


**occam 2.1 Toolset
386 PC Delivery
Manual
Release: D7405A**




72 TDS 499 00 – October 1995

© SGS-THOMSON Microelectronics Limited 1995. This document may not be copied, in whole or in part, without prior written consent of SGS-THOMSON Microelectronics.

 [®], **inmos** [®], IMS, occam and DS-Link [®] are trademarks of SGS-THOMSON Microelectronics Limited.

 is a registered trademark of the SGS-THOMSON Microelectronics Group.

 Part of the software in this product was supplied by the Centre for Development of Advanced Computing (CDAC) of Pune University Campus, Ganesh Khind, Pune – 411 007, India.

SGS-THOMSON Microelectronics is grateful to Andrew Schulman et al for permission to use code published in their excellent book 'Undocumented Windows'.

WATCOM is a trademark of WATCOM International Corporation.

Windows is a trademark of Microsoft Corporation.

This product incorporates innovative techniques which were developed with support from the European Commission under the ESPRIT Projects:

- P2701 PUMA (Parallel Universal Message-passing Architectures)
- P5404 GPMIMD (General Purpose Multiple Instruction Multiple Data Machines).
- P6290 HAMLET (High Performance Computing for Industrial Applications).
- P7250 TMP (Transputer Macrocell Project).
- P7267 OMI/STANDARDS.

Document Number: 72 TDS 499 00

Contents

1	Introduction	1
1.1	Layout of this manual	1
1.2	Prerequisites for running the toolset	1
2	Installing the release	3
2.1	Installation	3
2.2	Removing unnecessary components	3
2.3	Setting up the toolset for use	4
2.3.1	Set-up checklist	4
2.3.2	Setting the FILES variable	5
2.3.3	Setting the correct path	5
2.3.4	Configuring the DOS extender	5
2.3.5	Setting a file system search path	6
2.3.6	Setting the terminal definition file	7
2.3.7	Set up the ilaunch tool	7
2.3.8	Setting up the Host interface software	7
2.3.9	Environment space	9
3	Confidence testing	11
4	Release notes	13
4.1	General	13
4.1.1	Linking with old occam libraries after upgrading the toolset	13
4.1.2	New option to oconf	13
4.1.3	oconf behavior change	14
4.1.4	Developing AServer applications	14
4.1.5	irun and compound command line arguments	14
4.1.6	Preventing the irun output window from closing	15
4.1.7	ST20450 EMI configuration examples	15
4.2	Known problems	16
4.2.1	oconf and channel arrays at outer level	16
4.2.2	oconf and ST20 default MemStart value	17
4.2.3	Locating to incorrect line in INQUEST	18
4.2.4	DOS extender	18
4.2.5	irun timeout when booting code	19
4.2.6	imakef case sensitive command line	19
4.3	ST20450 workarounds	19
4.3.1	div/rem workarounds	19
4.3.2	ALU operations workarounds	20

Contents

4.3.3	Analyze behavior	20
Appendix		21
A	DOS/4GW additional notes	23
A.1	Changing the switch mode setting	23
	A.1.1 Fine control of memory usage	24
A.2	pminfo	25
A.3	VMM	27
	A.3.1 VMM default parameters	27
	A.3.2 Changing the defaults	28
	A.3.3 The .VMC file	28

1 Introduction

This manual provides installation instructions for the IMS D7405A occam 2.1 Toolset for an IBM 386 PC host (or compatible). In addition instructions for testing the release are given.

1.1 Layout of this manual

Chapter 2 provides installation instructions for this release.

Chapter 3 contains a simple procedure to check that the installation has been done correctly.

Chapter 4 contains release notes for the toolset.

Appendix A contains details of the DOS extender used with this release.

1.2 Prerequisites for running the toolset

In order to use the IMS D7405A occam 2.1 toolset you will require:

- An IBM 386 PC (or compatible) with at least 8 Mbytes RAM;
- DOS version 5.0 (or later) and Windows 3.1 running in enhanced mode;
- Approximately 14 Mbytes of free disk space;
- A transputer development board including associated board support software.

1.2 Prerequisites for running the toolset

2 Installing the release

This release of the IMS D7405A occam 2.1 toolset comes on five 1.44 Mbyte 3.5 inch diskettes.

2.1 Installation

To install the release first insert Disk 1 into your diskette drive. The installation program, `setup.exe`, should be executed from within Windows. Assuming your diskette drive is `A:`, you should run `setup.exe` from the program manager using the following command :

```
a:\setup
```

Follow the on screen instructions and the installation will install the toolset to your hard disk. By default the toolset is installed in the root directory of the destination drive in a directory named `D7405A`. The rest of this document assumes that this is the destination directory used. If you chose to install the toolset elsewhere then you will need to modify the following instructions accordingly.

The installation process creates the directory structure shown in Table 2.1.

Directory	Contents
<code>\D7405A\TOOLS</code>	The tools.
<code>\D7405A\LIBS</code>	The toolset libraries and include files.
<code>\D7405A\EXAMPLES</code>	Example sources.
<code>\D7405A\SOURCE</code>	Released sources.
<code>\D7405A\ITERMS</code>	ITERM files.
<code>\D7405A\ASLIBS</code>	AServer development libraries.

