Saturn SOC Support Package

Overview

Filename	:	Saturn 2 SOC Support.Package Overview.1.1f.External.mht
Version	:	1.1f External Issue (Package: Saturn 2 SOC Support Package 1.00.10.0032)
Issue Date	:	15 May 2012
Author	:	IMGWorks

Contents

1. Introduction 4

1.1. Prerequisites 4
1.1.1. cygwin 4
1.1.2. Metagence Toolkit 4
1.1.3. MeOS 4
1.2. Unpacking 4
1.3. readme 4
1.4. Package folder structure 4
1.4.1. \user\include 5
1.4.2. \user\lib\2.7 5
1.4.3. \user\docs 5
1.4.4. \user\utils 5
1.4.5. \reference\apis 5
1.4.6. \reference\apps 5
1.4.7. \reference\bootrom 6
1.4.8. \reference\docs 6
1.4.9. \reference\include 6
1.4.10. \reference\system 6
1.4.11. \reference\tools 6
1.5. API documentation 7
1.6. Reference build scripts 7
1.6.1. IMG_WORKROOT 7
1.6.2. Building the support libraries 7
1.6.3. Building the test applications 9

1. Introduction

This document provides an overview of the Saturn 2 SOC Support Package. This package contains the following:

- Pre-built libraries for the MeOS IP block drivers for META cores. The libraries are pre-built using selected versions of the meta toolkit.
- API header files
- Source and build scripts of the reference drivers and test applications for Saturn peripherals provided by IMGWorks.

1.1. Prerequisites

1.1.1. cygwin

The reference build scripts have been tested from within the cygwin environment.

1.1.2. Metagence Toolkit

The package assumes the presence of the Metagence toolkit release which is typically installed in (the exact location will depend on the version of the toolkit):

```
        c:\img\usr\metag\2.7
```

And the environment variable METAG_INST_ROOT has been set to point to the toolkit.

```
        METAG_INST_ROOT=/img/usr/metag/2.7
```

1.1.3. MeOS

The package assumes that a suitable version of MeOS has been installed along with the Meta Toolkit. The build files uses the METAG_INST_ROOT variable to locate the MeOS header files which are normally located in the folder metag-local\include\meos:

```
        c:\img\usr\metag\2.7\metag-local\include\meos
```

1.2. Unpacking

The Saturn 2 SOC Support Package must be unzipped to a folder on the same drive as the Metagence toolkit(s).

1.3. readme

A “readme” file can be found in:

```
        saturn_apis\readme.html
```

This contains a link to the Release Note and links to other documents.

1.4. Package folder structure

The package is divided into two branches at the top level – the “user” branch and the “reference” branch.

The “user” branch contains all the necessary C header files, pre-built libraries and other related files that are needed to build applications which use the support package features.

The “reference” branch contains the API source files, source files for test applications, built scripts and other related files that the “user” branch files are created from. As the name suggests, this branch is intended to be used purely as reference material - it should not be necessary for any rebuilding within this branch to be performed.

The following is a list of the key folders in the “user” branch of the package:

```
        saturn_apis\user\include
                   \user\lib
                   \user\docs
                   \user\utils
```

1.4.1. \user\include

This folder contains the published header files.

1.4.2. \user\lib\2.7

This folder contains the pre-built MeOS SOC support libraries for META, for DEBUG, TEST and RELEASE builds. These are built with toolkit version 2.7.

The following libraries can be found:

```
        libSaturn_debug_mtp-0gcc.a          - META (thread 0) debug library
        libSaturn_debug_mtp-1gcc.a          - META (thread 1) debug library
        libSaturn_test_mtp-0gcc.a           - META (thread 0) test library
        libSaturn_test_mtp-1gcc.a           - META (thread 1) test library
        libSaturn_mtp-0gcc.a                - META (thread 0) release library
        libSaturn_mtp-1gcc.a                - META (thread 1) release library
```

1.4.3. \user\docs

This folder contains the package documentation.

1.4.4. \user\utils

This folder contains any pre-built utility applications that may be of use to the package user.

The following is a list of the key folders in the “reference” branch of the package:

```
        saturn_apis\reference\apis
                  \reference\apps
                  \reference\bootrom
                  \reference\docs
                  \reference\include
                  \reference\system
```

1.4.5. \reference\apis

This folder contains the source of the reference drivers.

1.4.6. \reference\apps

This folder contains the source of the test applications.

1.4.7. \reference\bootrom

This folder contains the source of the bootrom.

1.4.8. \reference\docs

This folder contains the package documentation.

1.4.9. \reference\include

This folder contains a number of common and platform specific include files (files which define the primitive data types e.g. IMG_UINT32).

1.4.10. \reference\system

This folder contains the SoC system definitions.

1.4.11. \reference\tools

This folder contains non api specific tools

1.5. API documentation

The documentation for the APIs is in HTML format and generated using doxygen. The HTML documentation can be found in:

```
saturn_apis\user\docs\html
```

1.6. Reference build scripts

The package includes example build scripts for building the reference drivers and test applications. These build scripts are for GNU make.

The user\lib folder of the package contains all variants of pre-built libraries for Saturn's META core. Rebuilding the libraries within the source tree is only necessary if the user wishes to build the supplied test applications.

