Oracle® Java ME Embedded
Getting Started Guide for the Reference Platform (Keil)
Release 3.3.1
E38140-02

June 2013
This book describes how to install and run the Oracle Java ME Embedded software on the Keil MCBSTM32F200 reference platform, using the RTX operating system.
## Contents

**Preface** ................................................................................................................................. ix  
Audience ............................................................................................................................. ix  
Documentation Accessibility ......................................................................................... ix  
Related Documents ........................................................................................................ ix  
Operating System Commands ......................................................................................... ix  
Shell Prompts .................................................................................................................. x  
Conventions ....................................................................................................................... x

### 1 Running the Oracle Java ME Embedded Software on the Keil MCBSTM32F200 Board

- Downloading and Installing the MDK-ARM Lite Development Kit .......................................... 1-1  
- Downloading and Installing the PuTTY Terminal Emulator Program ........................................... 1-2  
- Connecting the Keil MCBSTM32F200 Board to the Computer ..................................................... 1-2  
- Setting Up the MicroSD Card ........................................................................................................ 1-3  
- Installing the Firmware on the Evaluation Board ......................................................................... 1-5  
- Connecting to the CLI and Logging Ports ....................................................................................... 1-6  
- Additional Methods of Obtaining Logging Information .................................................................. 1-11

### 2 Using NetBeans or Eclipse with the Keil MCBSTM32F200 Board

#### Using NetBeans with the Keil MCBSTM32F200 Board

- Install the Oracle Java ME SDK Plugin for NetBeans ................................................................. 2-1  
- Adding the Keil MCBSTM32F200 Board to the Device Selector .................................................... 2-2  
- Assigning the Keil MCBSTM32F200 Board to Your Project ............................................................ 2-3  
- Sample Source Code .................................................................................................................. 2-3  
- Accessing the Peripherals on the Keil MCBSTM32F200 ................................................................. 2-4  
- Method #1: Modifying the Security Policy File ............................................................................. 2-4  
- Method #2: Signing the Application with API Permissions ............................................................. 2-5  
- Debugging an IMlet on the Keil MCBSTM32F200 Board ................................................................. 2-6

#### Using Eclipse with the Keil MCBSTM32F200 Board

- Install the Oracle Java ME SDK Plugin for Eclipse ........................................................................ 2-7  
- Adding the Keil MCBSTM32F200 Board to the Device Selector .................................................... 2-8  
- Assigning the Keil MCBSTM32F200 Board to Your Project ............................................................ 2-9  
- Sample Source Code .................................................................................................................. 2-10  
- Accessing the Peripherals on the Keil MCBSTM32F200 ................................................................. 2-11  
- Method #1: Modifying the Security Policy File ............................................................................. 2-11
3 Troubleshooting

Installing the Firmware on the Board ................................................................. 3-1
Starting Oracle Java ME Embedded on the Board .................................................. 3-1
Using the Board with the Oracle Java ME SDK and the NetBeans IDE .................. 3-2
Other Problems ........................................................................................................ 3-3

A Keil MCBSTM32F200 Board Peripheral List

Analog-to-Digital Converter (ADC) ........................................................................................ A-1
Pulse Counter ................................................................................................................. A-2
Digital-to-Analog Converter (DAC) .................................................................................. A-4
GPIO Pins ...................................................................................................................... A-4
GPIO Ports ..................................................................................................................... A-7
Inter-Integrated Circuit (I2C) .......................................................................................... A-7
MMIO .......................................................................................................................... A-8
SPI ............................................................................................................................... A-9
UART ............................................................................................................................ A-10
Watchdog ...................................................................................................................... A-11

B AMS Installer Error Codes

Glossary ....................................................................................................................... 

Index
List of Examples

2–1  Sample Code to Access a GPIO Port .................................................................................. 2-3
2–2  Sample Code to Access a GPIO Port ................................................................................ 2-10
List of Figures

1–1 The Keil MCBSTM32F200 Board ................................................................. 1-2
1–2 Windows Formatter Settings ........................................................................ 1-3
1–3 The MCBSTM32F200 Touchscreen After Installing Java ME Embedded .... 1-6
1–4 PuTTY Options for Connecting to the Keil Board ......................................... 1-7
1–5 Logging and Command Line Interfaces ...................................................... 1-8
2–1 Adding API Permissions with NetBeans .................................................... 2-5
2–2 The Signing Pane in the NetBeans Project Properties ............................... 2-6
2–3 Debugging an IMlet on the Board Using NetBeans ................................... 2-7
2–4 Debugging an IMlet on the Board Using the Eclipse IDE ......................... 2-13
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–1</td>
<td>Ports Used by the Embedded Board</td>
<td>1-7</td>
</tr>
<tr>
<td>1–2</td>
<td>AMS CLI Commands</td>
<td>1-8</td>
</tr>
<tr>
<td>1–3</td>
<td>Additional System Commands Available in the AMS CLI</td>
<td>1-9</td>
</tr>
<tr>
<td>3–1</td>
<td>Problems and Solutions - Installing the Firmware on the Board</td>
<td>3-1</td>
</tr>
<tr>
<td>3–2</td>
<td>Problems and Solutions - Starting Oracle Java ME Embedded on the Board</td>
<td>3-2</td>
</tr>
<tr>
<td>3–3</td>
<td>Problems and Solutions - Oracle Java ME SDK and the NetBeans IDE</td>
<td>3-2</td>
</tr>
<tr>
<td>B–1</td>
<td>Installer Error Codes</td>
<td>B-1</td>
</tr>
</tbody>
</table>
Preface

This book describes how to install Oracle Java ME Embedded software onto a Keil MCBSTM32F200 embedded device. Readers using this guide must be familiar with the Information Module Profile - Next Generation (IMP-NG) 1.0 Specification.

Audience

This document is intended for developers who want to run Oracle Java ME Embedded software on embedded devices.

Documentation Accessibility

For information about Oracle’s commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For a complete list of documents with the Oracle Java ME Embedded software, see the Release Notes.

Operating System Commands

This document does not contain information on basic commands and procedures such as opening a terminal window, changing directories, and setting environment variables. See the software documentation that you received with your system for this information.
Shell Prompts

<table>
<thead>
<tr>
<th>Shell</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bourne shell and Korn shell</td>
<td>$</td>
</tr>
<tr>
<td>Windows</td>
<td>directory&gt;</td>
</tr>
</tbody>
</table>

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Running the Oracle Java ME Embedded Software on the Keil MCBSTM32F200 Board

The Keil MCBSTM32F200 board is the reference device of the Oracle Java ME Embedded software. This chapter documents how to install the Java ME Embedded software with RTX binary onto the Keil MCBSTM32F200 board, configuring the Java ME Embedded system, connecting to the command-line and logging interfaces, and installing and running a Java ME Embedded "IMlet" application.

The following items are required for developing on the Keil MCBSTM32F200 board:

- The Oracle Java ME Embedded RTX Software Distribution, Version 3.3.1
- A desktop computer running Windows XP or later with two USB ports
- A Keil MCBSTM32F200 Cortex Board
- A hardware debugger, such as the ULINK-ME or ULINK 2
- A MicroSD card (with an SD adapter, if necessary for connecting to the desktop computer)
- A networking LAN cable with RJ-45 interface
- A terminal emulation program, such as PuTTY

Downloading and Installing the MDK-ARM Lite Development Kit

To install the Oracle Java ME Embedded software on the reference board, first download and install version 4.54, 4.6, or 4.7 of the MDK-ARM development kit. The MDK-ARM development kit can be obtained from the following site:

https://www.keil.com/download/product

Be sure the download completes successfully: for example, the file size for the MDK-ARM Lite version is approximately 500 MB.

Once downloaded, install the MDK-ARM tool by double-clicking on the executable. After the tool install finishes, verify the installation by double-clicking on the Keil µVision 4 IDE executable.

WARNING: There is currently a bug in Version 4.60 of the MDK-ARM Development Kit that prevents an output file from being generated during binary flashing onto the Keil boards. If this occurs, please use a different version of the MDK-ARM Development Kit.
Downloading and Installing the PuTTY Terminal Emulator Program

Download the PuTTY Terminal Emulator Program (putty.exe) from the following site:

http://www.putty.org/

The terminal emulator executable is directly downloadable as putty.exe. The terminal emulator is used to connect to two separate sockets: one for the command-line interface (CLI) that issues commands to the board, and one for the logging or system output provided by the board.

Connecting the Keil MCBSTM32F200 Board to the Computer

The Keil MCBSTM32F200 board comes with two USB cables. Both cables must be connected to the desktop computer. The first USB cable attaches to the USBHS connector, and provides power to the board, as shown in Figure 1–1. The other connects to the hardware debugging unit, such as the ULINK-ME, which in turn connects to a compatible debugging port on the evaluation board. See the instructions that come with your specific hardware debugging unit for more information.

After the USB cables are connected, ensure that the board and the debugger are receiving power by verifying that the LEDs on both the board near the USB connection and the hardware debugger are lit. The LED near the USB connection should be lit green. With the Keil MCBSTM32F200, the backlight of the touchscreen display also lights up, even though it might not be displaying any information. See Figure 1–1.

Figure 1–1  The Keil MCBSTM32F200 Board

The first time the board is connected to the desktop computer, Windows typically installs the device driver software for the debugger. The ULINK-ME, for example, uses the standard set of USB drivers that comes with Microsoft Windows.
Setting Up the MicroSD Card

The MicroSD card must contain configuration files used by the Java ME Embedded native distribution, including the initialization properties and security policy files. Ensure that you are using a compatible SD card. This release has been tested with the Transcend 2 GB card. For a list of supported cards, please refer to the appropriate documentation for the RTX OS and the Keil F200 series embedded boards.

Note: All filenames on the MicroSD card must adhere to the 8.3 file format.

Follow these steps to prepare the card:

1. Insert the card on the Windows desktop computer, select the card in the My Computer window, and right-click and select Format...

2. Select File System as FAT32, Allocation Unit Size as the default allocation unit size, and ensure that Quick Format is not selected, as shown in Figure 1–2. The Volume label is optional. Press the Start button.

3. Once the formatting completes, copy all the files inside the directory sd_card/ in the Oracle Java ME Embedded distribution to the root directory of the SD card.

4. Edit the file rtc_upd.cfg on the root directory of the SD card. (This can be done by opening the file with a standard text editor.) Edit the current time and date that the device should use, in the format YYYY/MM/DD HH:MM:SS Zone. For example, the file rtc_upd.cfg could contain the following:

   2012/11/22 18:46:00 GMT

---

Figure 1–2 Windows Formatter Settings

![Windows Formatter Settings](image)
Next, rename the file `rtc_upd.cfg` to `rtc.cfg`.

5. Open the file `platform.cfg` on the root directory of the SD card to specify the appropriate networking information for the device.

Edit the MAC address if you have more than one board in your local network or if you think that there is a MAC address conflict. The following is an example of a MAC address:

```
#---------------------------------------------
# MAC Address
# - It should be unique value in the same LAN
#---------------------------------------------
mac.addr= 00:0A:0C:0E:07:0A
```

**Note:** The MAC address must comply with the IEEE 802.3 standard.

If your network uses dynamic IP addresses, uncomment the DHCP line under Dynamic IP Address and comment the lines under Static IP Address. If your network uses static IP addresses, uncomment and edit the lines under Static IP Address and comment the line under Dynamic IP Address.

