intel®

product release notes

System V/iRMK™ (RMKU)

CONTENTS

System V/iRMK™ (RMKU) Release Notes

Introduction

This release note provides information on product limitations, corrections to manuals, and cautionary notes.

The product is supplied in a CPIO formatted tape and installs with the standard System V/386 **installpkg** tool. The product tape contains the iRMK Kernel package and the Universal Development Interface (**UNXUDI**). Both are described below.

These packages have been evaluated on System V, release 3 and release 4.

iRMK™ Kernel I.3.1

The iRMK Kernel package includes the Kernel libraries, the include files, source files for various device drivers, and source files for examples. A detailed description of each example is supplied as a README file. The **installpkg** utility will install product files in the */usr/intel/rmk* directory. The second stage bootstrap loader, *stage2.rmk*, will install in the */msa* directory. **installpkg** creates required directories if they do not already exist.

In addition to the Kernel libraries, the package includes:

- the remote debug monitor (rdm) needed by the Soft-Scope® III debugger
- the remote console interface server (rci) needed by the Soft-Scope III debugger for communication over Multibus II
- debug libraries for the target

- the Multibus II MSA-related utilities: **reboot**, **bootstat**, and **nmi**
- **mread** and **mdir** utilities for access to DOS diskettes

Universal Development Interface (UNXUDI)

UNXUDI for System V/386 runs Intel-supplied DOS-based compilers and utilities on System V/386. **installpkg** places **UNXUDI** programs in the */usr/bin* and the */usr/lib* directories.

installpkg also installs scripts in the */usr/intel/bin* directory for invoking the following Intel compilers and utilities under **UNXUDI**:

- iC-386
- ASM386
- PL/M-386
- Fortran-386
- BND386
- BLD386
- LIB386
- MAP386

Installation Notes

The following notes address installation issues.

If you have previously installed *UNXUDI* or the iRMK I.3 product , do not use the System V/386 overinstall capability to install this package. Run **removepkg** first and then use **installpkg**. If the Soft-Scope III Debugger is installed and you remove the iRMK I.3 package, **removepkg** will issue messages stating that certain directories can not be removed. Ignore these messages.

If the following messages are displayed on the system console during installation, you can safely ignore them:

```
Invalid Buffer Parameter

rm: Cannot remove any directory in the current
working directory /usr/tmp/installnnn
```

When installing on System V/386 release 4, **installpkg** changes the permissions of the */usr/tmp* directory from the default (read/write/execute) to an arbitrary value. To correct this problem, enter the command `chmod 777 /usr/tmp` after installation of the product. Without this correction, the **rdm** utility will not connect to the target board for debugging.

Installation of **UNXUDI** rebuilds the UNIX kernel, requiring you to shut down and restart the system. **installpkg** will prompt for the shutdown.

You must be root to install the Soft-Scope III Debugger. It has been observed that the first attempt to install the Soft-Scope debugger after a power up may fail with an "end of media" error. The second attempt succeeds.

System V/383 release 4 provides the utilities **reset** and **reboot** in the */usr/ucb* directory. If you intend to use the Multibus utilities with the same names, make sure you specify the correct PATH variable. The Multibus utilities are in the */usr/lbin* and */usr/intel/rmk/bin* directories.

iRMK™ Kernel

The following notes address Kernel issues.

For an application in PL/M, the 82530 driver-based *stdio* (console I/O) requires the X313 version (or later) of the PL/M compiler.

Interfaces to PIC and PIT drivers will not change, as they are part of the product definition. However, interfaces to DMA driver sources (82258 ADMA driver, 82380 DMA driver) may change in the future. The DMA drivers are specific to Multibus II message passing. These interfaces are not part of the product definition.

The amount of memory and the number of GDT slots reserved with the builder depend upon the debugger used. The amount of memory reserved for Soft-Scope and iM III is 0H-1CFFFH. The GDT slots reserved are 3-63.

The Kernel internally creates stacks (4K each) for the idle task and message passing tasks in the data segment of the Kernel. Remove all references to `dl_stack_seg`, `mp_stack_seg` and `idle_stack_seg` from old build files (if protected stacks are not needed). Note that these symbols are only found in iRMK I.2 (and older) build files. The user can also create builder-defined stacks for these internal tasks, and modify the sizes of these stacks. For details, refer to the *iRMK™ Kernel Reference Manual.*

Previous versions of the Kernel assumed the GDT alias to be 8H and the IDT alias to be 10H. This version of the Kernel allows the GDT and IDT aliases to be specified by the user. The default is 8H (GDT slot#1) for the GDT and 10H (GDT slot#2) for the IDT. For information on changing defaults, see the *iRMK™ Kernel Reference Manual.*

If your application runs in a different subsystem from the Kernel, the pointers returned as part of structures from the Kernel (for example in `KN_get_descriptor_attributes`) are not translated. If you want a pointer to be based on your subsystem's data segment rather than that of the Kernel, use the following:

```
valid_ptr =
KN_linear_to_ptr(KN_ptr_to_linear(returned_ptr))
;
```

Repetitive alarms with an interval of zero do not work.

