inmos®

occam 2 Toolset Handbook

SGS-THOMSON MICROELECTRONICS

INMOS is a member of the SGS–THOMSON Microelectronics Group

INMOS document number: 72 TDS 377 00

Contents

Preface

Host versions

The documentation set which accompanies the occam 2 toolset is designed to cover all host versions of the toolset:

- IMS D7305 – IBM PC compatible running MS–DOS
- IMS D4305 – Sun 4 systems running SunOS.
- IMS D6305 – VAX systems running VMS.

Toolset documentation set

The documentation set comprises the following volumes:

- *72 TDS 366 01 occam 2 Toolset User Guide*
- *72 TDS 367 01 occam 2 Toolset Reference Manual*
- *72 TDS 368 01 occam 2 Toolset Language and Libraries Reference Manual*
- *72 TDS 379 00 Performance Improvement with the DX305 occam 2 Toolset*
- *72 TDS 377 00 occam 2 Toolset Handbook* (this document)
- *72 TDS 378 00 occam 2 Toolset Master Index*

Other documents

Other documents provided with the toolset product include:

- Delivery manual giving installation data, this document is host specific.
- Release notes, common to all host versions of the toolset.
- '*occam 2 Reference Manual*' published by Prentice Hall.
- '*A Tutorial Introduction to occam Programming*' published by BSP Professional Books.

Documentation conventions

The following typographical conventions are used in this manual:

Bold type	Used to emphasize new or special terminology.
`Teletype`	Used to distinguish command line examples, code fragments, and program listings from normal text.
Italic type	In command syntax definitions, used to stand for an argument of a particular type. Used within text for emphasis and for book titles.
Braces { }	Used to denote optional items in command syntax.
Brackets [ ]	Used in command syntax to denote optional items on the command line.
Ellipsis . . .	In general terms, used to denote the continuation of a series. For example, in syntax definitions denotes a list of one or more items.
|	In command syntax, separates two mutually exclusive alternatives.

occam 2 toolset

Tool	Description
oc	The occam 2 compiler. Generates object code for specific transputer targets or transputer classes.
occonf	The configurer which generates configuration binary files from configuration descriptions.
icollect	The code collector which generates executable code files.
idebug	The network debugger which provides post-mortem and interactive debugging of transputer programs.
idump	The memory dumper tool which dumps root transputer memory for post mortem debugging.
iemit	The memory configurer tool which helps to configure the transputer memory interface.
ieprom	The EPROM formatter tool which creates executable files for loading into ROM.
ilibr	The toolset librarian which creates libraries from compiled code files.
ilink	The toolset linker which links compiled code and libraries into a single unit.
ilist	The binary lister which displays binary files in a readable form.
imakef	The Makefile generator which creates Makefiles for toolset compilations.
imap	The map tool which generates a memory map for an executable file.
iserver	The host file server which loads programs onto transputer hardware and provides host communication.
isim	The T425 simulator which allows programs to be run without hardware.
iskip	The skip loader tool which loads programs over the root transputer.

Standard file extensions

Extension	Description
`.btl`	Bootable file which can be loaded onto a transputer or transputer network. Created by `icollect`.
`.btr`	Executable code without a bootstrap. Created by `icollect` and used as input to `ieprom`.
`.cfb`	Configuration binary file. Created by `occonf`.
`.clu`	Configuration linked unit. Created by `occonf`.
`.inc`	Include file. Input to `oc` and `occonf`.
`.lbb`	Library build files which specify the components of a library to `ilibr`.
`.lib`	Library file containing a collection of binary modules. Created by `ilibr`.
`.liu`	Library usage files. Created and used by `imakef`.
`.lku`	Linked unit. Created by `ilink`.
`.lnk`	Linker indirect files which specify the components of a program to be linked to `ilink`.
`.map`	Map file output by the collector. Can be used as input to `imap`.
`.occ`	occam 2 source files.
`.pgm`	Configuration description source file. Created by the user as a text file. Input to `occonf`.
`.rsc`	Dynamically loadable file. Created by `icollect`.
`.tco`	Compiled code file. Created by `oc`.

Tools

oc – occam 2 compiler

Compiles occam 2 source code.

Syntax: oc *filename* {*options*}

where: *filename* is the occam 2 program source code.

Options:

Option	Description
Transputer type	See page 24 for a list of options to specify transputer type.
H S X	Produces code in HALT mode. This is the default compilation mode and may be omitted for HALT mode programs. Produces code in STOP mode. Produces code in UNIVERSAL mode.
A	Prevents the compiler from performing alias checking.This option also disables usage checking. The default is to perform alias checking. When alias checking is enabled, the compiler may insert run-time alias checks. Details of alias and usage checking rules are given in Appendix C of the *Language and Libraries Reference Manual* and also in the *occam 2 Reference Manual.*
B	Displays messages in brief (single line) format.
C	Disables the generation of object code. The compiler performs syntax, semantic, alias and usage checking only.
CODE *nnn*	Specifies how large to make the code buffer. If not specified, the compiler will allocate 240 Kbytes. The code buffer is expressed as Kbytes, e.g. to allocate a buffer = 100kbytes, specify CODE 100.
D	Generates minimal debugging information. The default is to produce full debugging information. Debugging data is required by the debugger and by the transputer simulator.
E	Disables the use of the compiler libraries. This prevents the compilation of some programs which require 'complicated' arithmetic such as real arithmetic on a processor which does not have a floating point unit. If this option is used and the occam code requires use of the libraries, an error is reported.
G	Enables the compiler to recognize the restricted range of transputer instructions via the ASM construct, as listed in section C.8 of the *User Guide.*