Table 2.1 Installation directory structure

2.2 Removing unnecessary components

The release installation procedure installs everything onto the hard disk. Certain parts of the toolset release may be removed from the hard disk if disk space is limited. Table 2.2 indicates which parts of the release are essential for its correct operation. For the unnecessary components, the entire directory and its contents, including any sub-directories, may be deleted.

2.3 Setting up the toolset for use

Component	Directory	Necessary
Tools	\D7405A\TOOLS	yes
Libraries	\D7405A\LIBS	yes
Source code	\D7405A\SOURCE	no
Examples	\D7405A\EXAMPLES	no
ITERM files	\D7405A\ITERMS	yes
AServer development libraries	\D7405A\ASLIBS	no

Table 2.2 Essential components

2.3 Setting up the toolset for use

This section explains how to set up the environment necessary to use the toolset. It describes the basic changes to the system configuration file CONFIG.SYS which you should make before you attempt to use the toolset and shows how to set up the necessary environment variables.

Some tools in this toolset require Windows in order to execute; others are run from a DOS command line. The recommended use of the toolset is from a DOS prompt within Windows. For this reason it is important that the environment variables be set up such that they are visible to any Windows DOS prompts. The easiest way to achieve this is to set up the environment from DOS before starting Windows, then all DOS prompts will inherit the DOS environment. Setting up environment variables from Windows is difficult but if this is required then see the documentation on ilaunch and iset in your *Toolset Reference Manual*.^① s. 197 s. 297

The rest of this manual assumes that environment variables are set up from DOS. Note that the commands to achieve this can be added to your AUTOEXEC.BAT file so that the toolset will be set up whenever you switch on your PC. The installation program will give you the option of automatically updating the AUTOEXEC.BAT file with some of the required environment variables.

2.3.1 Set-up checklist

Table 2.3 is a checklist of the actions required to set up the toolset. The second column gives the section of this document where the action is described.

Action	Section
Set up the DOS FILES variable	2.3.2
Extend the system path	2.3.3
Configure the DOS extender	2.3.4
Set up the ISEARCH environment variable	2.3.5
Set up the ITERM environment variable	2.3.6
Set up the ilaunch tool	2.3.7
Set up the host interface software	2.3.8
Extend the DOS environment space if required	2.3.9

Table 2.3 Action checklist

2.3.2 Setting the FILES variable

The **FILES** command in your system configuration file **CONFIG.SYS** should specify at least 20 simultaneously open files. For example:

```
FILES=20
```

The PC must be rebooted to apply any change to the **FILES** command.

Any other file handling software used on the system (such as PC-NFS) should also be reset to accept at least 20 simultaneously open files.

2.3.3 Setting the correct path

To be able to use the tools you will need to add the directory **\D7405A\TOOLS** to your path.

For example, to set your path to your system commands and then the toolset (on drive **C:**), type:

```
PATH=C:\DOS;C:\D7405A\TOOLS
```

Note that the installation program will set up this environment variable if requested.

2.3.4 Configuring the DOS extender

The DOS command line based tools in this toolset run in protected mode on the PC and so require a DOS extender.

The DOS extender used by the toolset is **DOS/4GW**. This is a subset of the Rational Systems DOS extender **DOS/4G**, specially customized for use with **WATCOM C** packages.

- **Suppressing the DOS/4GW Banner**

2.3 Setting up the toolset for use

The banner that is displayed by DOS/4GW at startup can be suppressed by issuing the following command:

```
set DOS4G=quiet
```

Do not insert a space between `DOS4G` and the equals sign. A space to the right of the equals sign is optional.

- **Changing the Switch Mode Setting**

In almost all cases, DOS/4GW programs can detect the type of machine that it is running on, and automatically choose a real to protected mode switch technique. Table 2.4 lists those machines which may require further configuration of DOS/4GW. If you are using one of these machines then see Appendix A for details of the further configuration required.

Machine
NEC 98-series
Fujitsu FMR-60,-70
Hitachi B32
OKI #800
IBM PS/2 model 55

Table 2.4 Machines requiring extra configuration

- **Making use of virtual memory**

The DOS extender has a built in virtual memory manager. If you have problems with lack of memory then section A.3 describes how to make use of this feature.

2.3.5 Setting a file system search path

To enable the tools to find libraries and include files you must set up an environment variable called `ISEARCH`. This environment variable holds a path which is used by the tools to search for files. `ISEARCH` normally lists the library and 'include' file directory, `\D7405A\LIBS\`. Any user directories containing libraries or 'include' files should be added as required.

Unlike the DOS path you must add the closing backslash (`\`) to a directory name, since the string for a directory will be appended directly to the filename. Directories may be separated by a space or a semi-colon (`;`).

For example to set up `ISEARCH` to point to the standard 'include' files and libraries and to a user directory called `\MYDIR` type the following DOS command:

```
SET ISEARCH=C:\D7405A\LIBS\;C:\MYDIR\
```

Note that the installation program will set up this environment variable if requested.

2.3.6 Setting the terminal definition file

The interactive tools `iemit` and `imem450` need keyboard and screen mappings which are specified in what are known as ITERM files. The environment variable `ITERM` must be set to point at one of these. Several ITERM files are supplied including a general PC ITERM file which may be used in the following way:

```
SET ITERM=C:\D7405A\ITERMS\PCANSI.ITM
```

ITERM files are text files which describe the mappings between escape sequences and screen commands/keys. New ITERM files for non-standard terminals may be created by copying the supplied file, editing it and setting the `ITERM` environment variable accordingly.

