Integration Manual --

Table of Contents

[1 Dependencies 2](#_Toc358640599)

[1.1 SWCs 2](#_Toc358640600)

[1.2 Functions to be provided to Integration Project 2](#_Toc358640601)

[1.3 Functions to be provided by Integration Project 2](#_Toc358640602)

[2 Configuration 2](#_Toc358640603)

[2.1 Build Time Config 2](#_Toc358640604)

[2.2 Configuration Files to be provided by Integration Project 2](#_Toc358640605)

[2.2.1 startup\_cfg.h 3](#_Toc358640606)

[2.2.2 appinit\_cfg.h 3](#_Toc358640607)

[2.3 DaVinci Config Configuration Changes 3](#_Toc358640608)

[2.4 Manual Configuration Changes 3](#_Toc358640609)

[3 Integration 3](#_Toc358640610)

[3.1 Required Global Data Inputs 3](#_Toc358640611)

[3.2 Optional Global Data Inputs 3](#_Toc358640612)

[3.3 Specific Include Path present 3](#_Toc358640613)

[3.4 Build Exclusions 3](#_Toc358640614)

[3.5 Definition of Stacks 4](#_Toc358640615)

[3.6 Reset Causes 4](#_Toc358640616)

[3.7 Impact on Integration Project 6](#_Toc358640617)

[3.7.1 JTAG Debugger Considerations 6](#_Toc358640618)

[3.7.2 RAM Memory State 7](#_Toc358640619)

[3.7.3 ECC and Parity 7](#_Toc358640620)

[4 Runnable Scheduling 7](#_Toc358640621)

[5 Memory Mapping 7](#_Toc358640622)

[5.1 .resetcause Section 7](#_Toc358640623)

[5.2 Mapping 7](#_Toc358640624)

[5.3 Usage 7](#_Toc358640625)

[5.4 NvM Blocks 8](#_Toc358640626)

[6 Compiler Settings 8](#_Toc358640627)

[6.1 Preprocessor MACRO 8](#_Toc358640628)

[6.2 Optimization Settings 8](#_Toc358640629)

[6.3 Other Settings 8](#_Toc358640630)

[7 Revision Control Log 8](#_Toc358640631)

# Dependencies

## SWCs

|  |  |
| --- | --- |
| Module | Required Feature |
| StdDef | TMS570 Register Definitions |

## Functions to be provided to Integration Project

void \_fiqhandler(void);

uint32 \_coreGetDebugStatusAndControlRegister\_(void);

uint32 \_coreGetSecondaryAuxiliaryControlRegister\_(void);

void \_coreSetSecondaryAuxiliaryControlRegister\_(uint32 SecAuxCtrlRegVal\_Cnt\_T\_u32);

uint32 \_coreGetFPSCR\_(void);

## Functions to be provided by Integration Project

All common functionality required for a boot project startup is contained in “BootStartup.c”, and all common functionality for application project startup is contained in “AppStartup.c”. There may be instances, however, where a boot or application project will need to do special steps that are specific to a particular program, prior to the “main()” function call. To accommodate this, several functions are called, one at the start and one at the end of both BootStartup.c and AppStartup.c. These functions are

* void BootStartupCallout1(void)
* void BootStartupCallout2(void)
* void AppStartupCallout1(void)
* void AppStartupCallout2(void)

The integration projects are required to provide these functions, regardless of if there is content in them.

# Configuration

## Build Time Config

|  |  |  |
| --- | --- | --- |
| Modules | Notes | SWC |
| None |  |  |

## Configuration Files to be provided by Integration Project

Configuration files are needed to configure the startup sequence based on the needs of the program being implemented in. These configuration files are simply header files that define a set of build constants. Templates of these files are provided (located in the tools folder of this SWC) that can be adapted to the needs of the program. From the context of a boot project, startup\_cfg.h is required. From the context of an application project, appinit\_cfg.h is required.

### startup\_cfg.h

This file contains the startup configuration constants used in sys\_startup. The following constants can be enabled or disabled defining them as either STD\_OFF or STD\_O, descriptions are as follows:

### appinit\_cfg.h

## DaVinci Config Configuration Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | | Value | Notes |
| OsOSFIQHandler | \_fiqhandler | | Needs to be configured if more than one FIQ is configured in the system (e.g. when adding the floating point exception handling FIQ). When only one FIQ is configured, the value of this parameter should be the name of that one FIQ ISR. |

## Manual Configuration Changes

|  |  |  |
| --- | --- | --- |
| Constant | Notes | SWC |
| None |  |  |

# Integration

## Required Global Data Inputs

None

## Optional Global Data Inputs

None

## Specific Include Path present

Yes

## Build Exclusions

This component is designed to be included in a given program’s boot and application projects. The boot project is defined as the project which the hardware reset vector jumps to. A subset of this component’s files needs to be part of the build based on given context. The source files to be excluded from the build through code composer project settings are as follows:

Excluded from Boot Project:

* AppStartup.c

Excluded from the Application Project:

* BootStartup.c
* sys\_startup.c

## Definition of Stacks

The stacks (sizes and locations) need to be defined and reserved by both the boot and application integration projects. This can be done by defining symbols for the locations of the stack (typically in the linker file). The stack symbols (used to initialize the stack pointers) the startup code requires are:

* “\_StackSVC\_”
* “\_StackFIQ\_”
* “\_StackIRQ\_”
* “\_StackUSER\_”
* “\_StackABORT\_”
* “\_StackUND\_”

## Reset Causes

A variable of this SWC holds the reset cause and it is used to alter the processing of the startup sequence. This SWC defines a set of reset causes shown in the table below. The integration project is free to define its own reset cause values if necessary and write this value to the ResetCause\_Cnt\_Enum variable prior to performing a software reset; however, the names and values in the table below are reserved for use of this SWC. The ResetCause\_Cnt\_Enum variable can be read at any time by integration project components for information or diagnostic use.

|  |  |
| --- | --- |
| Reset Cause Name | Value |
| PWRONRESET | 0x0000FFFF |
| DEBUGRESET | 0x0001FFFE |
| CPURESET | 0x0002FFFD |
| SPPBISTFAILED | 0x0003FFFC |
| DPPBISTFAILED | 0x0004FFFB |
| EXTRESET | 0x0005FFFA |
| OSCFAIL | 0x0006FFF9 |
| SWRESET | 0x0007FFF8 |
| WDGFAIL | 0x0008FFF7 |
| CCMSTEFFAILED | 0x0009FFF6 |
| CCMSTFAILED | 0x000AFFF5 |
| CCMEFFAILED | 0x000BFFF4 |
| PBISTSCFAILED | 0x000CFFF3 |
| STCSCFAILED | 0x000DFFF2 |
| STCFAILED | 0x000EFFF1 |
| ESM3NONZERO | 0x000FFFF0 |
| EFCSTFAILED | 0x0010FFEF |
| EFCSTUCKZERO | 0x0011FFEE |
| EFCERROR | 0x0012FFED |
| FLSBUS2CORRFAILED | 0x0013FFEC |
| FLSBUS2ADDCAPFAILED | 0x0014FFEB |
| FLSBUS2MULBITDETFAILED | 0x0015FFEA |
| FLSBUS2SNGBITDETFAILED | 0x0016FFE9 |
| VIMPARFLGFAILED | 0x0017FFE8 |
| VIMPARADDERRFAILED | 0x0018FFE7 |
| VIMPARESMFAILED | 0x0019FFE6 |
| DCAN1PARESMFAILED | 0x001AFFE5 |
| DCAN2PARESMFAILED | 0x001BFFE4 |
| DCAN3PARESMFAILED | 0x001CFFE3 |
| DMAPARESMFAILED | 0x001DFFE2 |
| MIBADC1PARESMFAILED | 0x001EFFE1 |
| MIBADC2PARESMFAILED | 0x001FFFE0 |
| MIBSPI1PARESMFAILED | 0x0020FFDF |
| MIBSPI3PARESMFAILED | 0x0021FFDE |
| MIBSPI5PARESMFAILED | 0x0022FFDD |
| N2HET1PARESMFAILED | 0x0023FFDC |
| N2HET1TUPARESMFAILED | 0x0024FFDB |
| N2HET2PARESMFAILED | 0x0025FFDA |
| N2HET2TUPARESMFAILED | 0x0026FFD9 |
| B0MULBITRAMECCDETFAILED | 0x0027FFD8 |
| B1MULBITRAMECCDETFAILED | 0x0028FFD7 |
| B0SNGBITRAMECCDETFAILED | 0x0029FFD6 |
| B1SNGBITRAMECCDETFAILED | 0x002AFFD5 |
| SNGBITFLSECCDETFAILED | 0x002BFFD4 |
| MULBITFLSECCDETFAILED | 0x002CFFD3 |
| LPOTRIMERROR | 0x002DFFD2 |
| DATAMULBITRAMECCFAILED | 0x002EFFD1 |
| DATAMULBITFLSECCFAILED | 0x002FFFD0 |
| CPUDATAABORT | 0x0030FFCF |
| CPUPREFETCHABORT | 0x0031FFCE |
| PRFTCMULBITRAMECCFAILED | 0x0032FFCD |
| PRFTCMULBITFLSECCFAILED | 0x0033FFCC |
| UNDEFINST | 0x0034FFCB |
| CLOCKMONITOR | 0x0035FFCA |
| CCMFAILED | 0x0036FFC9 |
| FMCUNCORRERR | 0x0037FFC8 |
| B0UNCORRERR | 0x0038FFC7 |
| B1UNCORRERR | 0x0039FFC6 |
| B0ADDPARERR | 0x003AFFC5 |
| B1ADDPARERR | 0x003BFFC4 |
| FLSECCLIVELOCK | 0x003CFFC3 |
| VIMMULTBITFLT | 0x003DFFC2 |
| VIMPARTHRSHFLT | 0x003EFFC1 |
| UNUSEDINTERRUPT | 0x003FFFC0 |
| STACKOVERWRITE | 0x0040FFBF |
| MPUVIOLATION | 0x0041FFBE |
| WDGALIVEMONFAIL | 0x0042FFBD |
| WDGDEADLINEFAIL | 0x0043FFBC |
| WDGPROGFLOWFAIL | 0x0044FFBB |
| SWWDGFAIL | 0x0045FFBA |
| FPUDZCEXCP | 0x0046FFB9 |
| FPUOFCEXCP | 0x0047FFB8 |
| FPUIOCEXCP | 0x0048FFB7 |
| FPUUNKNOWNEXCP | 0x0049FFB6 |

## Impact on Integration Project

### JTAG Debugger Considerations

Integrating this project will impact the debugging capabilities. The normal startup which runs all of the diagnostic startup tests will be bypassed if a debugger connection is detected. Because of safety ramifications of potentially bypassing startup tests if the controller incorrectly reads a debugger being attached, a branch to self instruction is inserted in AppStartup.c if the reset cause is “DEBUGRESET”. In the scenario where a debugger is actually attached, the user running the debug session will have to manually move the program-counter past this branch instruction to continue debugging.

### RAM Memory State

This SWC will leave the RAM cleared to all zeros only on a Power-On Reset. All other resets will bypass any RAM manipulation. It is the responsibility of the boot project and application project to make certain the initial state of RAM is proper for all reset scenarios.

### ECC and Parity

The sys\_startup function will exit in a state with both RAM and Flash ECC turned on. If this is not the desired state for either the boot or application code, the callout functions could potentially be used to change this (e.g. the boot callout could turn off flash ECC, and the application callout could turn flash ECC back on).

The AppStartup can be configured to test parity on the applicable peripheral RAM. After the test, the parity will be left in the enabled state; however, the integration application code will need to configure response to failures (ISRs, etc).

# Runnable Scheduling

This section specifies the required runnable scheduling.

|  |  |  |
| --- | --- | --- |
| Init | Scheduling Requirements | Trigger |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Runnable | Scheduling Requirements | Trigger |
|  |  |  |

# Memory Mapping

## .resetcause Section

A “.resetcause” memory section must be defined in both the boot and application linker file. This holds a 32bit variable to be located in RAM memory. This variable is shared between the boot and the application projects, and therefore must be located in a shared, fixed memory location.

## Mapping

|  |  |  |
| --- | --- | --- |
| Memory Section | Contents | Notes |
|  |  |  |
|  |  |  |
|  |  |  |

\* Each …START\_SEC… constant is terminated by a …STOP\_SEC… constant as specified in the AUTOSAR Memory Mapping requirements.

## Usage

|  |  |  |
| --- | --- | --- |
| Feature | RAM | ROM |
| Full driver |  |  |

Table 1: ARM Cortex R4 Memory Usage

## NvM Blocks

|  |
| --- |
| Block Name |
|  |

# Compiler Settings

## Preprocessor MACRO

<Define all the preprocessor Macros needed and conditions when needed>.

## Optimization Settings

<Define Optimization levels that are needed and conditions when needed>.

## Other Settings

# Revision Control Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev #** | **Change Description** | **Date** | **Author Initials** |
| 1 | Initial version |  | LWW |
| 2 | Updated to latest Integration Manual Template; Updated reset cause table (section 3.5); Removed sys\_core.asm and sys\_memory.asm from list of modules needing special compilation (section 6.3) (handled by a directive in the asm files); added information on configuring \_fiqhandler | 6/10/2013 | KMC |
| 3 | Removed requirement of compilation of sys\_startup and AppStartup in arm mode | 08/02/13 | LWW |