Option	Description
I	Displays additional information as the compiler runs. This information includes target and error mode, and information about directives as they are processed. The default is not to display this information.
K	Disables run-time range checking. The default is to insert run-time range checks.
N	Disables usage checking. The default is to perform usage checking. Usage checking is also disabled by option 'A'. Details of alias and usage checking rules are given in Appendix C of the *Language and Libraries Reference Manual* and also in the *occam 2 Reference Manual.*
NA	Disables the insertion of run-time checks for calls to ASSERT.
NWCA	Disables warnings when CHAN OF ANY is used.
NWGY	Disables warnings when the obsolete construct GUY is used.
NWP	Disables warnings when function or procedure parameters are not used.
NWU	Disables warnings when declared variables or routines are not used.
O *outputfile*	Specifies the name of the output file. If no output file is specified the compiler uses the current directory and input filename and adds a .tco extension.
P *filename*	Generates a map file giving details of code mapping in memory. A filename must be specified. Map files can be displayed as normal text files and are read by the imap tool.
R *filename*	Redirects error messages to a file.
U	Disables the insertion of code to perform run-time error checks. The default is to perform run-time error checks.
V	Prevents the compiler from producing code which has a separate vector space requirement. The default is to produce code which uses separate vector space. See section B.1.4 in the *Language and Libraries Reference Manual.*
W	Enables the compiler to recognize the full range of transputer instructions via the ASM construct. Transputer instructions are listed in Appendix C of the *User Guide.*
WALL	Enable all warnings which are controlled from the command line.
WD	Provides a warning whenever a name is descoped.
WO	Provides a warning whenever a run-time alias check is generated.
WQUAL	Enables software quality warnings.
Y	Disables interactive debugging. **Note**: This option also disables the 'virtual routing' facilities of the configurer.

occonf – configurer

Generates configuration binary files from configuration descriptions

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

where: *filename* is the configuration description file.

Options:

Option	Description
B	Displays messages in brief (single line) format.
C	Disables the generation of object code. The configurer performs syntax, semantic, alias and usage checking only.
CODE *nnn*	Specifies how large to make the code buffer. If not specified, the configurer will allocate 40 Kbytes. *nnn* is a value in Kbytes i.e. the 'K' suffix is not required.
G	Enables the configurer to recognize the restricted range of transputer instructions, via the **ASM** construct. See Appendix C in the *User Guide.*
GA	Generates a configuration which can be debugged using the *INQUEST* debugger. This option is incompatible with the **RA**, **RO**, **NV**, **Y**, **PRE** and **PRU** options.
H	Produces code in HALT error mode. This is the default configuration mode and may be omitted for HALT error mode programs.
I	Displays extra information as the tool runs. This information includes target and error mode, and information about directives as they are processed. The default is not to display this information.
K	Disables run-time range checking. The default is to insert run-time range checking.
NA	Disables the insertion of run-time checks for calls to **ASSERT**.
NV	Disables the 'virtual routing' capabilities. If this option is specified, then processors may only communicate with adjacent processors, and no more than 1 channel in either direction may use any transputer link.
NWCA	Disables warnings when **CHAN OF ANY** is used.
NWGY	Disables warnings when the obsolete construct **GUY** is used.
NWP	Do not warn if declared parameters are not used.
NWU	Do not warn if declared variables or routines are not used.

Option	Description
O *outputfile*	Specifies an output filename. If no output file is specified the configurer uses the input filename and adds the extension `.cfb`.
PRE	Generates a configuration which can be profiled using the *INQUEST* network execution profiler. **Note**: This option cannot be used with the **GA**, **RA**, **RO** or **PRU** options.
PRU	Generates a configuration which can be profiled using the *INQUEST* network utilization profiler. **Note**: This option cannot be used with the **GA**, **RA**, **RO** or **PRE** options.
R *filename*	Redirects error and information messages to a file.
RA	Creates a file suitable for a boot-from-ROM application in which the code and data are both loaded into RAM. Interactive debugging must be disabled using the **Y** option.
RE	Enables re-ordering of code and data layout in memory, using the `order.code`, `order.vs` and `order.ws` attributes. Also enables the `location.code`, `location.ws`, and `location.vs` attributes. This option disables the ability to use `idebug` but not the INQUEST debugger.
RO	Creates a file suitable for a boot-from-ROM application in which the code is loaded into ROM and the data is loaded into RAM. Interactive debugging must be disabled using the **Y** option.
S	Produces code in STOP error mode.
U	Disables the insertion of all extra run-time error checking. The default is to insert run-time error checks. This is a 'stronger' option than **K**, and can be used to implement the occam UNDEFINED error mode.
V	Prevents the configurer from producing code which has a separate vector space requirement. The default is to produce code which does use separate vector space.
W	Enables the configurer to recognize the full range of transputer instructions, via the **ASM** construct. See section Appendix C in the *User Guide.*
WALL	Enable all warnings.
WD	Provides a warning whenever a name is descoped.
WO	Provides a warning whenever a run-time alias check is generated.
WQUAL	Enables software quality warnings.
X	Produces code in UNIVERSAL error mode.
Y	Disables interactive debugging with `idebug`.

`icollect` – code collector

Generates bootable code files. Also used to generate non-bootable files for dynamic loading or booting from ROM.

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

where: *filename* is a configuration data file created by a configurer or a single linked unit created by `ilink`.

