# High-Level Description

sys\_core provides assembly language functions for processor register data access and system startup.

# Figures

## Diagram – Function Data Sharing

No Shared Data

# Variable Data Dictionary

For details on module input / output variable, refer to the Data Dictionary for the application. Input / output variable names are listed here for reference.

(Note: Full variable names required in table.)

(Note: All global variables including End Of Line data used should be shown here)

|  |  |  |
| --- | --- | --- |
| Module Inputs | Module Outputs | |
| <None> | | <None> |
|  | |  |

## Module Internal Variables

This section identifies the name, range and resolutions for module specific data created by this module. If there are no range restrictions on the variable, the term “FULL” is placed into the table for legal range.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable Name | Resolution | Legal Range  (min) | Legal Range  (max) | Software Segment |
| <None> |  |  |  |  |
|  |  |  |  |  |

### User defined typedef definition/declaration

This section documents any user types uniquely used for the module.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Typedef Name | Element Name | User Defined Type | Legal Range  (min) | Legal Range  (max) |
| <None> |  |  |  |  |
|  |  |  |  |  |

# Constant Data Dictionary

## Calibration Constants

This section lists the calibrations used by the module. For details on calibration constants, refer to the Data Dictionary for the application.

|  |
| --- |
| Constant Name |
| <None> |
|  |

## Program(fixed) Constants

### Embedded Constants

All embedded constants whose values are provided in Eng units will be evaluated to the equivalent counts by using the FPM\_InitFixedPoint\_m() macro within the #define statement.

#### Local

|  |  |  |  |
| --- | --- | --- | --- |
| Constant Name | Resolution | Units | Value |
| <None> |  |  |  |

#### Global

This section lists the global constants used by the module. For details on global constants, refer to the Data Dictionary for the application.

|  |
| --- |
| Constant Name |
| <None> |

### Module specific Lookup Tables Constants

(This is for lookup tables (arrays) with fixed values, same name as other tables)

|  |  |  |  |
| --- | --- | --- | --- |
| Constant Name | Resolution | Value | Software Segment |
| <None> |  |  |  |

# Functions/Macros used by the Sub-Modules

## Library Functions / Macros

The library and functions / Macros that are called by the various sub modules are identified below,

1. <None>

## Data Hiding Functions

1. <None>

## Global Functions/Macros Defined by this Module

NOTE that all global functions in this module must be assembled in ARM mode. Therefore the .asm source file includes the .arm directive at the beginning of the file, applying the directive to all functions in the file.

### Global Function #1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_coreEnableVfp\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | None |  |  |  |  |  |

#### Description

Enables VFP co-processor unit.

.def \_coreEnableVfp\_

.asmfunc

\_coreEnableVfp\_

mrc p15, #0x00, r0, c1, c0, #0x02

orr r0, r0, #0xF00000

mcr p15, #0x00, r0, c1, c0, #0x02

mov r0, #0x40000000

fmxr fpexc, r0

bx lr

.endasmfunc

### Global Function #2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_coreInitRegisters\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | None |  |  |  |  |  |

#### Description

Initialize CPU Registers, including banked registers for all modes. This is required to be done at startup to ensure that both cores of the controller have identical starting register values to prevent a false trip of the core compare module. Included are all core registers as well as all floating point registers. Additionally, this function initializes the stack pointers for all modes of the uController. Therefore, this function needs to get executed prior to any stack usage and prior to any read usage of any core or floating point registers. The stack pointers are referenced by externally defined symbols (typically defined in the linker file) to allow for user configuration of the individual stack sizes.

.ref \_StackUSER\_

.ref \_StackFIQ\_

.ref \_StackUND\_

.ref \_StackIRQ\_

.ref \_StackABORT\_

.ref \_StackSVC\_

.def \_coreInitRegisters\_

.asmfunc

\_coreInitRegisters\_

; After reset, the CPU is in the Supervisor mode (M = 10011)

mov r0, lr

mov r1, #0x0000

mov r2, #0x0000

mov r3, #0x0000

mov r4, #0x0000

mov r5, #0x0000

mov r6, #0x0000

mov r7, #0x0000

mov r8, #0x0000

mov r9, #0x0000

mov r10, #0x0000

mov r11, #0x0000

mov r12, #0x0000

ldr sp, svcSp

; Switch to FIQ mode (M = 10001)