When a data chain is presented to the Kernel, all the data pointers and the pointer to the chain block should be either GDT-based or LDT-based. A mixture of GDT- and LDT-based pointers results in an error.

The builder-generated map displays the following warning when the Kernel standard I/O functions are used from a PL/M based application. This warning does not cause an error.

```
*** WARNING 126:  SYMBOL TYPES MISMATCH
    FILE:  kstdio.lnk
    MODULE:  KSTDIO
    SYMBOL:  PRINTF
```

The user has to initialize all the spurious interrupt handlers (level_x7 handlers) or check for spurious interrupts if there is a genuine level 7 handler. Otherwise, the vectors for these handlers may be not be initialized and may cause faults when such interrupts are encountered.

rci broadcasts to port 801H. Applications should not use this port ID for message passing.

Increasing the Initial Task Stack Size

When an application calls `KN_initialize`, the calling program becomes the initial task. The call does not specify how large the stack should be. Instead, the stack size is set by the compiler or BLD386. Most programs accept the stack size assigned by the compiler. The compiler assigned stack is usually large enough for the majority of programs to run correctly. However, it may be necessary to increase the initial task's stack size. For example, if there are multiple interrupts being serviced, there may not be enough room on the initial task's stack.

In small model, data and stack share the same segment. In compact model, there is a separate stack segment. If a small model program runs out of stack, data corruption or a stack fault may result. With a compact model program, a stack fault usually occurs.

Buildfile fragments are shown below. These demonstrate how to increase the initial task's stack. A compact model example and a small model example are shown.

Note that the plus sign (+) causes 512 bytes to be added to the existing segment. In the small model example, DS and SS are the same so DS is increased. In the compact model example, SS is increased.

Small Model

```
SEGMENT
        bist
                (DPL = Ø),
        bist.data
                (limit = +512);
```

Compact Model

```
SEGMENT
        bist
                (DPL = Ø),
        bist.stack
                (limit = +512);
```

UNXUDI Utility

UNXUDI is now able to use a directory other than */tmp* for temporary files. Use the `WORKDIR` environment variable to define an alternate directory. The */tmp* directory is the default.

Remote Development Monitor (rdm) Utility

The */usr/intel/rmk/bin/rdmcfg* configuration file is installed with valid entries for use with *rci*. If you use a serial port for debugging, rather than *rci*, you must add entries to the configuration file. See the *System V/iRMK™ Installation and User's Guide* for more information.

For multiple sessions of Soft-Scope (each as a different System V/386 user), invoke **rdm** as the super user. Otherwise, only the session that invokes **rdm** can use Soft-Scope.

If entering `rdm start` produces an error message indicating that a previous **rdm** session has not stopped, use `rdm stop` to kill the previous session. If the problem persists, find the running **rdm** processes by typing `ps -ef | grep rdm`. Kill these before invoking `rdm start` again.

The serial interface for **rdm** now works correctly with baud rates other than 9600. Previously, **rdm** ignored the baud rate specified in */usr/intel/rmk/bin/rdmcfg* and used 9600 baud.

Remote Console Interface (rci) Utility

If entering `rci start` produces an error message indicating that a previous **rci** session has not stopped, use `rci stop` to kill the previous session. If the problem persists, find the running **rci** processes by typing `ps -ef | grep rci`. Kill these before invoking `rci start` again.

Soft-Scope® III Debugger

The user selects the Soft-Scope target. This is normally done in the */usr/intel/rmk/bin/ss.set* file. However, if multiple sessions of Soft-Scope are needed, each user defines a target in a separate *ss.set* file and indicates the file with the System V/386 environment variable `SSSETPATH`.

When setting `SSSETPATH`, specify colons on either end of the path name.

The user cannot inspect the NDP registers until `KN_initialize_NDP()` is called.

The user needs to correctly set IDT entries in the build file for Soft-Scope to properly handle fault conditions. Please refer to the examples supplied.

