Theseus Simulator Installation Guide

Theseus Simulator Installation Guide

EMS-PS-0045

Issue: 2.0
July 2005

Notice

At the time of publication this document reflects the latest information on the Installation of the Theseus Simulator.

Edition 2005

Published by Emosyn, Spinners Court, 53-55 West End, Witney, Oxfordshire, OX28 1NH, U.K. **All Rights Reserved**.

Contents

1. Introduction

1.1 Overview

Theseus Simulator usage is explained in the Keil µVision book “Getting started with µVision2” version 02.2001.

Note This Installation Guide is a complement to the Keil book.

This page is intentionally left blank

2. Installation

2.1 Overview

This section outlines the installation of an Emosyn-compliant compilation and simulation environment.

2.2 System Minimum Requirement

The simulator requires as a minimum a 1GHz Pentium II with 500Mb of RAM and 10Gb of Hard Disc Space.

As simulation is processor-intensive, the recommended system would have a 3GHz or faster processor.

The following operating systems are supported:

- Windows 2000 workstation
- Windows XP
- Windows XP Professional

Note Windows 98 and earlier operating systems **are not** supported.

Keil µVision2 v02.2001 or newer **must** be installed before starting the Theseus Simulator installation.

The SwiftSIM card reader/writer is required for card programming and/or simulation through the ‘paddle board’.

2.3 Support for Emosyn devices

Registered Emosyn users may obtain the ‘Emosyn Device Database’ from the Emosyn.com web site or from your local Field Application Engineer (FAE). The package from the web site includes installation instructions.

2.4 Installing the Simulator

The Theseus®[1] Simulator is supplied as a self expanding executable file. The installation will utilize a directory named 'Emosyn' under the Keil installation.

Some system files may be updated. If the files in this installation are newer than those on the installation machine then it is usually safe to allow the replacement of the files.

If the files in this installation are older than the current installed files, perhaps because other Microsoft updates have been performed on the machine, then the installation files **must not** be used. This will not inhibit the Theseus Simulator installation.

The installation will update the 'tools.ini' file found in the top directory of the Keil installation, usually 'C:\Keil'. This will add the following line to the file:

- TDRV9=C:\Keil\Emosyn\TheseusKeil.dll("Theseus Simulator")

This will allow the Keil environment to find the files and the libraries that are needed to run the Theseus Simulator. See (Section 7) for further information.

The environment variable 'path' has the installation directory added to it. See (Section 7) for further information.

2.5 Installation problems

You must have 'Administrator' rights to install software on a PC.

The database of Emosyn chips must be installed before the Theseus Simulator is run. This can be obtained from the Emosyn.com web site along with installation details or see your local Field Application Engineer (FAE) for support for our newest chips, once the user is registered with Emosyn.

If the Keil environment has not been installed correctly, the target directory for the Theseus Simulator files will not be correct.

Note You can correct this by reinstalling the Keil environment.

Keil is normally installed in "c:\Keil". If another directory is selected which has a long pathname or a pathname which includes spaces, the simulator installation may fail.

In the "tools.ini" file, try replacing components of the path to TheseusKeil.dll with their DOS filename equivalents. E.g. 'Program Files' becomes 'Progra~1'.

If your installed Keil environment already has a line in the tools.ini file which starts 'TDRV9', then use an editor such as Notepad to add in the TDRV9 line above but using a different unused number (e.g. TDRV10).

WARNING Make a backup of the registry before making any of the changes below. See your Windows help files.

Check that the registry entry HKEY_CURRENT_USER\Environment\path has the installed 'Emosyn' directory pre-pended.

[1] Theseus is a registered trademark of Emosyn America Inc.

Also check that the registry entry HKEY_CURRENT_USER \Software\ Emosyn\Simulator\DLLPath is present.

If changes are made, ensure that the trailing backslash is included.

2.6 Modes of Operation

2.6.1 Register Viewing

The display of the Special Function Registers (SFR) and system registers in the “Register Window” is controlled by the encrypted configuration file. The displayed registers are different for each Emosyn device type. Additional registers may be displayed by contacting your Emosyn FAE with a request.

2.6.2 Normal Simulation

In this mode of operation, the PC simulates the execution of the target code and any output is displayed in the serial window. The same serial window is used to supply input characters to the running simulation.

