# Manual

for MPC5634M MCU Driver

Document Number: UM14MCUASR3.0R2.0.0
Rev. 1.1

freescale™

# Contents

**Section Number** **Title** **Page**

## Chapter 1
## Revision History

## Chapter 2
## Introduction



## Chapter 3
## Driver

## Chapter 4
## Tresos Configuration Plug-in

# Chapter 1
# Revision History

**Table 1-1. Revision History**

| Revision | Date | Author | Description |
|---|---|---|---|
| 1.0 | 01 - Feb - 2011 | Rosario Anchini | First released version |
| 1.1 | 12 - Dec - 2011 | Rosario Anchini | Added chapters Runtime errors, FMPLL parameters configuration and updated the chapter driver design summary |

# Chapter 2
# Introduction

This User Manual describes Freescale Semiconductor AUTOSAR Micro Controller Unit ( MCU ) for MPC5634M .

AUTOSAR MCU driver configuration parameters and deviations from the specification are described in MCU Driver chapter of this document. AUTOSAR MCU driver requirements and APIs are described in the AUTOSAR MCU driver software specification document.

## 2.1 Supported Derivatives

The software described in this document is intented to be used with the following microcontroller devices of Freescale Semiconductor .

**Table 2-1. MPC5634M Derivatives**

| | |
|---|---|
| Freescale Semiconductor | mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176 |

All of the above microcontroller devices are collectively named as MPC5634M .

## 2.2 Overview

**AUTOSAR (AUTomotive Open System ARchitecture)** is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.

- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

## 2.3 About this Manual

This Technical Reference employs the following typographical conventions:

**Boldface** type: Bold is used for important terms, notes and warnings.

*Italic* font: Italic typeface is used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

**Note**

This is a note.

## 2.4 Acronyms and Definitions

**Table 2-2. Acronyms and Definitions**

| Term | Definition |
|---|---|
| API | Application Programming Interface |
| AUTOSAR | Automotive Open System Architecture |
| ASM | Assembler |
| BSMI | Basic Software Make file Interface |
| CAN | Controller Area Network |
| DEM | Diagnostic Event Manager |
| DET | Development Error Tracer |
| C/CPP | C and C++ Source Code |
| VLE | Variable Length Encoding |
| N/A | Not Applicable |
| MCU | Micro Controller Unit |

## 2.5 Reference List

**Table 2-3. Reference List**

| # | Title | Version |
|---|---|---|
| 1 | AUTOSAR 3.0MCU Driver Software Specification Document. | V2.2.0 R3.0 Rev 0001 |
| 2 | MPC5634M Reference Manual | Rev. 6, 4 October 2011 |

# Chapter 3
# Driver

## 3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR 3.0MCU Driver Software Specification document (See Table Reference List ).

## 3.2 Driver Design Summary

The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required from other Microcontroller Abstraction Layer (MCAL) software modules.

Driver major features:

### 3.2.1 Initialization of MCU clock, PLL, MCU clock distribution

The following list summarizes the clock architecture implemented on the MPC5634M:

- **System clock** can be derived from the following sources:

  - FMPLL with ext reference called PllClockMode with external clock reference in the plug-in

  - FMPLL with crystal reference called PllClockMode with crystal clock in the plug-in

  - External reference called PllOffMode with external clock reference in the plug-in

  - Crystal reference called PllOffMode with crystal clock reference in the plug-in

  The system clock is adopted for generating the following clock sources:

- **INTC** clock for the interrupt controller

- **SIU** clock for the SIU module that controls the uC reset configuration, pad configuration, external interrupt, general purpose I/O, internal peripheral multiplexing and the system reset operation.

- **Peripheral set clock** represents the clock source of the peripherals

- **FlexCAN source** for the first clock domain of the FlexCAN

- **ClkOut** represents the external address/data bus clock for the calibration interface

- **MCKO** represents the Nexus auxiliary port clock

**NOTE**

A system clock divider is placed right at the output of the FMPLL and before the clock is used by any other circuits, including the other clock dividers.

- **XTAL_OSC** represents the oscillator clock output that can be adopted as clock reference for the FMPLL or as source for the second clock domains of the FlexCAN
- Frequency Modulated Phase-Locked Loop (**FMPLL**):

  - generates an high speed system clock from a common 4-20 Mhz input crystal oscillator source or external clock generator. Its frequency is software configurable

  - supports software configurable frequency modulation

  -is continuously monitored by the Lock detect circuitry which reports the lock status of the PLL

  - is continuously monitored by the clock quality monitor unit (CQM) which is able to detect the failure of the PLL.

- **Internal RC** oscillator whose major features are:

  - nominal frequency of 16 Mhz

  - glitch free oscillation

  - adopted as reference time base to measure the frequency of the crystal oscillator and the FMPLL output in order to monitor the quality of the crystal and FMPLL.

## 3.2.2 Initialization of RAM sections

### 3.2.3 Activation of MCU low power modes

The microcontroller offers several devices modes corresponding to different usage models of the device. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. In particular the following modes to reduce the activity of the uC for saving energy are supported:

- Module Disable mode is intended to be used by software to let the uC wait in a state of low power consumption at the cost of higher wakeup latency until the uC is required to do something and then to react quickly (i.e within a few system clock cycles of an interrupt event). When the uC enters in the Module Disable mode the clock to the following non-memory mapped registers within the following modules is gated off:

**Table 3-1. Module Disable mode support**

| Block name |
|---|
| DSPI_B |
| DSPI_C |
| EBI |
| eTPU |
| FlexCAN A |
| FlexCAN C |
| eMIOS |
| eSCI_A |
| eSCI_B |
| Decimation Filter |
| NPC |
| Flash Array |

actions programmable by software are performed:

- the clocks to the core and the system memory are disabled

- the clock to each peripheral may be disabled

- the clock sources such as the PLLs, external and internal oscillators can be configured to be switched off. However a source can not be disabled if it has been chosen as the system clock to be used into the HALT mode.

- Code flash memory may be switched off

- Module HALT mode is intended to be used by software to let the uC wait in a state of low power consumption in which the clock to all registers within each module can be completely halted.
- Standby Mode is intended to be used by software to let the uC wait in a state of low power consumption in which the power supplies are removed from all modules except the standby RAM.

### 3.2.4 Activation of a MCU reset

### 3.2.5 Provides a service to get the reset reason from hardware

### 3.2.6 DMA Channel n Priority Register (*DMA_CPRn*) are initialized by the MCU driver

The direct memory access (DMA) controller is able to perform movements of data block from a source to a destination with minimal interventation from the host processor. The source or destination can be a memory block or an I/O block capable of operation with the DMA. The transfer can be requested by a source like a peripheral via one of the DMA channels.

Multiple service requests from different sources Arbitration can be configured to use either fixed-priority or round-robin policy. In the former, the highest priority channel requesting service is selected to execute. In particular a unique priority is associated with each DMA channel and is expressed by a numeric value which belongs to the interval [0..15] where 0 means the lowest priority, 1 is the next higher priority and so on until 15 which is the highest priority. Channel preemption can be enabled to allow the executing channel's data transfers to be temporarily suspended in favor of starting a higher priority channel. After the preempting channel has completed, the preempted channel is restored and resumes execution. Nested preemption (attempting to preempt a preempting channel) is not supported. Preemption is only available when fixed arbitration is selected. In round-robin arbitration mode, the channel priorities are ignored and the channels are cycled through, from channel 15 down to channel 0. The DMA control register (***DMA_CR***) can be configured to enable the round-robin or the fixed priority arbitration mechanism. The DMA channel n Priority Register (***DCHPRIn***) can be programmed by software to estabilish the preemption (on/off) and the arbitration priority (0..15) of the DMA channel n.

## 3.3 Deviation from Requirements

The driver deviates from the AUTOSAR MCU Driver software specification in some places.

There are also some additional requirements (on top of requirements detailed in AUTOSAR MCU Driver software specification) which need to be satisfied for correct operation.

**Table 3-2. Deviations Status Column Description**

| Term | Definition |
|---|---|
| N/A | Not available |
| N/T | Not testable |
| N/S | Out of scope |
| N/I | Not implemented |
| N/F | Not fully implemented |

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, or out of scope for the driver.