For Soft-Scope to automatically find the listing files when you use the supplied examples, invoke Soft-Scope from the example *src* or *abs* directory. The absolute files generated by the examples assume the listing files are in the *../lst* directory.

If you use the **list** command and Soft-Scope can not find the listing file, it prompts you for the pathname. If you enter the correct path but it is longer than 45 characters, you will receive an error. This problem can usually be addressed by specifying a relative pathname rather than a complete pathname.

After invoking Soft-Scope, if the message `<unable to connect to the target>` appears, exit Soft-Scope and ensure that **rdm** and **rci** are running, and that the *ss.set* file in the */usr/intel/rmk/bin* directory is correct. Other items to check include:

- the target file is booted with an absolute file built with *rds.lnk*.
- the bootstrap configuration file contains two parameters for the target (`BL_debug_on_boot=iM, CC_console=ccrci`), refer to the *System V/iRMK™ Installation and User's Guide* for more information.

If multiple invocations of Soft-Scope fail, stop **rdm** and **rci**, and then restart them. This frees resources tied up by **rdm**.

If you use memory-mapped I/O, the version of iM III supplied performs only byte I/O operations. Even if you specify a word-sized read or write of the I/O port from Soft-Scope III, memory is accessed byte-by-byte, causing incorrect access of the device. Access of I/O devices in I/O space (for example, using the port commands) is done correctly.

The line numbers associated with the sample sessions in the *Soft-Scope® III Debugger User's Guide* are incorrect. The starting line number is 1293 for the C sample and 1423 for the PL/M sample.

To exit Soft-Scope while the application is running, press `<Ctrl-C>`. You may also use `<Ctrl-\>`.

It has been observed that <Ctrl-C> and <Ctrl-\> may not exit Soft-Scope properly when the **stty** "kill" character is set to the default (@). To correct this problem, set the kill character to some other key.

Remote Development Server (RDS)

This component is supplied in the *rds.lnk* file and contains the iM III debug monitor. It must be built with the application to enable debugging, but is transparent to the user.

RDS can be configured in the `KN_initialize_RDS` call.

Multibus Utilities

Multibus II utilities installed with the Kernel include **reboot**, **bootstat**, and **nmi**. These utilities are described in the *System V/iRMK™ Installation and User's Guide.*

After **reboot** is invoked, invoke **bootstat** to verify the board has booted correctly. The output is similar to the following:

```
Slot 0: boot ENABLED Stage 2 - Error status 0:
E_BL_OK
Slot 1: boot ENABLED Stage 2 - Error status 0:
E_BL_OK
Slot 4: boot ENABLED Stage 2 - Error status 0:
E_BL_OK
Slot 5: boot ENABLED Stage 2 - Error status 22:
E_BL_FNEXIST
Slot 6: boot ENABLED Stage 1 - Error status 0:
E_BL_OK
```

In the examples above, slots 4, 5, and 6 contain target hosts. Slot 4 booted correctly; the second-stage bootloader finished with error status 0. Slot 5 got to the second stage, but it could not find the boot file specified in the bootstrap configuration file. Slot 6 shows error status 0, but since this is only for the first-stage bootloader, the board has not yet booted. Invoke **bootstat** again to verify that Slot 6 finishes booting. (For more information about the error codes, refer to the *Firmware User's Guide for MSA Firmware* or the *Multibus II System Architecture Bootstrap Specification.*)

If the boot status indicates file access problems, check the access rights of the bootable binary file. It should be readable and executable by world. Also check the bootstrap configuration file (*/etc/default/bootserver/config* in the case of System V/386 release 3) to see whether the pathname of the bootable file is specified correctly. If the **reboot** utility fails to boot a valid and properly specified executable binary file, invoke */usr/lbin/initbp* as the super user.

Since changing the bootserver configuration file is risky, (a change in the System V/386 host specifications may prevent the host from booting), consider copying the application to a central directory with a common name so that the configuration file need not be changed when an application is generated.

The **reboot** command does not work on MIX and Intel486™ series boards on the first invocation. A second invocation of the command will boot the specified board.

Intel Languages and Tools

The product was evaluated with the following Intel languages and tools:

iC-386 V4.3
PL/M-386 X313
FTN386 V4.2
ASM386 V4.0
BND386 V1.5
BLD386 V1.6
LIB386 V1.3
MAP386 V1.3

The PL/M and Fortran languages are supplied as separate packages on DOS diskettes. They are not available on the development tools tape installable under System V/386.