Options:

Option	Description
`B` *filename*	Uses a user-defined bootstrap loader program in place of the standard bootstrap. The program is specified by *filename* and must conform to the rules described in appendix E of the *Toolset Reference Manual.* This option can only be used with the '`T`' option (unconfigured mode) and cannot be used with the '`RA`' and '`RO`' options.
`BM`	Instructs the tool to use a different bootstrapping scheme, which uses the bottom of memory. This option is only valid for configured programs i.e. when the '`T`' option is *not* used.
`C` *filename*	Specifies a name for the debug data file. A filename must be supplied and is used as given. This option can only be used with the '`T`' option (unconfigured mode) and cannot be used with the '`D`' or '`K`' options.
`CM`	Instructs the collector to add a bootstrap which will clear memory during the booting and loading of the transputer network. Intended for use with parity-checked memory.
`D`	Disables the generation of the debug data file for single transputer programs. This option can only be used with the '`T`' option (unconfigured mode).
`E`	Changes the setting of the transputer Halt On Error flag. HALT mode programs are converted so that they not stop when the error flag is set, and non HALT mode programs to stop when the error flag is set. This option can only be used with the '`T`' option (unconfigured mode).
`I`	Displays progress information as the collector runs.

Option	Description
`K`	Creates a single transputer file with no bootstrap code. If no file is specified the output file is named after the input filename and given the `.rsc` extension. This option can only be used with the '`T`' option (unconfigured mode).
`M` *memorysize*	Specifies the memory size available (in bytes) on the root processor for single transputer programs. *memorysize* is specified in bytes and may be given in decimal format (optionally followed by '`K`' or '`M`' to indicate Kilobytes or Megabytes respectively), or it may be specified in hexadecimal using the '`#`' or '`$`' prefixes. This option can only be used with the '`T`' option (unconfigured mode) and results in a smaller amount of code being produced.
`O` *filename*	Specifies the output file. A filename must be supplied and is used as given.
`P` *filename*	Specifies a name for the memory map file. A filename must be supplied and is used as given. The file extension `.map` should be used when the file is to be used as input to `imap`.
`RA`	Creates a file for processing by `ieprom` into a boot from ROM file to run in RAM. If no output file is specified the filename is taken from the input file and given the `.btr` extension. This option is only necessary when using the '`T`' option (unconfigured mode) to create a ROM code file.
`RO`	Creates a file for processing by `ieprom` into a boot from ROM file to run in ROM. If no output file is specified the filename is taken from the input file and given the `.btr` extension. This option is only necessary when using the '`T`' option (unconfigured mode) to create a ROM code file.
`RS` *romsize*	Specifies the size of ROM on the root processor in bytes. Only valid when used with the '`RA`' or '`RO`' options. *romsize* is specified in bytes and may be given in decimal format (optionally followed by '`K`' or '`M`' to indicate Kilobytes or Megabytes respectively), or it may be specified in hexadecimal using the '`#`' or '`$`' prefixes. This option is only necessary when using the '`T`' option (unconfigured mode) to create a ROM code file.

Option	Description
`S` *stacksize*	Specifies the extra runtime stack size in words for single transputer programs. (For occam programs this option refers to **`stack.buffer`**). *stacksize* is specified in words and may be given in decimal format (optionally followed by '`K`' or '`M`' to indicate Kilowords or Megawords respectively), or it may be specified in hexadecimal using the '`#`' or '`$`' prefixes. This option can only be used with the '`T`' option.
`T`	Creates a bootable file for a single transputer. The input file specified on the command line must be a linked unit. This option can not be used for C programs which are linked with the *reduced* runtime library.
`Y`	Disables interactive debugging with **`idebug`** and reduces the amount of memory used. This option can only be used with the '`T`' option (unconfigured mode).

`idebug` – network debugger

Provides post-mortem and breakpoint debugging.

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

where: *bootablefile* is the bootable file to be debugged

Options:

Option	Description
`A`	Assert INMOS subsystem **Analyse**. Directs the debugger to assert **Analyse** on the sub-network connected to the root processor. Required when using B004 type boards.
`AP`	A replacement for the **`A`** option when running programs on boards from certain vendors. Asserts **Analyse** on the network connected to the root processor. Contact your supplier to see whether this option is applicable to your hardware. It does not apply to boards manufactured by INMOS.
`B` *linknumber*	Interactive breakpoint debug a network that is connected to the root processor via link *linknumber*. **`idebug`** executes on the root processor. Must be accompanied by the **`iserver`** '`SR`' option.
`C` *type*	Specify a processor type (e.g. T425) instead of a class (e.g. TA) for programs that have not been configured.
`D`	Dummy debugging session. Can be used for familiarization with the debugger or establishing memory mappings. Must be accompanied by the **`iserver`** '`SR`' option.
`GXX`	Improves symbolic debugging support for C++ source code. Should be specified when debugging C++ programs.
`I`	Display debugger version string. Must be accompanied by the **`iserver`** '`SR`' option.

Option	Description
J *#hexdigits*	Takes a hexadecimal digit sequence of up to 16 digits and replicates it throughout the data regions of a program (stack, static, heap and vectorspace as appropriate) when interactive debugging. The digit sequence *must* be preceded by a hash, '#', character. Used when breakpoint debugging configured T426 programs.
K *#hexdigits*	As the J option but includes freespace. Used when interactive debugging non-configured T426 programs.
M *linknumber*	Postmortem debug a previous interactive debugging session. **idebug** executes on the root processor. Must be accompanied by the **iserver** 'SA' option.
N *filename*	Postmortem debug a program from a network dump file *filename*, created by **idebug**. The file is assumed to have the extension **.dmp** if none is specified. Must be accompanied by the **iserver** 'SR' option.
Q *variable*	Specify environment variable used to specify the ITERM file. The default is "ITERM".
R *filename*	Postmortem debug a program that uses the root transputer. *filename* is the file that contains the contents of the root processor (created by **idump** or **isim**). The file is assumed to have the extension **.dmp** if none is supplied.
S	Ignore subsystem signals when interactive debugging.
T *linknumber*	Postmortem debug a program that does not use the root processor, on a network that is connected to link *linknumber* of the root processor. **idebug** executes on the root processor. Must be accompanied by the **iserver** 'SA' option.
XQ	Causes the debugger to request confirmation of the Quit command.

