**Module Design Document**

**For**

**DMA**

**Document Identifier: <Project\_id>\_<Config Id>**

**VERSION: 4**

**DATE: 31-Jan-2015**

**Location:** The official version of this document is stored in the Nexteer Configuration Management System and is uniquely identified by: <Project\_ID>\_<Config Id>

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No.** | **Description** | **Author** | **Version** | **Date** |
| 1 | Initial version | Owen Tosh | 1 | 04-Apr-2014 |
| 2 | Updated to FDD52 v001 | Owen Tosh | 2 | 29-Apr-2014 |
| 3 | Updated to FDD52 v002 | Owen Tosh | 3 | 02-May-2014 |
| 4 | Updated to ES-52 v004 | Kathleen Creager | 4 | 31-Jan-2015 |

**Table of Contents**

[1 Abbrevations And Acronyms 5](#_Toc384116577)

[2 References 6](#_Toc384116578)

[3 DMA & High-Level Description 7](#_Toc384116579)

[4 Design details of software module 8](#_Toc384116580)

[4.1 Graphical representation of DMA 8](#_Toc384116581)

[5 Variable Data Dictionary 9](#_Toc384116582)

[5.1 User defined typedef definition/declaration 9](#_Toc384116583)

[5.2 Variable definition for enumerated types 9](#_Toc384116584)

[6 Constant Data Dictionary 10](#_Toc384116585)

[6.1 Program(fixed) Constants 10](#_Toc384116586)

[6.1.1 Embedded Constants 10](#_Toc384116587)

[6.1.1.1 Local 10](#_Toc384116588)

[6.1.1.2 Global 10](#_Toc384116589)

[6.1.2 Module specific Lookup Tables Constants 10](#_Toc384116590)

[6.1.3 Library Functions / Macros 10](#_Toc384116591)

[6.1.4 Data Hiding Functions 11](#_Toc384116592)

[7 Software Module Implementation 12](#_Toc384116593)

[7.1 Initialization Functions 12](#_Toc384116594)

[7.1.1 Init: Dma \_Init 12](#_Toc384116595)

[7.1.1.1 Design Rationale 12](#_Toc384116596)

[7.1.1.1.1 MPU Settings 12](#_Toc384116597)

[7.1.1.1.2 Priority Assignments 12](#_Toc384116598)

[7.1.1.1.3 FlsTst Group (Channels 0 and 1) 12](#_Toc384116599)

[7.1.1.2 Initialize DMA Registers 12](#_Toc384116600)

[7.1.1.3 Module Outputs 12](#_Toc384116601)

[7.1.1.4 Module Internal 13](#_Toc384116602)

[7.2 PERIODIC FUNCTIONS 13](#_Toc384116603)

[7.2.1 Per: Dma \_Per1 13](#_Toc384116604)

[7.2.1.1 Design Rationale 13](#_Toc384116605)

[7.2.1.2 Store Module Inputs to Local copies 13](#_Toc384116606)

[7.2.1.3 Clear DMA RAM Buffers 13](#_Toc384116607)

[7.2.1.4 Store Local copy of outputs into Module Outputs 13](#_Toc384116608)

[7.3 Interrupt Functions 13](#_Toc384116609)

[7.4 TRANSIENT FUNCTIONS 13](#_Toc384116610)

[7.5 Serial Communication Functions 13](#_Toc384116611)

[7.6 Local Function/Macro Definitions 13](#_Toc384116612)

[7.7 GLObAL Function/Macro Definitions 13](#_Toc384116613)

[7.7.1 Setup MtrCtrl Groups 13](#_Toc384116614)

[7.7.1.1 Description 13](#_Toc384116615)

[7.7.2 Setup FlsTst Blocks 14](#_Toc384116616)

[7.7.2.1 Description 14](#_Toc384116617)

[7.7.3 Enable FlsTst Block 14](#_Toc384116618)

[7.7.3.1 Description 14](#_Toc384116619)

[7.7.4 Disable FlsTst Block 14](#_Toc384116620)

[7.7.4.1 Description 14](#_Toc384116621)

[7.7.5 Report DMA MPU Error 14](#_Toc384116622)

[7.7.5.1 Description 14](#_Toc384116623)

[8 Known Limitations With Design 15](#_Toc384116624)

[9 UNIT TEST CONSIDERATION 16](#_Toc384116625)

[10 Appendix A – Configuration Schemes 17](#_Toc384116626)

# Abbrevations And Acronyms

|  |  |
| --- | --- |
| Abbreviation | Description |
| MDD | Module design Document |
| MtrCtrl ISR | Motor Control Interrupt Service Routine. This is the “fast” code loop that controls the main PWM signals. |

# References

This section Lists the title & version of all the documents that are referred for development of this document

|  |  |  |
| --- | --- | --- |
| Sr. No. | Title | Version |
| 1 | MDD Guidelines | 1 |
| 2 | Software Naming Conventions | 1 |
| 3 | Coding Standands | 1 |
| 4 | ES 52 – DMA | 004 |

# DMA & High-Level Description

DMA is a driver level module that performs flash, RAM, and peripheral reads and writes in the background, freeing up the CPU to do other work in parallel. It is equipped with parity checking and an MPU, but timing and data consistency must be considered with respect to CPU execution.

