# High-Level Description

Cd\_uDiagFPU provides diagnostic data on floating point exceptions. Based on “Unreleased draft of EA3.x FDD 32.5 uC Diagnostics Execution (0x02A-0x031) v002.docx” as saved on May 7, 2013 and “Unreleased Draft of EA3.x FDD 32.Appendices v002.docx” as saved on May 6, 2013.

On initialization, it sets up the Secondary Auxiliary Control Register to generate a floating point interrupt (VIM47) on the occurrence of any of these floating point exception flags: IOC (invalid operation), OFC (overflow), or DZC (divide by zero), and enables the floating point interrupt. When the floating point interrupt occurs, the module’s ISR saves program address information (to indicate where the exception occurred) and a reset reason indicating the type of exception, and causes a reset.

Cd\_uDiagFPU functionality is enabled/disabled by D\_ENABLEFPUDIAG\_CNT\_LGC. See Integration Manual for the configuration parameter that controls this #define.

# 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 | |
| ResetCause\_Cnt\_Enum | | ResetCause\_Cnt\_Enum |
|  | |  |

## 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 |
| FPUExceptionAddr\_Cnt\_D\_u32 | NA | FULL | FULL | See Integration Manual |
|  |  |  |  |  |

### 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 |
| D\_FPSCRBITIOC\_CNT\_U32 | N/A | N/A | 0x00000001 |
| D\_FPSCRBITDZC\_CNT\_U32 | N/A | N/A | 0x00000002 |
| D\_FPSCRBITOFC\_CNT\_U32 | N/A | N/A | 0x00000004 |
| D\_VFPEXCEPTIONOUTPUTEN\_CNT\_U32 | N/A | N/A | 0x00001600 |

#### 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 |
| FPUDZCEXCP |
| FPUOFCEXCP |
| FPUIOCEXCP |
| FPUUNKNOWNEXCP |
| D\_ENABLEFPUDIAG\_CNT\_LGC |

### 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. RednRpdShtdn()

## Data Hiding Functions

1. \_coreGetSecondaryAuxiliaryControlRegister\_()
2. \_coreSetSecondaryAuxiliaryControlRegister\_()
3. \_coreGetFPSCR\_()
4. EnableVFPInterrupt()
5. \_uDiagGetLinkRegForFiqIsr\_()

## Global Functions/Macros Defined by this Module

### Global Function #1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | uDiagFPU\_Init1 | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | N/A |  |  |  |  |  |

#### Design Rationale

When the reset cause was a floating point exception, need to ensure that shutdown can be reached without another floating point exception occurring . Therefore the floating point exceptions should only be enabled when the reset cause was NOT a floating point exception.

When built with D\_ENABLEFPUDIAG\_CNT\_LGC == STD\_ON, the function operates as shown in 5.3.1.2. When built with D\_ENABLEFPUDIAG\_CNT\_LGC == STD\_OFF, the function returns without doing anything.

#### Description



### Global Function #2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function Name** | uDiagFPU\_Init2 | Type | Dir. | Min | Max | UTP Tol. |
| **Arguments Passed** | None |  |  |  |  |  |
| **Return Value** | N/A |  |  |  |  |  |

#### Design Rationale

When the reset cause was a floating point exception, need to ensure that shutdown can be reached without another floating point exception occurring . Therefore the floating point interrupt should only be enabled when the reset cause was NOT a floating point exception.

When built with D\_ENABLEFPUDIAG\_CNT\_LGC == STD\_ON, the function operates as shown in 5.3.2.2. When built with D\_ENABLEFPUDIAG\_CNT\_LGC == STD\_OFF, the function returns without doing anything.

#### Description



## 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

### Isr: Mcu\_FpuIrq()

#### Design Rationale

The floating point interrupt is set up as an FIQ so that it is the highest priority interrupt in the system (see section 9.2).

When a floating point interrupt occurs, need to ensure that shutdown is reached even if floating point processing is not operational. Therefore, this ISR calls the Redundant Rapid Shutdown function with a reset reason indicating which type of floating point exception occurred. The shutdown function resets the processor, so control does not return from this floating point interrupt ISR. After the reset, the reset handler sets a DTC shutdown fault with parameter bit(s) set to indicate the type of floating point exception.

Only three of six possible floating point exceptions are enabled. Also, if this interrupt is not the highest priority interrupt in the system, it is possible that the Floating Point Status and Control Register will no longer indicate an exception on entry to this ISR (see section 9.2). Therefore, in addition to reset reasons for the three expected floating point exception types, an “Unknown FP Exception” type will be indicated if none of the three expected types is seen in the status register.

When built with D\_ENABLEFPUDIAG\_CNT\_LGC == STD\_OFF, this function will never be invoked because the interrupt will not be enabled in the uDiagFPU\_Init2() function.

#### Processing



## Serial Communication Functions

None

# Execution Requirements

See Integration Manual.

# 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 |
| uDiagFPU\_Init1 | N/A |
| uDiagFPU\_Init2 | N/A |
| Mcu\_FpuIrq | 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

Saved program address where exception occurred is actually the contents of the link register; it will usually give the address of the instruction at or shortly after where the exception occurred. However, under some circumstances the address may not indicate where the exception occurred.

## 

If a floating point exception occurs in an ISR that has higher priority than the floating point exception ISR, entry to the floating point ISR will not occur until after the return from the ISR where the actual exception occurred. Because the processor saves/restores the Link Register and the Floating Point Status and Control register to/from the stack on interrupt, these registers will no longer contain the relevant information about the floating point exception after the return from the higher priority ISR, so the information will not be available to the floating point exception ISR. In this case, the reset reason will indicate Unknown Floating Point Exception, and the saved program address information will not be related to the occurrence of the exception.

NOTE that D\_ENABLEFPUDIAG\_CNT\_LGC is controlled by a configuration parameter and will be defined in the private section of uDiag\_Cfg.h. For the D\_ENABLEFPUDIAG\_CNT\_LGC symbol to be properly defined on compilation of uDiagFPU.c, the uDiagFPU.c file must include these lines:

#define UDIAG\_C /\* to include module private section of header \*/

#include "uDiag.h"

and uDiag.h must include uDiag\_Cfg.h.

If the D\_ENABLEFPUDIAG\_CNT\_LGC symbol is not defined at compilation of this code, e.g. if the above includes are not properly set up, the floating point exception handling WILL be present and functional regardless of the configuration parameter setting..

# Revision Control Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev #** | **Change Description** | **Date** | **Author Initials** |
| 1.0 | Initial Revision | 6/6/2013 | KMC |
| 2.0 | Added D\_ENABLEFPUDIAG\_CNT\_LGC to enable/disable floating point exception diagnostic functionality. | 6/26/13 | KMC |