cps #17

mov r8, #0x0000

mov r9, #0x0000

mov r10, #0x0000

mov r11, #0x0000

mov r12, #0x0000

; Abort mode

cps #23

ldr sp, abortSp

mov lr, r0

; Undefined instruction mode

cps #27

ldr sp, undefSp

mov lr, r0

; FIQ mode

cps #17

ldr sp, fiqSp

mov lr, r0

; IRQ mode

cps #18

ldr sp, irqSp

mov lr, r0

; System mode

cps #31

ldr sp, userSp ; SYS mode shares stack with User mode

mov lr, r0

; Switch back to Supervisor Mode (M = 10011)

cps #19

fmdrr d0, r1, r1

fmdrr d1, r1, r1

fmdrr d2, r1, r1

fmdrr d3, r1, r1

fmdrr d4, r1, r1

fmdrr d5, r1, r1

fmdrr d6, r1, r1

fmdrr d7, r1, r1

fmdrr d8, r1, r1

fmdrr d9, r1, r1

fmdrr d10, r1, r1

fmdrr d11, r1, r1

fmdrr d12, r1, r1

fmdrr d13, r1, r1

fmdrr d14, r1, r1

fmdrr d15, r1, r1

bl next1

next1

bl next2

next2

bl next3

next3

bl next4

next4

bx r0

userSp .word \_StackUSER\_

svcSp .word \_StackSVC\_

fiqSp .word \_StackFIQ\_

irqSp .word \_StackIRQ\_

abortSp .word \_StackABORT\_

undefSp .word \_StackUND\_

.endasmfunc

### Global Function #3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_esmCcmErrorsClear\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | None |  |  |  |  |  |

#### Description

Clear ESM CCM errors. This function is required for ERRATA DEVICE#140 workaround found in gladiator silicon revision A. It will clear all CCM error flags in the ESM status registers, clear the ESMKEY register to reset the ESM error pin state, clear the ESMH VIM interrupt request flag, and clear the core compare status register compare error flag.

.def \_esmCcmErrorsClear\_

.asmfunc

\_esmCcmErrorsClear\_:

ldr r0, ESMSR1\_REG ; load the ESMSR1 status register address

ldr r2, ESMSR1\_ERR\_CLR

str r2, [r0] ; clear the ESMSR1 register

ldr r0, ESMSR2\_REG ; load the ESMSR2 status register address

ldr r2, ESMSR2\_ERR\_CLR

str r2, [r0] ; clear the ESMSR2 register

ldr r0, ESMSSR2\_REG ; load the ESMSSR2 status register address

ldr r2, ESMSSR2\_ERR\_CLR

str r2, [r0] ; clear the ESMSSR2 register

ldr r0, ESMKEY\_REG ; load the ESMKEY register address

mov r2, #0x5 ; load R2 with 0x5

str r2, [r0] ; clear the ESMKEY register

ldr r0, VIM\_INTREQ ; load the INTREQ register address

ldr r2, VIM\_INT\_CLR

str r2, [r0] ; clear the INTREQ register

ldr r0, CCMR4\_STAT\_REG ; load the CCMR4 status register address

ldr r2, CCMR4\_ERR\_CLR

str r2, [r0] ; clear the CCMR4 status register

bx lr

ESMSR1\_REG .word 0xFFFFF518

ESMSR2\_REG .word 0xFFFFF51C

ESMKEY\_REG .word 0xFFFFF538

ESMSSR2\_REG .word 0xFFFFF53C

CCMR4\_STAT\_REG .word 0xFFFFF600

CCMR4\_ERR\_CLR .word 0x00010000

ESMSR1\_ERR\_CLR .word 0x80000000

ESMSR2\_ERR\_CLR .word 0x00000004

ESMSSR2\_ERR\_CLR .word 0x00000004

VIM\_INT\_CLR .word 0x00000001

VIM\_INTREQ .word 0xFFFFFE20

.endasmfunc

### Global Function #4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_coreEnableRamEcc\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | None |  |  |  |  |  |

#### Description

Enable RAM ECC Support.