# Design details of software module

## Graphical representation of DMA



# Variable Data Dictionary

## User defined typedef definition/declaration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Typedef Name | Element Name | User Defined Type | Legal Range  (min) | Legal Range  (max) |
| DMADataType\_Str \* | DummyVarForAlignment\_Uls\_f64\*\* | float64 | FULL | FULL |
| FastSPI\_Cnt\_u16[D\_NUMFASTSPIWORDS\_CNT\_U16] | uint16 | 0 | 216 - 1 |
| SlowSPI\_Cnt\_u16[D\_NUMSLOWSPIWORDS\_CNT\_U16] | uint16 | 0 | 216 - 1 |
| SlowADC\_Cnt\_u16[D\_NUMFASTADCCHANNELS\_CNT\_U16] | uint16 | 0 | 216 - 1 |
| FastADC\_Cnt\_u16[D\_NUMSLOWADCCHANNELS\_CNT\_U16] | uint16 | 0 | 216 - 1 |
| PWMCmp\_Cnt\_u16[4][2] | uint16 | 0 | 216 - 1 |
| PWMPeriod\_Cnt\_u32 | uint32 | 0 | 232 - 1 |

\* Note that the elements of this structure are dependent on whether the respective peripherals are configured to use DMA. In the case that none of the ADC, SPI, or PWM groups are enabled in DMA, the type will not be defined.

\*\* Note that the boundaries of this struct type must be 64-bit aligned due to TMS570 DMA MPU operation, which in some cases allows access outside of configured MPU region boundaries up to a 64-bit boundary if the region boundaries are not 64-bit aligned. A float64 is used to force the alignment; per section 6.2.1.6 of SPNU151G–August 2011 (ARM Optimizing C/C++ Compiler v 4.9 User’s Guide) “Structures are aligned according to the member with the most restrictive alignment requirement. Structures are padded so that the size of the structure is a multiple of its alignment.”

## Variable definition for enumerated types

|  |  |  |
| --- | --- | --- |
| Enum Name | Element Name | Value |
| None |  |  |

# Constant Data Dictionary

## Program(fixed) Constants

## Embedded Constants

## Local

|  |  |  |  |
| --- | --- | --- | --- |
| Constant Name | Resolution | Units | Value |
| D\_DMAFLSTSTENABLED\_CNT\_ENUM | enum | Count | configurable |
| D\_FASTSPIGROUPENABLED\_CNT\_ENUM | enum | Count | configurable |
| D\_FASTADCGROUPENABLED\_CNT\_ENUM | enum | Count | configurable |
| D\_FASTPWMGROUPENABLED\_CNT\_ENUM | enum | Count | configurable |
| D\_SLOWADCGROUPENABLED\_CNT\_ENUM | enum | Count | configurable |
| D\_CRCCTRLREGSTART\_CNT\_U32 | 1 | Count | &(CRCCTRLREG->CRC\_REGL1) |
| D\_CRCPSASIGREGSTART\_CNT\_U32 | 1 | Count | &(CRCCTRLREG->PSA\_SIGREGL1) |
| D\_NUMFASTSPIWORDS\_CNT\_U16 | 1 | Count | D\_TGSIZE\_CNT\_U16 |
| D\_FASTSPISTARTADDR\_CNT\_U32 | 1 | Count | &(mibspiRAM3->rx[0].data) |
| D\_NUMFASTADCCHANNELS\_CNT\_U16 | 1 | Count | D\_ADC2G1BUFSZ\_CNT\_U08 |
| D\_FASTADCSTARTADDR\_CNT\_U32 | 1 | Count | (D\_ADC2RSLTBASEADR\_CNT\_U32 + (4u \* D\_ADC2EVTBUFSZ\_CNT\_U08) + 2u) |
| D\_DMANHETPERIODADDR\_CNT\_U32 | 1 | Count | &(HET\_PRD\_BUF1\_0.memory.data\_word) |
| D\_EPWMSTARTADDR\_CNT\_U32 | 1 | Count | &(ePWM1->CMPA) |
| D\_NUMSLOWADCCHANNELS\_CNT\_U16 | 1 | Count | D\_ADC1G2BUFSZ\_CNT\_U08 |
| D\_SLOWADCSTARTADDR\_CNT\_U32 | 1 | Count | (0xFF3E0000ul + (4u \* (D\_ADC1EVTBUFSZ\_CNT\_U08 + D\_ADC1G1BUFSZ\_CNT\_U08)) + 2u) |
| D\_NUMSLOWSPIWORDS\_CNT\_U16 | 1 | Count | D\_TGSIZE\_CNT\_U16 |
| D\_SLOWSPISTARTADDR\_CNT\_U32 | 1 | Count | &(mibspiRAM5->rx[0].data) |

## Global

|  |
| --- |
| Constant Name |
| STD\_OFF |
| STD\_ON |
| CRCCTRLREG |
| D\_TGSIZE\_CNT\_U16 |
| mibspiRAM3 |
| D\_ADC2G1BUFSZ\_CNT\_U08 |
| D\_ADC2RSLTBASEADR\_CNT\_U32 |
| D\_ADC2EVTBUFSZ\_CNT\_U08 |
| HET\_PRD\_BUF1\_0 |
| ePWM1 |
| D\_ADC1G2BUFSZ\_CNT\_U08 |
| D\_ADC1EVTBUFSZ\_CNT\_U08 |
| D\_ADC1G1BUFSZ\_CNT\_U08 |
| mibspiRAM5 |
| DMA\_PARITY\_ENABLE |

## Module specific Lookup Tables Constants

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

## Library Functions / Macros

The library and functions / Macros that are called by the various sub modules shall be identified in this section

None

## Data Hiding Functions

The Data hiding functions that’s uses RTE interface or other macro interfaces shall be listed in this section.

DMA\_REPORTERRORSTATUS(event, param, status)

# Software Module Implementation

## Initialization Functions

## Init: Dma\_Init

## Design Rationale

The DMA module must encapsulate many possible configurations. These configurations can be enabled or disabled per the configuration header file. All programs use DMA for the Flash CRC Test (part of the TMS570 uC diagnostics); some programs configure additional uses.

The DMA peripheral is reset by a system reset and can also be reset by a control register bit. The latest version of TMS\_570 Startup includes a test of DMA MPU functionality; at the conclusion of this test it sets the DMA peripheral reset bit. Per discussion with TI and draft updates to TI documentation, the reset bit should be checked to ensure the peripheral is out of reset before configuring registers.

This init function checks the reset bit; whether the peripheral was last reset by a system reset (when used with older versions of TMS\_570 Startup) or by the reset at the end of the startup test, the peripheral is expected to be out of reset by the time this init function executes. If the reset bit indicates the peripheral is not out of reset, a global variable flag is used to indicate a DMA reset failure. This flag is used in the TMS570\_uDiag component which controls the Peripheral Startup Fault NTC (0x037).

DMA is enabled at the end of the function. Once this is done, DMA will remain active for the remainder of the ignition cycle. Individual channels can be enabled or disabled as needed, however.

## MPU Settings

The nature of the DMA MPU is different from that of the CPU. The DMA has four configurable memory regions; anything outside of the declared regions is considered full access. The regions may overlap, however, and are treated by “priority” – that is, for addresses included in multiple regions, the settings in the lowest-numbered region are used. In light of this, the final memory region is used to cover the majority of the memory map. The other three regions are configured to allow access to specific regions. The following table shows the memory region allocations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Region** | **Access** | **Purpose** | **Start Address** | **End Address** |
| 0 | Read/Write | SPI, ADC, NHET | 0xFF0A0000 | 0xFF473FFF |
| 1 | Read/Write | ePWM, CRC | 0xFCF78C00 | 0xFE0001FF |
| 2 | Read/Write | RAM | &DMAData\_G\_str | &DMAData\_G\_str + sizeof(DMAData\_G\_str) - 1 |
| 3 | No Access | Everything but Flash | 0x00200000 | 0xFFFFFFFF |

This configuration is based on safety analysis (included in FDD 52) and the DMA MPU limitations.

Per discussion with TI and draft TI documentation updates, in some cases (depending on element size being transferred), the DMA MPU may allow access beyond the end of a DMA MPU region to the next 64-bit boundary if the region boundary is not 64-bit aligned (start address is 64-bit aligned, end address is the last byte of a 64-bit aligned space, i.e. end address is a 64-bit aligned address minus one). For this reason, a dummy 64-bit variable is included at the beginning of the DMADataType\_Str type.

When DMA is used for Flash CRC Test only, the DMA MPU settings are more restrictive, providing only the access needed by the Flash CRC test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Region** | **Access** | **Purpose** | **Start Address** | **End Address** |
| 0 | Write only | CRC | Address of first CRC register needed | Address of last byte of last CRC register needed |
| 1 | No access | Peripheral RAM and registers | 0xF0800000UL | 0xFFFFFFFFUL |
| 2 | Read only | Entire address space | 0x00000000UL | 0xFFFFFFFFUL |
| 3 | Region not enabled | N/A | N/A | N/A |

## Priority Assignments

Channels 0 and 1 are assigned low priority, while the other channels are assigned as high priority. As channels 0 and 1 are used for FlsTst functionality, they are designed to be run “in the background”, while the other channels need high priority to ensure the MtrCtrl ISR is run on schedule. Any unused channel is left at a default of low priority.

## Initialize DMA Registers

<flow chart>

## Module Outputs

None

## Module Internal

None

## PERIODIC FUNCTIONS

None

## Interrupt Functions

None

## TRANSIENT FUNCTIONS

None

## Serial Communication Functions

None

## Local Function/Macro Definitions

None

## GLObAL Function/Macro Definitions

## Check Validity of Slow ADC Group

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | Dma\_SlowADCGroupValidity | Type | Min | Max |
| **Arguments Passed** | None |  |  |  |
| **Return Value** | RetValue\_Cnt\_T\_lgc | boolean | FALSE | TRUE |

## Design Rationale

This function is provided as one part of the data integrity scheme. The FDD requires that the DMA buffer be checked before the data is read, and cleared after the data is read. This function is designed to be called before the ADC data is read, and returns FALSE if the data is invalid. The FDD specifies that the data be copied if it is valid; for throughput and memory reasons, this is left as the responsibility of the caller.

## Description

<flow chart>

## Invalidate Slow ADC Group Buffers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | Dma\_InvalidateSlowADCGroup | Type | Min | Max |
| **Arguments Passed** | None |  |  |  |
| **Return Value** | N/A |  |  |  |

## Design Rationale

To ensure that the DMA is running, this function is provided to be run after the DMA data is collected by the 2ms IoHwAbstraction component. The buffers are cleared to 0xFFFF with the expectation that the DMA will fill them with proper data before the data is collected again. If the DMA fails to do so, the cleared buffers will trigger the internal DMA diagnostic (implemented in Dma\_SlowADCGroupValidity).

This is intended to be used as part of a larger data consistency scheme. See Appendix A for more information on how this is designed to be used.

## Description

<flow chart>

## Setup MtrCtrl Groups

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | Dma\_SetupMtrCtrlGroups | Type | Min | Max |
| **Arguments Passed** | None |  |  |  |
| **Return Value** | N/A |  |  |  |

## Description

<flow chart>

## Setup FlsTst Blocks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | Dma\_SetupFlsTstBlock | Type | Min | Max |
| **Arguments Passed** | CRCAddr\_Cnt\_T\_u32 | uint32 | 0 | 232 - 1 |
|  | FlsAddr\_Cnt\_T\_u32 | uint32 | 0 | 232 - 1 |
|  | DmaFrameCount\_Cnt\_T\_u16 | uint16 | 0 | 216 - 1 |
|  | DmaElementCount\_Cnt\_T\_u16 | uint16 | 0 | 216 - 1 |
| **Return Value** | N/A |  |  |  |

## Description

<flow chart>

## Enable FlsTst Block

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | Dma\_EnableFlsTstBlock | Type | Min | Max |
| **Arguments Passed** | None |  |  |  |
| **Return Value** | N/A |  |  |  |

## Description

<flow chart>

## Disable FlsTst Block

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function Name** | Dma\_DisableFlsTstBlock | Type | Min | Max |
| **Arguments Passed** | None |  |  |  |
| **Return Value** | N/A |  |  |  |

## Description

<flow chart>

# Known Limitations With Design

1. This design only considers DMA transfers for a specific sensor configuration. Older or newer programs may have other DMA needs that are not considered, and the design will need to be updated to accommodate those sensor configurations in time.
2. This design has only been verified on Champion hardware. It would need to be validated on Gladiator hardware before use.

# UNIT TEST CONSIDERATION

None

# Appendix A – Configuration Schemes

The DMA module is intended to be configurable for any number of use cases. While each channel is specifically assigned, each channel can be enabled or disabled per program.

The following diagram shows the MtrCtrl ISR groups and where they are designed to run. The trigger points for each group, as well as non-configurable data formats and lengths, are based on this data and the corresponding peripherals.