For a description of ITERM files, see the appendices of the Toolset Reference Manual.

2.3.7 Set up the `ilaunch` tool

`ilaunch` requires a Windows driver to be installed in order for it to operate. If requested the installation program for this toolset will install the driver for you. If you prefer to do this manually then you need to add the following line to the `[386Enh]` section of Windows `system.ini` file:

```
DEVICE = C:\D7405A\TOOLS\VWRUN.386
```

The `ilaunch` tool must be running in order for the host server tool `irun` to execute from within a DOS prompt. We recommend that the `ilaunch` tool be installed in the Startup program group so that it is started automatically whenever Windows is started.

2.3.8 Setting up the Host interface software

This toolset contains host interface software collectively called the AServer which provides the means to load programs onto a processor and provides access to host resources, for example a file system. The AServer consists of a program, `irun`, and a number of ancillary services which provide access to host resources. `irun` is documented in more detail in the *Toolset Reference Manual*. Note that `irun` requires Windows in order to execute. If you intend to run `irun` from the command line within a DOS prompt then the `ilaunch` tool must also be running.

The host interface software must be configured to operate with the type of hardware interface in use. The procedure will vary depending on the interface, but it will always involve adding a line to the AServer Database file `aservdb` found in the `\D7405A\LIBS` directory of this release. An *Edit Example ASERVDB* icon is installed as part of the release to aid in modifying the `aservdb` file.

The AServer Database is a list of resources available to the `irun` server tool. In the simplest case these may be hardware resources consisting of processors connected to the host via some form of interface hardware. The AServer Database must be

2.3 Setting up the toolset for use

modified to reflect the hardware available on your system. The AServer Database is described in more detail in an appendix of the *Toolset Reference Manual*.

The general form of an AServer Database entry declaring a hardware resource is as follows:

```
| target |txcs tcp back@server |1|
```

This can be understood as follows. The name between the first two vertical bars is the name of the resource being defined (in this case `target`). This is an arbitrary string of alphanumeric characters (it may also contain underscores) and defines a name by which the resource will be known. Throughout this section the name `target` will be used. The string between the second and third vertical bars is the definition of the resource. The number between the third and fourth vertical bars is always 1 for a hardware resource. Note that all AServer Database entries are case significant.

The definition of a hardware resource connected via an OS link interface is always `txcs` followed by two parameters. `txcs` is in fact an AServer service built into `irun` which can control hardware connected via an OS-link. The two parameters select the type of interface and its parameters.

More precise instructions for configuring the various types of hardware interface are given below:

- **B300**

The B300 must be configured with the factory standard linkops port number (4047).

The AServer Database should have a line of the following form added to it:

```
| target |txcs tcp userlink@b300_server |1|
```

In this case `tcp` specifies that a B300 style interface is in use. `userlink@b300_server` specifies that the resource being defined is connected to a B300 with the internet name of `b300_server` via the subsystem with the name `userlink`.

- **B008**

The B008 should be installed according to the documentation supplied with it and the port and interrupt numbers used should be noted.

If requested the installation program for this toolset will install a Windows driver for the B008. If you prefer to do this manually then you need to add the following lines to the [386Enh] section of Windows `system.ini` file:

```
DEVICE=C:\D7405A\TOOLS\VB008ST.386  
B008BASE=200  
B008IRQ=3
```

where **200** is the port number of the B008 and **3** is the interrupt number that it will use.

The AServer Database should have a line of the following form added to it:

```
| B008 | txcs vlinkos B008 |1|
```

- **B008 in B004 mode**

The B008 may be used in B004 mode if desired. The B004 is a, now obsolete, single processor PC board. This may provide an easier set up option on some PCs since no IRQ is necessary, although the performance will not be as high as with the standard B008 driver. This setup may be required on some older B008 boards which have exhibited problems in some PCs when used with the B008 Windows driver. If you experience problems setting up the B008 to use the B008 Windows driver it is worth attempting to use its B004 mode of operation.

The B008 should be installed according to the documentation supplied with it and the port number used should be noted.

The AServer Database should have a line of the following form added to it:

```
| target |txcs wb004 #200 |1|
```

where **200** is the port number of the B008.

2.3.9 Environment space

The PC may not have enough environment space by default. This may need to be increased in order to run the toolset.

All versions of DOS allow the environment space to be increased to a maximum of 32 Kbytes. For the commands or procedures to use on your system consult the user documentation for the specific version of DOS you are using.

The **SHELL** command in the **CONFIG.SYS** file can be used to set up an environment size when the PC is booted. For example:

```
SHELL=COMMAND.COM /E:1024 /P
```

This example gives the name of the DOS command processor, sets the environment space to 1024 bytes and makes this version of the command processor permanently resident.

3 Confidence testing

This chapter describes a short procedure which may be followed to check that installation has been correctly completed. It assumes that you have already installed a transputer development or evaluation board and the associated board support software.

A simple example program is built for the T425. The example program source is supplied with the toolset, and is called `simple`.