The following is an example of configuring options for using static IP addresses:

```
#---------------------------------------------
# Dynamic IP address (DHCP)
#---------------------------------------------
#ip.method= dhcp

#---------------------------------------------
# static IP address
#---------------------------------------------
ip.method= static
ip.addr= 10.0.0.10
ip.netmask= 255.0.0.0
ip.gateway= 10.0.0.1
ip.dns= 10.0.0.1
```

6. In the file `platform.cfg`, disable the watchdog by setting its value to `false`:

```
#---------------------------------------------
# H/W Watchdog
#---------------------------------------------
watchdog.enable= false
```

The watchdog resets the system if a program appears to be stuck. However, the watchdog may trigger while the board communicates with the Oracle ME SDK and NetBeans, especially during debugging.

7. If you plan to use on-device debugging (ODD) with an IDE, edit the file `jwc_prop.ini` in the `java/` directory of the SD card. Set the property `odt_run_on_start` to `true` to enable ODD.
Installing the Firmware on the Evaluation Board

The Java ME Embedded distribution contains a utility that uses the MDK-ARM software to erase and flash the Oracle Java ME Embedded binary onto the evaluation board. Use the following procedure to prepare the SD card and install the firmware:

1. Use the **Safely Remove Hardware** function on the Windows desktop to remove the SD card from the computer. (In Windows XP, for example, the Safely Remove Hardware tool is present in the lower right system tray near the clock.) Alternatively, you can select the SD card in the My Computer window, right-click and select **Eject**. If necessary, remove the MicroSD card from the SD card housing.

2. Be sure that the board is not powered up by temporarily disconnecting the USB cables. Insert the MicroSD card into the SD card slot on the board.

3. Switch the "BAT/3V3" jumper near the RS-232C connection on the board to the left (BAT) position, outlined in Figure 1–3.

4. Connect the board to the Windows desktop computer as you did previously, including the network cable, the USB power cable, and the debugger with its USB cable. The board should now be receiving power.

5. On the desktop, check the file **flash.bat** inside the directory **flash/** in the Oracle Java ME Embedded software. Ensure that this file contains the correct path to the MDK-ARM tool binary. For example:

   ```
   C:\Keil\UV4\UV4.exe -f jmee.uvproj -o output.txt
   ```

6. With the board connected, run the batch file **flash.bat** to download the distribution to the evaluation board. The MDK-ARM tool notifies you when the download has completed. If the MDK-ARM tool requires a Flash Upgrade of the debugger, press **OK** to allow this, then restart the **flash.bat** when completed.

7. Once completed, push the **Reset** button on the board to start the Java ME Embedded native platform. The runtime will then use the configuration files on the MicroSD card to initialize itself. If successful, the touchscreen on the device should look similar to Figure 1–3. Note that after the time clock has been initialized, the runtime will rename the **rtc.cfg** file on the MicroSD card back to **rtc_upd.cfg** and ask you to reset the board to activate the clock’s settings.
Connecting to the CLI and Logging Ports

When the Java ME Embedded distribution is running, note the IP address that is shown on the touchscreen in green, and start two PuTTY executables. Use these to create raw socket connections to the IP address and two ports shown in Table 1–1. For example, the PuTTY settings to initiate a connection to the IP address of 10.0.0.10 and the port 65000 is shown in Figure 1–4. Note that the IP address of your embedded board may be different.
Running the Oracle Java ME Embedded Software on the Keil MCBSTM32F200 Board

Connecting to the CLI and Logging Ports

1-7

Figure 1–4  PuTTY Options for Connecting to the Keil Board

The two default ports that are used are shown in Table 1–1.

Table 1–1  Ports Used by the Embedded Board

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>65000</td>
<td>Logging / Java VM System Output</td>
</tr>
<tr>
<td>65002</td>
<td>Command-line interface</td>
</tr>
</tbody>
</table>

The window that outputs data from port 65000 provides logging and standard output information from the device. This is useful when determining if a Java ME embedded program ran successfully on the board. The window from port 65002 provides a command-line interface (CLI). Both the CLI and logging windows are shown in Figure 1–5. The logging port is defined in the jwc_prop.ini properties file using value assigned to the log.tcp.port property.
Connecting to the CLI and Logging Ports

In the CLI window, you can run the Application Management System (AMS) commands shown in Table 1–2. Note that the AMS syntax may change in future releases; entering `help [command]` is the best way to obtain the latest CLI syntax.

**Table 1–2  AMS CLI Commands**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`ams-list [INDEX or NAME</td>
<td>VENDOR]`</td>
</tr>
<tr>
<td><code>ams-install &lt;URL&gt; [username:password]</code></td>
<td>Install an IMlet using the specified JAR or JAD file, specified as a URL. An optional username and password can be supplied for login information as well.</td>
</tr>
<tr>
<td>`ams-update &lt;INDEX or NAME</td>
<td>VENDOR&gt;`</td>
</tr>
<tr>
<td>`ams-remove &lt;INDEX or NAME</td>
<td>VENDOR&gt;`</td>
</tr>
<tr>
<td>`ams-run &lt;INDEX or NAME</td>
<td>VENDOR&gt; [IMLET_ID] [-debug]`</td>
</tr>
<tr>
<td>`ams-stop &lt;INDEX or NAME</td>
<td>VENDOR&gt; [IMLET_ID]`</td>
</tr>
</tbody>
</table>

**WARNING:** The command-line interface (CLI) feature in this Oracle Java ME Embedded software release is provided only as a concept for your reference. It uses insecure connections with no encryption, authentication, or authorization.
When the `sysmenu` command is entered with the on option, additional system menu commands are available with the AMS CLI, as shown in Table 1–3.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`ams-suspend &lt;INDEX or NAME</td>
<td>VENDOR&gt; [IMLET_ID]`</td>
</tr>
<tr>
<td>`ams-resume &lt;INDEX or NAME</td>
<td>VENDOR&gt; [IMLET_ID]`</td>
</tr>
<tr>
<td>`ams-setup &lt;INDEX or NAME</td>
<td>VENDOR&gt;`</td>
</tr>
<tr>
<td>`ams-info &lt;INDEX or NAME</td>
<td>VENDOR&gt;`</td>
</tr>
<tr>
<td><code>ams-log &lt;command&gt; [param1, param2, ..., paramN]</code></td>
<td>Display the IMlet log or watchdog log if recorded by the watchdog handler in the platform</td>
</tr>
<tr>
<td><code>ams-log wdog</code></td>
<td></td>
</tr>
<tr>
<td>`ams-logger-list [INDEX or NAME</td>
<td>VENDOR]`</td>
</tr>
<tr>
<td>`ams-logger-info &lt;INDEX or NAME</td>
<td>VENDOR&gt; [LOGGER_NAME]`</td>
</tr>
<tr>
<td>`ams-logger-level-set &lt;INDEX or NAME</td>
<td>VENDOR&gt; [LOGGER_NAME] &lt;LOGGER_LEVEL&gt;`</td>
</tr>
<tr>
<td><code>help [command name]</code></td>
<td>Enable hidden system menu commands. Currently, the password is 12345.</td>
</tr>
<tr>
<td>`sysmenu &lt;on PASSWORD</td>
<td>off&gt;`</td>
</tr>
</tbody>
</table>

Table 1–2 (Cont.) AMS CLI Commands

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setprop &lt;KEY&gt; &lt;VALUE&gt;</code></td>
<td>Sets a property identified by &lt;KEY&gt; with the value &lt;VALUE&gt;</td>
</tr>
<tr>
<td><code>getprop &lt;KEY&gt;</code></td>
<td>Returns a property identified by &lt;KEY&gt;</td>
</tr>
<tr>
<td>`odd [on</td>
<td>off]`</td>
</tr>
<tr>
<td><code>shutdown [-r]</code></td>
<td>Perform either a shutdown of the board, or a reboot if the -r parameter has been passed. Note that the watchdog should be enabled in the platform.cfg file to successful reboot. (See Setting Up the MicroSD Card)</td>
</tr>
</tbody>
</table>

Here is a typical example of using the Application Management System (AMS) to install, list, run, and remove a Java ME Embedded application on the Cortex board.

Note: Note that the AMS syntax and output may change in future Oracle Java ME Embedded releases.
First, install the application using the `ams-install` command, specifying its location either on the MicroSD card (using the `Memorycard` directory, which maps to `memcard` on the filesystem) or across a network using a URL.

```
oracle>> ams-install file:///Memorycard/hello.jar
<<ams-install,start install,file:///Memorycard/hello.jar
<<ams-install,install status: stage 0, 5%
<<ams-install,install status: stage 3, 100%
<<ams-install,install status: stage 4, 100%
<<ams-install,OK,Install success
```

```
oracle>> ams-install http://www.example.com/netdemo.jar
<<ams-install,start install,http://www.example.com/netdemo.jar
<<ams-install,install status: stage 0, 5%
<<ams-install,install status: stage 3, 100%
<<ams-install,install status: stage 4, 100%
<<ams-install,OK,Install success
```

```
oracle>> ams-install http://www.example.com/notthere.jar
<<ams-install,start install,http://www.example.com/notthere.jar
<<ams-install,FAIL,errorCode=103(OTHER_ERROR)
```

Note that the final installation example failed with an error code and matching description. If the install process shows any error code, see Table B–1 in Appendix B for more information on how to resolve the error.

Once an IMlet is installed, verify it using the `ams-list` command. Here, we have added an additional IMlet: `rs232dem`. Each IMlet has been assigned a number for convenience.

```
oracle>> ams-list
<<ams-list,0.hello|Oracle,STOPPED
<<ams-list,1.netdemo|Oracle,STOPPED
<<ams-list,2.rs232dem|Oracle,RUNNING
<<ams-list,OK,3 suites are installed
```

The `ams-remove` command can be used to remove any installed IMlet.

```
oracle>> ams-remove 0
<<ams-remove,OK,hello removed
```

The results can again be verified with the `ams-list` command.

```
oracle>> ams-list
<<ams-list,1.netdemo|Oracle,STOPPED
<<ams-list,2.rs232dem|Oracle,RUNNING
<<ams-list,OK,2 suites are installed
```

Finally, start up the IMlet using the `ams-run` command. The application can be terminated with the `ams-stop` command.

```
oracle>> ams-run 1
<<ams-run,OK,started
```

```
oracle>> ams-list
<<ams-list,1.netdemo|Oracle,RUNNING
<<ams-list,2.rs232dem|Oracle,RUNNING
<<ams-list,OK,1 suites are installed
```

Subsequent chapters will discuss how to use either the NetBeans or Eclipse IDE to develop, install, and test applications directly on the board.
Additional Methods of Obtaining Logging Information

There are several additional ways to connect to the embedded board and obtain logging output. These include the following:

1. **UART - Through RS232**

   To utilize this method, connect to the board using an RS232 serial cable and use a tool such as PuTTY to create a serial connection from a terminal on the Windows desktop using the following options:
   - Baud rate: 115200
   - Data bits: 8
   - Stop bits: 1
   - Parity: None
   - Flow control: XON/XOFF

   Also, ensure that the `log.method` property from the `jwc_prop.ini` file contains the value **UART** in its list.