**Table 3-3. Driver Deviations Table**

| Requirement | Status | Description | Notes |
|---|---|---|---|
| MCU105 | N/A | The code file structure shall not be defined within this specification. | Not a requirement. |
| MCU127 | N/S | If not applicable, the MCU module's environment shall pass a NULL pointer to the function Mcu_Init. In this case the check for this NULL pointer has to be omitted. | Not a driver requirement. |
| MCU136 | N/S | The MCU module's environment shall call the function Mcu_InitRamSection only after the MCU module has been initialized using the function Mcu_Init. | Not a driver requirement.. |
| MCU139 | N/S | The MCU module's environment shall only call the function Mcu_InitClock after the MCU module has been initialized using the function Mcu_Init. | Not a driver requirement. |
| MCU141 | N/S | The MCU module's environment shall execute the function Mcu_DistributePllClock if the MCU module needs a separate request to activate the PLL clock after the PLL is locked. In this case, the function shall remove the current clock source (for example internal oscillator clock) from MCU clock distribution. | Not a driver requirement. |

*Table continues on the next page...*

**Table 3-3. Driver Deviations Table (continued)**

| Requirement | Status | Description | Notes |
|---|---|---|---|
| MCU142 | N/S | The MCU module's environment shall only call the function Mcu_DistributePllClock after the status of the PLL has been detected as locked by the function Mcu_GetPllStatus. | Not a driver requirement. |
| MCU145 | N/S | The MCU module's environment shall only call the function Mcu_PerformReset after the MCU module has been initialized by the function Mcu_Init. | Not a driver requirement. |
| MCU148 | N/S | The MCU module's environment shall only call the function Mcu_SetMode after the MCU module has been initialized by the function Mcu_Init. | Not a driver requirement. |

## 3.4 Software specification

The following sections contains driver software specifications.

### 3.4.1 Define Reference

Constants supported by the driver are as per AUTOSAR MCU Driver software specification Version 3.0 .

#### 3.4.1.1 Define MCU_CLOCKLOSS_ISR_USED

the ISR Mcu_ClockLoss_ISR is/isn't available (STD_ON/STD_OFF)

**Table 3-4. Define MCU_CLOCKLOSS_ISR_USED Description**

| | |
|---|---|
| **Name** | MCU_CLOCKLOSS_ISR_USED |
| **Initializer** | (STD_OFF) |

#### 3.4.1.2 Define MCU_DEV_ERROR_DETECT

Development error detection enabled/disabled (STD_ON/STD_OFF).

**Satisfied Requirements:** MCU118.

**Table 3-5. Define MCU_DEV_ERROR_DETECT Description**

| | |
|---|---|
| **Name** | MCU_DEV_ERROR_DETECT |
| **Initializer** | (STD_ON) |

### 3.4.1.3 Define MCU_ENTER_LOW_POWER_MODE

if enter low power mode transition enabled (STD_ON/STD_OFF)

**Table 3-6. Define MCU_ENTER_LOW_POWER_MODE Description**

| | |
|---|---|
| **Name** | MCU_ENTER_LOW_POWER_MODE |
| **Initializer** | (STD_ON) |

### 3.4.1.4 Define MCU_FLASH_CONFIGURATION_USED

the Api function Mcu_Flash_Configure is/isn't available (STD_ON/STD_OFF)

**Table 3-7. Define MCU_FLASH_CONFIGURATION_USED Description**

| | |
|---|---|
| **Name** | MCU_FLASH_CONFIGURATION_USED |
| **Initializer** | (STD_ON) |

### 3.4.1.5 Define MCU_LOCKLOSS_ISR_USED

the ISR Mcu_LockLoss_ISR complete is/isn't available (STD_ON/STD_OFF)

**Table 3-8. Define MCU_LOCKLOSS_ISR_USED Description**

| | |
|---|---|
| **Name** | MCU_LOCKLOSS_ISR_USED |
| **Initializer** | (STD_OFF) |

### 3.4.1.6 Define MCU_PERFORM_RESET_API

the Api function Mcu_PerformReset is/isn't available (STD_ON/STD_OFF)

**Table 3-9. Define MCU_PERFORM_RESET_API Description**

| | |
|---|---|
| **Name** | MCU_PERFORM_RESET_API |
| **Initializer** | (STD_ON) |

### 3.4.1.7 Define MCU_RESET_CALLOUT_USED

the callout reset is/isn't available (STD_ON/STD_OFF)

**Table 3-10. Define MCU_RESET_CALLOUT_USED Description**

| | |
|---|---|
| **Name** | MCU_RESET_CALLOUT_USED |
| **Initializer** | (STD_OFF) |

### 3.4.1.8 Define MCU_VERSION_INFO_API

version info API enabled/disabled (STD_ON/STD_OFF)

**Table 3-11. Define MCU_VERSION_INFO_API Description**

| | |
|---|---|
| **Name** | MCU_VERSION_INFO_API |
| **Initializer** | (STD_ON) |

## 3.4.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR MCU Driver software specification Version 3.0 .

### 3.4.2.1 Enumeration Mcu_ClockNotificationType

Clock failure notification.

**Details:**

Enable/disable clock failure interrupt generated by the MCU.

**Satisfied Requirements:** MCU054.

**Table 3-12. Enumeration Mcu_ClockNotificationType Values**

| Name | Initializer | Description |
|---|---|---|
| MCU_CLOCKNOTIFICATION_DISABLE | 0 | |
| MCU_CLOCKNOTIFICATION_ENABLE | | |

### 3.4.2.2 Enumeration Mcu_PllStatusType

Lock status of the PLL.

**Details:**

This is a status value returned by the function Mcu_GettPllStatus of the MCU module.

**Satisfied Requirements:** MCU054.

**Table 3-13. Enumeration Mcu_PllStatusType Values**

| Name | Initializer | Description |
|---|---|---|
| MCU_PLL_LOCKED | 0 | |
| MCU_PLL_UNLOCKED | | |
| MCU_PLL_STATUS_UNDEFINED | | |

### 3.4.2.3 Enumeration Mcu_ResetType

Contains the subset of reset sources.

**Satisfied Requirements:** MCU134.

**Table 3-14. Enumeration Mcu_ResetType Values**

| Name | Initializer | Description |
|---|---|---|
| MCU_POWER_ON_RESET | 0 | |

*Table continues on the next page...*

**Table 3-14. Enumeration Mcu_ResetType Values (continued)**

| Name | Initializer | Description |
|---|---|---|
| MCU_EXTERNAL_RESET | | |
| MCU_LOSS_OF_LOCK_RESET | | |
| MCU_LOSS_OF_CLOCK_RESET | | |
| MCU_WATCHDOG_RESET | | |
| MCU_CHECKSTOP_RESET | | |
| MCU_SW_WATCHDOG_RESET | | |
| MCU_SW_RESET | | |
| MCU_SW_EXT_RESET | | |
| MCU_RESET_UNDEFINED | | |

## 3.4.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR MCU Driver software specification Version 3.0 .

### 3.4.3.1 Function Mcu_DistributePllClock

This function activates the main PLL as the system clock source.

**Details:**

This function sets the main PLL clock as the system clock and also enables monitoring of the main PLL clock. A clock failure notification is generated when a mismatch with the PLL monitor occurs by enabling the interrupts from the current clock setting.

**Pre:** Function requires the status of the PLL has been detected as locked by the function `Mcu_GetPllStatus`.

**Note**

Function has to be executed if the MCU module needs a separate request to activate the PLL clock after the PLL is locked.

**Satisfied Requirements:** MCU156, MCU028, MCU053, MCU013, MCU100, MCU101, MCU016, MCU018, MCU125, MCU122.

**Prototype:** `void Mcu_DistributePllClock(void);`

### 3.4.3.2 Function Mcu_GetPllStatus

This function returns the lock status of the PLL.

**Details:**

This function returns the `MCU_PLL_STATUS_UNDEFINED` if this function is called prior to calling of the function Mcu_Init.

**Pre:** Function requires an execution of `Mcu_Init()` and `Mcu_InitClock()` before it can be used, otherwise it reports error to DET.

**Return:** Mcu_PllStatusType.

**Satisfied Requirements:** MCU157, MCU008, MCU132, MCU013, MCU100, MCU101, MCU016, MCU018

**Prototype:** `Mcu_PllStatusType Mcu_GetPllStatus(void);`

**Table 3-15. Mcu_GetPllStatus Return Values**

| Name | Description |
|---|---|
| MCU_PLL_LOCKED | PLL is locked. |
| MCU_PLL_UNLOCKED | PLL is unlocked. |
| MCU_PLL_STATUS_UNDEFINED | PLL Status in unknown. |

### 3.4.3.3 Function Mcu_GetResetRawValue

This function returns the raw reset value.

**Details:**

This function returns the raw reset value from the hardware register and return this reason if supported by the hardware.

**Pre:** Function requires an execution of `Mcu_Init()` before it can be used, otherwise it returns an implementation specific vlue which does not correspond to a valid value of the reset status register and is not equal to 0.

**Note**

If the hardware does not support the hardware detection of the reset reason, the function return value is always 0x0.

**Return:** Mcu_RawResetType.

**Satisfied Requirements:** MCU159, MCU006, MCU125, MCU013, MCU100, MCU101, MCU016, MCU018.

**Prototype:** `Mcu_RawResetType Mcu_GetResetRawValue(void);`

### 3.4.3.4 Function Mcu_GetResetReason

This function returns the reset reason.

**Details:**

This function returns the reset reason from the hardware and return this reason if supported by the hardware.

**Pre:** Function requires an execution of `Mcu_Init()` and `Mcu_InitClock()` before it can be used, otherwise it returns `MCU_RESET_UNDEFINED`.

**Note**

If the hardware does not support the hardware detection of the reset reason, the function return value is always MCU_POWER_ON_RESET.

**Return:** One of the possible reset reasons defined in Mcu_ResetType.

**Satisfied Requirements:** MCU158, MCU052, MCU005, MCU133, MCU125, MCU013, MCU100, MCU101, MCU016, MCU018.

**See:**
- Enumeration Mcu_ResetType

**Prototype:** `Mcu_ResetType Mcu_GetResetReason(void);`

### 3.4.3.5 Function Mcu_GetVersionInfo

This function returns the version information for the MCU module.

**Details:**

The version information includes:

- Module Id,
- Vendor Id,
- Vendor specific version numbers.

**Note**

The function is pre-compile time configurable on/off by the configuration parameter `McuVersionInfoApi`.

**Satisfied Requirements:** MCU103, MCU104, MCU162, MCU149.

**Prototype:** `void Mcu_GetVersionInfo(Std_VersionInfoType *versioninfo);`

**Table 3-16. Mcu_GetVersionInfo Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| Std_VersionInfoType * | versioninfo | **output** | Pointer for storing the version information of this module. |

### 3.4.3.6 Function Mcu_Init

MCU driver initialization function.

**Details:**

This routine initializes the MCU Driver. The intention of this function is to make the configuration setting for power down,clock and ram sections visible within the MCU Driver.

**Note**

Violates MISRA-C: 2004 19.1, 19.15 - See `Mcu_c_REF_1`, Mcu_c_REF_4

**Satisfied Requirements:** MCU153, MCU026, MCU126, MCU127, MCU013, MCU100, MCU101, MCU016, MCU018.

**Prototype:** `void Mcu_Init(const Mcu_ConfigType *ConfigPtr);`

**Table 3-17. Mcu_Init Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| const `Mcu_ConfigType` * | ConfigPtr | **input** | Pointer to MCU configuration structure. |

### 3.4.3.7 Function Mcu_InitClock

MCU driver clock initialization function.

**Details:**

This function intializes the PLL and MCU specific clock options. The clock setting is provided from the configuration structure.

**Pre:** Function requires an execution of `Mcu_Init()` before it can be used, otherwise it reports error to `DET` and returns `E_NOT_OK`.

**Return:** Std_ReturnType.

**Satisfied Requirements:** MCU155, MCU031, MCU107, MCU013, MCU100, MCU101, MCU016, MCU018, MCU019, MCU125, MCU112, MCU113, MCU017.

**Prototype:** `Std_ReturnType Mcu_InitClock(Mcu_ClockType ClockSetting);`

**Table 3-18. Mcu_InitClock Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Mcu_ClockType` | ClockSetting | **input** | Index of clock setting from config structure to be used. |

**Table 3-19. Mcu_InitClock Return Values**

| Name | Description |
|---|---|
| E_OK | Command has been accepted. |
| E_NOT_OK | Command has not been accepted. |

### 3.4.3.8 Function Mcu_InitRamSection

MCU driver initialization of ram sections.

**Details:**

Function intializes the ram section selected by RamSection parameter. The section base address, size and value to be written are provided from the configuration structure.

**Pre:** Function requires an execution of `Mcu_Init()` before it can be used, otherwise it reports error to `DET` and returns `E_NOT_OK`.

**Return:** Std_ReturnType.

**Satisfied Requirements:** MCU154, MCU030, MCU011, MCU013, MCU100, MCU101, MCU016, MCU018, MCU125, MCU017, MCU021, MCU018.

**Prototype:** `Std_ReturnType Mcu_InitRamSection(Mcu_RamSectionType RamSection);`

**Table 3-20. Mcu_InitRamSection Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Mcu_RamSectionType` | RamSection | **input** | Index of ram section from config structure to be initialized. |

**Table 3-21. Mcu_InitRamSection Return Values**

| Name | Description |
|---|---|
| E_OK | Command has been accepted. |
| E_NOT_OK | Command has not been accepted. |

### 3.4.3.9 Function Mcu_PerformReset

This function performs a microcontroller reset.

**Details:**

This function performs the reset type which is configured in the configuration set by using feature of the microcontroller.

**Pre:** Function requires an execution of `Mcu_Init()` before it can be used.

**Note**

The function is available if the parameter `McuPerformResetApi` is set to `TRUE`.

**Satisfied Requirements:** MCU160, MCU007, MCU125, MCU013, MCU100, MCU101, MCU016, MCU018, MCU146.

**Prototype:** `void Mcu_PerformReset(void);`

### 3.4.3.10 Function Mcu_SetMode

This function sets the MCU power mode.

**Details:**

This function sets MCU power modes configured in the configuration set.

**Pre:** Function requires an execution of `Mcu_Init()` before it can be used.

**Note**

If CPU is switched off, the function returs after it has performed a wake-up.

**Satisfied Requirements:** MCU161, MCU164, MCU001, MCU035, MCU013, MCU100, MCU101, MCU016, MCU018, MCU020, MCU125, MCU013, MCU163, MCU112

**Prototype:** `void Mcu_SetMode(Mcu_ModeType McuMode);`

**Table 3-22. Mcu_SetMode Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Mcu_ModeType` | McuMode | **input** | The parameter represents the MCU mode settings. |

## 3.4.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR MCU Driver software specification Version 3.0 .

### 3.4.4.1 Structure Mcu_ClockConfigType

Definition of a MCU mode section in the configuration structure.

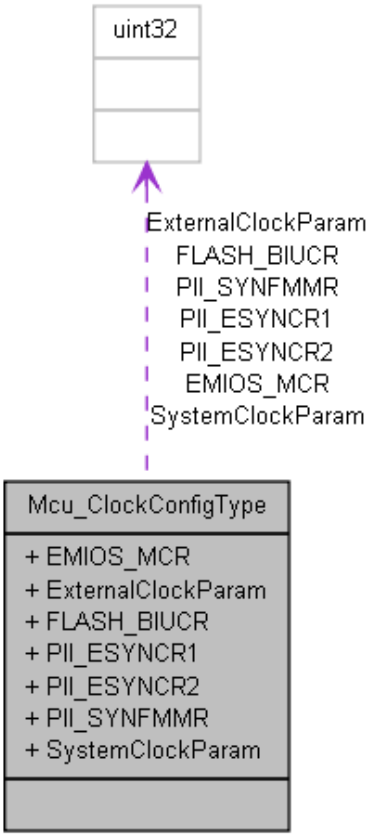



**Figure 3-1. Struct Mcu_ClockConfigType**

## Details:

Specifies the system behaviour during the selected target mode.

**Satisfied Requirements:** MCU035.

## Declaration:

```
typedef struct
                          {
                            uint32 EMIOS_MCR,
                                    uint32 ExternalClockParam,
                                    uint32 FLASH_BIUCR,
                                    uint32 Pll_ESYNCR1,
                                    uint32 Pll_ESYNCR2,
                                    uint32 Pll_SYNFMMR,
                                    uint32 SystemClockParam
                          } Mcu_ClockConfigType;
```

**Table 3-23. Structure Mcu_ClockConfigType member description**

| Member | Description |
|---|---|
| EMIOS_MCR | Specifies the configuration of the eMIOS200 block. It contains the configuration of the EMIOS_MCR register. |

*Table continues on the next page...*

**Table 3-23. Structure Mcu_ClockConfigType member description (continued)**

| Member | Description |
|---|---|
| ExternalClockParam | Specifies the timing relationship between the system clock and the external clock CLKOUT. It contains the configuration of the SIU_ECCR register. |
| FLASH_BIUCR | Specifies the global configuration of all attached flash arrays. It contains the configuration of the BIUCR register. |
| Pll_ESYNCR1 | Specifies the configuration of the PLL in terms of FMPLL Pre-divider, FMPLL feedback loop divider, FMPLL operating mode (Bypass/normal mode). It contains the configuration of the ESYNCR1 register. |
| Pll_ESYNCR2 | Specifies the configuration of the PLL in terms of FMPLL output divider, monitor of loss-of-clock, loss-of-lock reset, loss-of-clock reset, loss-of-lock interrupt request, Loss-of-clock interrupt request It contains the configuration of the ESYNCR2 register. |
| Pll_SYNFMMR | Specifies the configuration of the PLL modulation in terms of modulation period, modulation depth, modulation selection FMPLL It contains the configuration of the SYNFMMR register. |
| SystemClockParam | Specifies if the system clock divider is/isn't bypassed and the system clock divider. It contains the configuration of the SIU_SYSDIV register. |

### 3.4.4.2 Structure Mcu_ConfigType

The structure contains the initialization data for the MCU driver.

