
REAL64	DACOS	VAL REAL64 X
REAL64	DALOG	VAL REAL64 X
REAL64	DALOG10	VAL REAL64 X
REAL64	DASIN	VAL REAL64 X
REAL64	DATAN	VAL REAL64 X
REAL64	DATAN2	VAL REAL64 X, VAL REAL64 Y
REAL64	DCOS	VAL REAL64 X
REAL64	DCOSH	VAL REAL64 X
REAL64	DEXP	VAL REAL64 X
REAL64	DPOWER	VAL REAL64 X, VAL REAL64 Y
REAL64,INT64	DRAN	VAL INT64 X
REAL64	DSIN	VAL REAL64 X
REAL64	DSINH	VAL REAL64 X
REAL64	DTAN	VAL REAL64 X
REAL64	DTANH	VAL REAL64 X

T414/T425 maths library #USE "tbmaths.lib"

Contains the same functions as `snglmath.lib` and `dblmath.lib`, but optimised for the IMS T414 and IMS T425 procesoors.

String library #USE "string.lib"

Result	Function	Parameter Specifiers
`INT`	`char.pos`	`VAL BYTE search, VAL []BYTE str`
`INT`	`compare.strings`	`VAL []BYTE str1, str2`
`BOOL`	`eqstr`	`VAL []BYTE s1,s2`
`BOOL`	`is.digit`	`VAL BYTE char`
`BOOL`	`is.hex.digit`	`VAL BYTE char`
`BOOL`	`is.id.char`	`VAL BYTE char`
`BOOL`	`is.in.range`	`VAL BYTE char, bottom, top`
`BOOL`	`is.lower`	`VAL BYTE char`
`BOOL`	`is.upper`	`VAL BYTE char`
`INT, BYTE`	`search.match`	`VAL []BYTE possibles, str`
`INT, BYTE`	`search.no.match`	`VAL []BYTE possibles, str`
`INT`	`string.pos`	`VAL []BYTE search, str`

Procedure	Parameter Specifiers
`append.char`	`INT len, []BYTE str, VAL BYTE char`
`append.hex.int`	`INT len, []BYTE str, VAL INT number, field`
`append.hex.int64`	`INT len, []BYTE str, VAL INT64 number, VAL INT width`
`append.int`	`INT len, []BYTE str, VAL INT number, field`
`append.int64`	`INT len, []BYTE str, VAL INT64 number, VAL INT field`
`append.real32`	`INT len, []BYTE str, VAL REAL32 number, VAL INT Ip, Dp`
`append.real64`	`INT len, []BYTE str, VAL REAL64 number, VAL INT Ip, Dp`
`append.text`	`INT len, []BYTE str, VAL []BYTE text`
`delete.string`	`INT len, []BYTE str, VAL INT start, size, BOOL not.done`
`insert.string`	`VAL []BYTE new.str, INT len, []BYTE str, VAL INT start, BOOL not.done`
`next.int.from.line`	`VAL []BYTE line, INT ptr, number, BOOL ok`
`next.word.from.line`	`VAL []BYTE line, INT ptr, INT len, []BYTE word, BOOL ok`
`str.shift`	`[]BYTE str, VAL INT start, len, shift, BOOL not.done`
`to.lower.case`	`[]BYTE str`
`to.upper.case`	`[]BYTE str`