- 1 Set the current directory to a convenient directory for performing this test. For example:

```
C>cd \mine
```

```
C>
```

- 2 Copy the files `simple.c` and `simple.pgm` to the current directory:

```
C>copy c:\d7405a\examples\manuals\simple\simple.occ
1 File(s) copied
```

```
C>copy c:\d7405a\examples\manuals\simple\simple.pgm
1 File(s) copied
```

- 3 Compile the example for the T425 (Alternatively, replace the `-t425` with the relevant option for your particular processor type).

```
C>oc simple -t425
DOS/4GW Protected Mode Run-time Version 1.97
Copyright (c) Rational Systems, Inc. 1990-1994
```

```
C>
```

The DOS/4GW start up message will not appear if you have suppressed it using the `DOS4G` environment variable.

If you have not set the `DOS4G` environment variable and the DOS/4GW start up message does not appear, or is followed by a message from the DOS extender, check the instructions given in the installation chapter on configuring the DOS extender. Another possible problem is insufficient extended memory being available. If this is the case then you will have to reduce the amount of memory being used by other programs.

- 4 Link the resulting object file with the necessary parts of the library (note that if a different option to `-t425` was used on the compiler command line then the same option should replace `-t425` on the linker command line and the associated linker file should replace `occama.lnk` if required):

```
C>ilink simple.tco -f occama.lnk hostio.lib -t425
```

3 Confidence testing

- 5 Configure the program. This stage makes use of a configuration description file which describes how the program is mapped onto the transputer. The file `simple.pgm` is such a file and describes a simple network of a single T425 with 1M of memory which is connected to the host via link 0. You will need to edit this file if your hardware configuration is different. (See the chapter of the occam Toolset User Guide entitled Configuring Transputer Programs for details of configuration.)

The command to configure the example is as follows:

```
C>occonf simple.pgm
```

- 6 Add bootstrap code to the configured file. The bootstrap code loads the application onto the transputer and starts it executing. The bootstrap is added by the collector `icollect`:

```
C>icollect simple.cfb
```

This generates a bootable file, `simple.bt1`.

- 7 Finally, the program can be run on the resource called `target`:

```
C>irun -sl target simple.bt1 -zk
```

`target` specifies the name of the transputer resource on which the program will be executed. The server looks up this name in the AServer Database.

The output 'Please type your name :' should appear in a window.

After entering your name a greeting is displayed.

4 Release notes

4.1 General

This section gives general notes on the toolset, some of which were not documented in the main toolset manuals at the time of going to press.

4.1.1 Linking with old occam libraries after upgrading the toolset

If you have upgraded from an earlier occam release and are making use of occam libraries for which you do not have the source code, e.g. supplied by a third party, then you may have problems using them with this toolset.

Section 10.3.2 of the Toolset Reference Manual supplied with this toolset describes a course of action which may help in this situation.

4.1.2 New option to `occonf`

Option	Description
<code>GD</code>	Generates a configuration which can be debugged using the <i>INQUEST</i> debugger. It differs from <code>GA</code> in that only processors with the 'nodebug' attribute explicitly set to <code>FALSE</code> in the configuration description, will have debugging enabled. This option is incompatible with the <code>GA</code> , <code>RO</code> , <code>PRE</code> and <code>PRU</code> options.

The information given here supplements that given in section 4.14 of the Toolset reference manual supplied with this toolset.

The `GD` option generates a configuration which can be debugged by the *INQUEST* debugger in interactive mode.

When the `GD` option is used, the configurer will allocate debugging kernels to all processors which have been placed with at least one process. See figure 1.1.

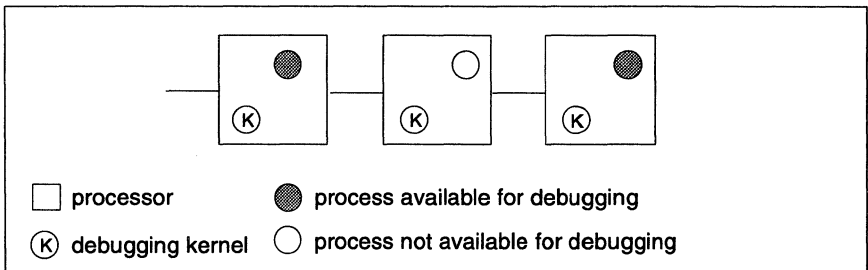


Figure 1.1 Allocation of debugging kernels using the `GD` option

4.1 General

A processor may be debugged provided the `nodebug` attribute (set in the `MAPPING` section of the configuration description) is set to `FALSE`. When the `GD` option is used the default for this attribute is `TRUE`.

The `GD` option is provided to support the debugging of dynamically loaded code and enables the user to force a debugging kernel to be loaded onto a processor even when the processor is not available for debugging.

The `GA` and `GD` options are mutually exclusive and must not be used with the `RO` option (where the processes on the root processor execute in ROM).

The `PRE` and `PRU` options are mutually exclusive and must not be used with `GD`.

The `GD` option affects the value of `LoadStart`.

4.1.3 `occonf` behavior change

In toolsets prior to this one, when `occonf` was run without specifying the `Y` option the configurator would ignore all channel placements since these would clash with the virtual routing system used by the old toolset debugger, `idebug`. In previous toolsets the `Y` option had the effect of disabling interactive debugging via `idebug`.

Now in this toolset `idebug` support has been removed in favour of `inquest` and so the check to ignore channel placements has been disabled. Thus programs that used to configure with versions of `occonf` supplied in earlier toolsets may not configure with the latest `occonf` since the configurator will now honour all the channel placements which were previously ignored and so may interfere with the normal operation of the virtual routing system.