To use the Theseus Simulator under Keil ensure that the appropriate Emosyn chip is selected as shown in Figure 1. Select: ‘Project->Options for Target’.

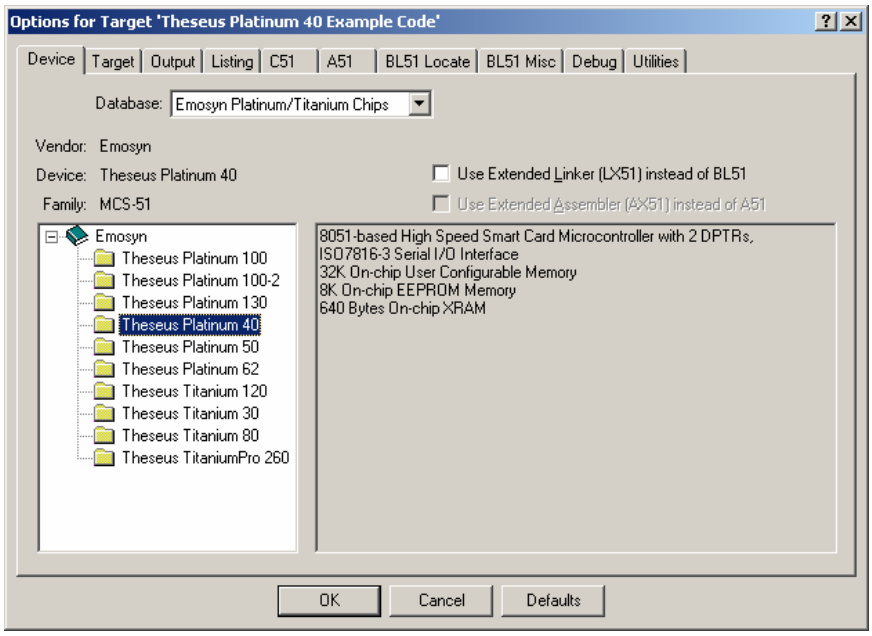


Figure 1: Chip selection

On the Debug tab, select ‘Use’ and use the dropdown list to select the ‘Theseus Simulator’. Select ‘Load Application at Start up’. Select any session values you wish to save and restore.

The dialog shown in Figure 2 will appear.

Note The driver and dialog dlls and the parameters may differ between Theseus family members.

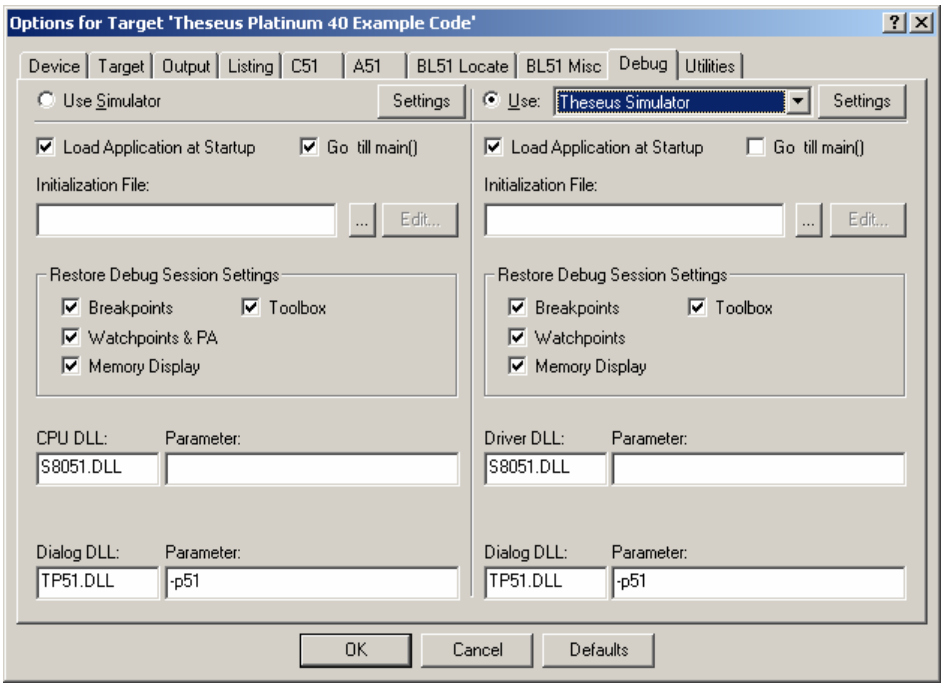


Figure 2: Dialog

2.6.3 Paddle board Simulation

In this mode of operation, the PC simulates the execution of the target code but any output or input is via the paddle board connected to a SwiftSIM®[2].

