inmos®

IMS D7305A IBM 386 PC occam 2 Toolset delivery manual

SGS-THOMSON MICROELECTRONICS

INMOS is a member of the SGS–THOMSON Microelectronics Group

INMOS document number: 72 TDS 389 01

Contents

1 Introduction

This manual provides installation instructions for the IMS D7305A occam 2 toolset for the IBM 386 PC (and compatibles). 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.

Appendix A contains some distribution notes for this release.

Appendix B shows the positions of the debugger and simulator function keys on a standard PC keyboard.

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

1.2 Prerequisites for running the toolset

In order to use the occam 2 toolset you will require:

- An IBM 386 PC (or compatible) with at least 4 Mbytes RAM.
- DOS version 5.0 or later.
- About 11 Mbytes of free disk space.
- An IMS B004 or B008 (or similar) transputer board with an INMOS 32-bit transputer such as an IMS T800 or T414 (Rev B or later), with 2 Mbytes or more of memory.

Note: For interactive debugging a 32-bit transputer is used to run the debugger, so to use this feature of the toolset at least one additional transputer will be needed.

1.3 Compatibility with previous releases

This release is source code compatible with the previous D7205A release of the toolset. Users should recompile all source code with this new toolset.

This release is compatible with the D7314A ANSI C toolset supplied by INMOS. Together these toolsets can be used for mixed language programming in C and occam.

2 Installing the release

This release of the IMS D7305A occam 2 toolset comes on three 1.2 Mbyte 5.25 inch floppy disks and three 1.44 Mbyte 3.5 inch floppy disks. The installation procedure is the same whether you install from the 1.2 Mbyte disks or the 1.44 Mbyte disks. You will require about 11 Mbytes of free disk space during the installation process, although the installed product only occupies about 9 Mbytes.

The whole release will be installed by the installation procedure, to make the process as easy as possible. You can delete some of the components after installation if you wish.

2.1 Installation

To install the release first insert Disk 1 in your floppy disk drive. Next run the batch file, `install.bat`, on Disk 1, giving as parameters the drive letter of the floppy disk drive and the drive on which the toolset will be installed.

For example, if your floppy disk drive is A, and the drive on which you want the toolset installed is C, type:

```
a:install a c
```

Installation will then proceed. Periodically you will be asked to insert further disks into the disk drive until all the software has been transferred to the PC hard disk.

If the installation has been successful the following messages will be displayed at the end of installation:

```
Installation complete

You may delete install2.bat
```

`INSTALL2.BAT` is the batch file which installed the release. It is no longer required and so may be deleted.

The installation procedure creates a directory called `\D7305A`. All the components of the toolset itself are copied into sub-directories of `\D7305A`, as shown in the following table:

Directory	Contents
`\D7305A\TOOLS`	The tools.
`\D7305A\LIBS`	The toolset libraries and include files.
`\D7305A\EXAMPLES`	Example sources.
`\D7305A\EXAMPLES\MANUALS`	Examples found in the user manuals.
`\D7305A\EXAMPLES\IDEBUG`	Examples for use with `idebug`.
`\D7305A\EXAMPLES\MIXED`	Mixed language example sources.
`\D7305A\EXAMPLES\MIXCONF`	Mixed language configuration example sources.
`\D7305A\EXAMPLES\OC`	occam compiler example sources.
`\D7305A\EXAMPLES\OCCONF`	Configurer example sources.
`\D7305A\SOURCE`	Released sources.
`\D7305A\SOURCE\IMAKEF`	Sources for the `imakef` tool.
`\D7305A\SOURCE\ISERVER`	Server sources.
`\D7305A\SOURCE\DRIVER`	Driver program sources.
`\D7305A\SOURCE\LIBS`	Library sources (in sub directories).
`\D7305A\SOURCE\BOOTSTRP`	Sources for the bootstraps.
`\D7305A\ITERMS`	Example ITERM files.
`\D7305A\CONNECT`	Sample iserver connection database files.

Users of the previous occam 2 toolset (IMS D7205) will notice that the `itools` and `iserver` directories no longer exist.

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 a problem. The following table indicates which parts of the release are essential for its correct operation.

Component	Necessary
Tools	yes
Libraries	yes
Iterms	yes
Connect	yes
Source code	no
Examples	no

The `source` and `examples` directories are not strictly needed for the operation of the toolset, but it is recommended that you leave them installed, at least initially, as they provide many useful examples of occam programs, some of which are referred to in the manual.

2.2 Hosted and non-hosted tools

Most of the tools in this release execute on the PC host. However three tools, namely `idebug`, `idump`, and `iskip` execute on a transputer.

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.

The commands described below to set up the toolset may be added to your `AUTOEXEC.BAT` file so that the toolset will be set up whenever you switch on your PC.

2.3.1 Setting the `FILES` variable

The `FILES` command in your system configuration file `config.sys` should be changed or added to specify at least 20 simultaneously open files. For example:

```
FILES=20
```

Note: 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.