`idump` – memory dumper

Writes the root transputer's memory to a file. Used in debugging programs that use the root transputer.

Syntax: **idump** *filename memorysize* [{ *startoffset length* }]

where: *filename* is the name of the dump file to be created.

memorysize is the number of bytes, starting at the bottom of memory, to be written to the file.

startoffset is an offset in bytes from the start of memory.

length is the amount of memory in bytes, starting at *startoffset*, to be dumped in addition to *memorysize*.

`iemit` – memory interface configurer

Evaluates memory configurations.

Syntax: `iemit` *options*

Options:

Option	Description
`A`	Produce ASCII output file.
`E`	Invoke interactive mode.
`F` *filename*	Specify input memory configuration file.
`I`	Select verbose mode. In this mode the user will receive status information about what the tool is doing during operation for example, reading or writing to a file.
`O` *filename*	Specify output filename.
`P`	Produce PostScript output file.

`ieprom` – EPROM program convertor

Formats bootable code for installation by ROM loaders.

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

where: *filename* is the name of the control file.

Options:

Option	Description
`I`	Selects verbose mode. In this mode the user will receive status information about what the tool is doing during its operation, for example reading or writing to a file.
`R`	Directs `ieprom` to display the absolute address of the code reference point. This address can be used to locate within the memory map created by the `icollect` '`P`' option.

ilibr – librarian

Builds libraries of code from separate files.

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

where: *filenames* is a list of input files separated by spaces.

Options:

Option	Description
`F` *filename*	Specifies a library indirect file.
`I`	Displays progress information as the library is built.
`O` *filename*	Specifies an output file. If no output file is specified the name is taken from the first input file and a `.lib` extension is added.

ilink – linker

Links object files together, resolving external references to create a single linked unit.

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

where: *filenames* is a list of compiled files or library files.

Options:

Option	Description
Transputer type	See page 24 for a list of options to specify transputer type.
`EX`	Allows the extraction of modules without linking them.
`F` *filename*	Specifies a linker indirect file.
`H`	Generates the linked unit in HALT mode. This is the default mode for the linker and may be omitted for HALT mode programs. This option is mutually exclusive with the '`S`' option.
`I`	Displays progress information as the linking proceeds.
`KB` *memorysize*	Specifies virtual memory required in Kilobytes.
`LB`	Specifies that the output is to be generated in LFF format, for use with the `iboot` bootstrap tool and `iconf` configurer tool used in earlier INMOS toolsets. (Pre–TCOFF toolsets e.g. the Dx05 occam toolset).
`LC`	Specifies that the output is to be generated in LFF format, for use with the `iconf` tool used in earlier INMOS toolsets. (Pre–TCOFF toolsets e.g. the Dx05 occam toolset).
`ME` *entryname*	Specifies the name of the main entry point of the program and is equivalent to the `#mainentry` linker directive.
`MO` *filename*	Generates a module information file with the specified name.
`O` *filename*	Specifies an output file.
`S`	Generates the linked unit in STOP mode. This option is mutually exclusive with the '`H`' option.
`T`	Specifies that the output is to be generated in TCOFF format. This format is the default format.
`U`	Allows unresolved references.
`X`	Generates the linked unit in UNIVERSAL error mode, which can be mixed with HALT and STOP modes.
`Y`	Disables interactive debugging for occam code. Used when linking in occam modules compiled with interactive debugging disabled.

`ilist` – binary lister

Decodes and displays information from object files and bootable files.

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

where: *filenames* is a list of one or more files to be displayed.

Options:

Option	Description
`A`	Displays all the available information on the symbols used within the specified modules.
`C`	Displays the code in the specified file as hexadecimal. This option also invokes the '`T`' option by default.
`E`	Displays all exported names in the specified modules.
`H`	Displays the specified file(s) in hexadecimal format.
`I`	Displays full progress information as the lister runs.
`M`	Displays module data.
`N`	Displays information from the library index.
`O` *filename*	Specifies an output file. If more than one file is specified the last one specified is used.
`P`	Displays any procedural interfaces found in the specified modules.
`R` *reference*	Displays the library module(s) containing the specified reference. This option is used in conjunction with other option to display data for a specific symbol. If more than one library file is specified the last one specified is used.
`T`	Displays a full listing of a file in any file format.
`W`	Causes the lister to identify a file. The filename (including the search path if applicable) is displayed followed by the file type. This is the default option.
`X`	Displays all external references made by the specified modules.

`imakef` – Makefile generator

Creates Makefiles for toolset compilations.

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

where: *filenames* is a list of target files for which makefiles are to be generated.

Options:

Option	Description
`C`	This option is used when incorporating C or FORTRAN modules into the program. It specifies that the list of files to be linked is to be read from a linker indirect file. This option *must* be specified for correct C or FORTRAN operation.
`D`	Disables the generation of debugging information in compilations. The default is to compile with full debugging information.
`I`	Displays full progress information as the tool runs.
`M`	Produce compiler, linker and collector map files for `imap`.
`NI`	Files in the directories in `ISEARCH` are not put into the makefile. This means that system files are not present, making it much easier to read.
`O` *filename*	Specifies an output file. If no file is specified the output file is named after the target file and given the `.mak` extension.
`R`	Writes a deletion rule into the makefile.
`Y`	Disables interactive (breakpoint) debugging in all compilations. The default is to compile with full breakpoint debugging information.