2. **USB/COM - Through a USB COM port and an appropriate driver**

   In order to use this method, connect a USB cable from your desktop to the Keil board. A USB to COM driver must also be installed on your Windows desktop machine. Drivers’ installers and installation instructions can be found in the driver directory of the Keil bundle. Finally, ensure that the `log.method` property from the `jwc_prop.ini` file contains the value **USBCOM** in its list. Then, create a serial connection from your terminal with the following options:
   - Baud rate: 115200
   - Data bits: 8
   - Stop bits: 1
   - Parity: None
   - Flow control: XON/XOFF

3. **ITM - Through ITM trace supported by MCU**

   An instrumentation trace can be used as supported by the ARM microcontroller. See the embedded board documentation for more information on this method.

   Also, ensure that the `log.method` property from the `jwc_prop.ini` file contains the value **ITM** in its list.
Additional Methods of Obtaining Logging Information
Using NetBeans or Eclipse with the Keil MCBSTM32F200 Board

Developers can run and debug IMlets on the Keil MCBSTM32F200 board directly from the NetBeans IDE or Eclipse IDE using the Oracle Java ME SDK. This chapter describes how to add the board to the Device Selector in the Oracle Java ME SDK and how to debug an IMlet on the board from both the NetBeans IDE and the Eclipse IDE.

Using NetBeans with the Keil MCBSTM32F200 Board

Running and debugging IMlet projects on the Keil MCBSTM32F200 board using the NetBeans IDE requires the following software:

- NetBeans IDE 7.3 with Java ME enabled
- Oracle Java ME SDK
- Oracle Java ME SDK NetBeans Plugin

Install the Oracle Java ME SDK Plugin for NetBeans

After installing NetBeans, use these steps to install the remaining software.

1. Ensure that Java ME is enabled in NetBeans. This can be done by selection Tools -> Plugins and selecting the Installed pane. Activate the Java ME plugin if it is not already activated.

2. Install the Java ME SDK distribution. See the Java ME SDK documentation for details.

3. Install the Oracle Java ME SDK NetBeans plugin. This is a downloadable ZIP file that consists of a number of NetBeans modules (.nbm files) that can be added using the Tools -> Plugins dialog and selecting the Downloaded pane. The Oracle Java ME SDK NetBeans plugin is required to use the Device Selector to connect to the board. See the Oracle Java ME SDK Release Notes for installation instructions: [http://docs.oracle.com/javame/dev-tools/jme-sdk-3.3/release-notes/toc.htm](http://docs.oracle.com/javame/dev-tools/jme-sdk-3.3/release-notes/toc.htm)

4. Ensure that the Oracle Java ME Embedded 3.3 appears in the list of Java ME platforms. In the NetBeans IDE, go to Tools -> Java Platforms. If the Oracle Java ME Embedded 3.3 does not appear in the list of J2ME platforms, follow these steps:
   - Click on Add Platform.
   - Select Java ME CLDC Platform Emulator and click Next.
Select the folder where the Oracle Java ME SDK 3.3 runtime for Keil MCBSTM32F200 resides and follow the instructions to install it. Then, click Finish to close the dialog.

5. Ensure that the Keil MCBSTM32F200 board is running the Oracle Java ME Embedded distribution. See Chapter 1 for more information on how to install the runtime distribution on the Keil MCBSTM32F200 board.

Adding the Keil MCBSTM32F200 Board to the Device Selector

Follow these steps to add the board to the Device Selector in the Oracle Java ME SDK:

1. Ensure that the property `odt_run_on_start` is set to true in the file `java/jwe_prop.ini` on the Keil MCBSTM32F200.

2. Ensure that the board is receiving power and the Oracle Java ME Embedded software is running.

3. If necessary, open TCP port 55123 in the firewall settings of your computer. The exact procedure to open a port differs depending on your version of Windows or the firewall software that is installed on your computer.

4. Start the NetBeans IDE. In the NetBeans IDE, go to Tools -> Java ME -> Device Selector.

5. On the Device Selector, click on the Add a Device button at the top of the Device Selector window.

6. Write the IP address of the Keil MCBSTM32F200 board in the IP Address field and click Next. You can find the IP address of the Keil MCBSTM32F200 board by looking at the touchscreen on the board.
Using NetBeans with the Keil MCBSTM32F200 Board

7. Once the device is detected, click **Finish** on the Device Detection screen.
The list of devices in the Device Selector should now include IMPNGExternalDevice1.

**Assigning the Keil MCBSTM32F200 Board to Your Project**

If you already have an existing NetBeans project with an IMlet that you want to run or debug on the board, follow these steps:

1. Right-click on your project and choose **Properties**.
2. Select the **Platform** category on the properties window.
3. Select **IMPNGExternalDevice1** from the device list.

If you are creating a new NetBeans project from scratch, follow these steps:

1. Select **File -> New Project**.
2. Select the **Java ME** category and **Embedded Application** in Projects. Click **Next**.
3. Provide a project name and click **Next**. Be sure that the **Create Default Package and IMlet Class** option is checked.
4. Ensure the Emulator Platform is **Oracle Java ME Embedded 3.3**. Then, select **IMPNGExternalDevice1** from the device list and click **Finish**.

After you assign the board to your project, the IMlets run on the board instead of on the emulator when you click on **Run Project** on the NetBeans IDE.

**Sample Source Code**

Once the project is created, use the source code in Example 2–1 for the default IMlet.java source file.

**Example 2–1  Sample Code to Access a GPIO Port**

```java
package embeddedapplication1;

import com.oracle.deviceaccess.PeripheralManager;
```
import com.oracle.deviceaccessPeripheralNotAvailableException;
import com.oracle.deviceaccessPeripheralNotFoundException;
import com.oracle.deviceaccess.gpio.GPIOPin;
import java.io.IOException;
import javax.microedition.midlet.*;

class IMlet extends MIDlet {
    public void startApp() {
        try {
            GPIOPin pin = (GPIOPin)PeripheralManager.open(2);
            boolean b = pin.getValue();
        } catch (PeripheralNotAvailableException ex) {
            ex.printStackTrace();
        } catch (PeripheralNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}

public class IMlet extends MIDlet {
    public void startApp() {
        try {
            GPIOPin pin = (GPIOPin)PeripheralManager.open(2);
            boolean b = pin.getValue();
        } catch (PeripheralNotAvailableException ex) {
            ex.printStackTrace();
        } catch (PeripheralNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}

This sample application will obtain an object representing GPIO pin 1 from the PeripheralManager, and attempt to obtain its high/low value.

Accessing the Peripherals on the Keil MCBSTM32F200

There are two ways to allow access to the peripherals on the Keil MCBSTM32F200. The first is to use unsigned applications and modify the security policy file, and the second is to digitally sign the application with the appropriate API permissions requested in the JAD file.

Method #1: Modifying the Security Policy File

Modifying the security policy file is only necessary in the event that a user must manually install the application on the board, at which point the unsigned application will be installed in the untrusted security domain.

With this method, simply add the line "allow: device_access" to the "untrusted" domains of the security policy file. By default, this is located on the SD card in the appdb/_policy.txt file, but be sure to check the security.policy file entry in the java/jwc_prop.ini file to verify the current file name.

Note that if an application is installed on the board using NetBeans or Eclipse during development, the application will automatically be installed in the maximum security domain as a convenience. Manual installation, however, will install the unsigned application into the untrusted security domain. Note that after development is finished, you should publish your applications with signed API permissions.
Method #2: Signing the Application with API Permissions

The second method is more complex, but is the preferred route for production applications that are widely distributed. First, the JAD file must have the proper API permissions. Right-click the project name (EmbeddedApplication1 in this example) and choose Properties. Select Application Descriptor, then in the resulting pane, select API Permissions. Click the Add... button, and add the com.oracle.deviceaccess/gpio API, as shown in Figure 2–1. Click OK to close the project properties dialog.

Figure 2–1 Adding API Permissions with NetBeans

Applications that access the Device Access APIs must also be signed. Here are the instructions on how to setup a keystore with a local certificate that can be used to sign the applications.

1. Generate a new self-signed certificate with the following command on the desktop, using the keytool that is shipped with the Java SE JDK.

   keytool -genkey -v -alias mycert -keystore mykeystore.ks -storepass spass -keypass kpass -validity 360 -keyalg rsa -keysize 2048 -dname "CN=thehost"

   This command will generate a 2048-bit RSA key pair and a self-signed certificate, placing them in a new keystore with a keystore password of "spass" and a key password of "kpass" that is valid for 360 days.

2. Modify the appdb/_main.ks keystore file from the Keil Micro SD card by performing the following mekeytool.exe command (or alternatively java -jar MEKeyTool.jar... if your distribution contains only that) that ships with the Oracle Java ME SDK 3.3 distribution.

   {mekeytool} -import -MEkeystore _main.ks -keystore mykeystore.ks -storepass spass -alias mycert -domain trusted
This will import the information in `mykeystore.ks` you just created to the `_main.ks` keystore.

Use the following steps to sign your application before deploying to the Keil MCBSTM32F200 board.

1. Right click your project and select **Properties**.
2. Choose the **Signing** option under the **Build** category.
3. Open the **Keystores Manager** and import the `mykeystore.ks` file that you created.
4. Check the **Sign Distribution** box. If you wish, unlock the keystore and the key with the passwords that you specified earlier. This is shown in Figure 2–2.
5. When the project is built and run, it will be digitally signed and deployed to the Keil MCBSTM32F200.

*Figure 2–2  The Signing Pane in the NetBeans Project Properties*

---

**Debugging an IMlet on the Keil MCBSTM32F200 Board**

Follow these steps to debug an IMlet using NetBeans:

1. Open your IMlet class on the NetBeans editor.
2. Click once directly on the line number where you want to set a breakpoint. The line number is replaced by a red square and the line is highlighted in red.
3. Select **Debug -> Debug Project** or use the Debug button on the toolbar.

The debugger connects to the debug agent on the board and the program execution stops at your breakpoint, as shown in *Figure 2–3.*
Figure 2–3  Debugging an IMlet on the Board Using NetBeans

![NetBeans Debugging Environment](image-url)

Figure 2–3 shown an entire NetBeans debugging environment that allows the programmer to execute a program step by step as well as add and remove variables from a watch list on the bottom of the screen.

For more information on using the device access APIs, please see the Device Access API Guide and the associated javadocs.

Using Eclipse with the Keil MCBSTM32F200 Board

Running and debugging IMlet projects on the Keil MCBSTM32F200 board using the Eclipse IDE requires the following software:

- Eclipse 3.7 Indigo or Eclipse 4.2 Juno
- Oracle Java ME SDK
- Oracle Java ME SDK Eclipse Plugin

Install the Oracle Java ME SDK Plugin for Eclipse

After installing Eclipse, use these steps to install the remaining software.