Type conversion library `#USE "convert.lib"`

Procedure	Parameter Specifiers
BOOLTOSTRING	INT len, []BYTE string, VAL BOOL b
HEXTOSTRING	INT len, []BYTE string, VAL INT n
HEX16TOSTRING	INT len, []BYTE string, VAL INT16 n
HEX32TOSTRING	INT len, []BYTE string, VAL INT32 n
HEX64TOSTRING	INT len, []BYTE string, VAL INT64 n
INTTOSTRING	INT len, []BYTE string, VAL INT n
INT16TOSTRING	INT len, []BYTE string, VAL INT16 n
INT32TOSTRING	INT len, []BYTE string, VAL INT32 n
INT64TOSTRING	INT len, []BYTE string, VAL INT64 n
REAL32TOSTRING	INT len, []BYTE string, VAL REAL32 X, VAL INT Ip, Dp
REAL64TOSTRING	INT len, []BYTE string, VAL REAL64 X, VAL INT Ip, Dp
STRINGTOBOOL	BOOL Error, b, VAL []BYTE string
STRINGTOHEX	BOOL Error, INT n, VAL []BYTE string
STRINGTOHEX16	BOOL Error, INT16 n, VAL []BYTE string
STRINGTOHEX32	BOOL Error, INT32 n, VAL []BYTE string
STRINGTOHEX64	BOOL Error, INT64 n, VAL []BYTE string
STRINGTOINT	BOOL Error, INT n, VAL []BYTE string
STRINGTOINT16	BOOL Error, INT16 n, VAL []BYTE string
STRINGTOINT32	BOOL Error, INT32 n, VAL []BYTE string
STRINGTOINT64	BOOL Error, INT64 n, VAL []BYTE string
STRINGTOREAL32	BOOL Error, REAL32 X, VAL []BYTE string
STRINGTOREAL64	BOOL Error, REAL64 X, VAL []BYTE string

Block CRC library `#USE "crc.lib"`

Result	Function	Parameter Specifiers
INT	CRCFROMLSB	VAL []BYTE InputString VAL INT PolynomialGenerator, INT OldCRC
INT	CRCFROMMSB	VAL []BYTE InputString, VAL INT PolynomialGenerator, INT OldCRC

Link handling library `#USE "xlink.lib"`

Procedure	Parameter Specifiers
InputOrFail.c	CHAN OF ANY c, []BYTE mess CHAN OF INT kill, BOOL aborted
InputOrFail.t	CHAN OF ANY c, []BYTE mess, TIMER TIME, VAL INT t, BOOL aborted
OutputOrFail.c	CHAN OF ANY c, VAL []BYTE mess, CHAN OF INT kill, BOOL aborted
OutputOrFail.t	CHAN OF ANY c, VAL []BYTE mess, TIMER TIME, VAL INT t, BOOL aborted
Reinitialise	CHAN OF ANY c

Process library `#USE "process.lib"`

Procedure	Parameter Specifiers
debug.timer	CHAN OF INT stop
ss.B00x.stream.driver	CHAN OF SS from.user.screen, CHAN OF KS to.user.kbd, VAL INT board.type, port, baud.rate, screen.type

Compiler library user functions

Function	Result(s)	Parameter specifiers
ABS	REAL32	VAL REAL32 x
ARGUMENT.REDUCE	BOOL, INT32, REAL32	VAL REAL32 x, y, y.err
ASHIFTRIGHT	INT	VAL INT operand, places
ASHIFTLEFT	INT	VAL INT argument, places
COPYSIGN	REAL32	VAL REAL32 x, y
DABS	REAL64	VAL REAL64 x
DARGUMENT.REDUCE	BOOL, INT32, REAL64	VAL REAL64 x, y, y.err
DCOPYSIGN	REAL64	VAL REAL64 x, y
DDIVBY2	REAL64	VAL REAL64 x
DFLOATING.UNPACK	INT, REAL64	VAL REAL64 x
DFPINT	REAL64	VAL REAL64 x
DIEEECOMPARE	INT	VAL REAL64 x, y
DISNAN	BOOL	VAL REAL64 x
DIVBY2	REAL32	VAL REAL32 x
DLOGB	REAL64	VAL REAL64 x
DMINUSX	REAL64	VAL REAL64 x
DMULBY2	REAL64	VAL REAL64 x
DNEXTAFTER	REAL64	VAL REAL64 x, y
DNOTFINITE	BOOL	VAL REAL64 x
DORDERED	BOOL	VAL REAL64 x, y
DSCALEB	REAL64	VAL REAL64 x, VAL INT n
DSQRT	REAL64	VAL REAL64 x
FLOATING.UNPACK	INT, REAL32	VAL REAL32 x
FPINT	REAL32	VAL REAL32 x