`imap` – memory mapper

Generates a memory map for an executable file.

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

where: *filename* is the name of the file containing the map output from the collector.

Options:

Option	Description
`A`	Displays the list of symbols produced by the linker, including those symbols the linker identifies as not being used. This option will not override the '`R`' option if it is used.
`I`	Displays progress information as `imap` processes information from the input files, such as the filenames of files as they are opened and closed.
`O` *filename*	Specifies an output file.
`R`	This option reduces the amount of detail generated by `imap` in two ways: • the Module memory usage table only displays details for user modules i.e. 'USER' and 'SHARED_USER' processes. • the Symbol table excludes those symbols containing a '%' character in their name. Such symbols are normally internal symbols e.g. C runtime library symbols.
`ROM` *hex offset*	This option is only applicable to, and must be specified for, code targetted at ROM. It enables a hexadecimal offset to be specified which represents the start address of the code in ROM. This offset will be added to the start address of any code which is to run in ROM, in `imap`'s output.

`iserver` – server/loader

Loads programs onto transputers and transputer boards and serves host communications.

Syntax: `iserver` {*options*}

Options:

Option	Description
`SA`	Analyses the root transputer and peeks 8K of its memory.
`SB` *filename*	Boots the program contained in the named file.
`SC` *filename*	Copies the named file to the root transputer link.
`SE`	Terminates the server if the transputer error flag is set or a control link error message is received.
`SI`	Displays progress information as the program is loaded.
`SK` *interval*	Specifies the number of seconds between attempts to access the resource.
`SL` *name*	Specifies the capability name.
`SM`	Invokes the session manager interface.
`SP` *n*	Sets the size of memory to peek on **Analyse** to *n* Kbytes.
`SR`	Resets the root transputer and its subsystem.
`SS`	Serves the link, i.e. provides host system support to programs communicating on the host link.
`ST`	All of the following command line is passed directly to the booted program as parameters.
Option '`SB` *filename*' is equivalent to '`SR SS SI SC` *filename*'.	

`isim` – T425 simulator

Simulates the execution of a program on the IMS T425.

Syntax: `isim` *program* *[programparameters]* { *options* }

where: *program* is the program bootable file.

programparameters is a list of parameters to the program. The list of parameters may follow the `isim` 'N' option and parameters must be separated by spaces.

Options:

Option	Description
`B`	Batch mode operation. The simulator runs in line mode i.e. full display data is not provided. Commands are read in from the input stream e.g. the keyboard and executed. The commands are not echoed to the output stream e.g. the display screen, as they are executed.
`BQ`	Batch Quiet mode. The simulator automatically executes the program specified on the command line and then terminates. If an error occurs, the appropriate message will be displayed. The debugging facilities of the simulator are not available in this mode.
`BV`	Batch Verify mode. Similar to batch mode, except that the commands and prompts displayed when running the simulator in interactive mode are echoed to the output stream e.g. the display.
`I`	Displays information about the simulator as it runs.
`N`	No more options for the simulator. Any options entered after this option will be assumed to be program parameters to be passed to the program running on the simulator.

`iskip` – skip loader

Allows programs to be loaded onto transputer networks beyond the root transputer.

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

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

Options:

Option	Description
`E`	Directs `iskip` to monitor the subsystem error status and terminates when it becomes set.
`I`	Displays detailed progress information as the tool loads.
`R`	Reset subsystem.Resets all transputers connected downstream of link *linknumber*. Does *not* reset the root transputer.
`RP`	A replacement for the `R` option when running programs on boards from certain vendors. Contact your supplier to see whether this option is applicable to your hardware. It does not apply to boards manufactured by INMOS.

Transputer targets – options for `oc` & `ilink`

Option	Description
`TA`	Specifies target transputer class TA (T400, T414, T425, T426, T800, T801, T805).
`TB`	Specifies target transputer class TB (T400, T414, T425, T426)
`T212`	Specifies a T212 target processor.
`T222`	Specifies a T222 target processor. Same as `T212`
`M212`	Specifies a M212 target processor. Same as `T212`
`T2`	Same as `T212`, `T222` and `M212`
`T225`	Specifies a T225 target processor.
`T3`	Same as `T225`.
`T400`	Specifies a T400 target processor. Same as `T425`.
`T414`	Specifies a T414 target processor. This is the default processor type and may be omitted when the target processor is a T414 processor.
`T4`	Same as `T414` (default).
`T425`	Specifies a T425 target processor.
`T426`	Specifies a T426 target processor.
`T5`	Same as `T400`, `T425` and `T426`.
`T800`	Specifies a T800 target processor.
`T8`	Same as `T800`.
`T801`	Specifies a T801 target processor. Same as `T805`.
`T805`	Specifies a T805 target processor.
`T9`	Same as `T801` and `T805`.

Debugger commands

Debugger symbolic functions

BACKTRACE	Locate to the calling function or procedure.
END OF FILE	Go to the last line in the file.
CHANGE FILE	Display a different source file.
CHANNEL	Locate to the process waiting on a channel.
CONTINUE FROM †	Restart a stopped process from the current line.
ENTER FILE	Change to an included source file.
EXIT FILE	Return to the enclosing source file.
FINISH	Quit the debugger.
GET ADDRESS	Display the location of a source line in memory.
GOTO LINE	Go to a specific line in the file.
HELP	Display a summary of commonly used symbolic functions.
INFO	Display process information (e.g. instruction pointer, process descriptor, process name).
INSPECT	Display the type and value of a source code symbol.
INTERRUPT †	Force the debugger into the Monitor page without stopping the program.
MODIFY †	Change the value of a variable in memory.
MONITOR	Change to the monitor page.
RELOCATE	Locate back to the last location line.
RESUME †	Resume a process stopped at a breakpoint.
RETRACE	Undo a BACKTRACE.
SEARCH	Search for a specified string.
TOGGLE BREAK †	Set or clear a breakpoint on the current line.
TOGGLE HEX	Enables/disables hex-oriented display of constants and variables for C.
TOP	Locate back to the error or last source code location.
TOP OF FILE	Go to the first line in the file.