Mixed case file names are not supported by all the Intel languages and tools listed above. To be safe, use lower case file names.

The following sections address specific limitations in language support.

iC-386 Notes

You cannot use the NDP emulation library with the current iC-386 compiler. The board must contain an NDP coprocessor for evaluation.

The strings used in a `pragma` should be enclosed in double quotes. Examples are:

```
# pragma varparams ("KN_token_to_ptr",
"KN_create_task", "KN_delete_task")

# pragma interrupt ("function_name")
```

Fortran-386 Notes

Fortran-386 does not support pointers. If you need to access interfaces requiring pointers as parameters, you need to implement part of your application in another language that provides pointer support.

Fortran-386 does not support structures.

Either iC-386 or PL/M-386 is required to make the Fortran-386 examples provided with the product.

ISV Languages

The product has been evaluated with the System V/386 versions of the Metaware High-C compiler (Version 1.5) and the PharLap LinkLoc linker (Version 2.2d). The supplied example (in the */usr/intel/rmk/examples/highc* directory) shows the compiler and the linker invocation.

The user can use other C compilers (or other languages) with Linkloc if they support CodeView control and Easy-OMF output. This is required for Soft-Scope III debugger support.

System V/386 Commands

Two System V/386 utilities provided with this product are not documented in the manual set. The commands, **mdir** and **mread**, access files on DOS diskettes. They are installed in the */usr/intel/rmk/bin* directory.

The commands are described in the following sections.

mdir Command

mdir displays contents of a DOS directory

Synopsis:

```
mdir        [-w] dosdirectory
```

or

```
mdir        [-w] dosfile [dosfiles...]
```

Description:

mdir displays the contents of a directory on an MS-DOS formatted diskette. The second form of this command displays information for the specified files, such as file size and creation date. Subdirectories are supported with either the "/" or "\" separator. The use of the "\" separator or wildcards requires the names to be enclosed in quotation marks, to protect them from being misinterpreted by the shell.

The command line option is:

`-w` Wide output. This option will print the file names across the page without displaying the file size or creation date.

In the first form of this command, an error occurs if a component of the path is not a directory. **mdir** does not follow the DOS convention of allowing only one directory argument.

The environmental variable MCWD is used to establish a current working directory (relative to the DOS diskette).

Examples

Place a DOS diskette in drive A. Type the following./

`mdir`
: Displays the root directory of the disk in drive A. The output format is the same as for the DOS dir command.

`mdir /`*`dirname`*
: Displays the *dirname* directory.

mread Command

mread copies a DOS file to System V/386.

Synopsis:

```
mread       [-t | -n] msdosfile [unixfile]
or
mread       [-t | -n] msdosfile [msdosfiles...]
            unixdirectory
```

Description:

The first version of **mread** copies the specified DOS file to the named System V/386 file. The second form of the command copies multiple DOS files to the named directory. In this form, since target filenames are not specified, filenames will be copied to upper-case names.

DOS subdirectories are supported with either the "/" or "\" separator. The use of the "\" separator or wildcards requires the names to be enclosed in quotation marks, to protect them from being misinterpreted by the shell.

The command line options are as follows:

- `-t` Text file transfer. **mread** will translate incoming carriage return/line feed characters into line feed characters.
- `-n` No warning. **mread** will not warn the user when overwriting an existing file.

If the target file already exists and the `-n` option is not in effect, **mread** asks if it should overwrite the file.

The environmental variable MCWD is used to establish a current working directory (relative to the DOS diskette).

Examples

Two files, *plm386.exe* and *plm386.lib*, are on a DOS diskette inserted in drive A. To copy the files to hard disk, first change to the target directory on the hard disk. Then use the **mread** command for each file. Enter:

```
cd /usr/intel/bin
mread plm386.exe plm386.exe
cd /usr/intel/lib
mread plm386.lib plm386.lib
```

Specify both the source and the target file parameters. If the target file name is not specified, **mread** will make the target file name the same as the source file. However, uppercase characters will be used for the target file name.

Code Examples

Makefiles for the examples that can run on a MIX baseboard, such as */usr/intel/rmk/examples/plm/msg_pass.cpt* and */usr/intel/rmk/examples/ic386/msg_pass.cpt*, show the configuration changes needed for the MIX processor board (in comments). To execute an application on a MIX processor board, change the makefiles as shown in the comments and rebuild the application.

The *stdio* examples do not run on a MIX baseboard, since the baseboard does not include a serial port.

Due to a Fortran-386 compiler problem, the far references in the small rom model do not conform to the requirements of Intel tools. As a result, the Fortran *bist.sro* example is not rommable.