1. Install the Java ME SDK distribution. See the Java ME SDK documentation for details.
2. Install the Oracle Java ME SDK Eclipse plugin. This is required to use the Device Selector to connect to the board. See the Oracle Java ME SDK Release Notes for installation instructions:
   
   http://docs.oracle.com/javame/dev-tools/jme-sdk-3.3/release-notes/toc.htm

3. Ensure that the Keil MCBSTM32F200 board has the Oracle Java ME Embedded 3.3 runtime. See Chapter 1 for more information on how to install the runtime distribution on the Keil MCBSTM32F200 board.
4. Ensure that the Oracle Java ME Embedded 3.3 appears in the list of Java ME platforms. If it doesn’t appear, follow these steps for your project properties.
   - Under the Java ME category, select Device Management. In the Device Management window, press the Manual Install... button.
   - Browse to the base directory of the Java ME SDK environment and press the OK button. After the platform is scanned and the devices are installed, close each of the respective dialogs.

**Adding the Keil MCBSTM32F200 Board to the Device Selector**

Follow these steps to add the board to the Device Selector in the Oracle Java ME SDK:

1. Ensure that the property odt_run_on_start is set to true in the file java/jwc_prop.ini on the Keil MCBSTM32F200.
2. Ensure that the board is receiving power and the Oracle Java ME Embedded software is running.
3. If necessary, open TCP port 55123 in the firewall settings of your computer. The exact procedure to open a port differs depending on your version of Windows or the firewall software that is installed on your computer.
4. Start the Eclipse IDE. In the Eclipse IDE, go to Window -> Show View -> Other. In the popup window that appears, expand the Java ME node and select Device Selector.

5. On the Device Selector, click on the Add a Device button at the top of the Device Selector window.
Using NetBeans or Eclipse with the Keil MCBSTM32F200 Board

6. Write the IP address of the Keil MCBSTM32F200 board in the IP Address field and click Next. You can find the IP address of the Keil MCBSTM32F200 board by looking at the touchscreen on the board.

7. Once the device is detected, click Finish on the Add Device screen. The list of devices in the Device Selector should now include IMPNGExternalDevice1.

Assigning the Keil MCBSTM32F200 Board to Your Project

If you already have an existing Eclipse project with an IMlet that you want to run or debug on the board, follow these steps:
1. Right-click on your project and choose Properties.
2. Select the Java ME category on the properties window.
3. Select IMPNGExternalDevice1 from the device list. If the device is not shown, add it using the Add... button, selecting Oracle Java ME Embedded 3.3 as the SDK and IMPNGExternalDevice1 as the device.

If you are creating a new Eclipse project from scratch, follow these steps:

1. Select New -> Other. Then expand the Java ME tree node, and create a new MIDlet Project.
2. Expand the Java ME tree node, and create a new MIDlet Project.
3. In the Configuration pane of the creation dialog, select IMPNGExternalDevice1 from the device list.
4. Select the appropriate Profile and Configuration for your project.

After you assign the board to your project, the IMlets run on the board instead of on the emulator when you click on Project -> Run on the Eclipse IDE.

Sample Source Code

Once the project is created, use the source code in Example 2–2 for a default source file.

Example 2–2   Sample Code to Access a GPIO Port

```java
package embeddedapplication1;

import com.oracle.deviceaccess.PeripheralManager;
import com.oracle.deviceaccess.PeripheralNotAvailableException;
import com.oracle.deviceaccess.PeripheralNotFoundException;
import com.oracle.deviceaccess.gpio.GPIOPin;
import java.io.IOException;
import javax.microedition.midlet.*;

public class IMlet extends MIDlet {

    public void startApp() {
        try {
            GPIOPin pin = (GPIOPin)PeripheralManager.open(2);
            boolean b = pin.getValue();
        } catch (PeripheralNotAvailableException ex) {
            ex.printStackTrace();
        } catch (PeripheralNotFoundException ex) {
            ex.printStackTrace();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```
This sample application will obtain an object representing GPIO pin 1 from the PeripheralManager, and attempt to obtain its high/low value.

**Accessing the Peripherals on the Keil MCBSTM32F200**

There are two ways to allow access to the peripherals on the Keil MCBSTM32F200. The first is to use unsigned applications and modify the security policy file, and the second is to digitally sign the application with the appropriate API permissions requested in the JAD file.

**Method #1: Modifying the Security Policy File**

Modifying the security policy file is only necessary in the event that a user must manually install the application on the board, at which point the unsigned application will be installed in the untrusted security domain.

With this method, simply add the line "allow: device_access" to the "untrusted" domains of the security policy file. By default, this is located on the SD card in the appdb/_policy.txt file, but be sure to check the security.policy file entry in the java/jwc_prop.ini file to verify the current file name.

Note that if an application is installed on the board using NetBeans or Eclipse during development, the application will automatically be installed in the maximum security domain as a convenience. Manual installation, however, will install the unsigned application into the untrusted security domain. Note that after development is finished, you should publish your applications with signed API permissions.

**Method #2: Signing the Application with API Permissions**

The second method is more complex, but is the preferred route for applications that are widely distributed. Open the Application Descriptor for your project in the Packages window, and select the Application Descriptor pane. You will need to manually add or change the following lines in the Application Descriptor.