To avoid the problem described above make sure that all channel placements which are not required are removed from your configuration description.

4.1.4 Developing AServer applications

The libraries and header files required to allow users to develop their own AServer applications can be found in the directory `aslib`.

When building hosted AServer services the `aslib` directory should be specified as an include path to the host 'C' compiler and a library search path to the host linker. When building transputer AServer programs the `aslib` directory should be included on the `ISEARCH` path.

Note that in order to use the transputer AServer Development Libraries you will also require the Dx414A ANSI C Toolset.

4.1.5 `irun` and compound command line arguments

In certain cases, it may be necessary to pass a string containing spaces to `irun` as a single argument. For example, if using the `sad` option, the argument will inevitably

contain spaces. Such compound arguments must be delimited by the character pair `\ "`. The double quote character allows `irun` to see the string as a single argument while the back-slash character prevents the command line interpreter from stripping the quote character out.

For example to define a service called `abc` using the `sad` option use the following:

```
-sad \"|abc| myservice |1|\"
```

This applies whether the option is entered on the command line within a DOS prompt or from within a dialogue box.

If it is necessary to pass a compound argument to an application program running on the target hardware then note that the command line will be parsed twice, once by `irun` itself and once by the service which supplies the command line to the application. Thus if the target program is to receive the following as a single command line argument:

```
compound argument
```

then the service which supplies the command line argument to the application must receive the following from `irun`:

```
\ "compound argument\ "
```

which means that the argument passed to `irun` must be as follows:

```
-st \\ "compound argument\\ "
```

4.1.6 Preventing the `irun` output window from closing

When serving an application `irun` displays all output written on `stdout` and `stderr` in a window. The default behavior is for `irun` to close this window when the application exits thereby losing this output. In order to make the output window persist you can use the `znt` option. The window is then closed by using the `Exit` option in the `File` menu. Alternatively all output can be saved to a file using the `sof` option.

4.1.7 ST20450 EMI configuration examples

This toolset contains example applications which allow a ST20450 EMI to be programmed from link and ROM. If you wish to use these applications then they should be placed on the path and `ISEARCH`. To add these examples to the path along with the standard tools use the following command:

```
PATH=C:\DOS;C:\D7405A\TOOLS;C:\D7405A\EXAMPLES\MEMAPP\TOOLS
```

If the above results in a path which is too long then either make use of the DOS `subst` command to abbreviate the long directory path names or copy the contents of the `\D7405A\EXAMPLES\MEMAPP\TOOLS` directory to the `\D7405A\TOOLS` directory.

To add these examples to `ISEARCH` along with the standard libraries use the following command:

```
SET ISEARCH=C:\D7405A\LIBS;C:\D7405A\EXAMPLES\MEMAPP\LIBS\
```

4.2 Known problems

If the above results in an ISEARCH path which is too long then either make use of the DOS `subst` command to abbreviate the long directory path names or copy the contents of the `\D7405A\EXAMPLES\MEMAPP\LIBS` directory to the `\D7405A\LIBS` directory.

4.2 Known problems

This section describes known problems and, where possible, workarounds which are present in this release of the toolset.

4.2.1 `occonf` and channel arrays at outer level

The occam configurer may fail to configure your network correctly if the following conditions are met:

- 1 An array of channels is defined at the outer level of the configuration.
- 2 Not all elements of the array are used.
- 3 The array element accesses use replicator index variables as subscripts.

For example:

```
CONFIG
[10]CHAN OF INT pipe:
PAR
  PROCESSOR tram
  PAR
    INT any :
    SEQ
      any := 999
      pipe[0] ! 1
      pipe[1] ? any
      pipe[5] ! any
      pipe[6] ? any
    SEQ
      PAR i = 0 FOR 1
        INT any :
        SEQ
          pipe[i] ? any
          pipe[i+1] ! any
      PAR i = 5 FOR 1
        INT any :
        SEQ
          pipe[i] ? any
          pipe[i+1] ! any
  :
```

In the above there is a "hole" in the array which is not used. i.e. elements 2 to 4. The above is not configured correctly. There are three possible workarounds:

- 1 Move the definition of the array within the processor declaration. For example:

```
CONFIG
PAR
  PROCESSOR tram
    [10]CHAN OF INT pipe:
  PAR
    ...
```

- 2 Remove the “hole” in the array by removing those elements which are not used. For example:

```
CONFIG
  [4]CHAN OF INT pipe:
  PAR
    PROCESSOR tram
  PAR
    ...
```

In the original example all accesses to elements 5 and 6 of pipe are changed to accesses to elements 2 and 3.

- 3 Define slices of the array for those elements that you require to access. For example:

```
CONFIG
  [10]CHAN OF INT pipe:
  pipe1 IS [pipe FOR 2]:
  pipe2 IS [pipe FROM 5 FOR 2]:
  PAR
    PROCESSOR tram
  PAR
    ...
```

In the original example all accesses to elements 0 and 1 of pipe are changed to accesses of pipe1 and all accesses to elements 5 and 6 of pipe are changed to accesses of pipe2.

Reference : INSDi04200

4.2.2 `occonf` and ST20 default MemStart value

`occonf` uses a default value of MemStart when the ST20 processor type is used and the MemStart value is not defined in the configuration file. This is a bug as there should be no default value and the default value used is incorrect. When using the ST20 processor type always specify Memstart explicitly.