Note: † = Functions only available in interactive mode.

Debugger monitor page commands

Key	Meaning	Description
A*	ASCII	View a region of memory in ASCII.
B†*	Breakpoint	Display the Breakpoint menu enabling breakpoints to be set, cleared or listed.
C	Compare	Compare the code on the network with the code that should be there to ensure that the code has not been corrupted.
D*	Disassemble	Display the transputer instructions at a specified area of memory.
E	Next Error	Switch the current display information to that of the next processor in the network which has halted with its error flag set.
F*	Select file	Select a source file for symbolic display using the file-name of the object file produced for it.
G	Goto process	Goto symbolic debugging for a particular process.
H*	Hex	View a region of memory in hexadecimal.
I*	Inspect	View a region of memory in a symbolic type. Types are expressed as standard occam types.
J†*	Jump	Start or resume the application program.
K	Processor names	Display the names and types of all processors in the network.
L	Links	Display instruction pointers and process descriptors for the processes currently waiting for input or output on a transputer link, or for a signal on the Event pin.
M	Memory map	Display the memory map of the current processor.
N*	Network dump	Copy the entire state of the transputer network into a 'network dump' file in order to allow continued (off-line) debugging at a later date.
O*	Specify process	Resume the source level symbolic features of the debugger for a particular process.
P*	Processor	Switch the current display information to that of another processor.
Q	Quit	Leave the debugger and return to the host operating system.
† = Interactive mode only. * = String editing functions available for these commands.		

Key	Meaning	Description
R	Run queues	Display instruction pointers and process descriptors of the processes on either the high or low priority active process queue.
S†	Show messages	Display the Messages menu enabling the default actions of the debugger to debug support functions to be changed.
T	Timer queues	Display instruction pointers, the process descriptors and the wake-up times of the processes on either the high or low priority timer queue.
U†	Update	Update the monitor page display to reflect the current state of the processor.
V	Process names	Display the memory map of processes on the current processor.
W†*	Write	Write to any portion of memory in a symbolic type. Types are expressed as standard occam types.
X	Exit	Return to symbolic mode.
Y†	Postmortem	Change an interactive breakpoint debugging session into a post-mortem debug session.
Z	Virtual links	Display instruction pointers and process descriptors for processes waiting on the configurer's software virtual links.
?	Help	Display help information.
† = Interactive mode only. * = String editing functions available for these commands.		

Simulator commands

Key	Meaning	Description
A	ASCII	Displays a portion of memory in ASCII.
B	Break points	Breakpoint menu.
D	Disassemble	Displays transputer instructions at a specified area of memory.
G	Go	Runs (or resumes) the program.
H	Hex	Displays a portion of memory in hexadecimal.
I	Inspect	Displays a portion of memory in any occam type.
J	Jump into program	Runs (or resumes) the program. Same as **G**.
L	Links	Displays `Iptr` and `Wptr` for processes waiting for input or output on a link, or for a signal on the **Event** pin.
M	Memory map	This option is not supported for the current toolset.
N	Create dump file	Creates a core dump file.
P	Program boot	Simulates a program 'boot' onto the transputer.
Q	Quit	Quits the simulator.
R	Run queue	Displays `Iptr` and `Wptr` for processes on the high or low priority active process queues.
S	Single step	Executes the next transputer instruction.
T	Timer queue	Displays `Iptr`, `Wptr`, and wake-up times for processes on the high or low priority timer queues.
U	Assign register	Assigns a value to a register.
?	Help	Displays help information.
?†	Query state	Displays values of registers and queue pointers.
.†	Where	Displays next `Iptr` and transputer instruction.
† Batch mode commands.		

▲, ▼, PAGE UP, PAGE DOWN Scroll the display.

HELP, ? Display help information.

REFRESH Redraw the screen.

FINISH Quit the simulator.

Libraries

User libraries

Library	Description
`convert.lib`	String conversion library
`crc.lib`	Block CRC library
`dblmath.lib`	Double length mathematical functions
`debug.lib`	Debugging support library
`hostio.lib`	Host file server library
`msdos.lib`	DOS specific hostio library
`snglmath.lib`	Single length mathematical functions
`streamio.lib`	Stream I/O library
`string.lib`	String handling library
`tbmaths.lib`	T400/T414/T425/T426 optimized maths
`xlink.lib`	Extraordinary link handling library

Include files

File	Description
`hostio.inc`	Constants for the host file server interface (*hostio* library)
`linkaddr.inc`	Addresses of transputer links
`mathvals.inc`	Maths constants
`msdos.inc`	DOS specific constants
`streamio.inc`	Constants for the stream i/o interface (*streamio* library)
`ticks.inc`	Rates of the two transputer clocks

Compiler libraries

File	Processor types supported
`occam2.lib`	T212/T222/T225/M212
`occam8.lib`	T800/T801/T805
`occama.lib`	T400/T414/T425/T426/TA/TB
`occamutl.lib`	All
`virtual.lib`	All

`occamutl.lib` contains routines which are called from within some of the other compiler libraries and `virtual.lib` is used to support interactive debugging. These two libraries support all processor types and error modes.