Function	Result(s)	Parameter specifiers
IEEE32OP	BOOL, REAL32	VAL REAL32 x, VAL INT rm, op, VAL REAL32 y
IEEE64OP	BOOL, REAL64	VAL REAL64 x, VAL INT rm, op, VAL REAL64 y
IEEECOMPARE	INT	VAL REAL32 x, y
ISNAN	BOOL	VAL REAL32 x
LOGB	REAL32	VAL REAL32 x
LONGADD	INT	VAL INT left, right, carry.in
LONGDIFF	INT, INT	VAL INT left, right, borrow.in
LONGDIV	INT, INT	VAL INT dividend.hi, dividend.lo, divisor
LONGPROD	INT, INT	VAL INT left, right, carry.in
LONGSUB	INT	VAL INT left, right, borrow.in
LONGSUM	INT, INT	VAL INT left, right, carry.in
MINUSX	REAL32	VAL REAL32 x
MULBY2	REAL32	VAL REAL32 x
NEXTAFTER	REAL32	VAL REAL32 x, y
NORMALISE	INT, INT, INT	VAL INT hi.in, lo.in
NOTFINITE	BOOL	VAL REAL32 x
ORDERED	BOOL	VAL REAL32 x, y
REAL32EQ	BOOL	VAL REAL32 x, y
REAL32GT	BOOL	VAL REAL32 x, y
REAL32OP	REAL32	VAL REAL32 x, VAL INT op, VAL REAL32 y

Function	Result(s)	Parameter specifiers
REAL32REM	REAL32	VAL REAL32 x, REAL32 y
REAL64EQ	BOOL	VAL REAL64 x, y
REAL64GT	BOOL	VAL REAL64 x, y
REAL64OP	REAL64	VAL REAL64 x, VAL INT op, VAL REAL64 y
REAL64REM	REAL64	VAL REAL64 x, y
ROTATELEFT	INT	VAL INT argument, places
ROTATERIGHT	INT	VAL INT argument, places
SCALEB	REAL32	VAL REAL32 x, VAL INT n
SHIFTLEFT	INT, INT	VAL INT hi.in, lo.in, places
SHIFTRIGHT	INT, INT	VAL INT hi.in, lo.in, places
SQRT	REAL32	VAL REAL32 x

2D block moves

Procedure	Parameter Specifiers
CLIP2D	VAL [][]BYTE Source, VAL INT sx, sy, [][]BYTE Dest, VAL INT dx, dy, width, length
DRAW2D	VAL [][]BYTE Source, VAL INT sx, sy, [][]BYTE Dest, VAL INT dx, dy, width, length
MOVE2D	VAL [][]BYTE Source, VAL INT sx, sy, [][]BYTE Dest, VAL INT dx, dy, width, length

Low level programming

Procedure	Parameter Specifiers
CAUSEERROR	–
KERNEL.RUN	VAL []BYTE code, VAL INT entry.offset, []INT workspace, VAL INT no.of.parameters
LOAD.BYTE.VECTOR	INT here, []BYTE b.vec
LOAD.INPUT.CHANNEL	INT here, CHAN OF ANY in
LOAD.INPUT.CHANNEL.VECTOR	INT here, []CHAN OF ANY in.vec
LOAD.OUTPUT.CHANNEL	INT here, CHAN OF ANY out
LOAD.OUTPUT.CHANNEL.VECTOR	INT here, []CHAN OF ANY out.vec

INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
U.K.
Telephone (0454) 616616
TLX 444723

INMOS Corporation
P.O. Box 16000
Colorado Springs
Colorado 80935
U.S.A.
Telephone (719) 630 4000
TLX (Easy Link) 62944936

INMOS SARL
Immeuble Monaco
7 rue Le Corbusier
SILIC 219
94518 Rungis Cedex
France
Telephone (1) 46.87.22.01
TLX 201222

INMOS Japan K.K.
4th Floor No 1 Kowa Bldg
11-41 Akasaka 1-chome
Minato-ku
Tokyo 107
Japan
Telephone 03-505-2840
TLX J29507 TEI JPN

INMOS GmbH
Danziger Strasse 2
8057 Eching
West Germany
Telephone (089) 319 10 28
TLX 522645

72 TDS 199 00 April 1989