The C Library examples (*clib.cpt* and *clib.sml*) use a default of */dev/tty354* for the terminal port. This tty string is valid on System V/386 V3.2.2. On System V/386 V4.0.3, use */dev/tty001*. In addition, the port on both versions of System V/386 must have read/write permissions for the user who invokes the C Server.

The *seg_gate.cpt* example for iC-386 may only run for the first sub-task cycle. However, the *seg_gate.cpt* is still a valid example of using multiple subsystems.

Port of Existing Applications

Refer to the *System V/iRMK™ Installation and User's Guide* for more information on program development and moving an existing application to this release of the Kernel.

C Libraries

The following sections apply to the C Libraries.

_defaults.c

The */usr/intel/rmk/system/src/_defaults.c* file contains default environmental variables. These values are used by the `getenv()` and `putenv()` functions. Place values you wish to have in the environment prior to execution in the structure contained in this file.

A compiled version of *_defaults.c* already exists in the *C Libraries*. If you wish to modify your environment, change the appropriate entries in *_defaults.c* and then re-compile. Use the command syntax, listed below, appropriate for the model that you are using.

Small Model

```
/usr/intel/bin/ic386 -si
/usr/intel/rmk/system/include/ -sm
            -ram _defaults.c
```

Compact Model

```
/usr/intel/bin/ic386 -si
/usr/intel/rmk/system/include/ -cp
            -rom _defaults.c
```

When binding your application, bind *_defaults.obj* before *CLIBS*.

CLIB_CONFIGURATION_STRUC structure

A new field has been added to the C library initialization structure. The field indicates whether the C Server needs to be initialized for the host. The variable, named `init_server`, has a value of 1 if the C Server is needed by the host, and a value of 0 if the server not needed.

C Server

You must start the C Server before running an application that uses it, such as the *C Library* examples provided for iC-386. Use the invocation `c_server start`. You only need to start the C Server once for multiple iRMK hosts. When you start or stop the C Server, the shell prompt returns followed by a message from a C Server process. This makes it appear that the shell prompt has not returned. Press `<RETURN>` to verify that you have the prompt.

Each iRMK host has its copy of the C Server running on the System V/386 host. When a host is rebooted, the server detects this and aborts the previous running server for that host and starts a new one.

Since there is only one copy of the C Server per iRMK host, blocking reads (such as those from a terminal) will cause all I/O for that host to be suspended. If this is a problem, use the function `kbhit()`. `kbhit()` checks the device input queue for available characters; if there are none, a value of 0 is returned. If characters are available, a value of 1 is returned. A task can call `kbhit()` prior to issuing a read command that blocks I/O for the host.

Corrections to Manuals

iRMK™ Kernel Reference Manual

On page 2-7, the description of the `flags` parameter to the **attach_protocol_handler** system call is missing. The `flags` parameter specifies whether the protocol handler is in the same or a different subsystem from the Kernel. These literals are defined for the flag:

```
KN_HANDLER_CONVENTION_MASK
KN_CALL_NEAR
KN_CALL_FAR
```

The tab for Chapter 5 of the *iRMK™ Kernel Reference Manual* reads "Configuration and Installation." It should be "Configuration and Initialization."

System V/iRMK™ Installation and User's Guide

On page 2-2, the last paragraph in the first section states that **UNXUDI** is installed in the */usr/bin* directory. A file is also installed in the */usr/lib* directory. This information should also be added to the first paragraph on page 2-4.

On page 2-6, Step 4, and page 2-7, Step 6, the instruction is to remove the installation tape when the shell prompt returns. Because **UNXUDI** installation rebuilds the System V/386 kernel, the **installpkg** utility prompts to shut down and reboot the system. Remove the tape when the shutdown process begins.

Page 2-8 should state that you must be the root user to install Soft-Scope III.

Page 2-23 describes using a prebuilt example file called *testss.abs*. The correct name of this file is *bistss.abs*. This change should also be made in the last paragraph of page 5-4 and the last entry in the bootstrap configuration file on page 5-5.

Table 3-1 on page 3-2 shows a *rom_ex* example in the *ic386* example directory. This example is only supplied for PL/M. This information should also be changed in the last paragraph of page B-1.

Page 3-4 describes using the **make** command for examples. It does not describe using the "make clean" syntax to remove files previously generated. For the correct invocation, refer to the README files in the examples.

Page 5-4 should state that you must be the root user to edit the bootstrap configuration file.