```plaintext
MIDlet-Permissions: com.oracle.deviceaccess.gpio
Microedition-Profile: IMP-NG
```

Applications that access the Device Access APIs must also be signed. Here are the instructions on how to setup a keystore with a local certificate that can be used to sign the applications.

1. Generate a new self-signed certificate with the following command on the desktop, using the keytool that is shipped with the Java SE JDK.

   ```sh
   keytool -genkey -v -alias mycert -keystore mykeystore.ks -storepass spass -keypass kpass -validity 360 -keyalg rsa -keysize 2048 -dname "CN=thehost"
   ```

   This command will generate a 2048-bit RSA key pair and a self-signed certificate, placing them in a new keystore with a key password of "spass" and a key password of "kpass" that is valid for 360 days.

2. Copy the appdb/_main.ks keystore file from the Keil MCBSTM32F200 over to the desktop and perform the following command using the mekeytool.exe command (or alternatively java -jar MEKeyTool.jar... if your distribution contains only that) that ships with the Oracle Java ME SDK 3.3 distribution.
{mekeytool} -import -MEkeystore _main.ks -keystore mykeystore.ks
-storepass spass -alias mycert -domain trusted

This will import the information in mykeystore.ks you just created to the _main.ks keystore. Once this is completed, copy the _main.ks file back to its original location on the Keil MCBSTM32F200.

Use the following steps to sign your application before deploying to the Keil MCBSTM32F200 board:

1. Right click your project and select Properties.
2. Choose the Signing option under the Java ME category.
3. Check the Enable Project Specific Settings checkbox. Import the mykeystore.ks file that you created as an External... keystore file. Provide the keystore and key passwords that you created earlier. Ensure that the mycert key alias is present.
4. Ensure that the project is being signed in the project’s Application Descriptor. When the project is built and run, it will be digitally signed when deployed to the Keil MCBSTM32F200.

### Debugging an IMlet on the Keil MCBSTM32F200 Board

After you assign the board to your project, follow these steps to debug an IMlet:

1. Open your IMlet class on the Eclipse editor.
2. Click once directly on the line number where you want to set a breakpoint. The line number has a small circle next to it to indicate a breakpoint.
3. Select Run -> Debug or use the Debug button on the toolbar.

The debugger connects to the debug agent on the board and the program execution stops at your breakpoint, as shown in Figure 2–4.
Figure 2–4 shown an entire Eclipse debugging environment that allows the programmer to execute a program step by step as well as add and remove variables from a watch list on the bottom of the screen.

For more information on using the device access APIs, please see the Device Access API Guide and the associated javadocs.
This chapter contains a list of common problems that you may encounter while installing and running the Oracle Java ME Embedded software on the Keil MCBSTM32F200 board. This chapter provides information on the causes of these problems and possible solutions for them.

The common problems in this chapter are grouped in four categories:

- Installing the Firmware on the Board
- Starting Oracle Java ME Embedded on the Board
- Using the Board with the Oracle Java ME SDK and the NetBeans IDE
- Other Problems

## Installing the Firmware on the Board

Table 3–1 contains information about problems and solutions when installing the firmware on the board.

### Table 3–1 Problems and Solutions - Installing the Firmware on the Board

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>flash.bat</code> script fails and the firmware is not installed on the board.</td>
<td>The debugger is not connected.</td>
<td>Connect the debug cable to the board and to the computer. Both the power cable and the debug cable must be connected to install the firmware on the board.</td>
</tr>
</tbody>
</table>

(continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>An output file specified in the <code>flash.bat</code> is not generated.</td>
<td>There is a bug in version 4.6 of the MDK-ARM toolkit.</td>
<td>Use MDK-ARM 4.54 or 4.7</td>
</tr>
</tbody>
</table>

## Starting Oracle Java ME Embedded on the Board

Table 3–2 contains information about problems and solutions when starting the runtime on the board.
Table 3–2  Problems and Solutions - Starting Oracle Java ME Embedded on the Board

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Java ME Embedded fails to start on the board, even after the firmware is installed correctly.</td>
<td>The SD card is not inserted correctly.</td>
<td>Eject the SD card from the board, insert it again and press the Reset button.</td>
</tr>
<tr>
<td>(continued)</td>
<td>The SD card is not formatted correctly using FAT32.</td>
<td>Format the SD card using the Windows format tool and copy the files from the Oracle Java ME Embedded distribution. See the section Setting Up the MicroSD Card in Chapter 1. Insert the SD card on the board and press the Reset button.</td>
</tr>
<tr>
<td>(continued)</td>
<td>Some files in the SD card are marked as read only.</td>
<td>Change the attributes of all files in the SD card and assign them write permissions. Insert the SD card on the board and press the Reset button.</td>
</tr>
<tr>
<td>(continued)</td>
<td>In rare cases, the file system of the SD card may be corrupted.</td>
<td>Re-format and re-install the contents of the SD card. See the section Setting Up the MicroSD Card in Chapter 1. Insert the SD card on the board and press the Reset button.</td>
</tr>
<tr>
<td>An authorization failure is given when an IMlet attempts to access any Device Access API.</td>
<td>The IMlet is not signed.</td>
<td>Sign the IMlet using a keystore with a trusted certificate authority (CA).</td>
</tr>
<tr>
<td>(continued)</td>
<td>The date on the board may invalidate the certificate used to authenticate the digital signature.</td>
<td>See the section Setting Up the MicroSD Card in Chapter 1 for the procedure on how to reset the date on the board by modifying the rtc_upd.cfg file.</td>
</tr>
<tr>
<td>Oracle Java ME Embedded fails to initialize the network on the board.</td>
<td>The network configuration is incorrect.</td>
<td>Edit the file platform.cfg in the SD card and edit the network configuration parameters. Insert the SD card on the board and press the Reset button.</td>
</tr>
<tr>
<td>Oracle Java ME Embedded fails to reset the clock on the board.</td>
<td>The rtc_upd.cfg configuration file was not renamed prior to board initialization.</td>
<td>Rename the rtc_upd.cfg file on the SD card to rtc.cfg and reset the board.</td>
</tr>
</tbody>
</table>

Using the Board with the Oracle Java ME SDK and the NetBeans IDE

Table 3–3 contains information about problems and solutions when using the board with the Oracle Java ME SDK and the NetBeans IDE.

Table 3–3  Problems and Solutions - Oracle Java ME SDK and the NetBeans IDE

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Java ME menu does not appear; unable to find the Device Selector.</td>
<td>The Oracle Java ME SDK plugin for the IDE is not installed.</td>
<td>Install the Oracle Java ME SDK plugin that is specifically for your development IDE, as described in Chapter 2.</td>
</tr>
<tr>
<td>The board is not detected when adding a new device to the Device Selector.</td>
<td>On-device debugging is not enabled.</td>
<td>Edit the file java/jwc_prop.ini and set the property odt_run_on_start to true. Insert the SD card on the board and press the Reset button.</td>
</tr>
<tr>
<td>The runtime on the board has problems accessing files or it is unstable.</td>
<td>The SD card is not supported or it is not formatted correctly.</td>
<td>Ensure that you are using a supported SD card and that you format it using the Windows formatting tool. See the section Setting Up the MicroSD Card in Chapter 1. Insert the SD card on the board and press the Reset button.</td>
</tr>
</tbody>
</table>
Other Problems

The TCP log of the Oracle Java ME Embedded software can help you diagnose problems that arise when running IMlets on the board. The TCP log is covered in the section Connecting to the CLI and Logging Ports in Chapter 1. If you cannot view the TCP log, the system log of the Keil MCBSTM32F200 board can provide you with the same information. Follow these steps to view the system log:

1. Connect a serial cable from the computer to the board. Use the upper DB9 connector.

2. Ensure that the log.method property from the jwc_prop.ini file contains the value UART in its list.

3. Open a terminal emulator on the computer, such as PuTTY.

4. Choose a serial connection and set the following options:
   - Speed: 115200
   - Data bits: 8
   - Stop bits: 1
   - Parity: None
   - Flow control: XON/XOFF
   In PuTTY these options are in the category Connection > Serial.

5. Open the connection. The system log appears on the terminal.

### Table 3–3  (Cont.) Problems and Solutions - Oracle Java ME SDK and the NetBeans IDE

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The board resets itself, especially when installing large IMlets or debugging applications.</td>
<td>In some cases the watchdog may trigger while the board communicates with the Oracle Java ME SDK and the NetBeans IDE.</td>
<td>Disable the watchdog on the file platform.cfg. See the section Setting Up the MicroSD Card in Chapter 1. Insert the SD card on the board and press the Reset button.</td>
</tr>
<tr>
<td>The debugging session freezes, disconnects unexpectedly, or shows error messages.</td>
<td>The firewall on the computer is blocking some debugging traffic.</td>
<td>Open TCP port 2808 on your firewall configuration settings. The exact procedure to open a port differs depending on your version of Windows or your firewall software.</td>
</tr>
<tr>
<td>(continued)</td>
<td>Thunderbird is using a port that is needed for communication with the board.</td>
<td>Close thunderbird.exe during the debugging session.</td>
</tr>
</tbody>
</table>

Other Problems
This appendix describes the proper ID and names for the various peripheral ports and
buses for the Keil MCBSTM32F200 embedded board, which are accessible using the
Device Access APIs. Note that any IMlet that accesses the Device Access APIs must be
digitally signed using a trusted certificate authority. An IMlet that is not signed will
encounter an authentication error when attempting to access the Device Access APIs.

The Keil Reference Manual for the MCBSRM32F200 board provides additional
technical information about the peripherals accessible via the Device Access APIs.

Analog-to-Digital Converter (ADC)

The following ADC channels are pre-configured.

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| 110                 | ADC1                  | On-board Potentiometer connected to PF9 | converterNumber = 1  
channelNumber = 7  
resolution = 12  
samplingInterval = 14857  
samplingTime = 15000 (ignored) |

The following additional ADC channels are available ad-hoc.

<table>
<thead>
<tr>
<th>Mapped To</th>
<th>Converter(s)</th>
<th>Channel</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF6</td>
<td>2</td>
<td>4</td>
<td>Not connected</td>
</tr>
<tr>
<td>PF7</td>
<td>2</td>
<td>5</td>
<td>Not connected</td>
</tr>
<tr>
<td>PF8</td>
<td>2</td>
<td>6</td>
<td>Not connected</td>
</tr>
<tr>
<td>PF10</td>
<td>2</td>
<td>8</td>
<td>Not connected</td>
</tr>
<tr>
<td>PA3</td>
<td>0-2</td>
<td>3</td>
<td>USBHS (USBHS_EN J2)</td>
</tr>
<tr>
<td>PA5</td>
<td>0-1</td>
<td>5</td>
<td>Microphone (M_DATA J5), DAC2</td>
</tr>
<tr>
<td>PA6</td>
<td>0-1</td>
<td>6</td>
<td>Camera (CAM_EN J17)</td>
</tr>
<tr>
<td>PB0</td>
<td>0-1</td>
<td>8</td>
<td>USBHS (USBHS_EN J2)</td>
</tr>
<tr>
<td>PB1</td>
<td>0-1</td>
<td>9</td>
<td>USBHS (USBHS_EN J2)</td>
</tr>
</tbody>
</table>
Pulse Counter

There are four pulse counters on the board. Each has a GPIO pin source: PA0, PB3, PB10, and PA3. Note that each source has only one counter, so the "number" property is not used for the Keil board.

The following pulse counters are pre-configured:

<table>
<thead>
<tr>
<th>Mapped To</th>
<th>Converter(s)</th>
<th>Channel</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC0</td>
<td>0-2</td>
<td>11</td>
<td>USBHS (USBHS_EN J2)</td>
</tr>
<tr>
<td>PF9</td>
<td>2</td>
<td>7</td>
<td>Potentiometer</td>
</tr>
</tbody>
</table>

Additional notes:

- There are 48 total channels. For more information on assigned external pins, see the manufacturer board schematics.
- For the converter number, acceptable values range from 0 to 2, which map to ADC1 to ADC3 on the device. For the channel number, acceptable values range from 0 to 15. Note that only one channel may be opened on each converter at a time. Consequently, only three channels may be in operation at any one time.