.def \_coreEnableRamEcc\_

.asmfunc

\_coreEnableRamEcc\_

mrc p15, #0x00, r0, c1, c0, #0x01

orr r0, r0, #0x0C000000

dmb

mcr p15, #0x00, r0, c1, c0, #0x01

isb

bx lr

.endasmfunc

### Global Function #5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_coreDisableRamEcc\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | None |  |  |  |  |  |

#### Description

Disable RAM ECC Support.

.def \_coreDisableRamEcc\_

.asmfunc

\_coreDisableRamEcc\_

mrc p15, #0x00, r0, c1, c0, #0x01

bic r0, r0, #0x0C000000

dmb

mcr p15, #0x00, r0, c1, c0, #0x01

isb

bx lr

.endasmfunc

### Global Function #6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_coreGetDebugStatusAndControlRegister\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | Value of DBGDSCR register | uint32 |  | FULL | FULL |  |

#### Description

Get contents of debug status and control register.

.def \_coreGetDebugStatusAndControlRegister\_

.asmfunc

\_coreGetDebugStatusAndControlRegister\_:

mrc p14, #0x00, r0, c0, c1, #0x00

bx lr

.endasmfunc

### Global Function #7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_coreGetSecondaryAuxiliaryControlRegister\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | Value of Secondary Auxiliary Control Register | uint32 |  | FULL | FULL |  |

#### Description

Get contents of Secondary Auxiliary Control Register.

.def \_coreGetSecondaryAuxiliaryControlRegister\_

.asmfunc

\_coreGetSecondaryAuxiliaryControlRegister\_:

mrc p15, #0, r0, c15, c0, #0

bx lr

.endasmfunc

### Global Function #8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_coreSetSecondaryAuxiliaryControlRegister\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | Value to be stored in Secondary Auxiliary Control Register | uint32 |  | FULL | FULL |  |
| **Return Value** | None |  |  |  |  |  |

#### Description

Set contents of Secondary Auxiliary Control Register.

.def \_coreSetSecondaryAuxiliaryControlRegister\_

.asmfunc

\_coreSetSecondaryAuxiliaryControlRegister\_:

mcr p15, #0, r0, c15, c0, #0

bx lr

.endasmfunc

### Global Function #9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | \_coreGetFPSCR\_ | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | Value of FPSCR register | uint32 |  | FULL | FULL |  |

#### Description

Get contents of Floating Point Status and Control Register.

.def \_coreGetFPSCR\_

.asmfunc

\_coreGetFPSCR\_:

fmrx r0, FPSCR

bx lr

.endasmfunc

## Local Functions/Macros Used by this MDD only

None

# Software Module Implementation

## Runtime Environment (RTE) Initial Values

This section lists the initial values of data written by this module but controlled by the RTE. After RTE initialization, the data in this table will contain these values.

|  |  |
| --- | --- |
| Data | Value |
| <None> |  |

## Initialization Functions

None

## Periodic Functions

None

## Fault Recovery Functions

None

## Shutdown Functions

None

## Interrupt Functions

None

## Serial Communication Functions

None

# Execution Requirements

## Execution Sequence of the Module

(Describe in words relevant details about the execution sequence of the different sub modules.)

## Execution Rates for sub-modules called by the Scheduler

This table serves as reference for the Scheduler design

|  |  |  |
| --- | --- | --- |
| Function Name | Calling Frequency | System State(s) in which the function is called |
| <None> |  |  |

## Execution Requirements for Serial Communication Functions

|  |  |
| --- | --- |
| Function Name | Sub-Module called by (Serial Comm Function Name) |
| <None> |  |

# Memory Map Definition Requirements

## Sub Modules (Functions)

This table identifies the software segments for functions identified in this module.

|  |  |
| --- | --- |
| Name of Sub Module | Software Segment |
| N/A |  |

## Local Functions

This table identifies the software segments for local functions identified in this module.

|  |  |
| --- | --- |
| Name of Local Function | Software Segment |
| None |  |
|  |  |

# Known Issues / Limitations With Design

None

# Revision Control Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Rev #** | **Change Description** | **Date** | **Author Initials** |
| 1 | 1.0 | Initial Revision of MDD | 6/6/2013 | KMC |
| 2 | 2.0 | Added disable RAM ECC function | 10/03/13 | LWW |