Note that there should be no need to use the ST20 processor type as the only ST20 family processor supported by this toolset is the ST20450 and the processor type T450 exists explicitly to support this processor.

Reference : INSSy00087

4.2 Known problems

4.2.3 Locating to incorrect line in INQUEST

If a conditional statement is the last statement in a procedure then INQUEST may locate to an incorrect line when the procedure is single stepped and the conditional statement is exited. For example consider the following program:

```
PROC s()  
  SEQ  
  INT q,b:  
  SEQ  
  q := 5  
  CASE q  
  1  
    SEQ  
    b := 10  
  10  
    SEQ  
    b := 3  
  5  
    SEQ  
    b := 0  
  :
```

In the above example the debugger will locate to the statement “b := 3” when it exits the CASE statement. This is because there is no debug information associated with the area between the end of the CASE statement and the end of the procedure. The debugger does the best it can and finds the last piece of debug information generated before the end of the procedure. This debug information just happens to be associated with the second case branch.

The workaround is to place a SKIP statement between the end of the conditional statement and the end of the procedure.

Reference : INSco00037

4.2.4 DOS extender

This toolset makes use of the DOS4GW DOS extender as supplied by the WATCOM International Corporation. The version used is 1.97. In most situations this has been shown to work correctly however we have encountered some problems which seem to be cured by using an earlier version of the DOS extender (version 1.95). The particular problem encountered causes some tools to fail to operate correctly outside of Windows if Windows has been entered and exited since the last reboot of the machine.

Because of the above this toolset also contains version 1.95 of the DOS extender. This is contained in the `EXTOLD` directory which is below the `TOOLS` directory of the release tree.

If you encounter problems with any of the tools similar to that described above, it may be worth trying this earlier version of the DOS extender in the first instance to see if the problem is cured. To use it, make sure it is on the path before the version shipped in the `TOOLS` directory.

4.2.5 irun timeout when booting code

The version of `irun` in this product has a timeout when booting code onto a target network. By default if the `.btl` file fails to be copied completely, `irun` timeouts after one second and terminates. If necessary this timeout can be varied by defining an environment variable `IrunBootTestTimeout`. The value of this environment variable is the timeout period in seconds.

The `ilaunch` tool must be used to define this environment variable within Windows. In this case defining the environment variable in DOS before starting Windows will NOT work.

Reference : INSco00093

4.2.6 imakef case sensitive command line

`imakef` can fail to recognize a bootable file type if upper case characters are used in the filename extension. For example:

```
imakef HELLO.BTL
```

causes the following error message to be displayed:

```
Error-imakef-HELLO.BTL-unknown type of file
```

The workaround is to specify the file extension using all lower case characters as follows:

```
imakef HELLO.btl
```

Reference : INSco00090

4.3 ST20450 workarounds

This release of the toolset has been verified on ST20450A silicon, and contains software workarounds for some of the bugs in this revision of silicon. Later revisions of the device will fix these problems.

The workarounds are generated by the compiler when compiling code written in occam. However, no workarounds are generated when code is written in assembly language embedded in occam code using the `ASM` statement. This means that the user is responsible for ensuring the presence of the workarounds in such code.

The compiler includes an option (`ZT450AWA`) which disables these workarounds.

4.3.1 div/rem workarounds

The `div` and `rem` instructions do not work correctly in some situations if the dividend is negative.

4.3 ST20450 workarounds

The compiler works around this by replacing each `div` and `rem` instruction with a code sequence which first checks the sign of the dividend and, if it is negative, flips the sign, performs the operation and then flips the sign of the result.

Where these instructions are used explicitly in other toolset software, e.g. libraries, the workarounds have been hard coded. Unlike compiler generated workarounds, it is not possible to disable such hard coded workarounds.

Reference : INSsy00086

4.3.2 ALU operations workarounds

If a micro interrupt occurs in the first cycle of a multi-cycle ALU operation that immediately follows another ALU operation then the multi-cycle instruction is skipped and not executed.

The compiler works around this by inserting a `nop` between such pairings of ALU instructions thus ensuring that the problem does not occur.

Where these instructions are used explicitly in other toolset software, e.g. libraries, the workarounds have been hard coded. Unlike compiler generated workarounds, it is not possible to remove such hard coded workarounds from the toolset.

Reference : INSsy00085

4.3.3 Analyze behavior

ST20450A silicon can not be correctly analyzed when a high priority process or interrupt handler is executing. There are no workarounds for this bug. The profiling tools work by means of a high priority process periodically sampling the processor execution state. This means that there is a chance that the profiling tools may fail even though there are no high priority processes in the system under test.

Reference : RBSDCPU-00234

The Analyze code of `inquest/iprof/imon/iline` does not work on ST20450 TRAMS. The initial part of the boot code fails to communicate back to the host. When the error flag is set the boot operation fails. There are no workarounds to this bug, ST20450 TRAMS cannot be postmortem debugged or profiled.

Reference : INSco00089

Appendix

A DOS/4GW additional notes

This appendix describes the extra configuration of DOS/4GW which may be required on some machines. You should only need to read this appendix if you were referred here from the installation chapter.

A.1 Changing the switch mode setting

In almost all cases, DOS/4GW programs can detect the type of machine that it is running on, and automatically choose a real to protected mode switch technique. For the few cases in which this default setting does not work the `DOS16M` DOS environment variable is provided, which overrides the default setting.