2.3.2 Setting the correct PATH

To be able to use the tools you will need to add the directory `C:\D7305A\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:\D7305A\TOOLS
```

Note : If you are using more than one INMOS toolset then make sure that the newer toolset appears first on the path. This makes sure that you will always use the latest version of any tool which is common to all toolsets, e.g. `ilink`. To determine which toolset is the newer you can invoke the common tools from each toolset until you find an instance where the version dates differ, the tool with the newer version date belongs to the newer toolset.

2.3.3 Configuring the DOS extender

The majority of the tools which execute on the PC run in protected mode, and so require a DOS extender. This toolset is supplied with the `DOS/4GW` DOS extender. It can be found in the directory `D7305A\tools` and is executed automatically whenever you invoke one of the PC hosted tools.

`DOS/4GW` should be able to determine automatically what type of machine it is running on, and set the switch mode settings (how to transfer from real to protected mode) appropriately. However for a few machines `DOS/4GW` must be configured manually.

This toolset includes a program, `PMINFO`, which can be used to check the switch mode settings on your machine. `PMINFO` can be found in the `\D7305A\TOOLS` directory.

Run the `PMINFO` program as follows:

```
C>pminfo
```

If all is well `PMINFO` displays a report of the protected-mode resources available to your programs. The last line of the report will be something like the following (although possibly with a different switch mode value):

```
using DOS/16M switch mode 11 (VCPI)
```

If `PMINFO` does not work correctly then you may have to configure the DOS extender manually. Please read appendix C, section C.3, for details of how to do this.

`PMINFO` is described in more detail in section C.4.

Whenever a PC hosted tool starts up a message is displayed by the DOS extender. If you want to disable this message then set the `DOS4G` environment variable as follows:

```
C>set DOS4G=quiet
```

See section C.3.1 for more details.

2.3.4 Setting up the iserver

The iserver is a program which runs on the host machine and provides access to the facilities of that machine, e.g. the file system.

Selecting the required iserver

Two versions of the iserver are provided with this toolset. One which works with PC-NFS and one which works with PC/TCP. These iservers are found in the `\D7305A\TOOLS` directory with the following names:

`ISERVER.NFS`	: iserver for use with PC-NFS.
`ISERVER.TCP`	: iserver for use with PC/TCP.

If you have neither PC-NFS nor PC/TCP installed on your PC then INMOS recommend the use of the PC-NFS version.

Having selected the required iserver make a copy of it in the `\D7305A\TOOLS` directory with the name `ISERVER.EXE`.

Example: If you require the PC-NFS version of the iserver then the following commands would work:

```
C>cd \d7305a\tools

C>copy iserver.nfs iserver.exe
```

Special notes for users of the PC-NFS iserver

In order to make use of the PC-NFS iserver it may be necessary to upgrade your version of NFS. INMOS supply a program which checks your NFS installation and, if necessary, upgrades it to the level required by iserver. The upgrade entails replacing some of the files in your NFS installation with newer versions.

The upgrade program, called `upgrade.exe`, can be found in `\D7305A`. All users of the PC-NFS version of the iserver should run the `upgrade` program as shown below:

```
C>cd \d7305a

C>upgrade c:\d7305a\ nfs-mods.exe
```

The `upgrade` program asks some questions regarding your PC-NFS installation and if required will upgrade NFS. **Note**: `upgrade` will only upgrade PC-NFS if you elect to do so.

Notes common to both versions of the iserver

The iserver supplied with this toolset supersedes any previous version. Unlike previous versions which required a separate executable file for each link device supported, this iserver contains drivers for all link devices in a single executable. This means that each time the iserver is used, it must be told which driver to use. This is achieved by use of a Connection Database.

A connection database is a text file which contains a description of all the links available to the iserver, and enough information to allow the iserver to choose the correct one. Each link is given a name called the capability name.

When the iserver is executed a capability name must be supplied by the user (either explicitly or via an environment variable). The capability name is translated into a link name via the connection database.

Sample connection databases containing descriptions of some common INMOS hardware are supplied with this toolset. They can be found in the directory `C:\D7305A\CONNECT`, and are listed in the following table:

File	Supported hardware	Capability name(s)
`B004.DAT`	IMS B004 board	B004
`B008.DAT`	IMS B008 board	B008

Thus the capability name for the IMS B004 is **B004**, for the IMS B008 is **B008**. The entries in the sample connection databases are correct for boards which were installed at the default addresses and/or with the default names as described in the relevant Installation guide.

For detailed information on the format of a connection database see the iserver chapter in the *occam 2 Toolset Reference Manual*.

The iserver makes use of two environment variables, `ICONDB` and `TRANSPUTER`.

`ICONDB` specifies the whereabouts of the connection database file. This should be set to refer to the connection database which matches your hardware. e.g. to set up the connection database for a B004 use the following:

```
set ICONDB=C:\D7305A\CONNECT\B004.DAT
```

`TRANSPUTER` specifies the default capability name to use. So for the B004, `TRANSPUTER` should be set as follows:

```
set TRANSPUTER=B004
```

The default capability name can be overridden using the iserver `SL` option.

See the iserver chapter in the *occam 2 Toolset Reference Manual* for more details about the use of the iserver.

Note for users of the IMS B008 motherboard

The B008 motherboard can be accessed either directly from the iserver or via the B008 device driver, supplied with IMS S708b. If you have installed the device driver (**note** this often clashes with Mouse Drivers and is no longer recommended), then you should use a connection database based on the example `B008.DAT` file. If, however, you do not have the S708b device driver installed (this is the recommended method), then you should use a connection database based on the example `B004.DAT` file. An entry in such a connection database would be something like:

```
|B008    |T|localhost|#150|b004|||Description ...|
```

Note the use of `b004` in the 5th field. For historical reasons the iserver which accesses the B008 motherboard directly is known as the `B004` iserver.

2.3.5 Use of the iserver by transputer tool driver programs

Driver programs are supplied for use with the transputer hosted tools. These drivers invoke the iserver in order to boot the tools onto the transputer.

Normally the iserver invoked by the transputer tool driver programs is the same as that which is found on the system path. It can be changed to a different server by defining the environment variable `ISERVER`. If `ISERVER` is defined on the system then the driver programs use the iserver specified within it, otherwise the server is searched for on the default path.

The `ISERVER` variable should contain the full directory path and filename of the alternative iserver. For example, to use a server called `MYSERVER.EXE` from your `\BIN` directory on drive C, use the following definition:

```
set ISERVER=C:\BIN\MYSERVER.EXE
```

Note: Setting `ISERVER` does not affect the iserver which is used when `iserver` is invoked from the command line; it only affects use of the iserver from within driver programs.

2.3.6 Setting the board memory size

Before you can use any tool which runs on your transputer evaluation board you must set up an environment variable, `IBOARDSIZE`, giving the size of the memory on the board (in bytes). To do this use the DOS `set` command. For example, to set a board size to 2 Mbytes type:

```
set IBOARDSIZE=#200000
```

You may give either a decimal or hexadecimal (preceded by '#' and using upper case letters only) number. On keyboards without '#', the '$' character can be used instead. Leading and trailing spaces are prohibited.

If `IBOARDSIZE` is specified incorrectly, for example as a character, string or with leading or trailing spaces, the system defaults to a board size of 0 (zero) and transputer programs will fail to run. If `IBOARDSIZE` is explicitly set to a very small value a similar error may occur.

2.3.7 Setting root memory size for `idebug`

The amount of memory on the root transputer must be defined for `idebug`, using the environment variable `IDEBUGSIZE`. This variable is set up in the same way as `IBOARDSIZE` (see section 2.3.6) and should be set to the available memory. Leading and trailing spaces are prohibited.

The debugger requires at least 1 Mbyte of available root transputer memory: it is strongly recommended that 2 Mbytes or more be available.

2.3.8 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 will normally give the standard library and include file directory (`\D7305A\LIBS\`) and any user directories as required. Note that unlike the DOS path you must add the closing backslash, '\', to a directory name. 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:\D7305A\LIBS\;C:\MYDIR\
```

Note : All INMOS toolsets use `ISEARCH`. If you are using more than one INMOS toolset then `ISEARCH` should be set up to point to the libraries and include files for each toolset plus any other user directories. The most recent toolset should appear first in the list.

2.3.9 Setting the device driver and terminal definition file

In order to use the interactive tools it is necessary to install a device driver for the screen. The device driver to use is ANSI.SYS which is supplied as part of DOS. It is likely that ANSI.SYS is already installed on your PC, if not the following is an example of the line which is required in your `CONFIG.SYS` file :–

```
DEVICE=C:\DOS\ANSI.SYS
```

You will need to re-boot the PC in order for the `ANSI.SYS` device driver to be installed.

The interactive tools `idebug` and `isim` 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. An ITERM file suitable for use with the PC and ANSI.SYS is supplied and may be set up in the following way:

```
set ITERM=C:\D7305A\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 appendix D in the *occam 2 Toolset Reference Manual.*

2.3.10 Environment space

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

In DOS version 5.0 (and compatible versions) 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.

Alternatively a new command interpreter can be started from inside the normal one, specifying the new environment size:

```
COMMAND /e:1024
```

This command interpreter can be terminated by typing `EXIT` to return to the resident one. However this technique will reduce the memory available as two copies of the command interpreter will be running.

2.3.11 Setup checklist

The following 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	2.3.1
Extend the system `path`	2.3.2
Configure the DOS extender	2.3.3
Select the required version of the iserver	2.3.4
Run the PC-NFS upgrade program (PC-NFS iserver only)	2.3.4
Set up the `ICONDB` environment variable	2.3.4
Set up the `TRANSPUTER` environment variable	2.3.4
Set up the `ISERVER` environment variable (optional)	2.3.5
Set up the `IBOARDSIZE` environment variable	2.3.6
Set up the `IDEBUGSIZE` environment variable	2.3.7
Set up the `ISEARCH` environment variable	2.3.8
Set up the `ANSI.SYS` device driver	2.3.9
Set up the `ITERM` environment variable	2.3.9
Extend the DOS environment space if required	2.3.10

3 Confidence testing

This chapter describes a short procedure which may be followed to check that installation has been done correctly.

A simple example program is compiled for the TA processor class and executed on a T425 transputer. Compiling for the TA class allows the program to be run on any 32-bit transputer such as the T425.

If there is no transputer available then the program may be executed using the simulator as long as it is built for a T425 target (which this example allows because it is compiled for the TA class).

3.1 Building the example

1 Set the current disk to the same disk as the toolset has been installed on. For example, if the compiler has been installed in directory `C:\D7305A`, do this:

```
D>c:

C>
```

2 Set the current directory to a convenient directory for doing this test. For example:

```
C>mkdir \mine

C>cd \mine

C>
```

3 Copy the example files `simple.occ` and `simple.pgm` to the current directory:

```
C>copy \d7305a\examples\manuals\simple\simple.occ
    1 File(s) copied

C>copy \d7305a\examples\manuals\simple\simple.pgm
    1 File(s) copied

C>
```

4 Compile the example for the TA processor class:

```
C>oc /ta simple.occ
DOS/4GW Protected Mode Run-time   Version 1.8
Copyright (c) Rational Systems, Inc. 1990-1992

C>
```

The DOS/4GW start up message will not appear if you have suppressed it using the `DOS4G` environment variable (see section C.3.1).

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 section C.3 on the settings for the `DOS16M` environment variable. 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.

5 Link the resulting binary file with the necessary parts of the run-time library:

```
C>ilink /ta simple.tco hostio.lib /f occama.lnk

C>
```

6 Configure the program. This stage makes use of a configuration description file which describes the hardware that the program is to run on. 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 through link 0 (it also suffices for programs which will be run on the simulator). You will need to edit this file if your hardware configuration is different (see the chapter of the *occam 2 Toolset User Guide* entitled Configuring Transputer Programs for details of configuration). This example has been compiled for the TA transputer class which is suitable for a T425 transputer.

The command is as follows:

```
C>occonf simple.pgm

C>
```

7 Add bootstrap code to the configured file. The bootstrap code loads the application onto the transputer and starts it executing :

```
C>icollect simple.cfb

C>
```

3.2 Running the example program

Finally, the program can be run on a transputer:

```
C>iserver /sb simple.btl
```

This will display the following if the name 'Bruce' is given:

```
Please type your name :Bruce
Hello Bruce

C>
```

The output '`Hello Bruce`' comes from the `simple.occ` example program.

If, instead of the `C>` prompt, the computer outputs an error message reporting problems with the link resource name or a message similar to:

```
Failed to open connection to transputer because:
     ...
```

is displayed, then check that the iserver is correctly set up (section 2.3.4). The reason displayed by the error message should indicate where the problem is.

Another potential source of problems is that the iserver cannot communicate with the transputer hardware. This will result in an error message similar to:

```
Error - iserver - protocol error ...
```

In particular, please check that any wire links, accessible from the back of the PC, have been correctly installed, and any jumpers have been set correctly. The transputer board's documentation should help with this.

3.3 Using the simulator

The example program can instead be executed on the transputer simulator which is supplied as part of the toolset. The simulator simulates an IMS T425 transputer. Since we compiled the example for `ta`, which means it will run on any 32-bit transputer, we can use the same executable program:

```
C>isim /bq simple.btl

C>
```

Display produced when the name 'Bruce' is used:

```
Please type your name :Bruce
Hello Bruce
```

A Distribution notes

A.1 iserver source

The directory `/D7305A/SOURCE/ISERVER/` contains the sources for the host file server. The server source is partitioned into three components which must be built in the following order:

linkios
linkops
iserver

Each component contains a single **`src`** directory which contains the source files and makefiles for that component. The makefiles are designed in such a way that they work within a copy of the **`src`** directory which should appear at the same level as **`src`** but with a name taken from the host dependent suffix to the makefile. e.g. In order to build a PC version of the iserver each component should contain a copy of the **`src`** directory called **`pc`** in which the makefile **`makefile.pc`** should be invoked.

A.2 Driver program errors

The transputer based tools are executed through a driver program which itself generates error messages.