- The internal sampling values range from [14857, 46857]. However, the sampling time is ignored.
There are no pulse counters available ad-hoc.

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>COUNTER_PA0</td>
<td>“WAKEUP” button connected to PA0</td>
<td>counterNumber - not supported / ignored type = PulseCounterConfig.TYPE_FALLING_EDGE_ONLY portNumber = 0 pinNumber = 0</td>
</tr>
<tr>
<td>202</td>
<td>COUNTER_PB3</td>
<td>PB3</td>
<td>counterNumber - not supported / ignored type = PulseCounterConfig.TYPE_FALLING_EDGE_ONLY portNumber = 1 pinNumber = 3</td>
</tr>
<tr>
<td>203</td>
<td>COUNTER_PB10</td>
<td>PB10</td>
<td>counterNumber - not supported / ignored type = PulseCounterConfig.TYPE_FALLING_EDGE_ONLY portNumber = 1 pinNumber = 10</td>
</tr>
<tr>
<td>204</td>
<td>COUNTER_PA3</td>
<td>PA3</td>
<td>counterNumber - not supported / ignored type = PulseCounterConfig.TYPE_FALLING_EDGE_ONLY portNumber = 0 pinNumber = 3</td>
</tr>
</tbody>
</table>
Digital-to-Analog Converter (DAC)

The following DAC devices are pre-configured:

<table>
<thead>
<tr>
<th>DAAP Peripheral ID</th>
<th>DAAP Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| 120                | DAC1                 | PA4, connected to Analog input 'AIN1A' of CODEC IC11 | converterNumber = 0  
channelNumber = 0  
resolution = 12  
samplingInterval = 1000 |
| 121                | DAC2                 | PA5, connected to 'CLK' input of HS USB OTG transceiver IC6. To use DAC2 the IC should be disabled by disconnecting J2 'USBHS_EN' jumper | converterNumber = 0  
channelNumber = 1  
resolution = 12  
samplingInterval = 1000 |

There are no DAC channels available ad-hoc.

GPIO Pins

The following GPIO pins are pre-configured:
<table>
<thead>
<tr>
<th>DAAPID Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LED 1</td>
<td>LED PH3</td>
<td>portNumber = 7 pinNumber = 3 direction = GPIOPinConfig.DIR_OUTPUT_ONLY mode = GPIOPinConfig.MODE_OUTPUT_PULL trigger = GPIOPinConfig.TRIGGER_NONE initValue = false</td>
</tr>
<tr>
<td>2</td>
<td>LED 2</td>
<td>LED PH6</td>
<td>portNumber = 7 pinNumber = 6 direction = GPIOPinConfig.DIR_OUTPUT_ONLY mode = GPIOPinConfig.MODE_OUTPUT_PULL trigger = GPIOPinConfig.TRIGGER_NONE initValue = false</td>
</tr>
<tr>
<td>3</td>
<td>LED 3</td>
<td>LED PH7</td>
<td>portNumber = 7 pinNumber = 7 direction = GPIOPinConfig.DIR_OUTPUT_ONLY mode = GPIOPinConfig.MODE_OUTPUT_PULL trigger = GPIOPinConfig.TRIGGER_NONE initValue = false</td>
</tr>
<tr>
<td>4</td>
<td>LED 4</td>
<td>LED PI10</td>
<td>portNumber = 8 pinNumber = 10 direction = GPIOPinConfig.DIR_OUTPUT_ONLY mode = GPIOPinConfig.MODE_OUTPUT_PULL trigger = GPIOPinConfig.TRIGGER_NONE initValue = false</td>
</tr>
<tr>
<td>5</td>
<td>WAKEUP BUTTON or BUTTON 1</td>
<td>&quot;WAKEUP&quot; button</td>
<td>portNumber = 0 pinNumber = 0 direction = GPIOPinConfig.DIR_INPUT_ONLY mode = GPIOPinConfig.MODE_INPUT_PULL_DOWN trigger = GPIOPinConfig.TRIGGER_RISING_EDGE initValue - ignored</td>
</tr>
</tbody>
</table>
The following GPIO pins are available ad-hoc. Note that some GPIO pins are used at the system level, so opening those pins may affect board behavior.

- Accepted port numbers are PA=0, PB=1, PC=2, PD=3, PE=4, PF=5, PG=6, PH=7, PI=8.
- Accepted pins number are from 0 to 15.
- Depending on the board, PB0 and PB1 may be unused.

---

**WARNING:** The information in this appendix is presented in addition to the board schematics provided by the manufacturer. Be sure to familiarize yourself with the manufacturer's documentation for all of the GPIO pins. Failure to do so can result in damaged peripherals or even a damaged embedded board.

---

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| 6                   | TAMPER BUTTON or BUTTON 2 | “TAMPER” button | portNumber = 2  
                        |                       | pinNumber = 13  
                        |                       | direction = GPIOPinConfig.DIR_INPUT_ONLY  
                        |                       | mode = GPIOPinConfig.MODE_INPUT_PULL_UP  
                        |                       | trigger = GPIOPinConfig.TRIGGER_FALLING_EDGE  
                        |                       | initValue - ignored  |
| 7                   | USER BUTTON or BUTTON 3 | “USER” button | portNumber = 6  
                        |                       | pinNumber = 15  
                        |                       | direction = GPIOPinConfig.DIR_INPUT_ONLY  
                        |                       | mode = GPIOPinConfig.MODE_INPUT_PULL_UP  
                        |                       | trigger = GPIOPinConfig.TRIGGER_FALLING_EDGE  
                        |                       | initValue - ignored  |
GPIO Ports

The following GPIO ports are pre-configured:

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>LEDS</td>
<td>PH3, PH6, PH7, PI10</td>
<td>direction = GPIOPortConfig.DIR_OUTPUT_ONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>initValue = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mode of pins = GPIOPinConfig.MODE_OUTPUT_PUSH_PULL</td>
</tr>
</tbody>
</table>

The following GPIO ports are available ad-hoc:
- All combinations of available pins

Inter-Integrated Circuit (I2C)

The following I2C devices are pre-configured:

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>I2C_JOYSTICK</td>
<td>On-board Joystick connected via I2C bus</td>
<td>busNumber = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>address = 68</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>clockFrequency = 100000</td>
</tr>
</tbody>
</table>
The following I2C devices are available ad-hoc using connectors PB8 and PB9:

- Any address on bus 0

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| 51                  | LIS3DH                | LIS3DH MEMS motion sensor (On-board 3-axis accelerometer connected via I2C bus) | busNumber = 0  
address = 24  
clockFrequency = 100000 |

The following MMIO peripherals are available:

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>TESTDEVICE</td>
<td>byteOrdering = Peripheral.LITTLE_ENDIAN</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>WDOGLOG</td>
<td>byteOrdering = Peripheral.LITTLE_ENDIAN</td>
<td></td>
</tr>
<tr>
<td>999</td>
<td>RTC</td>
<td>byteOrdering = Peripheral.LITTLE_ENDIAN</td>
<td></td>
</tr>
</tbody>
</table>

The MMIO raw memories are shown here:
Note that the addresses [0x2001FE00 - 20020000) are part of the external RAM reserved for MMIO, and any type will be allowed. In addition, the addresses [0x40000000 - 0x50060C00) and [0xA0000000 - 0xA0001000) are peripheral IO registers. Be sure to check the manufacturer’s documentation for actual sizes, type and valid values. With each of the addresses above, the final address is non-inclusive.

There are no devices with event support.

### SPI

The following SPI devices are pre-configured:

<table>
<thead>
<tr>
<th>DA-API Peripheral ID</th>
<th>DA-API Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>SPI2_Slave</td>
<td>MOSI on PB15</td>
<td>busNumber = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MISO on PB14</td>
<td>address = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCK on PB13</td>
<td>clockFrequency = 2000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CS on PB12</td>
<td>clockMode = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>wordLength = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bitOrdering = 1</td>
</tr>
</tbody>
</table>

Configuration is tested with external SPI Digital thermometer (DS1722, Dallas Semiconductor)

The following SPI devices are available ad-hoc:

- SPI bus numbers: SPI1 = 0; SPI2 = 1; SPI3 = 2

  Clock frequency: 2MHz = 20000000; 25MHz = 25000000; 50MHz = 50000000; 100MHz = 100000000

  Clock polarity and phase: 0 = CPOL_Low | CPHA_1Edge; 1 = CPOL_Low | CPHA_2Edge; 2 = CPOL_High | CPHA_1Edge; 3 = CPOL_High | CPHA_2Edge
Bit ordering for data transfers: MSB = 1; LSB = 0

The following UART devices are pre-configured:

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| 14                  | UART                  | Connected to SubD 9F connector if jumpers J13 and J14 are set to TX3/4 and RX3/4 (for 1.1 boards version only) | uartNumber = 0  
baudRate = 19200  
dataBits = UARTConfig.DATABITS_8 or UARTConfig.DATABITS_9  
parity = UARTConfig.PARITY_NONE or UARTConfig.PARITY_EVEN or UARTConfig.PARITY_ODD  
stopBits = UARTConfig.STOPBITS_1 or UARTConfig.STOPBITS_2  
flowcontrol = UARTConfig.FLOWCONTROL_NONE |

There are no UART devices available ad-hoc.

The deviceaccess.uart.prefix property in the jwc_prop.ini file may contain a prefix for easy conversion of the UARTConfig.portNumber value to a platform-specific port name. For example, the property may be set to "COM" in a Windows environment,
Watchdog

The following watchdog devices are pre-configured:

<table>
<thead>
<tr>
<th>DAAPI Peripheral ID</th>
<th>DAAPI Peripheral Name</th>
<th>Mapped To</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>WDG</td>
<td>Platform Watchdog</td>
<td></td>
</tr>
</tbody>
</table>

There are no watchdog devices available ad-hoc.
Table B–1 lists the error codes that the AMS command-line interface shows when the installation of an IMlet fails. The description of each code contains more information about the problem that caused the error.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALAA_ALIAS_NOT_FOUND</td>
<td>78</td>
<td>Application Level Access Authorization: The alias definition is missing.</td>
</tr>
<tr>
<td>ALAA_ALIAS_WRONG</td>
<td>80</td>
<td>Application Level Access Authorization: The alias definition is wrong.</td>
</tr>
<tr>
<td>ALAA_MULTIPLE_ALIAS</td>
<td>79</td>
<td>Application Level Access Authorization: An alias has multiple entries that match.</td>
</tr>
<tr>
<td>ALAA_TYPE_WRONG</td>
<td>77</td>
<td>Application Level Access Authorization: The MIDlet-Access-Auth-Type has missing parameters.</td>
</tr>
<tr>
<td>ALREADY_INSTALLED</td>
<td>39</td>
<td>The JAD matches a version of a suite already installed.</td>
</tr>
<tr>
<td>APP_INTEGRITY_FAILURE_DEPENDENCY_CONFLICT</td>
<td>69</td>
<td>Application Integrity Failure: two or more dependencies exist on the component with the same name and vendor, but have different versions or hashes.</td>
</tr>
<tr>
<td>APP_INTEGRITY_FAILURE_DEPENDENCY_MISMATCH</td>
<td>70</td>
<td>Application Integrity Failure: there is a component name or vendor mismatch between the component JAD and IMlet or component JAD that depends on it.</td>
</tr>
<tr>
<td>APP_INTEGRITY_FAILURE_HASH_MISMATCH</td>
<td>68</td>
<td>Application Integrity Failure: hash mismatch.</td>
</tr>
<tr>
<td>ATTRIBUTE_MISMATCH</td>
<td>50</td>
<td>A attribute in both the JAD and JAR manifest does not match.</td>
</tr>
<tr>
<td>AUTHORIZATION_FAILURE</td>
<td>49</td>
<td>Application authorization failure, possibly indicating that the application was not digitally signed.</td>
</tr>
<tr>
<td>CA_DISABLED</td>
<td>60</td>
<td>Indicates that the trusted certificate authority (CA) for this suite has been disabled for software authorization.</td>
</tr>
<tr>
<td>CANCELED</td>
<td>101</td>
<td>Canceled by user.</td>
</tr>
<tr>
<td>CANNOT_AUTH</td>
<td>35</td>
<td>The server does not support basic authentication.</td>
</tr>
<tr>
<td>Constant</td>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CIRCULAR_COMPONENT_DEPENDENCY</td>
<td>64</td>
<td>Circular dynamic component dependency.</td>
</tr>
<tr>
<td>COMPONENT_DEPS_LIMIT_EXCEEDED</td>
<td>65</td>
<td>Dynamic component dependencies limit exceeded.</td>
</tr>
<tr>
<td>COMPONENT_NAMESPACE_COLLISION</td>
<td>72</td>
<td>The namespace used by a component is the same as another.</td>
</tr>
<tr>
<td>CONTENT_HANDLER_CONFLICT</td>
<td>55</td>
<td>The installation of a content handler would conflict with an already installed handler.</td>
</tr>
<tr>
<td>CORRUPT_DEPENDENCY_HASH</td>
<td>71</td>
<td>A dependency has a corrupt hash code.</td>
</tr>
<tr>
<td>CORRUPT_JAR</td>
<td>36</td>
<td>An entry could not be read from the JAR.</td>
</tr>
<tr>
<td>CORRUPT_PROVIDER_CERT</td>
<td>5</td>
<td>The content provider certificate cannot be decoded.</td>
</tr>
<tr>
<td>CORRUPT_SIGNATURE</td>
<td>8</td>
<td>The JAR signature cannot be decoded.</td>
</tr>
<tr>
<td>DEVICE_INCOMPATIBLE</td>
<td>40</td>
<td>The device does not support either the configuration or profile in the JAD.</td>
</tr>
<tr>
<td>DUPLICATED_KEY</td>
<td>88</td>
<td>Duplicated JAD/manifest key attribute</td>
</tr>
<tr>
<td>EXPIRED_CA_KEY</td>
<td>12</td>
<td>The certificate authority’s public key has expired.</td>
</tr>
<tr>
<td>EXPIRED_PROVIDER_CERT</td>
<td>11</td>
<td>The content provider certificate has expired.</td>
</tr>
<tr>
<td>INCORRECT_FONT_LOADING</td>
<td>82</td>
<td>A font that is contained with the JAR cannot be loaded.</td>
</tr>
<tr>
<td>INSUFFICIENT_STORAGE</td>
<td>30</td>
<td>Not enough storage for this suite to be installed.</td>
</tr>
<tr>
<td>INVALID_CONTENT_HANDLER</td>
<td>54</td>
<td>The MicroEdition-Handler-&lt;n&gt; JAD attribute has invalid values.</td>
</tr>
<tr>
<td>INVALID_JAD_TYPE</td>
<td>37</td>
<td>The server did not have a resource with the correct type or the JAD downloaded has the wrong media type.</td>
</tr>
<tr>
<td>INVALID_JAD_URL</td>
<td>43</td>
<td>The JAD URL is invalid.</td>
</tr>
<tr>
<td>INVALID_JAR_TYPE</td>
<td>38</td>
<td>The server did not have a resource with the correct type or the JAR downloaded has the wrong media type.</td>
</tr>
<tr>
<td>INVALID_JAR_URL</td>
<td>44</td>
<td>The JAR URL is invalid.</td>
</tr>
<tr>
<td>INVALID_KEY</td>
<td>28</td>
<td>A key for an attribute is not formatted correctly.</td>
</tr>
<tr>
<td>INVALID_NATIVE_LIBRARY</td>
<td>85</td>
<td>A native library contained within the JAR cannot be loaded.</td>
</tr>
<tr>
<td>INVALID_PACKAGING</td>
<td>87</td>
<td>A dependency cannot be satisfied.</td>
</tr>
<tr>
<td>INVALID_PAYMENT_INFO</td>
<td>58</td>
<td>Indicates that the payment information provided with the IMlet suite is incomplete or incorrect.</td>
</tr>
<tr>
<td>INVALID_PROVIDER_CERT</td>
<td>7</td>
<td>The signature of the content provider certificate is invalid.</td>
</tr>
</tbody>
</table>
Table B–1 (Cont.) Installer Error Codes

<table>
<thead>
<tr>
<th>Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_RMS_DATA_TYPE</td>
<td>76</td>
<td>The server did not have a resource with the correct type or the JAD downloaded has the wrong media type.</td>
</tr>
<tr>
<td>INVALID_RMS_DATA_URL</td>
<td>73</td>
<td>The RMS data file URL is invalid.</td>
</tr>
<tr>
<td>INVALID_SERVICE_EXPORT</td>
<td>86</td>
<td>A LIBlet that exports a service with a LIBlet Services attribute does not contain the matching service provider configuration information.</td>
</tr>
<tr>
<td>INVALID_SIGNATURE</td>
<td>9</td>
<td>The signature of the JAR is invalid.</td>
</tr>
<tr>
<td>INVALID_VALUE</td>
<td>29</td>
<td>A value for an attribute is not formatted correctly.</td>
</tr>
<tr>
<td>INVALID_VERSION</td>
<td>16</td>
<td>The format of the version is invalid.</td>
</tr>
<tr>
<td>IO_ERROR</td>
<td>102</td>
<td>A low-level hardware error has occurred.</td>
</tr>
<tr>
<td>JAD_MOVED</td>
<td>34</td>
<td>The JAD URL for an installed suite is different than the original JAD URL.</td>
</tr>
<tr>
<td>JAD_NOT_FOUND</td>
<td>2</td>
<td>The JAD was not found.</td>
</tr>
<tr>
<td>JAD_SERVER_NOT_FOUND</td>
<td>1</td>
<td>The server for the JAD was not found.</td>
</tr>
<tr>
<td>JAR_CLASSES_VERIFICATION_FAILED</td>
<td>56</td>
<td>Not all classes within JAR package can be successfully verified with class verifier.</td>
</tr>
<tr>
<td>JAR_IS_LOCKED</td>
<td>100</td>
<td>Component or MIDlet or IMlet suite is locked by the system.</td>
</tr>
<tr>
<td>JAR_NOT_FOUND</td>
<td>20</td>
<td>The JAR was not found at the URL given in the JAD.</td>
</tr>
<tr>
<td>JAR_SERVER_NOT_FOUND</td>
<td>19</td>
<td>The server for the JAR was not found at the URL given in the JAD.</td>
</tr>
<tr>
<td>JAR_SIZE_MISMATCH</td>
<td>31</td>
<td>The JAR downloaded was not the same size as given in the JAD.</td>
</tr>
<tr>
<td>MISSING_CONFIGURATION</td>
<td>41</td>
<td>The configuration is missing from the manifest.</td>
</tr>
<tr>
<td>MISSING_DEPENDENCY_HASH</td>
<td>67</td>
<td>A dependency hash code is missing.</td>
</tr>
<tr>
<td>MISSING_DEPENDENCY_JAD_URL</td>
<td>66</td>
<td>A dependency JAD URL is missing.</td>
</tr>
<tr>
<td>MISSING_JAR_SIZE</td>
<td>21</td>
<td>The JAR size is missing.</td>
</tr>
<tr>
<td>MISSING_JAR_URL</td>
<td>18</td>
<td>The URL for the JAR is missing.</td>
</tr>
<tr>
<td>MISSING_PROFILE</td>
<td>42</td>
<td>The profile is missing from the manifest.</td>
</tr>
<tr>
<td>MISSING_PROVIDER_CERT</td>
<td>4</td>
<td>The content provider certificate is missing.</td>
</tr>
<tr>
<td>MISSING_SUITE_NAME</td>
<td>13</td>
<td>The name of MIDlet or IMlet suite is missing.</td>
</tr>
<tr>
<td>MISSING_VENDOR</td>
<td>14</td>
<td>The vendor is missing.</td>
</tr>
<tr>
<td>MISSING_VERSION</td>
<td>15</td>
<td>The version is missing.</td>
</tr>
<tr>
<td>NEW_VERSION</td>
<td>32</td>
<td>This suite is newer than the one currently installed.</td>
</tr>
</tbody>
</table>
Table B-1  (Cont.) Installer Error Codes

<table>
<thead>
<tr>
<th>Constant</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>0</td>
<td>No error.</td>
</tr>
<tr>
<td>NOT_YET_VALID_PROVIDER_CERT</td>
<td>89</td>
<td>A certificate is not yet valid.</td>
</tr>
<tr>
<td>NOT_YET_VALID_CA_KEY</td>
<td>90</td>
<td>A CA’s public key is not yet valid.</td>
</tr>
<tr>
<td>OLD_VERSION</td>
<td>17</td>
<td>This suite is older that the one currently installed.</td>
</tr>
<tr>
<td>OTHER_ERROR</td>
<td>103</td>
<td>Other errors.</td>
</tr>
<tr>
<td>PROXY_AUTH</td>
<td>51</td>
<td>Indicates that the user must first authenticate with the proxy.</td>
</tr>
<tr>
<td>PUSH_CLASS_FAILURE</td>
<td>48</td>
<td>The class in a push attribute is not in Midlet-&lt;n&gt; attribute.</td>
</tr>
<tr>
<td>PUSH_DUP_FAILURE</td>
<td>45</td>
<td>The connection in a push entry is already taken.</td>
</tr>
<tr>
<td>PUSH_FORMAT_FAILURE</td>
<td>46</td>
<td>The format of a push attribute has an invalid format.</td>
</tr>
<tr>
<td>PUSH_PROTO_FAILURE</td>
<td>47</td>
<td>The connection in a push attribute is not supported.</td>
</tr>
<tr>
<td>REVOKED_CERT</td>
<td>62</td>
<td>The certificate has been revoked.</td>
</tr>
<tr>
<td>RMS_DATA_DECRYPT_PASSWORD</td>
<td>83</td>
<td>Indicates that a password is required to decrypt RMS data.</td>
</tr>
<tr>
<td>RMS_DATA_ENCRYPT_PASSWORD</td>
<td>84</td>
<td>Indicates that a password is required to encrypt RMS data.</td>
</tr>
<tr>
<td>RMS_DATA_NOT_FOUND</td>
<td>75</td>
<td>The RMS data file was not found at the specified URL.</td>
</tr>
<tr>
<td>RMS_DATA_SERVER_NOT_FOUND</td>
<td>74</td>
<td>The server for the RMS data file was not found at the specified URL.</td>
</tr>
<tr>
<td>RMS_INITIALIZATION_FAILURE</td>
<td>81</td>
<td>Failure to import RMS data.</td>
</tr>
<tr>
<td>SUITE_NAME_MISMATCH</td>
<td>25</td>
<td>The MIDlet or IMlet suite name does not match the one in the JAR manifest.</td>
</tr>
<tr>
<td>TOO_MANY_PROPS</td>
<td>53</td>
<td>Indicates that either the JAD or manifest has too many properties to fit into memory.</td>
</tr>
<tr>
<td>TRUSTED_OVERWRITE_FAILURE</td>
<td>52</td>
<td>Indicates that the user tried to overwrite a trusted suite with an untrusted suite during an update.</td>
</tr>
<tr>
<td>UNAUTHORIZED</td>
<td>33</td>
<td>Web server authentication failed or is required.</td>
</tr>
<tr>
<td>UNKNOWN_CA</td>
<td>6</td>
<td>The certificate authority (CA) that issued the content provider certificate is unknown.</td>
</tr>
<tr>
<td>UNKNOWN_CERT_STATUS</td>
<td>63</td>
<td>The certificate is unknown to OCSP server.</td>
</tr>
<tr>
<td>UNSUPPORTED_CERT</td>
<td>10</td>
<td>The content provider certificate has an unsupported version.</td>
</tr>
<tr>
<td>UNSUPPORTED_CHAR_ENCODING</td>
<td>61</td>
<td>Indicates that the character encoding specified in the MIME type is not supported.</td>
</tr>
<tr>
<td>Constant</td>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UNSUPPORTED_PAYMENT_INFO</td>
<td>57</td>
<td>Indicates that the payment information provided with the MIDlet or IMlet suite is incompatible with the current implementation.</td>
</tr>
<tr>
<td>UNTRUSTED_PAYMENT_SUITE</td>
<td>59</td>
<td>Indicates that the MIDlet or IMlet suite has payment provisioning information but it is not trusted.</td>
</tr>
<tr>
<td>VENDOR_MISMATCH</td>
<td>27</td>
<td>The vendor does not match the one in the JAR manifest.</td>
</tr>
<tr>
<td>VERSION_MISMATCH</td>
<td>26</td>
<td>The version does not match the one in the JAR manifest.</td>
</tr>
</tbody>
</table>
Access Point
A network-connectivity configuration that is predefined on a device. An access point can represent different network profiles for the same bearer type, or for different bearer types that may be available on a device, such as WiFi or bluetooth.

ADC
Analog-to-Digital Converter. A hardware device that converts analog signals (time and amplitude) into a stream of binary numbers that can be processed by a digital device.

AMS
Application Management System. The system functionality that completes tasks such as installing applications, updating applications, and managing applications between foreground and background.

APDU
Application Protocol Data Unit. A communication mechanism used by SIM Cards and smart cards to communicate with card reader software or a card reader device.

API
Application Programming Interface. A set of classes used by programmers to write applications that provide standard methods and interfaces and eliminate the need for programmers to reinvent commonly used code.

ARM
Advanced RISC Machine. A family of computer processors using reduced instruction set (RISC) CPU technology, developed by ARM Holdings. ARM is a licensable instruction set architecture (ISA) and is used in the majority of embedded platforms.

AT commands
A set of commands developed to facilitate modem communications, such as dialing, hanging up, and changing the parameters of a connection. Also known as the Hayes command set, AT means attention.

AXF
ARM Executable Format. An ARM executable image generated by ARM tools.

BIP
Bearer Independent Protocol. Allows an application on a SIM Card to establish a data channel with a terminal, and through the terminal, to a remote server on the network.
CDMA
Code Division Multiple Access. A mobile telephone network standard used primarily in the United States and Canada as an alternative to GSM.

CLDC
Connected Limited Device Configuration. A Java ME platform configuration for devices with limited memory and network connectivity. It uses a low-footprint Java virtual machine such as the CLDC HotSpot Implementation, and several minimalist Java platform APIs for application services.

Configuration
Defines the minimum Java runtime environment (for example, the combination of a Java virtual machine and a core set of Java platform APIs) for a family of Java ME platform devices.

DAC
Digital-to-Analog Converter. A hardware device that converts a stream of binary numbers into an analog signal (time and amplitude), such as audio playback.

ETSI
European Telecommunications Standards Institute. An independent, non-profit group responsible for the standardization of information and communication technologies within Europe. Although based in Europe, it carries worldwide influence in the telecommunications industry.

GCF
Generic Connection Framework. A part of CLDC, it is a Java ME API consisting of a hierarchy of interfaces and classes to create connections (such as HTTP, datagram, or streams) and perform I/O.

GPIO
General Purpose Input/Output. Unassigned pins on an embedded platform that can be assigned or configured as needed by a developer.

GPIO Port
A group of GPIO pins (typically 8 pins) arranged in a group and treated as a single port.

GSM
Global System for Mobile Communications. A 3G mobile telephone network standard used widely in Europe, Asia, and other parts of the world.

HTTP
HyperText Transfer Protocol. The most commonly used Internet protocol, based on TCP/IP that is used to fetch documents and other hypertext objects from remote hosts.

HTTPS

ICCID
Integrated Circuit Card Identification. The unique serial number assigned to an individual SIM Card.
IMP-NG
Information Module Profile Next Generation. A profile for embedded "headless" devices, the IMP-NG specification (JSR 228) is a subset of MIDP 2.0 that leverages many of the APIs of MIDP 2.0, including the latest security and networking+, but does not include graphics and user interface APIs.

IMEI
International Mobile Equipment Identifier. A number unique to every mobile phone. It is used by a GSM or UMTS network to identify valid devices and can be used to stop a stolen or blocked phone from accessing the network. It is usually printed inside the battery compartment of the phone.

IMlet
An application written for IMP-NG. An IMlet does not differ from MIDP 2.0 MIDlet, except by the fact that an IMlet can not refer to MIDP classes that are not part of IMP-NG. An IMlet can only use the APIs defined by the IMP-NG and CLDC specifications.

IMlet Suite
A way of packaging one or more IMlets for easy distribution and use. Similar to a MIDlet suite, but for smaller applications running in an embedded environment.

IMSI
International Mobile Subscriber Identity. A unique number associated with all GSM and UMTS network mobile phone users. It is stored on the SIM Card inside a phone and is used to identify itself to the network.

I2C
Inter-Integrated Circuit. A multi-master, serial computer bus used to attach low-speed peripherals to an embedded platform.

ISA
Instruction Set Architecture. The part of a computer’s architecture related to programming, including data type, addressing modes, interrupt and exception handling, I/O, and memory architecture, and native commands. Reduced instruction set computing (RISC) is one kind of instruction set architecture.

JAD file
Java Application Descriptor file. A file provided in a MIDlet or IMlet suite that contains attributes used by application management software (AMS) to manage the MIDlet or IMlet life cycle, and other application-specific attributes used by the MIDlet or IMlet suite itself.

JAR file
Java Archive file. A platform-independent file format that aggregates many files into one. Multiple applications written in the Java programming language and their required components (class files, images, sounds, and other resource files) can be bundled in a JAR file and provided as part of a MIDlet or IMlet suite.

JCP
Java Community Process. The global standards body guiding the development of the Java programming language.
JDTS
Java Device Test Suite. A set of Java programming language tests developed specifically for the wireless marketplace, providing targeted, standardized testing for CLDC and MIDP on small and handheld devices.

Java ME platform
Java Platform, Micro Edition. A group of specifications and technologies that pertain to running the Java platform on small devices, such as cell phones, pagers, set-top boxes, and embedded devices. More specifically, the Java ME platform consists of a configuration (such as CLDC) and a profile (such as MIDP or IMP-NG) tailored to a specific class of device.

JSR
Java Specification Request. A proposal for developing new Java platform technology, which is reviewed, developed, and finalized into a formal specification by the JCP program.

Java Virtual Machine
A software “execution engine” that safely and compatibly executes the byte codes in Java class files on a microprocessor.

KVM
A Java virtual machine designed to run in a small, limited memory device. The CLDC configuration was initially designed to run in a KVM.

LCDUI
Liquid Crystal Display User Interface. A user interface toolkit for interacting with Liquid Crystal Display (LCD) screens in small devices. More generally, a shorthand way of referring to the MIDP user interface APIs.

MIDlet
An application written for MIDP.

MIDlet suite
A way of packaging one or more MIDlets for easy distribution and use. Each MIDlet suite contains a Java application descriptor file (.jad), which lists the class names and files names for each MIDlet, and a Java Archive file (.jar), which contains the class files and resource files for each MIDlet.

MIDP
Mobile Information Device Profile. A specification for a Java ME platform profile, running on top of a CLDC configuration that provides APIs for application life cycle, user interface, networking, and persistent storage in small devices.

MSISDN
Mobile Station Integrated Services Digital Network. A number uniquely identifying a subscription in a GSM or UMTS mobile network. It is the telephone number to the SIM Card in a mobile phone and used for voice, FAX, SMS, and data services.

MVM
Multiple Virtual Machines. A software mode that can run more than one MIDlet or IMlet at a time.
Obfuscation
A technique used to complicate code by making it harder to understand when it is decompiled. Obfuscation makes it harder to reverse-engineer applications and therefore, steal them.

Optional Package
A set of Java ME platform APIs that provides additional functionality by extending the runtime capabilities of an existing configuration and profile.

Preemption
Taking a resource, such as the foreground, from another application.

Preverification
Due to limited memory and processing power on small devices, the process of verifying Java technology classes is split into two parts. The first part is preverification which is done off-device using the preverify tool. The second part, which is verification, occurs on the device at runtime.

Profile
A set of APIs added to a configuration to support specific uses of an embedded or mobile device. Along with its underlying configuration, a profile defines a complete and self-contained application environment.

Provisioning
A mechanism for providing services, data, or both to an embedded or mobile device over a network.

Pulse Counter
A hardware or software component that counts electronic pulses, or events, on a digital input line, for example, a GPIO pin.

Push Registry
The list of inbound connections, across which entities can push data. Each item in the list contains the URL (protocol, host, and port) for the connection, the entity permitted to push data through the connection, and the application that receives the connection.

RISC
Reduced Instruction Set Computing. A CPU design based on simplified instruction sets that provide higher performance and faster execution of individual instructions. The ARM architecture is based on RISC design principles.

RL-ARM
Real-Time Library. A group of tightly coupled libraries designed to solve the real-time and communication challenges of embedded systems based on ARM processor-based microcontroller devices.

RMI
Remote Method Invocation. A feature of Java SE technology that enables Java technology objects running in one virtual machine to seamlessly invoke objects running in another virtual machine.
RMS
Record Management System. A simple record-oriented database that enables an IMlet or MIDlet to persistently store information and retrieve it later. MIDlets can also use the RMS to share data.

RTOS
Real-Time Operating System. An operating system designed to serve real-time application requests. It uses multi-tasking, an advanced scheduling algorithm, and minimal latency to prioritize and process data.

RTSP
Real Time Streaming Protocol. A network control protocol designed to control streaming media servers and media sessions.

RTX
The real-time operating system used on the Keil MCBSTM32F200 embedded platform. The Oracle Java ME Embedded software runs on the Keil platform.

SCWS
Smart Card Web Server. A web server embedded in a smart card (such as a SIM Card) that allows HTTP transactions with the card.

SD card
Secure Digital cards. A non-volatile memory card format for use in portable devices, such as mobile phones and digital cameras, and embedded systems. SD cards come in three different sizes, with several storage capacities and speeds.

SIM
Subscriber Identity Module. An integrated circuit embedded into a removable SIM card that securely stores the International Mobile Subscriber Identity (IMSI) and the related key used to identify and authenticate subscribers on mobile and embedded devices.

Slave Mode
Describes the relationship between a master and one or more devices in a Serial Peripheral Interface (SPI) bus arrangement. Data transmission in an SPI bus is initiated by the master device and received by one or more slave devices, which cannot initiate data transmissions on their own.

Smart Card
A card that stores and processes information through the electronic circuits embedded in silicon in the substrate of its body. Smart cards carry both processing power and information. A SIM Card is a special kind of smart card for use in a mobile device.

SMS
Short Message Service. A protocol allowing transmission of short text-based messages over a wireless network. SMS messaging is the most widely-used data application in the world.

SMSC
Short Message Service Center. The SMSC routes messages and regulates SMS traffic. When an SMS message is sent, it goes to an SMS center first, then gets forwarded to the destination. If the destination is unavailable (for example, the recipient embedded
board is powered down), the message is stored in the SMSC until the recipient becomes available.

**SOAP**
Simple Object Access Protocol. An XML-based protocol that enables objects of any type to communicate in a distributed environment. It is most commonly used to develop web services.

**SPI**
Serial Peripheral Interface. A synchronous bus commonly used in embedded systems that allows full-duplex communication between a master device and one or more slave devices.

**SSL**
Secure Sockets Layer. A protocol for transmitting data over the Internet using encryption and authentication, including the use of digital certificates and both public and private keys.

**SVM**
Single Virtual Machine. A software mode that can run only one MIDlet or IMlet at a time.

**Task**
At the platform level, each separate application that runs within a single Java virtual machine is called a task. The API used to instantiate each task is a stripped-down version of the Isolate API defined in JSR 121.

**TCP/IP**
Transmission Control Protocol/Internet Protocol. A fundamental Internet protocol that provides for reliable delivery of streams of data from one host to another.

**Terminal Profile**
Device characteristics of a terminal (mobile or embedded device) passed to the SIM Card along with the IMEI at SIM Card initialization. The terminal profile tells the SIM Card what values are supported by the device.

**UART**
Universal Asynchronous Receiver/Transmitter. A piece of computer hardware that translates data between serial and parallel formats. It is used to facilitate communication between different kinds of peripheral devices, input/output streams, and embedded systems, to ensure universal communication between devices.

**UICC**
Universal Integrated Circuit Card. The smart card used in mobile terminals in GSM and UMTS networks. The UICC ensures the integrity and security of personal data on the card.

**UMTS**
Universal Mobile Telecommunications System. A third-generation (3G) mobile communications technology. It utilizes the radio spectrum in a fundamentally different way than GSM.
**URI**
Uniform Resource Identifier. A compact string of characters used to identify or name an abstract or physical resource. A URI can be further classified as a uniform resource locator (URL), a uniform resource name (URN), or both.

**USAT**
Universal SIM Application Toolkit. A software development kit intended for 3G networks. It enables USIM to initiate actions that can be used for various value-added services, such as those required for banking and other privacy related applications.

**USB**
Universal Serial Bus. An industry standard that defines the cables, connectors, and protocols used in a bus for connection, communication, and power supply between computers and electronic devices, such as embedded platforms and mobile phones.

**USIM**
Universal Subscriber Identity Module. An updated version of a SIM designed for use over 3G networks. USIM is able to process small applications securely using better cryptographic authentication and stronger keys. Larger memory on USIM enables the addition of thousands of contact details including subscriber information, contact details, and other custom settings.

**WAE**
Wireless Application Environment. An application framework for small devices, which leverages other technologies, such as Wireless Application Protocol (WAP).

**WAP**
Wireless Application Protocol. A protocol for transmitting data between a server and a client (such as a cell phone or embedded device) over a wireless network. WAP in the wireless world is analogous to HTTP in the World Wide Web.

**Watchdog Timer**
A dedicated piece of hardware or software that "watches" an embedded system for a fault condition by continually polling for a response. If the system goes offline and no response is received, the watchdog timer initiates a reboot procedure or takes other steps to return the system to a running state.

**WCDMA**
Wideband Code Division Multiple Access. A detailed protocol that defines how a mobile phone communicates with the tower, how its signals are modulated, how datagrams are structured, and how system interfaces are specified.

**WMA**
Wireless Messaging API. A set of classes for sending and receiving Short Message Service (SMS) messages.

**XML Schema**
A set of rules to which an XML document must conform to be considered valid.
Index

A
Application Management System (AMS)
  Commands,  1-8
  Examples,  1-10

C
Command Line Interface (CLI),  1-6

L
Logging,  1-6

M
MDK-ARM
  Installation,  1-1
MicroSD Card
  Configuration,  1-3

P
PuTTY
  Installation,  1-2

S
STM32F200
  BAT/3V3 Jumper,  1-5
  Connecting to Computer,  1-2
  Installing Firmware,  1-5
  Windows Drivers,  1-2