The switch mode setting can be changed by issuing the following command:

```
set DOS16M=value
```

Do not insert a space between `DOS16M` and the equals sign. A space to the right of the equals sign is optional.

Table A.1 lists the machines and the settings you would use with them. The status column indicates if the setting will be automatically recognized (marked auto) or if the `DOS16M` variable must be set (marked required). For IBM PS/2 model 55's the variable may need to be set for certain machines, and so is marked optional.

Machine	Status	Setting	Comment
IBM 386 or 486 PC with DPML	auto	0	Set automatically if DPML active
NEC 98-series	required	1†	Must be set for 98-series
PS/2	auto	2	Set automatically for PS/2
IBM 386 or 486 PC	auto	3‡	Set automatically for IBM 386 or 486 PC
IBM 386 PC	auto	INBOARD	80386 with Intel Inboard
Fujitsu FMR-70	required	5	Must be set for Fujitsu FMR-70
IBM 386 or 486 PC with VCPI	auto	11	Set automatically if VCPI detected
Hitachi B32	required	14	Must be set for Hitachi B32
OKI if800	required	15	Must be set for OKI if800
IBM PS/2 model 55	optional	16	May be needed for some PS/2 model 55s

† The mnemonic "9801" may be used instead of the number.

‡ The mnemonics "386" or "80386" may also be used.

Table A.1 `DOS16M` settings

The following procedure shows how to test the switch mode settings.

- 1 If you have one of the machines listed in Table A.1, set the `DOS16M` environment variable to the value shown for the machine and specify a range of extended

A.1 Changing the switch mode setting

memory. For example, if your machine is an NEC 98-series, set `DOS16M=1@2M-4M`. See section A.1.1 for more information about setting memory usage.

Machine	Setting
NEC 98-series	1
Fujitsu FMR-60,-70	5
Hitachi B32	14
OKI if800	15

Figure A.1 `DOS16M` settings

Before running toolset applications, check the mode setting by following this procedure:

- 2 Run `pminfo` and note the switch setting reported by the last line of the display. (`pminfo`, which reports on the protected-mode resources available to your programs, is described in section A.2)

If `pminfo` runs, the setting is usable on your machine.

- 3 Add the new setting to your `AUTOEXEC.BAT` file if you needed to change them.

`pminfo` will run successfully on 80286 machines. If a program from the toolset does not run, and `pminfo` does, check the CPU type reported by the first line of the display.

A.1.1 Fine control of memory usage

In addition to setting the switch mode portion as described above, the `DOS16M` environment variable enables you to specify which portion of extended memory DOS/4GW will use. The variable also allows you to instruct DOS/4GW to search for extra memory and use it if it is present.

Specifying a range of extended memory

Normally, you do not need to specify a range of memory to use with the `DOS16M` environment variable. You must use the variable, however, in the following cases:

- You are running on a Fujitsu FMR-series, NEC 98-series OKI if800-series or Hitachi B-series machine.
- You have older programs that use extended memory, but don't follow one of the standard disciplines.

If neither of these conditions applies to you, you can skip this section.

The general syntax is:

```
set DOS16M=[switch mode][@start_address [-end_address]][[:size]]
```

In the syntax shown above, *start_address*, *end_address* and *size* represent numbers, expressed in decimal or in hexadecimal (hex requires a `0x` prefix). The number may end in a `K` to indicate an address or size in kilobytes, or an `M` to indicate megabytes. If no suffix is given then the address or size is assumed to be in kilobytes. If both a size and a range are specified, then the more restrictive interpretation is used.

The most flexible strategy is to specify only a size. However, if you are running with other software that does not follow a convention for indicating its use of extended memory, and these other programs start before DOS/4GW, you will have to calculate the range of extended memory used by the other programs and specify a range for DOS/4GW applications to use.

DOS/4GW ignores specifications (or parts of specifications) that conflict with other information about extended memory use. Below are some examples of memory control:

<code>set DOS16M=1@2m-4m</code>	Mode 1, for NEC 98-series machines, and use extended memory between 2.0 and 4.0MB.
<code>set DOS16M=:1M</code>	Use the last full megabyte of extended memory, or as much as available limited to 1MB.
<code>set DOS16M=@2M</code>	Use any extended memory available above 2MB.
<code>set DOS16M=@0-5m</code>	Use any available extended memory from 0.0 (really 1.0) to 5.0MB.
<code>set DOS16M=:0</code>	Use no extended memory.

As a default condition toolset applications take all available extended memory that is not otherwise in use. The default memory allocation strategy is to use extended memory if available, and overflow into DOS (low) memory.

In a VCPI or DPML environment, the *start_address* and *end_address* arguments are not meaningful. DOS/4GW memory under these protocols is not allocated according to specific addresses because VCPI and DPML automatically prevent address conflicts between extended memory programs. You can specify a *size* for memory managed by VCPI or DPML, but DOS/4GW will not necessarily allocate this memory from the highest available extended memory address, as it does for memory allocated under other protocols.

A.2 pminfo

Purpose: Measures the performance of protected/real-mode switching and extended memory.

Syntax: `pminfo`

Notes: The time-based measurements made by `pminfo` may vary slightly from run to run.