Compiler library user functions

Maths functions

Result(s)	Function name	Parameter specifiers
REAL32	ABS	VAL REAL32 x
BOOL, INT32, REAL32	ARGUMENT.REDUCE	VAL REAL32 x, y, y.err
INT	ASHIFTLEFT	VAL INT argument, places
INT	ASHIFTRIGHT	VAL INT argument, places
REAL32	COPYSIGN	VAL REAL32 x, y
REAL64	DABS	VAL REAL64 x
BOOL, INT32, REAL64	DARGUMENT.REDUCE	VAL REAL64 x, y, y.err
REAL64	DCOPYSIGN	VAL REAL64 x, y
REAL64	DDIVBY2	VAL REAL64 x
INT,REAL64	DFLOATING.UNPACK	VAL REAL64 x
REAL64	DFPINT	VAL REAL64 x
INT	DIEEECOMPARE	VAL REAL64 x, y
BOOL	DISNAN	VAL REAL64 x
REAL32	DIVBY2	VAL REAL32 x
REAL64	DLOGB	VAL REAL64 x
REAL64	DMINUSX	VAL REAL64 x
REAL64	DMULBY2	VAL REAL64 x
REAL64	DNEXTAFTER	VAL REAL64 x, y
BOOL	DNOTFINITE	VAL REAL64 x
BOOL	DORDERED	VAL REAL64 x, y
REAL64	DSCALEB	VAL REAL64 x, VAL INT n
REAL64	DSQRT	VAL REAL64 x
INT, REAL32	FLOATING.UNPACK	VAL REAL32 x
REAL32	FPINT	VAL REAL32 x
BOOL,REAL32	IEEE32OP	VAL REAL32 x, VAL INT rm, op, VAL REAL32 y
BOOL,REAL32	IEEE32REM	VAL REAL32 x, y
BOOL, REAL64	IEEE64OP	VAL REAL64 x, VAL INT rm, op, VAL REAL64 y

Result(s)	Function name	Parameter specifiers
BOOL, REAL64	IEEE64REM	VAL REAL64 x, y
INT	IEEECOMPARE	VAL REAL32 x, y
BOOL	ISNAN	VAL REAL32 x
REAL32	LOGB	VAL REAL32 x
INT	LONGADD	VAL INT left, right, carry.in
INT, INT	LONGDIFF	VAL INT left, right, borrow.in
INT, INT	LONGDIV	VAL INT dividend.hi, dividend.lo, divisor
INT, INT	LONGPROD	VAL INT left, right, carry.in
INT	LONGSUB	VAL INT left, right, borrow.in
INT	LONGSUM	VAL INT left, right, carry.in
REAL32	MINUSX	VAL REAL32 x
REAL32	MULBY2	VAL REAL32 x
REAL32	NEXTAFTER	VAL REAL32 x, y
INT, INT, INT	NORMALISE	VAL INT hi.in, lo.in
BOOL	NOTFINITE	VAL REAL32 x
BOOL	ORDERED	VAL REAL32 x, y
BOOL	REAL32EQ	VAL REAL32 x, y
BOOL	REAL32GT	VAL REAL32 x, y
REAL32	REAL32OP	VAL REAL32 x, VAL INT op, VAL REAL32 y
REAL32	REAL32REM	VAL REAL32 x, y
BOOL	REAL64EQ	VAL REAL64 x, y
BOOL	REAL64GT	VAL REAL64 x, y
REAL64	REAL64OP	VAL REAL64 x, VAL INT op, VAL REAL64 y
REAL64	REAL64REM	VAL REAL64 x, y
INT	ROTATELEFT	VAL INT argument, places
INT	ROTATERIGHT	VAL INT argument, places
REAL32	SCALEB	VAL REAL32 x, VAL INT n

Result(s)	Function name	Parameter specifiers
INT, INT	SHIFTLEFT	VAL INT hi.in, lo.in, places
INT, INT	SHIFTRIGHT	VAL INT hi.in, lo.in, places
REAL32	SQRT	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

Bit manipulation functions

Result	Function name	Parameter Specifiers
INT	BITCOUNT	VAL INT Word, CountIn
INT	BITREVNBITS	VAL INT x, n
INT	BITREVWORD	VAL INT x

CRC functions

Result	Function name	Parameter Specifiers
INT	CRCBYTE	VAL INT data, CRCIn, generator
INT	CRCWORD	VAL INT data, CRCIn, generator

Floating point arithmetic support functions

Result(s)	Function name	Parameter Specifiers
INT	FRACMUL	VAL INT x, y
INT	ROUNDSN	VAL INT Yexp, Yfrac, Yguard
INT, INT, INT	UNPACKSN	VAL INT x

Dynamic code loading support

Procedures

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

Functions

Result(s)	Function name	Parameter Specifiers
INT	WSSIZEOF	routinename
INT	VSSIZEOF	routinename

Transputer-related procedures

Procedure	Parameter Specifiers
CAUSEERROR	()
RESCHEDULE	()

Miscellaneous operations

Procedure	Parameter Specifiers
ASSERT	VAL BOOL test

T400/T414/T425/T426 maths library `#USE "tbmaths.lib"`

Contains the same functions as `snglmath.lib` and `dblmath.lib`, but optimized for the IMS T400, T414, T425 and T426 processors.