```
Fatal-driver- unable to execute 'idebug', Arg list too long
```

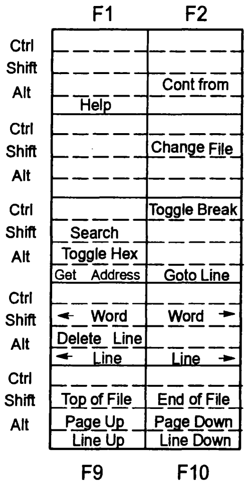
In this example the messages indicates that the DOS limit on the length of the command line has been exceeded.

Driver errors are generated for limitations or errors such as a command line too long, denial of read/write access to a file, and file or directory not found.

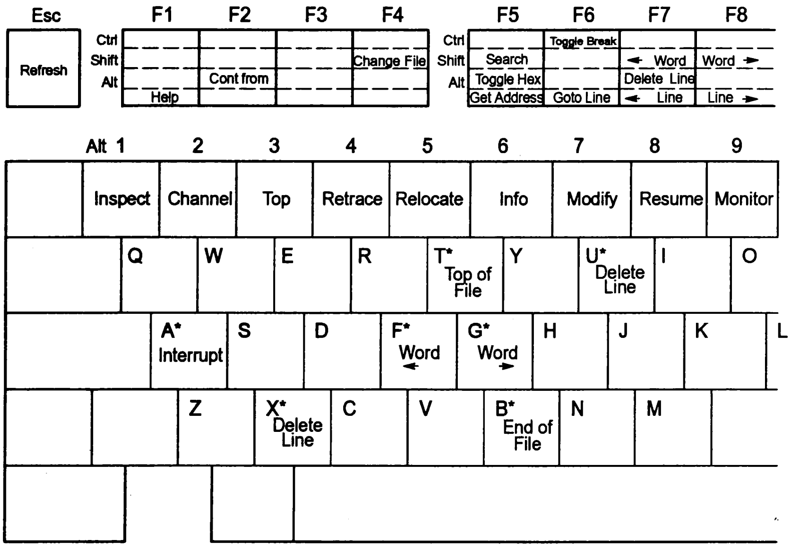
B Debugger function keys

This appendix gives the keyboard assignments for the debugger symbolic functions for both the IBM 386 PC and compatibles (`PCANSI.ITM`) Some of the keys are applicable to the simulator as well.

B.1 IBM 386 PC LH-keypad

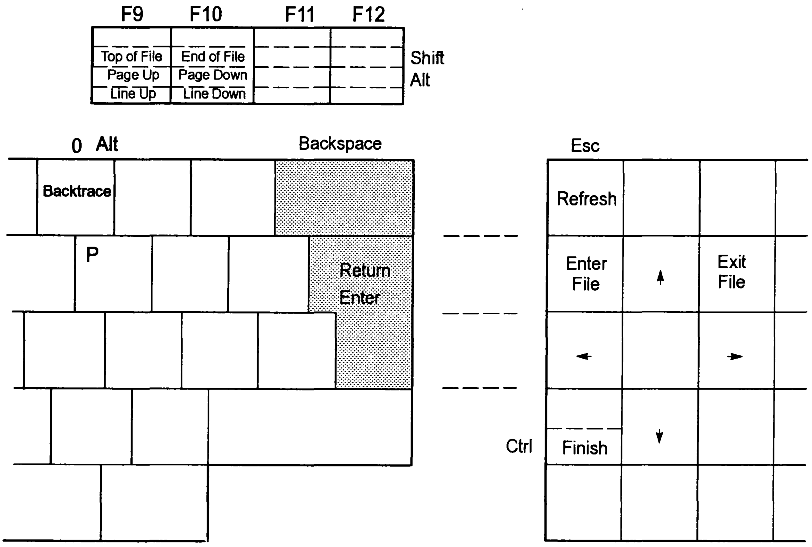


B.2 Keyboard layout for IBM 386 PC – left



* Ctrl + key

B.3 Keyboard layout for IBM 386 PC – right



C DOS Extender

All of the tools provided in this toolset will only run on a '386 or higher PC. This is because they use the full 32 bit capability of the '386 processor, to give access to a large memory space and 32 bit registers. To do this, and still remain compatible with MS-DOS, the applications use a DOS extender, and possibly an extended memory manager.

C.1 Installation

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 the WATCOM C/386 package, which is used to generate the tools in the toolset. To use this it is simply necessary to ensure that the DOS extender executable (`DOS4GW.EXE`) is accessible through the `PATH` environment variable. The installation script places this file in the `TOOLS` subdirectory, which should be on the path anyway.

Note that the version of the DOS extender supplied with the toolset is 1.8. This version, or a later one, is required for reliable operation of the toolset, so if a WATCOM product is also installed on the PC, ensure that the correct DOS extender is found.

A limited amount of configuration may be required before tools from the toolset can be run. When doing this it can be very useful to run PMINFO. This program is provided by Rational Systems and will display useful information about the system. If PMINFO will not run then consult section C.3 for further details of configuration options.

The most common problem which may be faced when starting to run the toolkit is ensuring that sufficient extended memory is available. Few applications use this directly, and so it may be used by disk caches, RAM disks, or expanded memory simulators. If this is the case some of these will have to be removed from the system, or their memory requirements reduced.

If DOS/4GW finds that an extended memory manager which uses one of the main disciplines is already installed, it will co-operate with it to use extended memory. To find out if there is an extended memory manager present, run PMINFO. In the final line of the output will be shown the switch method. This is the technique which is used to switch between real mode and protected modes (see section C.2.1 for the meaning of these terms). If there is no memory manager, then PMINFO will use switch method 3 (386), and DOS/4GW will manage the memory itself.

C.2 '386 Background

Most of the time the use of the DOS extender will be invisible to the user. However if problems do occur the following sections give some background on the concepts involved, before discussing the options which can be used to control the DOS extender.

C.2.1 '386 basics

The '386 is able to operate in three basic modes:

real mode This is the default mode, in which the '386 emulates an 8086. All registers are 16 bits wide, and so this is sometimes called 16 bit mode. This is the mode in which MS-DOS runs, limiting a program to 640K memory.

protected mode This mode supports hardware memory protection and paging. On a '386 this enables an application to access up to 4Gbytes memory, and all registers are extended to 32 bits, so it is sometimes known as 32 bit mode. This is the mode in which tools from the toolset run.

virtual 86 mode In this mode the '386 emulates an 8086, however it is implemented as part of the protected mode, allowing the memory management hardware to be used, and execution of certain instructions to be controlled. This is used by operating systems to support DOS applications, which think they are running on a real 8086, however all input and output is actually being redirected via the operating system. This mode is used for the DOS prompt in MS-Windows. We won't consider this mode any further.

Because it is possible to switch between real mode and protected mode, an application can be started from MS-DOS, running in real mode, switch to protected mode to run, and then return to MS-DOS for operating system support. This switching is controlled by software called DOS extenders, which control the execution of the application in protected mode, and make DOS calls available to the application.

C.2.2 Memory

Since the introduction of the original 8086 based PC, various methods have been used to make more memory available. As a result there are now a large number of 'types' of memory possible on a PC. Figure C.1 shows a memory map of a typical PC.

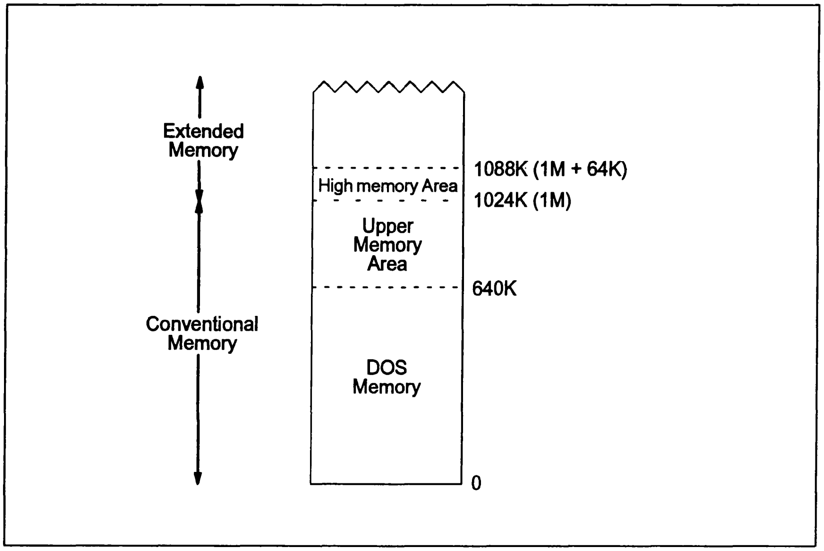


Figure C.1 Memory layout

Conventional Memory

The original 8086 had a 1Mbyte address space, that is, it could directly access up to 1Mbyte of memory. When IBM used the 8086 in the first PC they divided this address space into 640K which could be filled with RAM, and the remaining 384K was reserved for memory mapped hardware (such as graphic cards) and ROMs. MS-DOS was designed to run on this hardware and so contains a memory manager for the lower 640K. Thus this memory is called 'conventional memory' as it is supported on all systems.

The upper memory area does not normally contain memory, and so cannot normally be used for storing programs or data. However techniques do exist which allow backfilling of this address space so that device drivers can be placed there.

Expanded Memory