1.6.1. IMG_WORKROOT

The scripts make reference to an environment variable "IMG_WORKROOT" to locate the root folder of the package.

To set create/set the environment variable under XP; select "My Computer", "Properties" – select the "Advanced" tab and click on "Environment Variables" then use "New" or "Edit" to add or change the variable "IMG_WORKROOT". Alternatively, this can be set within the cygwin environment using the SET or export command, or any other appropriate method.

For example, if the package has been unzipped to the folder "c:\mypackage" then the IMG_WORKROOT variable must be set to "c:\mypackage\saturn_apis\reference".

1.6.2. Building the support libraries

Navigate to the following folder:

```
saturn_apis\reference\build\saturn\make
```

Issue the "make" commands from this folder. The following make commands can be issued:

```
“make libclean”        -  prepares to fully rebuild the support libraries.
“make libs”            -  rebuilds the support libraries.
```

Note: the options above are case-sensitive, as illustrated.

Note: the "libclean" target must always be built first as this causes several folders to be created. Simply building the "libs" target without ever having built the "libclean" target will fail.

By default the libraries are built for the META processor thread 0 in "debug" mode – ie no optimisation and debug symbols are generated. The following build options can be used to generate optimised code with or without debug symbols for the META processor thread 0, as below:

For optimised code with debug symbols …

```
“make libclean RELEASE=test”
“make libs RELEASE=test”
```

For fully optimised code without debug symbols …

```
       “make libclean RELEASE=release”
       “make libs RELEASE=release”
```

The optional control “IMG_CPU=meta” can be specified to explicitly select to build for the META processor, as below, but this is not mandatory as this is the default.

```
       “make libclean IMG_CPU=meta”
       “make libs IMG_CPU=meta”

or

       “make libclean IMG_CPU=meta RELEASE=test”
       “make libs IMG_CPU=meta RELEASE=test”

or

       “make libclean IMG_CPU=meta RELEASE=release”
       “make libs IMG_CPU=meta RELEASE=release”
```

The optional control “THREAD_ID=<thread id>” can be used to explicitly select which META thread to build for; in most cases there is no difference in the generated library members but there are a few exceptions that result in it being important to link against the library for the appropriate META thread.

By default thread 0 is selected. To explicitly build for META thread 0 specify “THREAD_ID=0”; to build for thread 1 of the META processor specify “THREAD_ID=1”.

```
       “make libclean IMG_CPU=meta THREAD_ID=0”
       “make libs IMG_CPU=meta THREAD_ID=0”

or

       “make libclean IMG_CPU=meta THREAD_ID=0 RELEASE=test”
       “make libs IMG_CPU=meta THREAD_ID=0 RELEASE=test”

or

       “make libclean IMG_CPU=meta THREAD_ID=0 RELEASE=release”
       “make libs IMG_CPU=meta THREAD_ID=0 RELEASE=release”

and

       “make libclean IMG_CPU=meta THREAD_ID=1”
       “make libs IMG_CPU=meta THREAD_ID=1”

or

       “make libclean IMG_CPU=meta THREAD_ID=1 RELEASE=test”
       “make libs IMG_CPU=meta THREAD_ID=1 RELEASE=test”

or

       “make libclean IMG_CPU=meta THREAD_ID=1 RELEASE=release”
       “make libs IMG_CPU=meta THREAD_ID=1 RELEASE=release”
```

1.6.3. Building the test applications

Having first built the libraries (see above) navigate to the test application folder under \apps. The available test application folders are shown below. It should be noted, though, that many of these tests require external hardware and/or software which is not provided as part of this package; these applications are only provided as useful examples in the use of the APIs for the associated blocks. Those test applications that can be run on Saturn without specialized external hardware/software are in bold type below.:

```
saturn_apis\reference\apps\saturn\bring-up\gpio
saturn_apis\reference\apps\saturn\bring-up\scbm
saturn_apis\reference\apps\saturn\bring-up\spis
saturn_apis\reference\apps\saturn\bring-up\sys_stress_test\meta
saturn_apis\reference\apps\saturn\bring-up\uart_bootapp
saturn_apis\reference\apps\saturn\bring-up\usb_isoc
saturn_apis\reference\apps\saturn\bring-up\usbd
```

Issue the "make" commands from this folder. The following make commands can be issued:

```
"make clean"           -  prepares to fully rebuild the test application.
"make"                 -  rebuilds the test application.
```

By default the test applications are built in "debug" mode – ie no optimisation and debug symbols are generated. Two other build options can be used to generate optimised code with or without debug symbols, as below:

For optimised code with debug symbols …

```
"make clean RELEASE=test"
"make RELEASE=test"
```

For fully optimised code without debug symbols …

```
"make clean RELEASE=release"
"make RELEASE=release"
```

By default the applications are built for the META core thread 0. To build libraries and applications for thread 1 use "THREAD_ID=1" below …

```
"make clean THREAD_ID=1"
"make THREAD_ID=1"
```

Note: the options above are case-sensitive, as illustrated.

Note: the libraries only need to be built once – there is no need to rebuild them for each individual test application.