The paddle board is inserted into any third party smartcard reader and the PC, SwiftSIM and Paddle board behaves as a smartcard.

WARNING The smartcard reader must supply an ISO clock of 3.578MHz to the SwiftSIM and be capable of supplying a current of 6.2mA

To enter the mode, select ‘Settings’ on the dialog shown in Figure 2.

The checkbox ‘Enable SwiftSIM’, the connected COM port and whether you are using SwifSIM1 or SwiftSIM2 are selected, see Figure 3.

[2] SwiftSIM is a registered trademark of Emosyn America Inc.

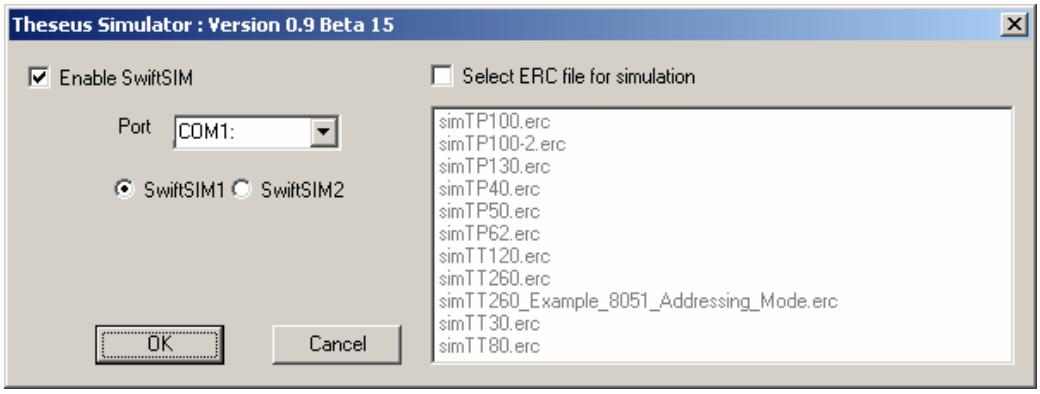


Figure 3: Enable SwiftSIM

2.6.4 Configuration File Selection

The simulator is configured for chip operation by an encrypted file with an '.erc' extension, as shown in Figure 4. The correct file is selected automatically from information in the Keil project settings. However, there are times when a modified configuration file needs to be used. When needed, the 'Select ERC file for simulation' should be checked and then the appropriate file selected.

WARNING **Incorrect selection of an '.erc' file could result in failure to simulate or other problems.**

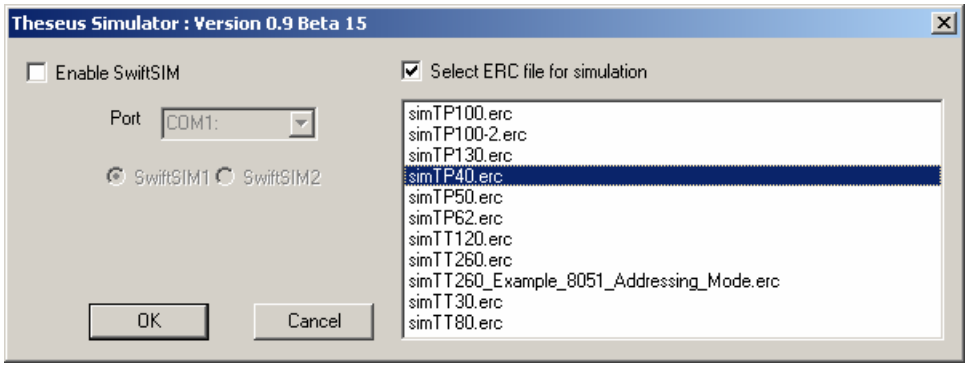


Figure 4: '.erc' extension

To return to the default '.erc' configuration file simply uncheck the 'Select ERC file for simulation' checkbox and then 'OK'

2.6.5 Stand alone simulation

Note Whenever there is a reference to a menu item, the appropriate toolbar icon can be used if preferred.

Obtain the 'Emosyn Generic Code' from the 'Emosyn.com' (registered users) or your local Field Application Engineer (FAE).

Install the code in a suitable location.

Using your file browser, navigate to Emosyn_Generic_Platinum_Code\TP40\Generic_TP40_Example_Code.Uv2 and click on it. With a correctly installed Keil environment this will cause the Keil environment to start up.

Examine the project properties and ensure that the Theseus Platinum 40 chip is selected. Examine the debug tab in the properties and ensure that the Theseus Simulator is selected as described above.

Select 'Project->Rebuild all Target Files'. This will build the project completely. Ensure that no warnings or errors are indicated.

From the Keil menu bar select 'Debug/Start Stop Debug Session'. The simulator is now ready to run.

If 'Serial Window 1' is not visible, select it from the view menu. You should see two lines indicating the configuration file in use and that the debug environment is running.

Use 'Debug->Go (F5)' to start execution.

The ATR generated by the example code appears in the 'Serial 1' window shown in Figure 5.

Note The ATR may be different if another chip example code is selected.

Figure 5: Serial 1

Examine the example code and display the file 'TP40_Send_ATR.A51' and ensure that the correct ATR was sent.

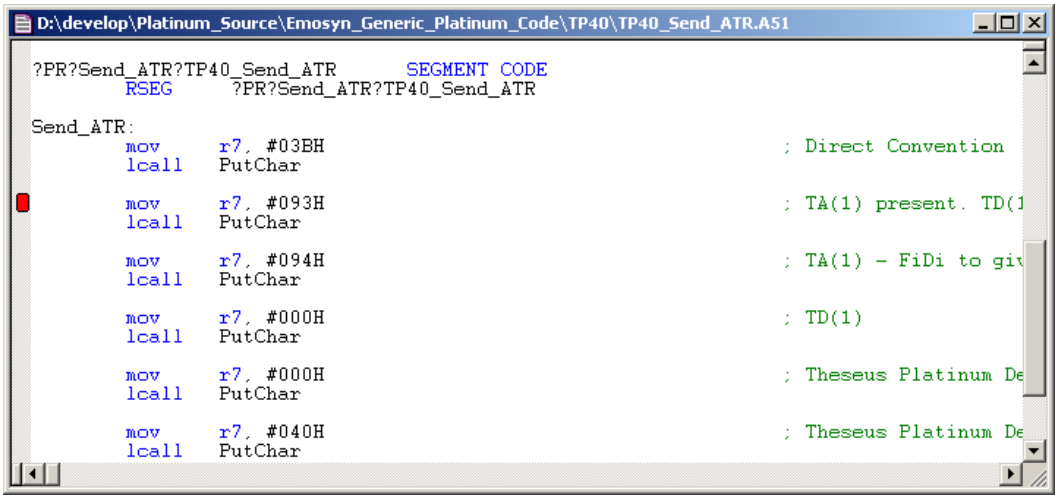


Figure 6: Source Code

Stop the simulation (debug menu) and put a breakpoint in the file shown in Figure 6 at the point where the second ATR byte is about to be transmitted. Refer to your Keil µVision2 manual to do this.

Now run the simulation again. The breakpoint is hit and the simulation stops. The serial window does not show the first byte of the ATR yet as the character has been sent to the simulated serial output register, Serial Buffer (SBUF), but the character has not yet been clocked out.

Using 'Step Over' (F10), step though the code and observe the characters of the ATR being displayed in the serial window.

At this time, registers, Special Function Registers (SFR's) and memory can be viewed and changed. Refer to your Keil µVision documentation.

- Using the 'Go' (F5) command, allow the example code simulation to run
- Now type: 'E5 0A 00 00 08' into the serial window. This is an example APDU which generates the pseudo chip ID response (See the document 'Generic Theseus Family Software Specification – Public Version' available on the web site 'Emosyn.com')
- Stop the execution of the simulated code by the 'Debug->Stop Execution'
- Terminate the debug session with 'Debug->Start/Stop Debug Session'

2.6.6 Paddle board simulation

Select paddle board simulation by selecting the Debug tab of the project properties. Click on 'Settings' and then select the COM port to which the SwiftSIM is connected. Select 'SwiftSIM1 (SwiftSIM2 is not currently available).

From the Keil menu bar select 'Debug/Start Stop Debug Session'. The simulator is now ready to run. **DO NOT USE** Debug-Go (F5).

The simulator is now controlled by the paddle board. When power, clock are supplied and the reset line goes high the simulation is started in the same way it would be on a smart card.

With a smartcard reader and application running, Plug the paddle board into the reader. The simulation will start and an ATR will be sent to the reader. When the reader applies the reset line the simulation will stop.

Code debugging can take place in the same way as the stand alone simulation with breakpoints set and unset and memory and registers viewed and changed.

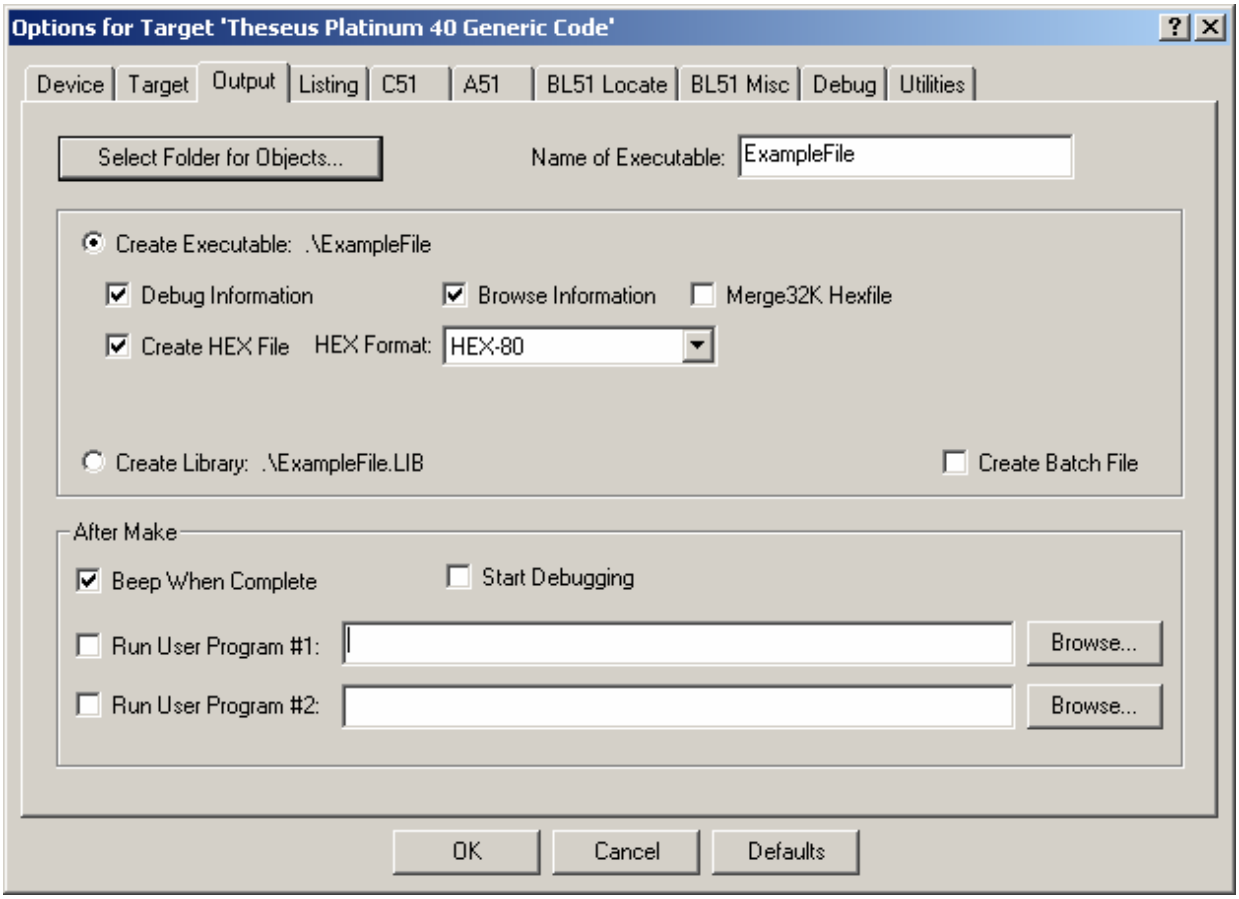
The application connected to the smartcard reader with the paddle board reader can be a scripting application. Emosyn has a scripting environment available on the web site.

This page is intentionally left blank

3. Card Flashing

3.1 Overview

Card programming is achieved through the simulator interface. The ‘hex’ file of the code is converted into an Emosyn proprietary format in preparation for the stand alone tool ‘SwiftSIM’ (supplied) to program the card. You must select “Create HEX File” to get Keil to create a hex file after each compilation.



To enable the use of SwiftSIM under Keil ensure that the Theseus Simulator is selected as shown in Figure 7. Select: 'Project->Options for Target->Utilities' and deselect "Enable SwiftSIM" on the 'debug/settings' tab if only one SwiftSIM is in use.

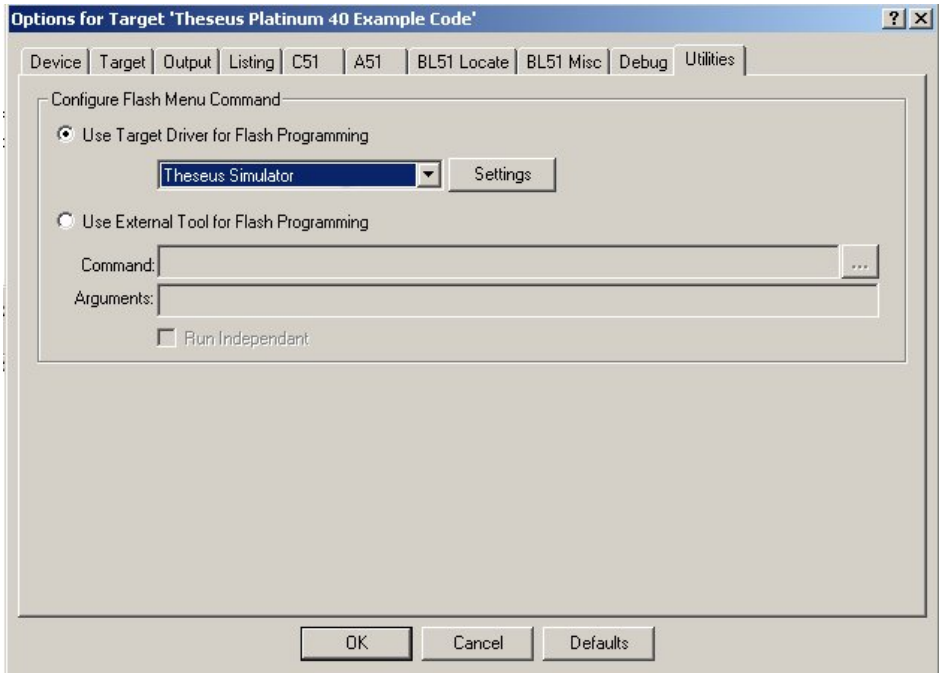


Figure 7: Selection

Use the 'Settings' button to select the SwiftSIM boot loader property, see Figure 8.

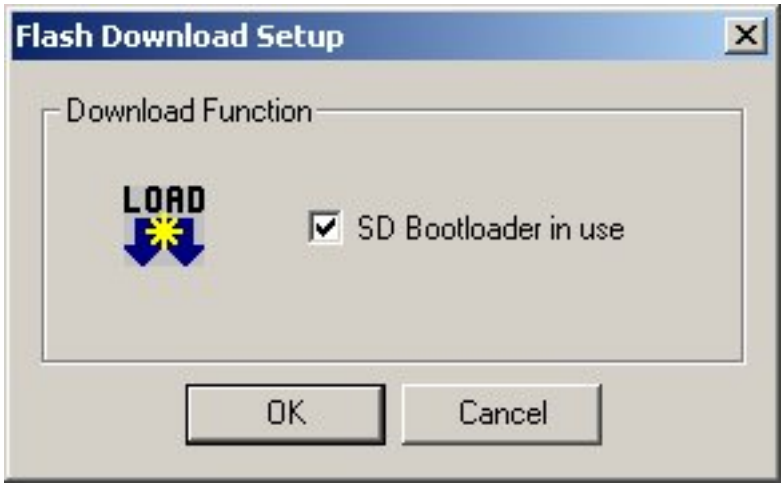
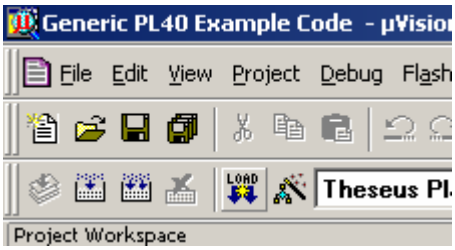


Figure 8: Settings

Theseus Titanium 30 & 80 must be programmed with 'SD Bootloader in use' selected.

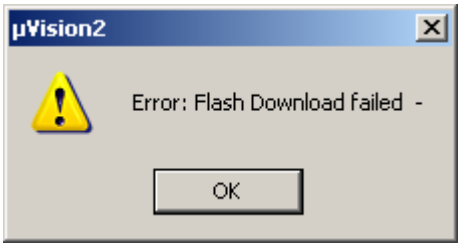
Variants of the Theseus Platinum 40 & 62 chips may or may not have the SD bootloader present. See your FAE for details.

To download to a card, click on the “Load” icon on the main Keil window.



This will parse the hex file and create a GNG. SwiftSIM will then be opened. Click on Program to load the newly created GNG onto the card.

If you get the following error message, then it means that the “Create Hex File” checkbox hasn’t been checked.



Theseus Platinum 100-2 & 130 require the use of the MMU Calculator (supplied) to generate the loader file which can then be programmed into a card with the SwiftSIM stand alone utility.

This page is intention ally left blank

4. Important General Issues

4.1 Overview

Timing / CPU resource issue:

- Due to important processor resource usage, the more powerful the computer can be, the better it is. A 2GHz processor seems to be a minimum system requirement
- In cases where no output is seen on the serial window, please exit from the Keil environment and restart it to try again. This can help
- It is quite likely that your APDU software and reader is going to need some integration with the environment. Given the number of readers on the market and different software this is inevitable. If the communication between your scripting software and the simulator is not performed properly, please contact the Emosyn technical support for advice

This page is intentionally left blank

5. Theseus Simulator Debugging features

5.1 Code modification during simulation

Files are displayed in the project workspace window. You can modify your code whilst you are in a debug session but have to stop it (debug, start-stop debug session) if you want to recompile.

5.2 Code execution

To run your code, you have a reset, run button on the left, step by step button:

5.3 Scripting facilities

Scripts can be executed through the simulator's IO window. The usage of it is described in the "standalone simulation part".

Scripts can be executed from any script software connected to a smart card reader. The IO traffic will go through this smartcard reader and the SwiftSIM and paddle-board hardware connected to it. The usage of it is described in "Simulation" with external scripting software. Please also refer to the (Section 4) concerning the scripting software support.

5.4 Breakpoints

Breakpoints can be set in any part of the code. The simulation needs to be **stopped** before adding or removing any breakpoint. They cannot be set dynamically during simulation.

5.5 Mouse over

When putting the mouse pointer on a variable, it will display the variable value.

5.6 Watch window

Local variables are displayed when you code is stopped in a function.

Global variables:

You can either type the name of the variable or put your mouse over the variable and right click to insert it in a watch window

5.7 Memory window

The simulation needs to be stopped to update a memory window.

In the bottom right or (view -> memory window): in the type-in window, you can write the address you want to see such as:

- **C:**0x8000 : code memory
- **I:**100 : idata, here address 0x64
- **Bx:** 0xA000 : code bank
- **D:** : internal RAM
- **X:** : Xdata RAM

5.8 Symbol window

View-> symbol displays the address of all the variables.

6. Not supported features

No code performance analysis is currently supported by the simulator.

The “trace” mode and code percentage executed is not supported.

8. Document Control

This section details the current document control/version number:

Version	Date	Name	Comment
1.0	22-04-05	Glenn Marsh	Release of Initial Document
2.0	26-07-05	Michael Herman Glenn Marsh	Section 3.1 revised Various edits and comments

7. Known Simulation Issues

See the Release Note.txt in the Keil installation location Emosyn directory.

This page is intentionally left blank

8. Document Control

This section details the current document control/version number:

Version	Date	Name	Comment
1.0	22-04-05	Glenn Marsh	Release of Initial Document
2.0	26-07-05	Michael Herman Glenn Marsh	Section 3.1 revised Various edits and comments