The first solution to the memory problem was Expanded Memory. This provided an application with up to 32 Mbytes of additional memory. The problem was that the memory could only be accessed through four 'pages', each of 16K which were mapped into a fixed place in the address space, usually in upper memory. Access to this memory was via another memory manager, the interface to which was defined by Lotus, Intel and Microsoft, and so is sometimes called the LIM standard, although it is more correctly named the EMS (Expanded Memory Specification).

This system had the advantage that it could be used on the 8086, but required additional hardware. Once the '386 had been introduced it became possible to implement expanded memory in software, using device drivers such as `EMM386.SYS`. However it is mainly used now for backward compatibility.

You may also come across the Enhanced Expanded Memory System (EEMS). This is a superset of EMS which allows more flexibility in where the additional memory will be mapped into the address space. In particular the number of pages, their size, and the addresses at which they may be mapped are all variable.

Extended Memory

Once the '286 was introduced, memory above 1Mbyte became directly accessible. In the case of the '286 and '386SX this was up to 16 Mbytes, for the '386DX and '486s it is 4Gbytes. The problem is that most of it is only available to programs written to use these chips in protected mode, which MS-DOS was not, and so most programs are still limited to 640K.

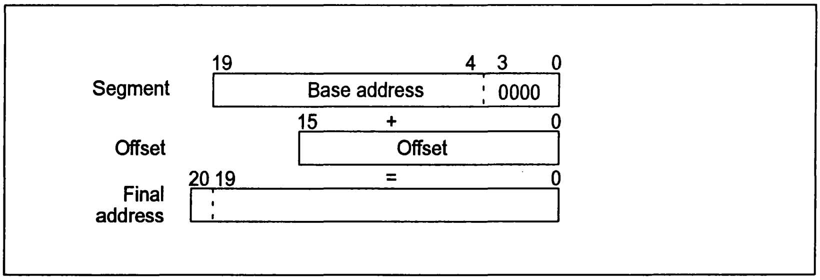
When an application is running in protected mode it is freed from the 640K limit of DOS, thus all the memory on your PC is available to the application. However the problem arises that several programs may want to use this memory at the same time, and so a memory manager is required to ensure that the same block of memory is only being used by one program at a time. While MS-DOS will manage the memory below 640K an external manager is required for extended memory. Several standards have evolved for these, in particular:

XMS Extended Memory Specification. This is the most common memory manager as it is distributed with DOS and MS-Windows in the form of `HIMEM.SYS`. It will also manage memory between 640K and 1Mbyte on machines that support it, and will handle the Higher Memory Area (see below).

VCPI Virtual Control Program Interface. This is an extension of EMS which as well as managing memory will switch between real mode and protected mode to allow code to be run outside the first 1Mbyte. Thus it will usually require a DOS extender. This interface to extended memory is most commonly provided by running `EMM386.SYS`.

DPMI DOS Protected Mode Interface. This is similar to VCPI but also provides limited communication between the protected application and DOS, as well as providing some protection to memory and devices. DPMI is notable because it is provided by Windows 3, so an application running in the DOS Prompt window can make DPMI calls.

One area of extended memory is accessible to programs running in real mode, despite the apparent 1Mbyte limit. This is called the High Memory Area (HMA) and is the first 64K of extended memory. This is directly accessible because of the 80x86's segmented addressing scheme, in which the address is generated by adding the segment address to an offset:



If the segment base address is set to the maximum possible value, this is an address 16 bytes short of 1Mbyte. However the segment can extend for 64K past this address, and so most of the first 64K of extended memory is accessible to a program running in real memory. On an 8086 this address would ‘wrap-round’ to the bottom of memory, and so to retain compatibility most ’286 and higher PC’s provide hardware which simulates this behavior (indeed it is implemented directly on a ’486). Because this behavior occurs when the 20th address bit is set, the control logic is called the ‘A20 gate’.

C.2.3 DOS Extenders

Although the ’386 provides the program running on it with a large number of facilities, it still requires an operating system to support it, most commonly MS-DOS. However this introduces some problems, since MS-DOS runs in real mode, and only provides memory management for conventional memory. To allow a protected mode program to run, an interface between real mode and protected mode is required – this is the DOS extender.

The DOS extender provides several features, the most important of which are:

- Load the application into memory and starts it running.
- Provide the application with operating system services, which it then passes on to DOS for execution.
- Provide the application with memory management functions, independently of the actual memory manager which is being used.

C.3 Configuring DOS/4GW

The DOS extender which is used with the INMOS toolset is DOS/4GW. This section describes how to use the `DOS4G` environment variable to suppress the banner that is displayed by DOS/4GW at startup. It also explains how to use the `DOS16M` environment variable to select the switch mode setting, if necessary, and to specify the range of extended memory in which DOS/4GW will operate. DOS/4GW is based on Rational Systems’ DOS/16M 16-bit protected mode support; hence the `DOS16M` environment variable name remains unchanged.

C.3.1 Suppressing the DOS/4GW Banner

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.

C.3.2 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.

The table below 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
'386/'486 with DPMI	auto	0	Set automatically if DPMI active
NEC 98-series	required	1†	Must be set for 98-series
PS/2	auto	2	Set automatically for PS/2
'386/'486	auto	3‡	Set automatically for '386 or '486
'386	auto	INBOARD	80386 with Intel Inboard
Fujitsu FMR-70	required	5	Must be set for Fujitsu FMR-70
'386/'486 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.			

If you have one of the machines listed below, set the DOS16M environment variable to the value shown for the machine and specify a range of extended memory.

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

For example, if your machine is an NEC 98-series set the DOS16M environment variable as follows:

```
C>set DOS16M=1@2M-4M
```

Note: The meaning of the characters "@2M-4M" is described in section C.3.3, "Fine Control of Memory Usage", later in this chapter.

Remember to add this command to your AUTOEXEC.BAT file so that the DOS16M is set up correctly each time you switch on your PC.

Before running toolset applications, check the mode setting by running the PMINFO program which can be found in the D7305A\TOOLS directory.

If all is well PMINFO displays a report of the protected-mode resources available to your programs. The last line will be something like the following (although possibly with a different switch mode value):

```
using DOS/16M switch mode 11 (VCPI)
```

Note: in some cases PMINFO may not run correctly even though it appears that the DOS extender has been configured properly. However the tools may still work correctly (this is because DOS/4GW has built in VCPI or DPMI host systems that will allow it to run while PMINFO does not). If PMINFO does not run try executing one of the INMOS tools supplied with this toolset. For example, the linker:

```
C>ilink
```

If the linker help page is displayed then the set up is probably OK. If the help page is not displayed then recheck your set up.

Note: 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.

C.3.3 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 don't 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, 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 than the address of size is assumed to be in kilobytes. If both a size and a range are specified, than 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 rage 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:

`set DOS16M=1@2m-4m`	Mode 1, for NEC 98-series machines, and use extended memory between 2.0 and 4.0Mbytes.
`set DOS16M=:1M`	Use the last full megabyte of extended memory, or as much as available limited to 1Mbyte.
`set DOS16M=@2M`	Use any extended memory available above 2Mbytes.
`set DOS16M=@0-5m`	Use any available extended memory from 0.0 (really 1.0) to 5.0Mbytes.
`set DOS16M=:0`	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 DPMI 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 DPMI automatically prevent address conflicts between extended memory programs. You can specify a *size* for memory managed by VCPI or DPMI, 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.

Using Extra Memory

Some machines contain extra non-extended, non-conventional memory just below 16Mbytes. When DOS/4GW runs on a Compaq 386, it automatically uses this memory because the memory is allocated according to a certain protocol, which DOS/4GW follows. Other machines have no protocol for allocating this memory. To use the extra memory that may exist on these machines, set `DOS16M` with the + options:

```
set DOS16M=+
```

Setting the + option causes DOS/4GW to search for memory in the range from FA0000 to FFFFFF and determine whether the memory is usable. DOS/4GW does this by writing into the extra memory, and reading what it has written. In some cases, this memory is mapped for DOS or BIOS usage, or for other system uses. If DOS/4GW finds extra memory that is mapped this way, and is not marked read-only, it will write into that memory. If this memory is in use, this will cause a crash, but won't damage your system.

C.3.4 Setting Runtime Options

The `DOS16M` environment variable sets certain runtime options for all DOS/4GW programs running on the same system. Most of these are only useful when developing applications under DOS/4GW, and are listed only for completeness. However, they may be useful when coercing toolset applications to run on machines that are not fully AT-compatible.

To set the environment variable, the syntax is:

set DOS16M=[*switch_mode_setting*]^*options*

Note: Some command line editing TSRs, such as CED, use the caret (^) as a delimiter. If you want to set `DOS16M` using the above syntax above while one of these TSRs is resident, modify the TSR to use a different delimiter.

The options are as follows:

`0x01` *check A20 line*. This option forces DOS/4GW to wait until the A20 line is enabled before switching to protected mode. When DOS/4GW switches to real mode, this option suspends your program's execution until the A20 line is disabled, unless an XMS manager (such as `HIMEM.SYS`) is active. If an XMS manager is running, your program's execution is suspended until the A20 line is restored to the state it had when the CPU was last in real mode.

Specify this option if you have a machine that runs DOS/4GW but is not truly AT-compatible. For more information on the A20 line, see section C.3.5, "Controlling Address line 20".

`0x02` *prevent initialization of VCPI.* By default, DOS/4GW searches for a VCPI server and, if one is present, forces it on. This option is useful if your application does not use EMS explicitly, is not a resident program, and may be used with '386-based EMS simulator software.

`0x04` *directly pass down keyboard status calls.* When this option is set, status requests are passed down immediately and unconditionally. When disabled pass-downs are limited so the 8042 auxiliary processor does not become overloaded by keyboard polling loops.

`0x10` *restore only changed interrupts.* Normally, when a DOS/4GW program terminates, all interrupts are restored to the values they had at the time of program startup, When you use this option, only the interrupts changed by the DOS/4GW program are restored.

`0x20` *set new memory to 00.* When DOS/4GW allocates a new segment or increases the size of a segment, the memory is zeroed. This can help you find bugs having to do with uninitialized memory. You can also use it to provide a consistent working environment regardless of what programs were run earlier. This option only affects segment allocations or expansions which are made through the DOS/4GW kernel (with DOS function 48H or 4AH). This option does not affect memory allocated with a compiler's `malloc()` function.

`0x40` *set new memory to FF.* When DOS/4GW allocates a new segment or increases the size of a segment, the memory is set to 0xFF bytes. This is helpful in making reproducible cases of bugs caused by uninitialized memory. This option only affects segment allocations or expansions which are made through the DOS/4GW kernel (with DOS function 48H or 4AH). This option does not affect memory allocated with a compiler's `malloc()` function.

`0x80` *new selector rotation.* When DOS/4GW allocates a new selector, it usually looks for the first available (unused) selector in numerical order starting with the highest selector used when the program was loaded. When the option is set, the new selector search begins after the last selector was allocated. This causes new selectors to rotate through the range. Use this option to find references to *stale* selectors, i.e., segments that have been canceled or freed.

C.3.5 Controlling Address line 20

This section describes how DOS/4GW uses address line 20 (A20) and describes the related `DOS16M` environment variable settings. It is unlikely that you will need to use these settings.

Because the 8086 and 8088 chips have a 20-bit address space, their highest addressable memory location is one byte below 1Mbyte. If you specify an address at 1Mbyte or over, which would require a twenty-first bit to be set, the address wraps back to zero. Some parts of DOS depend on this wrap, so on the 80286 and 80386, the twenty-first address line is disabled. To address extended memory, DOS/4GW enables the twenty-first address bit (the A20 line). The A20 line must be enabled for the CPU to run in protected mode, but it may be either enabled or disabled in real mode.

By default, when DOS/4GW returns to real mode, it disables the A20 line. Some software depends on the line being enabled. DOS/4GW recognizes the most common software in this class, the XMS managers (such as `HIMEM.SYS`), and enables the A20 line when it returns to real mode if an XMS manager is present. For other software that requires the A20 line to be enabled, use the `A20` option. The `A20` option makes DOS/4GW restore the A20 line to the setting it had when DOS/4GW switched to protected mode. Set the environment variable as follows:

```
    set DOS16M=A20
```

To specify more than one option on the command line, separate the options with spaces.

The `DOS16M` variable also lets you specify the length of the delay between a DOS/4GW instruction to change the status of the A20 line and the next DOS/4GW operation. By default, this delay is 1 loop instruction when DOS/4GW is running on a '386 machine. In some cases, you may need to specify a longer delay for a machine that will run DOS/4GW but is not truly AT-compatible. To change the delay, set the `DOS16M` to the desired number of loop instructions, preceded by a comma:

```
   set DOS16M=,loops
```

C.4 PMINFO

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

Syntax `PMINFO.EXE`

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.

C.5 RMINFO

Purpose Supplies configuration information and the basis for real/protected-mode switching in your machine.

Syntax `RMINFO.EXE`

Notes RMINFO starts up DOS/4GW, but stops your machine just short of switching from real mode to protected mode and displays configuration information about your computer. The information shown by RMINFO can help determine why DOS/4GW applications won't run on a particular machine. Run RMINFO if PMINFO does not run to completion.

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

```
Configuration information:

Machine configuration: Intel 80486 processor(delay= unknown)
        Has an PC or XT BIOS:   No
        Triple fault flag:      On
        Address line 20 rigor:  No
        Address line 20:        Off
        Address line 20 orig.:  Off
Operating system is DOS 5.0
        VDISK device found:     No
        QEXT device found:      No
        XMS system found:       No
Memory
        Configured memory:      Base 1024K, Size 15360K, Top 16384K
        Actual memory:          Base 1024K, Top 8192K
        Unallocated memory:     7168K
DOS/16M version 3.95
        This is the first copy of DOS/16M.
        Switch control flags:   0000
        Switch method:          3 (80386)
        Available memory:       7168K [0]
        No VCPI page table.
        VCPI not in use.
End of configuration information
```

The information provided by RMINFO includes: