# Module --

# High-Level Description

This is module implements the FLASH Memory testing requirements specified in “[EA3.x FDD 32 - uC Diagnostics 000D](http://misagweb01.nexteer.com/eRoomReq/Files/erooms8/NextGeneration/0_14dbbf/EA3.x%20FDD%2032%20%20-%20uC%20Diagnostics%20000D.docx)” according to the AUTOSAR “[Specification of Flash Test v1.2.0](http://www.autosar.org/download/R4.0/AUTOSAR_SWS_FlashTest.pdf) ” API definition. This SWC is TI TMS570 target specific.

This module extends the AUTOSAR API definition by including a hardware Self Test function and including support for hardware IRQ’s.

# Figures

## Diagram – Function Data Sharing

This diagram shows all data that is shared between functions within the module.

(Note – If no data is shared between functions, the Text “No Shared Data” can be used in place of a graphic. Also note that init functions need not be shown unless they compute non-zero data to be used by other functions in the module).

### Diagram – Function (Name)

This diagram describes the functional characteristics and data flow of a given function.

(Note – This is not mandatory, only used where a graphical representation helps explain the function. It is left to the author’s discretion. New headers of this level (Level 3) should be created for each function.

# 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 | |
| <VarName\_Units\_Type> | | <VarName\_Units\_Type> |
|  | |  |

## 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) |
| (Name given for the user defined typdef of type struct/union)  (Variable name qualified similar to all other variables) | (Variable name qualified similar to all other variables) | as other variables |  |  |
|  | (Variable name qualified similar to all other variables) |  |  |  |

# 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

### Global Function #1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | (Exact name used) | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | (if none, write None) |  |  |  |  |
|  | (Insert more rows for additional passed arguments) |  |  |  |  |
| **Return Value** | (if no value returned, write N/A) |  |  |  |  |

#### Description

(Place flowchart/design for local function)

## Local Functions/Macros Used by this MDD only

### Local Function #1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function Name** | (Exact name used) | Type | Min | Max | UTP Tol. |
| **Arguments Passed** | (if none, write None) |  |  |  |  |
|  | (Insert more rows for additional passed arguments) |  |  |  |  |
| **Return Value** | (if no value returned, write N/A) |  |  |  |  |

#### Description

(Place flowchart/design for local function)

# 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

(Note: For multiple init functions, insert new headers at the “Header 2” level – subset of “5.1 Initialization Functions” and follow the same sub-section design shown below)

### Init: \_L\_Init(n)

#### Design Rationale

(Add design rationale specifically related to this FUNCTION. If none required, place the text “None”)

#### Module Outputs

(Initialize all module outputs in this section)

#### Module Internal

(Initialize all module internal variables in this section)

## Periodic Functions

(Note: For multiple periodic functions, insert new headers at the “Header 2” level – subset of “5.2 Periodic Functions” and follow the same sub-section design shown below)

### Per: \_L\_Per(n)

#### Design Rationale

(Add design rationale specifically related to this FUNCTION. If none required, place the text “None”)

#### Program Flow Start

(If program flow is required by the module, the function to store the unique identifier to a temporary variable is done here – start of the function)

#### Store Module Inputs to Local copies

(If not required based on design, insert text “None”)

#### (Processing of function)………

(Breakdown the function into smaller sections to add clarity to the design).

#### Store Local copy of outputs into Module Outputs

(If not required based on design, insert text “None”)

#### Program Flow End

(If program flow is required by the module, the function to add the temporary variable to the global accumulator is done here)

## Fault Recovery Functions

(Note: For multiple functions, insert new headers at the “Header 2” level – subset of “5.3 Fault Recovery Functions” and follow the same sub-section design shown below)

### FaultRec: \_L\_FaultRec(n)

#### Design Rationale

(Add design rationale specifically related to this FUNCTION. If none required, place the text “None”)

#### Program Flow Start

(If program flow is required by the module, the function to store the unique identifier to a temporary variable is done here – start of the function)

#### Store Module Inputs to Local copies

(If not required based on design, insert text “None”)

#### (Processing of function)………

(Breakdown the function into smaller sections to add clarity to the design).

#### Store Local copy of outputs into Module Outputs

(If not required based on design, insert text “None”)

#### Program Flow End

(If program flow is required by the module, the function to add the temporary variable to the global accumulator is done here)

## Shutdown Functions

(Note: For multiple functions, insert new headers at the “Header 2” level – subset of “5.4 Shutdown Functions” and follow the same sub-section design shown below)

### Shtdn: \_L\_Shtdn(n)

#### Design Rationale

(Add design rationale specifically related to this FUNCTION. If none required, place the text “None”)

#### Program Flow Start

(If program flow is required by the module, the function to store the unique identifier to a temporary variable is done here – start of the function)

#### Store Module Inputs to Local copies

(If not required based on design, insert text “None”)

#### (Processing of function)………

(Breakdown the function into smaller sections to add clarity to the design).

#### Store Local copy of outputs into Module Outputs

(If not required based on design, insert text “None”)

#### Program Flow End

(If program flow is required by the module, the function to add the temporary variable to the global accumulator is done here)

## Interrupt Functions

(Note: For multiple functions, insert new headers at the “Header 2” level – subset of “5.5 Interrupt Functions” and follow the same sub-section design shown below)

### Isr: \_L\_Isr(n)

#### Design Rationale

(Add design rationale specifically related to this FUNCTION. If none required, place the text “None”)

#### (Processing of the ISR function)…..

(Note: Multiple headings can be used to break apart the functionality)

## Serial Communication Functions

(Note: For multiple functions, insert new headers at the “Header 2” level – subset of “5.6 Serial Communication Functions” and follow the same sub-section design shown below)

### SComm: \_L\_SComm(n)

#### Design Rationale

(Add design rationale specifically related to this FUNCTION. If none required, place the text “None”)

#### Program Flow Start

(If program flow is required by the module, the function to store the unique identifier to a temporary variable is done here – start of the function)

#### Store Module Inputs to Local copies

(If not required based on design, insert text “None”)

#### (Processing of function)………

(Breakdown the function into smaller sections to add clarity to the design).

#### Store Local copy of outputs into Module Outputs

(If not required based on design, insert text “None”)

#### Program Flow End

(If program flow is required by the module, the function to add the temporary variable to the global accumulator is done here)

# 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 |
|  |  |

## Local Functions

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

|  |  |
| --- | --- |
| Name of Sub Module | Software Segment |
|  |  |

# Known Issues / Limitations With Design

1. (Item #1)

# Revision Control Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Rev #** | **Change Description** | **Date** | **Author Initials** |
| 1 | 1.0 | Initial creation | 7/4/2012 | JJW |