Example

The following example shows the output of the `pminfo` program on an 80486 AT-compatible machine.

```
Protected Mode and Extended Memory Performance Measurement -- 3.95
Copyright 1988, 1989, 1990 by Rational Systems, Inc.

DOS memory   Extended memory   CPU is 33.7 MHz 80486.
-----
      639             7040   K bytes configured (according to BIOS).
      640             7168   K bytes physically present (SETUP).
      477             7040   K bytes available for DOS/16M programs.
                               (DOS/16M memory range 1088K to 8128K)
21.2 (0.0)     21.2 (0.0)  MB/sec word transfer rate (wait states).
42.1 (0.0)     40.1 (0.5)  MB/sec 32-bit transfer rate (wait states).

Overall cpu and memory performance (non-floating point) for typical
DOS programs is 8.32 +/- 0.67 times an 8MHz IBM PC/AT.

Protected/Real switch rate = 16124/sec (62 usec/switch, 32 up + 29 down),
using DOS/16M switch mode 3 (386).
```

The top information line shows that the CPU is an Intel 80486 processor running at 33.7MHz. Below are the configuration and timings for both the DOS memory and the extended memory. If the computer is not equipped with extended memory, or none is available for DOS/4GW, the extended memory measurements may be omitted (“-”).

The line “according to BIOS” shows the information provided by the BIOS (interrupts 21h and 15h function 88h). The line “SETUP”, if displayed, is the configuration obtained directly from the CMOS RAM as set by the computers setup program. It is only displayed if the numbers are different from those in the BIOS line. They will be different for computers where the BIOS has reserved memory for itself or if another program has allocated some memory and is intercepting the BIOS configuration requests to report less memory available than is physically configured. The “DOS/16M memory range”, if displayed, shows the low and high addresses available to DOS/4GW in extended memory.

Below the configuration information is information on the memory speed (*transfer rate*). `pminfo` tries to determine the memory architecture. Some architectures will perform well under certain circumstances and poorly under others; `pminfo` will show both the best and worst cases. The architectures detected are cache, interleaved, page-mode (or static column), and direct. Measurements are made using 32-bit accesses and reported as the number of megabytes per second that can be transferred. The number of wait states is reported in parentheses. The wait states can be a fractional number, like 0.5, if there is a wait state on writes but not on reads. Memory bandwidth (i.e. how fast the CPU can access memory) accounts for 60% to 70% of the performance for typical programs (that are not heavily dependent on floating-point arithmetic).

A performance metric developed by Rational Systems is displayed, showing the expected throughput for the computer relative to a standard 8MHz IBM PC/AT (disk

accesses and floating point are excluded). Finally the speed with which the computer can switch between real and protected mode is displayed, both as the maximum number of round-trip switches that can occur per second, and the time for a single round trip switch, broken out into the real-to-protected (up) and protected-to-real (down) components.

A.3 VMM

The Virtual Memory Manager (VMM) uses a swap file on disk to augment RAM. With VMM you can use more memory than your machine actually has. When RAM is not sufficient, part of your program is swapped out to the disk file until it is needed again. The combination of the swap file and available RAM is the *virtual memory*.

Your program can use VMM if you set the DOS environment variable, **DOS4GVM**, as follows. To set the **DOS4GVM** environment variable, use the format shown below.

```
set DOS4GVM= [option[#value]] [option[#value]]
```

A “#” is used with options that take values since the DOS command shell will not accept “=“.

If you set **DOS4GVM** equal to 1, the default parameters are used for all options.

For example:

```
C>set DOS4GVM=1
```

A.3.1 VMM default parameters

VMM parameters control the options listed below:

MINMEM	The minimum amount of RAM managed by VMM. The default is 512KB.
MAXMEM	The maximum amount of RAM managed by VMM. The default is 4MB.
SWAPMIN	The minimum or initial size of the swap file. If this option is not used, the size of the swap file is based on VIRTUALSIZE (see below).
SWAPINC	The size by which the swap file grows.
SWAPNAME	The swap file name. The default name is DOS4GVM.SWP . By default the file is in the root directory of the current drive. Specify the complete path name if you want to keep the swap file somewhere else.
DELETESWAP	Whether the swap file is deleted when your program exits. By default the file is not deleted. Program startup is quicker if the file is not deleted.
VIRTUALSIZE	The size of the virtual memory space. The default is 16MB.

A.3.2 Changing the defaults

You can change the defaults in two ways.

- 1 Specify different parameter values as arguments to the `DOS4GVM` environment variable, as shown in the example below.

```
set DOS4GVM=deleteswap maxmem#8192
```

- 2 Create a configuration file with the filetype extension ".VMC", and call that as an argument to the `DOS4GVM` environment variable, as shown below.

A.3.3 The .VMC file

A ".VMC" file contains VMM parameters and settings as shown in the example below. Comments are permitted. Comments on lines by themselves are preceded by an exclamation point (!). Comments that follow option settings are preceded by white space. Do not insert blank lines because processing stops at the first blank line.

```
!Sample .VMC file
!This file shows the default parameter values.
minmem = 512           At least 512k bytes of RAM is required.
maxmem = 4096         Uses no more than 4MB of RAM.
virtualsize = 16384   Swap file plus allocated memory is 16MB
!To delete the swap file automatically when the program exits,
!add
!deleteswap
!To store the swap file in a directory called SWAPFILE, add
!swapname = c:\swapfile\dos4gvm.swp
```