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"

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

Hostio library #USE "hostio.lib"

Procedure	Parameter Specifiers
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 bytes.read, []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
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

Procedure	Parameter Specifiers
so.fwrite.hex.int32	CHAN OF SP fs, ts, VAL INT32 streamid, n VAL INT 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, width, BYTE result
so.fwrite.int32	CHAN OF SP fs, ts, VAL INT32 streamid, n, VAL INT width, BYTE result
so.fwrite.int64	CHAN OF SP fs, ts, VAL INT32 streamid, VAL INT64 n, VAL INT width, 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.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
so.overlapped.pri.multiplexor	CHAN OF SP fs, ts, []CHAN OF SP from.user, to.user, CHAN OF BOOL stopper []INT queue
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

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

Procedure	Parameter Specifiers
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.pri.multiplexor	CHAN OF SP fs, ts, []CHAN OF SP from.user, to.user, CHAN OF BOOL stopper
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.int32	CHAN OF SP fs, ts, INT32 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.int32	CHAN OF SP fs, ts, INT32 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
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

Procedure	Parameter Specifiers
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, VAL BOOL days.first, [so.time.string.len]BYTE string
so.today.date	CHAN OF SP fs, ts, [so.date.len]INT date
so.version	CHAN OF SP fs, ts, 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.nl	CHAN OF SP fs, ts,
so.write.hex.int32	CHAN OF SP fs, ts, VAL INT32 n, VAL INT width
so.write.hex.int64	CHAN OF SP fs, ts, VAL INT64 n, VAL INT width
so.write.hex.int	CHAN OF SP fs, ts, VAL INT n, width

Procedure	Parameter Specifiers
so.write.int32	CHAN OF SP fs, ts, VAL INT32 n, VAL INT width
so.write.int64	CHAN OF SP fs, ts, VAL INT64 n, VAL INT width
so.write.int	CHAN OF SP fs, ts, VAL INT n, width
so.write.real32	CHAN OF SP fs, ts, VAL REAL32 r, VAL INT Ip, Dp
so.write.real.64	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.int64	CHAN OF KS source, INT64 number, INT char
ks.read.int	CHAN OF KS source, INT number, 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
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.delete.chl	CHAN OF SS scrn
ss.delete.chr	CHAN OF SS scrn

Procedure	Parameter Specifiers
ss.del.line	CHAN OF SS scrn
ss.down	CHAN OF SS scrn
ss.goto.xy	CHAN OF SS scrn, VAL INT x, y
ss.insert.char	CHAN OF SS scrn, VAL BYTE ch
ss.ins.line	CHAN OF SS scrn
ss.left	CHAN OF SS scrn
ss.right	CHAN OF SS scrn
ss.scrstream.copy	CHAN OF SS scrn.in, scrn.out
ss.scrstream.fan.out	CHAN OF SS scrn, screen.out1, screen.out2
ss.scrstream.from.array	CHAN OF SS scrn, VAL []BYTE buffer
ss.scrstream.multiplexor	[]CHAN OF SS screen.in, CHAN OF SS screen.out, CHAN OF INT stopper
ss.scrstream.sink	CHAN OF SS scrn
ss.scrstream.to.array	CHAN OF SS scrn, []BYTE buffer
ss.up	CHAN OF SS scrn
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, width
ss.write.hex.int64	CHAN OF SS scrn, VAL INT64 number, VAL INT width
ss.write.int	CHAN OF SS scrn, VAL INT number, width
ss.write.int64	CHAN OF SS scrn, VAL INT64 number, VAL INT width
ss.write.nl	CHAN OF SS scrn
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.string	CHAN OF SS scrn, VAL []BYTE str
ss.write.text.line	CHAN OF SS scrn, VAL []BYTE str

String library

#USE "string.lib"

Procedure	Parameter Specifiers
append.char	INT len, []BYTE str, VAL BYTE char
append.hex.int	INT len, []BYTE str, VAL INT number, width
append.hex.int64	INT len, []BYTE str, VAL INT64 number, VAL INT width
append.int	INT len, []BYTE str, VAL INT number, width
append.int64	INT len, []BYTE str, VAL INT64 number, VAL INT width
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.word.from.line	VAL []BYTE line, INT ptr, len, []BYTE word, BOOL ok
next.int.from.line	VAL []BYTE line, INT ptr, number, BOOL ok
str.shift	[]BYTE str, VAL INT start, len, shift, BOOL not.done
to.lower.case	[]BYTE str
to.upper.case	[]BYTE str

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.id.char	VAL BYTE char
BOOL	is.in.range	VAL BYTE char, bottom, top
BOOL	is.hex.digit	VAL BYTE char
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

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, VAL INT OldCRC
INT	CRCFROMMSB	VAL []BYTE InputString, VAL INT PolynomialGenerator, VAL 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 t, VAL INT time, 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 t, VAL INT time, BOOL aborted
Reinitialise	CHAN OF ANY c

Debugging support library #USE "debug.lib"

Procedure	Parameter Specifiers
DEBUG.ASSERT	VAL BOOL assertion
DEBUG.MESSAGE	VAL []BYTE message
DEBUG.STOP	()
DEBUG.TIMER	CHAN OF INT stop

DOS specific hostio library #USE "msdos.lib"

Procedure	Parameter Specifiers
dos.call.interrupt	CHAN OF SP fs, ts, VAL INT16 interrupt, VAL[dos.interrupt.regs.size]BYTE register.block.in, BYTE carry.flag, [dos.interrupt.regs.size]BYTE register.block.out, BYTE result
dos.port.read	CHAN OF SP fs, ts, VAL INT16 port.location, BYTE value, result
dos.port.write	CHAN OF SP fs, ts, VAL INT16 port.location, VAL BYTE value, BYTE result
dos.read.regs	CHAN OF SP fs, ts, [dos.read.regs.size] BYTE registers, BYTE result
dos.receive.block	CHAN OF SP fs, ts, VAL INT32 location, INT bytes.read, []BYTE block, BYTE result
dos.send.block	CHAN OF SP fs, ts, VAL INT32 location, VAL []BYTE block, INT len, BYTE result