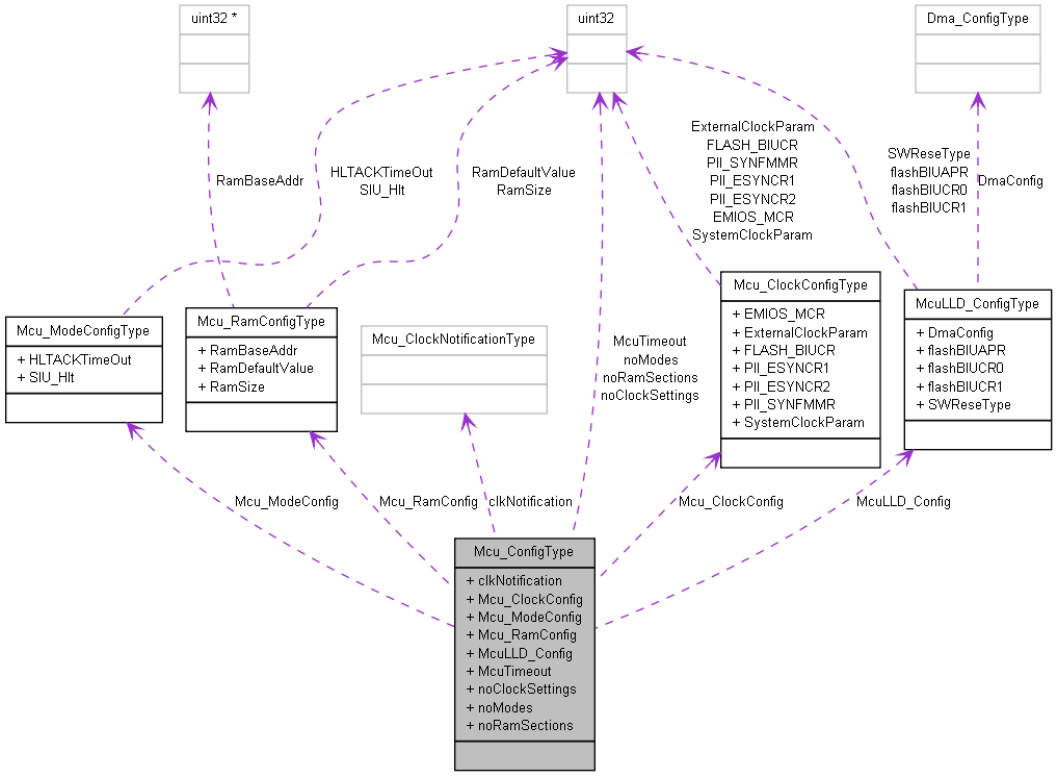



**Figure 3-2. Struct Mcu_ConfigType**

## Details:

A pointer to such a structure is provided to the MCU initialization routines for configuration.

**Satisfied Requirements:** MCU131, MCU054, MCU035, MCU031, MCU030

## Declaration:

```
typedef struct
                        {
                          Mcu_ClockNotificationType clkNotification,
                                  const Mcu_ClockConfigType * Mcu_ClockConfig,
                                  const Mcu_ModeConfigType * Mcu_ModeConfig,
                                  const Mcu_RamConfigType * Mcu_RamConfig,
                                  const McuLLD_ConfigType * McuLLD_Config,
                                  uint32 McuTimeout,
                                  Mcu_ClockType noClockSettings,
                                  Mcu_ModeType noModes,
                                  Mcu_RamSectionType noRamSections
                        } Mcu_ConfigType;
```

**Table 3-24. Structure Mcu_ConfigType member description**

| Member | Description |
|---|---|
| clkNotification | Clock failure notification. |

*Table continues on the next page...*

**Table 3-24. Structure Mcu_ConfigType member description (continued)**

| Member | Description |
|---|---|
| Mcu_ClockConfig | Identifies a system clock configuration. |
| Mcu_ModeConfig | Identifies a mode configuration in the configuration structure. |
| Mcu_RamConfig | Identifies a Ram setting in the configuration structure. |
| McuLLD_Config | Mcu driver configuration. |
| McuTimeout | |
| noClockSettings | Identifies a clock setting in the configuration structure. |
| noModes | Identifies a MCU mode in the configuration structure. |
| noRamSections | Identifies a RAM section in the configuration structure. |

### 3.4.4.3 Structure Mcu_RamConfigType

Definition of a RAM section within the configuration structure.

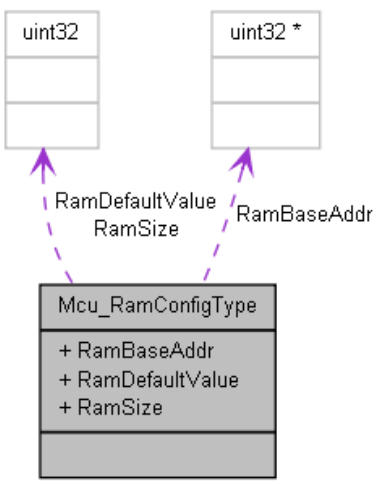



**Figure 3-3. Struct Mcu_RamConfigType**

**Satisfied Requirements:** MCU030.

**Declaration:**

```
typedef struct
                        {
                          uint32 * RamBaseAddr,
                                   uint32 RamDefaultValue,
                                   uint32 RamSize
                        } Mcu_RamConfigType;
```

**Table 3-25. Structure Mcu_RamConfigType member description**

| Member | Description |
|---|---|
| RamBaseAddr | RAM section base address. |
| RamDefaultValue | RAM default value. |
| RamSize | RAM section size. |

### 3.4.4.4 Structure McuLLD_ConfigType

Mcu driver configuration structure.

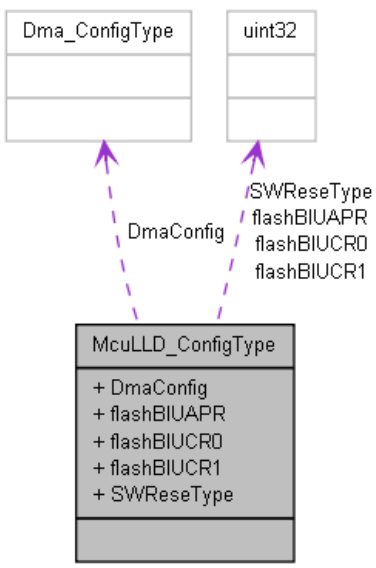



**Figure 3-4. Struct McuLLD_ConfigType**

**Satisfied Requirements:** MCU131

**Declaration:**

```
typedef struct
                        {
                          Dma_ConfigType DmaConfig,
                                  uint32 flashBIUAPR,
                                  uint32 flashBIUCR0,
                                  uint32 flashBIUCR1,
                                  uint32 SWReseType
                        } McuLLD_ConfigType;
```

**Table 3-26. Structure McuLLD_ConfigType member description**

| Member | Description |
|---|---|
| DmaConfig | Specifies the configuration of the DMA. |

*Table continues on the next page...*

**Table 3-26. Structure McuLLD_ConfigType member description (continued)**

| Member | Description |
|---|---|
| flashBIUAPR | Controls the read or write access permissions to the flash. It affects the value of the BIUAPR register. |
| flashBIUCR0 | Specifies the configuration of the flash in terms of Address pipelining control, Write Wait State Control and Read Wait State Control. It affects the value of the BIUCR register. |
| flashBIUCR1 | Defines the logical partitioning of the four page buffers. It affects the value of the BIUCR2 register. |
| SWReseType | Field for the RESET Options. |

## 3.4.5 Types Reference

Types supported by the driver are as per AUTOSAR MCU Driver software specification Version 3.0 .

### 3.4.5.1 Typedef Mcu_ClockType

Identifies a clock setting in the configuration structure.

**Details:**

The range is dependent on the the number of clock settings provided in the configuration structure.

**Type:** uint32

The range is dependent on the the number of clock settings provided in the configuration structure.

### 3.4.5.2 Typedef Mcu_ModeType

Identifies a MCU mode in the configuration structure.

**Details:**

The range is dependent on the the number of MCU modes provided in the configuration structure.

**Type:** uint32

The range is dependent on the the number of MCU modes provided in the configuration structure.

### 3.4.5.3 Typedef Mcu_RamSectionType

Identifies a RAM section in the configuration structure.

**Details:**

The range is dependent on the number of RAM sections provided in the configuration structure.

**Type:** uint32

The range is dependent on the number of RAM sections provided in the configuration structure.

### 3.4.5.4 Typedef Mcu_RawResetType

Identifies the reset reason in raw status register format.

**Type:** uint32

## 3.4.6 Variables Reference

Variables supported by the driver are as per AUTOSAR MCU Driver software specification Version 3.0 .

#### 3.4.6.1 Variable Mcu_Cfg_Ptr

Pointer to initialization structure.

**Note**

Violates MISRA-C: 2004 19.1, 19.15 - See `Mcu_c_REF_1`, Mcu_c_REF_4

**Satisfied Requirements:** MCU126

**Declaration:**

```
const Mcu_ConfigType* Mcu_Cfg_Ptr
```

## 3.5 Symbolic Names Disclaimer

All containers having the symbolic name tag set as true in the Autosar schema will generate defines like:

#define <Container_Short_Name> <Container_ID>

For this reason it is forbidden to duplicate the name of such containers across the MCAL configuration, or to use names that may trigger other compile issues (e.g. match existing #ifdefs arguments).

# Chapter 4
# Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the MCU Driver. The most of the parameters are described below.

## 4.1 Configuration elements of Mcu

**Included forms :**

- IMPLEMENTATION_CONFIG_VARIANT
- McuGeneralConfiguration
- CommonPublishedInformation
- McuModuleConfiguration

**Table 4-1. Revision table**

| Revision | Date |
|---|---|
| 2.0.0 | 2011-02-02T10:30:00 |

## 4.2 Form IMPLEMENTATION_CONFIG_VARIANT

**Table 4-2. Detailed description.**

| | |
|---|---|
| VariantPreCompile | Only precompile time configuration parameters. If Config Variant is equal to VariantPreCompile, the files Mcu_Cfg.h and Mcu_Cfg.c should be used. |
| VariantPostBuild | Mix of precompile and postbuild time configuration parameters. If Config Variant is equal to VariantPostBuild, the files Mcu_Cfg.h and Mcu_PBcfg.c should be used. |


**Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION_CONFIG_VARIANT form.**

**Table 4-3. Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description**

| Property | Value |
|---|---|
| Label | Config Variant |
| Default | VariantPreCompile |
| Range | VariantPreCompile<br>VariantPostBuild |

## 4.3 Form McuGeneralConfiguration

This container contains the configuration (parameters) of the MCU driver.

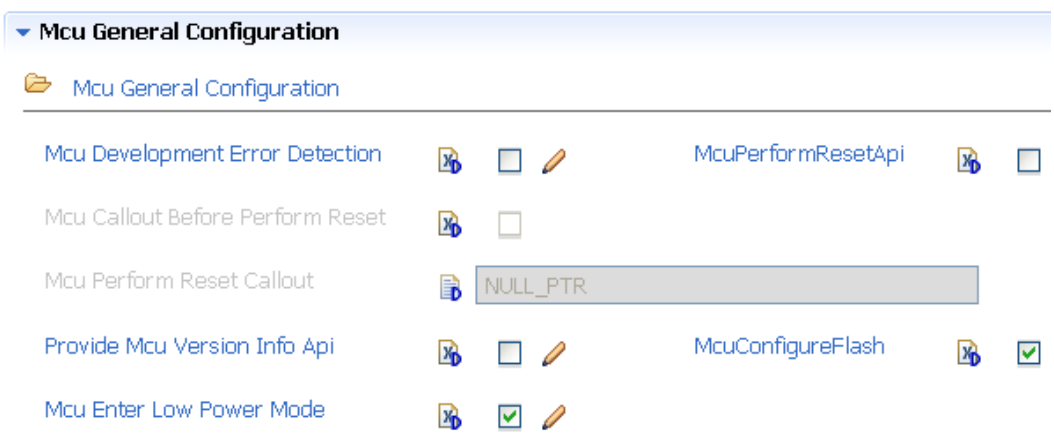



**Figure 4-2. Tresos Plugin snapshot for McuGeneralConfiguration form.**

### 4.3.1 McuDevErrorDetect (McuGeneralConfiguration)

Pre-processor switch for enabling the development error detection and reporting to the DEM. The switch McuDevErrorDetect shall activate or deactivate the detection of all development errors. The detection of development errors is configurable (ON/OFF) at precompile time. The detection of production code errors cannot be switched off.

**Table 4-4. Attribute McuDevErrorDetect (McuGeneralConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Development Error Detection |
| Type | BOOLEAN |
| Origin | AUTOSAR_ECUC |

*Table continues on the next page...*

**Table 4-4. Attribute McuDevErrorDetect (McuGeneralConfiguration) detailed description (continued)**

| Property | Value |
|---|---|
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.3.2 McuPerformResetApi (McuGeneralConfiguration)

Pre-processor switch to enable/disable the use of the function Mcu_PerformReset().

**Table 4-5. Detailed description.**

| | |
|---|---|
| 0 | Mcu_PerformReset() API is not used |
| 1 | Mcu_PerformReset() API is used |

**Table 4-6. Attribute McuPerformResetApi (McuGeneralConfiguration) detailed description**

| Property | Value |
|---|---|
| Type | BOOLEAN |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.3.3 McuCalloutBeforePerformReset (McuGeneralConfiguration)

Check this if you want a callout function, called by MCU right before Mcu_PerformReset().

Note:Implementation Specific Parameter

**Table 4-7. Attribute McuCalloutBeforePerformReset (McuGeneralConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Callout Before Perform Reset |

*Table continues on the next page...*

**Table 4-7. Attribute McuCalloutBeforePerformReset (McuGeneralConfiguration) detailed description (continued)**

| Property | Value |
|---|---|
| Type | BOOLEAN |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.3.4 McuPerformResetCallout (McuGeneralConfiguration)

Function name of callout. The field is editable only if McuCalloutBeforePerformReset is true.

Note:Implementation Specific Parameter

**Table 4-8. Attribute McuPerformResetCallout (McuGeneralConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Perform Reset Callout |
| Type | FUNCTION-NAME |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | NULL_PTR |

## 4.3.5 McuVersionInfoApi (McuGeneralConfiguration)

Pre-processor switch to enable / disable the API to read out the modules version information.

**Table 4-9. Attribute McuVersionInfoApi (McuGeneralConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Provide Mcu Version Info Api |
| Type | BOOLEAN |
| Origin | AUTOSAR_ECUC |

*Table continues on the next page...*

**Table 4-9. Attribute McuVersionInfoApi (McuGeneralConfiguration) detailed description (continued)**

| Property | Value |
|---|---|
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.3.6 McuConfigureFlash (McuGeneralConfiguration)

**Table 4-10. Detailed description.**

| | |
|---|---|
| 0 | Global Flash Settings including latency timings are not configured by the driver (reset values will not be updated). |
| 1 | Global Flash Settings including latency timings are configured by the driver |

Note:Implementation Specific Parameter

**Table 4-11. Attribute McuConfigureFlash (McuGeneralConfiguration) detailed description**

| Property | Value |
|---|---|
| Type | BOOLEAN |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | true |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.3.7 McuEnterLowPowerMode (McuGeneralConfiguration)

**Table 4-12. Detailed description.**

| | |
|---|---|
| 0 | Transition to Mcu in LOW POWER mode is performed in customer function. |
| 1 | Transition to Mcu in LOW POWER mode is performed by Mcu_SetMode API. |

**Table 4-13. Attribute McuEnterLowPowerMode (McuGeneralConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Enter Low Power Mode |
| Type | BOOLEAN |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | true |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.4 Form CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

### Included forms :

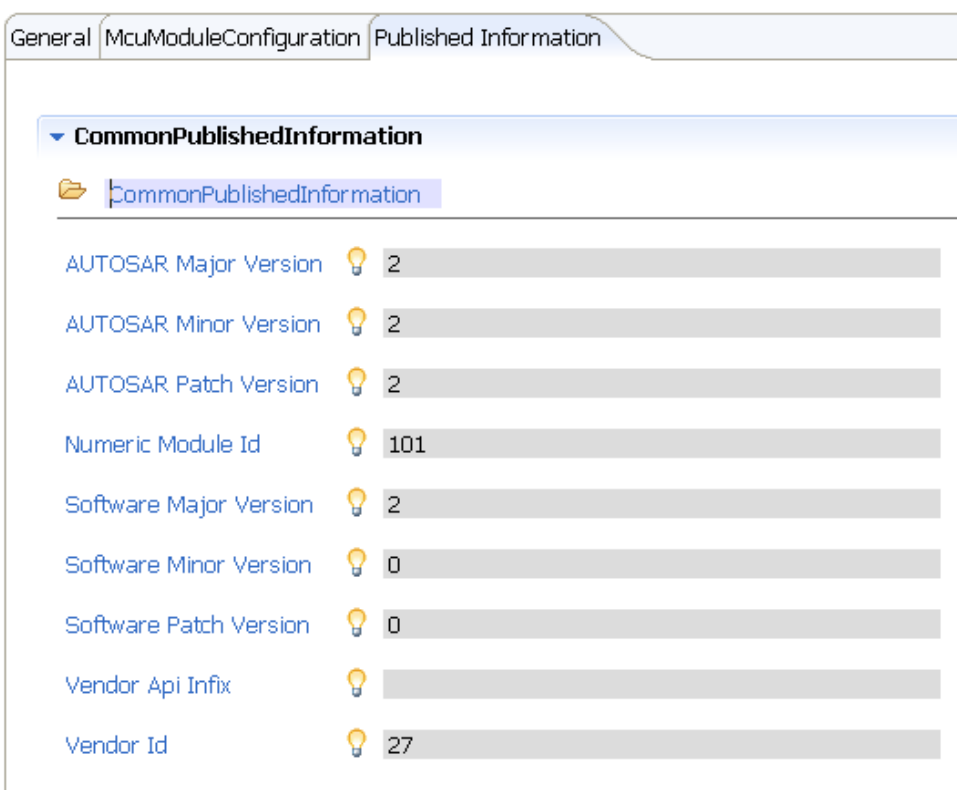



**Figure 4-3. Tresos Plugin snapshot for CommonPublishedInformation form.**

### 4.4.1 ArMajorVersion (CommonPublishedInformation)

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-14. Attribute ArMajorVersion (CommonPublishedInformation) detailed description**

| Property | Value |
|---|---|
| Label | AUTOSAR Major Version |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 2 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=2<br><=2 |

### 4.4.2 ArMinorVersion (CommonPublishedInformation)

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-15. Attribute ArMinorVersion (CommonPublishedInformation) detailed description**

| Property | Value |
|---|---|
| Label | AUTOSAR Minor Version |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 2 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=2<br><=2 |

### 4.4.3 ArPatchVersion (CommonPublishedInformation)

Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-16. Attribute ArPatchVersion (CommonPublishedInformation) detailed description**

| Property | Value |
|---|---|
| Label | AUTOSAR Patch Version |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 2 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=2<br><=2 |

### 4.4.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

**Table 4-17. Attribute ModuleId (CommonPublishedInformation) detailed description**

| Property | Value |
|---|---|
| Label | Numeric Module Id |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 101 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=101<br><=101 |

### 4.4.5 SwMajorVersion (CommonPublishedInformation)

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-18. Attribute SwMajorVersion (CommonPublishedInformation) detailed description**

| Property | Value |
|---|---|
| Label | Software Major Version |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 2 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=2<br><=2 |

## 4.4.6 SwMinorVersion (CommonPublishedInformation)

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-19. Attribute SwMinorVersion (CommonPublishedInformation) detailed description**

| Property | Value |
|---|---|
| Label | Software Minor Version |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 0 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=0<br><=0 |

## 4.4.7 SwPatchVersion (CommonPublishedInformation)

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-20. Attribute SwPatchVersion (CommonPublishedInformation) detailed description**

| Property | Value |
|---|---|
| Label | Software Patch Version |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 0 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=0<br><=0 |

## 4.4.8 VendorId (CommonPublishedInformation)

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

**Table 4-21. Attribute VendorId (CommonPublishedInformation) detailed description**

| Property | Value |
|---|---|
| Label | Vendor Id |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 27 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=27<br><=27 |

## 4.4.9 VendorApiInfix (VendorApiInfix)

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows: <ModuleName>_>VendorId>_<VendorApiInfix><Api name from SWS>. E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a

VendorApiInfix of "v11r456" a api name Can_Write defined in the SWS will translate to Can_123_v11r456Write. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

**Table 4-22. Attribute VendorApiInfix (VendorApiInfix) detailed description**

| Property | Value |
|---|---|
| Label | Vendor Api Infix |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | |
| Lower Multiplicity | 0 |
| Upper Multiplicity | 1 |

## 4.5 Form McuModuleConfiguration

This container contains the configuration (parameters) of the MCU driver.

**Included forms :**

- Form McuClockSettingConfig
- Form McuInterrupt
- Form McuEDMA_A_Config
- Form McuFlashBIUCR
- Form McuFlashBIUAPR
- Form McuFlashBIUCR2
- Form McuResetSource

- Form McuModeSettingConf
- Form McuRamSectorSettingConf

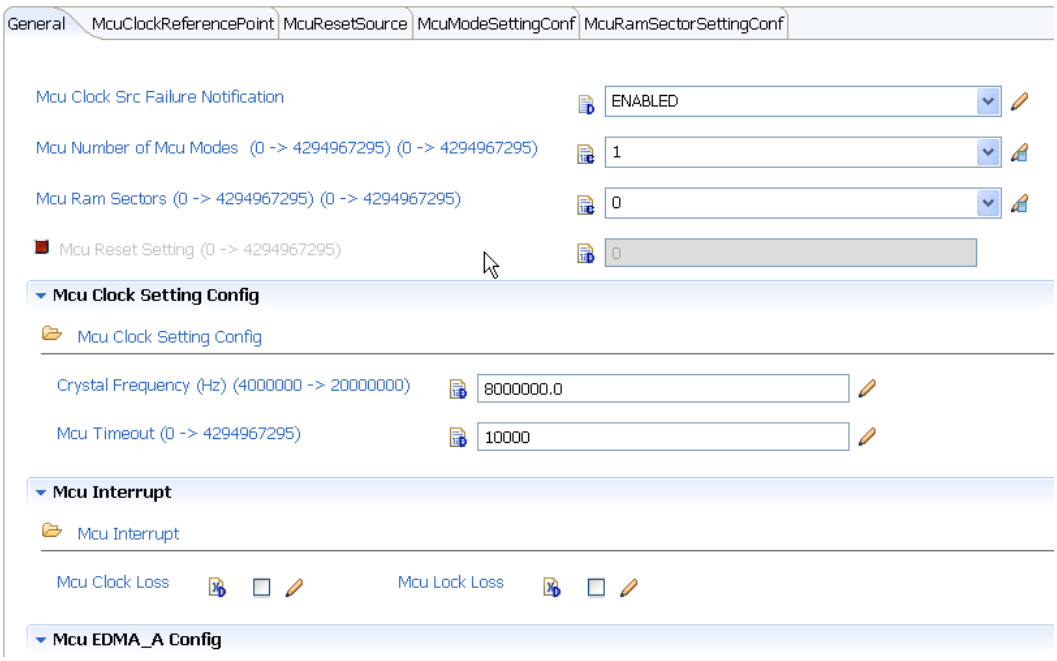



**Figure 4-4. Tresos Plugin snapshot for McuModuleConfiguration form.**

## 4.5.1 McuClockSrcFailureNotification (McuModuleConfiguration)

Enables/Disables clock failure notification.

**Table 4-23. Attribute McuClockSrcFailureNotification (McuModuleConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Clock Src Failure Notification |
| Type | ENUMERATION |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | ENABLED |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Range | ENABLED<br>DISABLED |

## 4.5.2 McuNumberOfMcuModes (McuModuleConfiguration)

This parameter shall represent the number of Modes available for the MCU.
CalculationFormula = Number of configured McuModeSettingConf.

**Table 4-24. Attribute McuNumberOfMcuModes (McuModuleConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Number of Mcu Modes (0 -> 4294967295) |
| Type | INTEGER |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.5.3 McuRamSectors (McuModuleConfiguration)

This parameter shall represent the number of RAM sectors available for the MCU. ClculationFormula = Number of configured McuRamSectorSettingConf.

**Table 4-25. Attribute McuRamSectors (McuModuleConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Ram Sectors (0 -> 4294967295) |
| Type | INTEGER |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.5.4 McuResetSetting (McuModuleConfiguration)

This parameters applies to the function Mcu_PerformReset, which performs a microcontroller reset using the hardware feature of the microcontroller.

Note: This parameter is not used by the current Implementation. Software Reset occurs when Mcu_PerformReset fucntion is called. The Reset is not user configurable parameter.

**Table 4-26. Attribute McuResetSetting (McuModuleConfiguration) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Reset Setting |
| Type | INTEGER |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 0 |
| Lower Multiplicity | 0 |
| Upper Multiplicity | 1 |
| Enable | false |

## 4.5.5 Form McuClockSettingConfig

This container contains the configuration (parameters) for the Clock settings of the MCU.

Note:Please see MCU031 for more information on the MCU clock settings.

**Is included by form :** Form McuModuleConfiguration

**Included forms :**

- Form McuClockReferencePoint

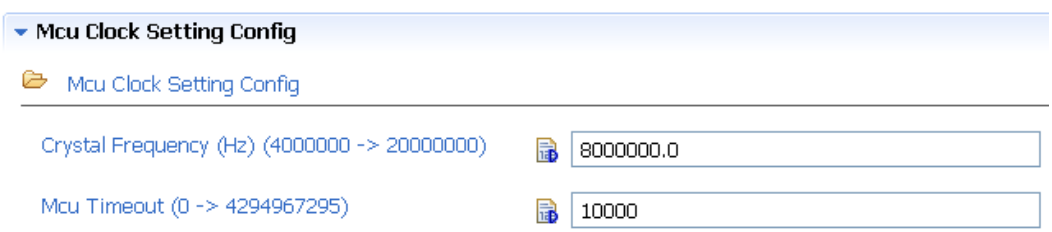



**Figure 4-5. Tresos Plugin snapshot for McuClockSettingConfig form.**

### 4.5.5.1 CrystalFrequencyHz (McuClockSettingConfig)

Crystal Frequency or External Reference Frequency [Hz]. CrystalFrequencyHz has to be in range 4 MHz up to 20 MHz.

Note: Implementation specific Parameter.

**Table 4-27. Attribute CrystalFrequencyHz (McuClockSettingConfig) detailed description**

| Property | Value |
|---|---|
| Label | Crystal Frequency (Hz) |
| Type | FLOAT |
| Origin | Custom |
| Symbolic Name | false |
| Default | 8000000 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=4000000<br><=20000000 |

### 4.5.5.2 McuTimeout (McuClockSettingConfig)

This parameter is used for internal loop, oscillator stability. It represents the maximum number of loops for blocking functionality.

Note: Implementation specific Parameter.

**Table 4-28. Attribute McuTimeout (McuClockSettingConfig) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Timeout |
| Origin | Custom |
| Symbolic Name | false |
| Default | 10000 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br>>=0<br><=4294967295 |

### 4.5.5.3 Form McuClockReferencePoint

This container defines a reference point in the Mcu Clock tree. It defines the frequency which then can be used by other modules as an input value. Lower multiplictiy is 1, as even in the simpliest case (only one frequency is used), there is one frequency to be defined.

**Is included by form :** Form McuClockSettingConfig

**Included forms :**

- Form GeneralClockSettings
- Form McuExternalClock
- Form McuPll
- Form McuEMIOSSettings
- Form McuFlashBIUCR0

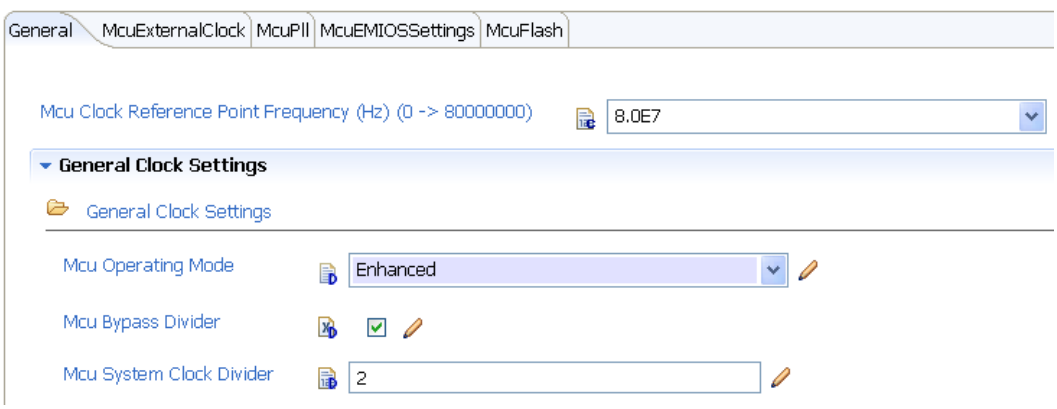



**Figure 4-6. Tresos Plugin snapshot for McuClockReferencePoint form.**

#### 4.5.5.3.1 McuClockReferencePointFrequency (McuClockReferencePoint)

This is the frequency for the specific instance of the McuClockReferencePoint container. It shall be given in Hz. McuClockReferencePointFrequency has to be less than or equal to 80 MHz.

**Table 4-29. Attribute McuClockReferencePointFrequency (McuClockReferencePoint) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Clock Reference Point Frequency (Hz) |
| Type | FLOAT |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Lower Multiplicity | 1 |

*Table continues on the next page...*

**Table 4-29. Attribute McuClockReferencePointFrequency (McuClockReferencePoint) detailed description (continued)**

| Property | Value |
|---|---|
| Upper Multiplicity | 1 |
| Invalid | Range<br><=80000000<br>>=0 |

#### 4.5.5.3.2 Form GeneralClockSettings

This container contains the SPECIFIC configuration (parameters) of the GeneralClockSettings Configuration.

Note: Implementation specific Container.

**Is included by form :** Form McuClockReferencePoint

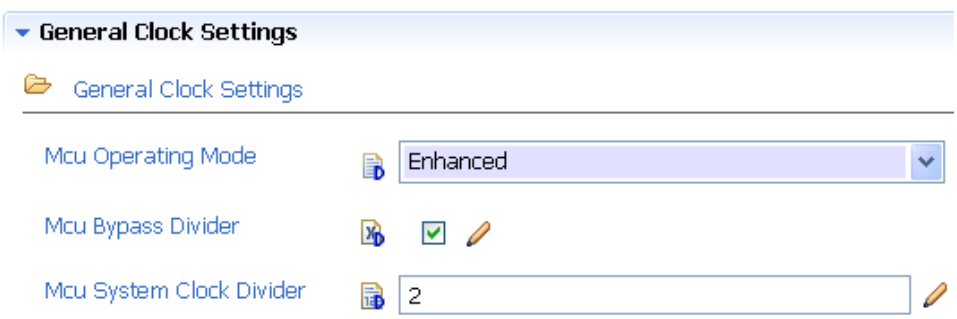



**Figure 4-7. Tresos Plugin snapshot for GeneralClockSettings form.**

##### 4.5.5.3.2.1 McuOperatingMode (GeneralClockSettings)

Clock configuration: Legacy/Enhanced Mode model. Not used in the current implementation. Note: implementation specific parameter.

**Table 4-30. Attribute McuOperatingMode (GeneralClockSettings) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Operating Mode |
| Type | ENUMERATION |
| Origin | Custom |
| Symbolic Name | false |
| Default | Enhanced |
| Lower Multiplicity | 1 |

*Table continues on the next page...*

**Table 4-30. Attribute McuOperatingMode (GeneralClockSettings) detailed description (continued)**

| Property | Value |
|---|---|
| Upper Multiplicity | 1 |
| Range | Enhanced |

#### 4.5.5.3.2.2 McuBypassDivider (GeneralClockSettings)

This parameter determines wether the system clock divider is bypassed or not. If it is selected, the divider is bypassed. Note: implementation specific parameter.

**Table 4-31. Attribute McuBypassDivider (GeneralClockSettings) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Bypass Divider |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | true |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.2.3 McuSystemClockDivider (GeneralClockSettings)

This parameter represents the system clock divider. Note: implementation specific parameter.

**Table 4-32. Attribute McuSystemClockDivider (GeneralClockSettings) detailed description**

| Property | Value |
|---|---|
| Label | Mcu System Clock Divider |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 2 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

*Table continues on the next page...*

**Table 4-32. Attribute McuSystemClockDivider (GeneralClockSettings) detailed description (continued)**

| Property | Value |
|---|---|
| Invalid | Range<br>=2<br>=4<br>=8<br>=16 |

### 4.5.5.3.3 Form McuExternalClock

This container contains the SPECIFIC configuration (parameters) of the Clock Output Configuration. Note: Implementation specific Container.

**Is included by form :** Form McuClockReferencePoint

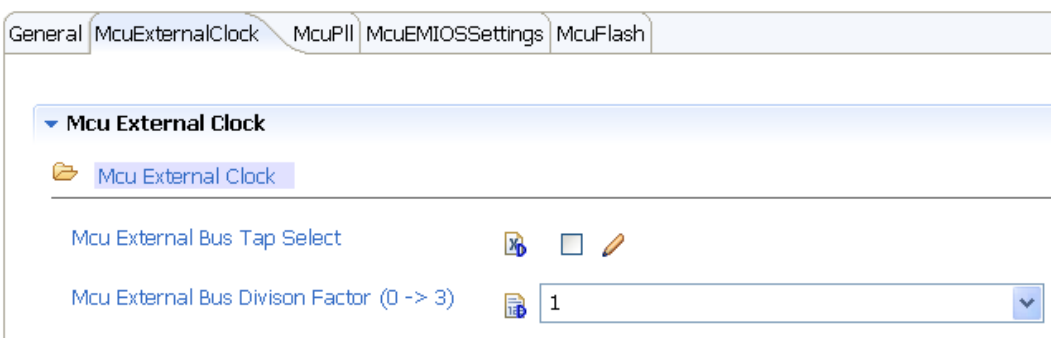



**Figure 4-8. Tresos Plugin snapshot for McuExternalClock form.**

#### 4.5.5.3.3.1 McuExternalBusTapSelect (McuExternalClock)

External bus tap select. This parameter correspond to EBTS bit. Changes the phase relationship between the system clock and CLKOUT. Changing the phase relationship so that CLKOUT is advanced in relation to the system clock increases the output hold time of the external bus signals to a non-zero value. It also increases the output delay times, increases the input hold times to non-zero values, and decreases the input setup times. Refer to the Electrical Specifications for how the EBTS bit affects the external bus timing.Note: Implementation specific Parameter.

**Table 4-33. Attribute McuExternalBusTapSelect (McuExternalClock) detailed description**

| Property | Value |
|---|---|
| Label | Mcu External Bus Tap Select |
| Type | BOOLEAN |

*Table continues on the next page...*

**Table 4-33. Attribute McuExternalBusTapSelect (McuExternalClock) detailed description (continued)**

| Property | Value |
|---|---|
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.3.2 McuExternalBusDivisonFactor (McuExternalClock)

External bus division factor. Specifies the frequency ratio between the system clock and the external clock, CLKOUT. Do not change EBDF during an external bus access or while an access is pending. The CLKOUT frequency is divided from the system clock frequency according to the descriptions below.

**Table 4-34. Detailed description.**

| | |
|---|---|
| 0 | External Bus Division Factor 1. |
| 1 | External Bus Division Factor 2 |
| 3 | External Bus Division Factor 4 |

Note: Implementation specific Parameter.

**Table 4-35. Attribute McuExternalBusDivisonFactor (McuExternalClock) detailed description**

| Property | Value |
|---|---|
| Label | Mcu External Bus Divison Factor |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 1 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.5.3.4 Form McuPll

This container contains the SPECIFIC configuration (parameters) of the MCU PLL Configuration. Note: Implementation specific Container.

**Is included by form :** Form McuClockReferencePoint

**Included forms :**

- Form McuPllParameter

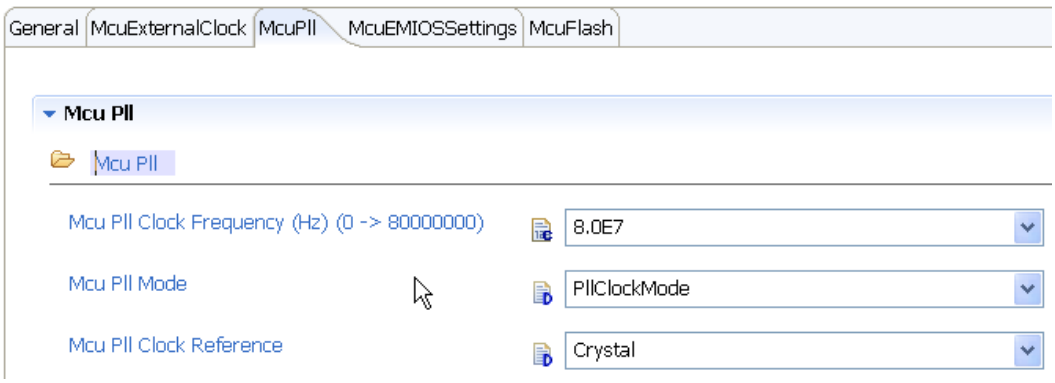



**Figure 4-9. Tresos Plugin snapshot for McuPll form.**

#### 4.5.5.3.4.1 McuPllClockFrequency (McuPll)

This is the PLL frequency for the specific instance of the McuClockReferencePoint container. It is expressed in Hz.

**Table 4-36. Attribute McuPllClockFrequency (McuPll) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Pll Clock Frequency (Hz) |
| Type | FLOAT |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=80000000<br>>=0 |

#### 4.5.5.3.4.2 McuPllMode (McuPll)

Clock Configuration. This parameter corresponds to CLKCFG[0:2] bits that are writable versions of the MODE, PLLSEL, and PLLREF bits in the SYNSR. These change the clock mode, after reset has negated, via software. CLKCFG[0:2] map directly to MODE, PLLSEL, and PLLREF to control the system clock mode. Note: Implementation specific Parameter.

**Table 4-37. Attribute McuPllMode (McuPll) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Pll Mode |
| Origin | Custom |
| Symbolic Name | false |
| Default | PllClockMode |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Enable | true |
| Range | PLLOffMode<br>PllClockMode |

#### 4.5.5.3.4.3 McuPllClockReference (McuPll)

This parameter specifies if the input clock of the PLL is crystal or external clock. Note: Implementation specific Parameter.

**Table 4-38. Attribute McuPllClockReference (McuPll) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Pll Clock Reference |
| Type | ENUMERATION |
| Origin | Custom |
| Symbolic Name | false |
| Default | Crystal |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Range | External<br>Crystal |

#### 4.5.5.3.4.4 Form McuPllParameter

**Is included by form :** Form McuPll

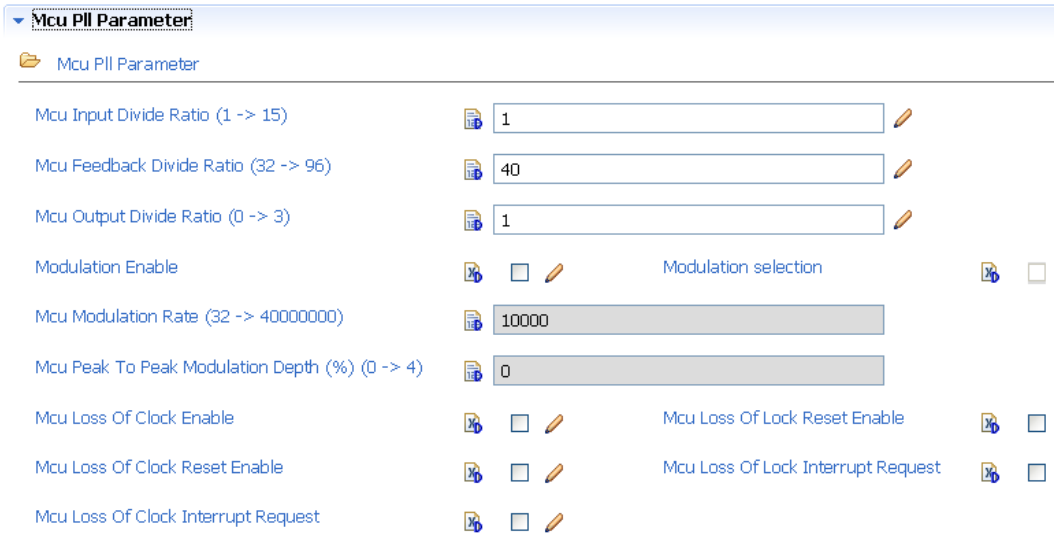



**Figure 4-10. Tresos Plugin snapshot for McuPllParameter form.**

#### *4.5.5.3.4.4.1 McuInputDivideRatio (McuPllParameter)*

Enhanced Pre-Divider. This parameter corresponds to (EPREDIV + 1). The EPREDIV bits control the value of the divider on the input clock. The output of the pre-divider circuit generates the reference clock to the PLL analog loop. When the EPREDIV bits are changed, the PLL immediately loses lock. In PLL bypass mode, the EPREDIV bits have no effect. Note: Implementation specific Parameter.

**Table 4-39. Attribute McuInputDivideRatio (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Input Divide Ratio |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 1 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=15<br>>=1 |

#### 4.5.5.3.4.4.2 McuFeedbackDivideRatio (McuPllParameter)

Enhanced Multiplication Factor Divider. This parameter correspond to (EMFD). The EMFD bits control the value of the divider in the PLL feedback loop. The value specified by the EMFD bits establish the multiplication factor applied to the reference frequency. In PLL Off mode, the EMFD bits have no effect. Note: Implementation specific Parameter.

**Table 4-40. Attribute McuFeedbackDivideRatio (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Feedback Divide Ratio |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 40 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=96<br>>=32 |

#### 4.5.5.3.4.4.3 McuOutputDivideRatio (McuPllParameter)

Enhanced reduced frequency divider. This 2-bit field controls a divider at the output of the FMPLL. The value specified by the ERFD bits establishes the division factor applied to the FMPLL frequency.

**Table 4-41. Detailed description.**

| | |
|---|---|
| 00 | Divide by 2 |
| 01 | Divide by 4 |
| 10 | Divide by 8 |
| 11 | Divide by 16 |

**Table 4-42. Attribute McuOutputDivideRatio (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Output Divide Ratio |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |

*Table continues on the next page...*

**Table 4-42. Attribute McuOutputDivideRatio (McuPllParameter) detailed description (continued)**

| Property | Value |
|---|---|
| Default | 1 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=3<br>>=0 |

#### *4.5.5.3.4.4.4 MODEN (McuPllParameter)*

Modulation enable. This bit enables the frequency modulation.

**Table 4-43. Detailed description.**

| | |
|---|---|
| 0 | Frequency modulation disabled. |
| 1 | Frequency modulation enabled |

**Table 4-44. Attribute MODEN (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Modulation Enable |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |

#### *4.5.5.3.4.4.5 MODSEL (McuPllParameter)*

Modulation selection. This bit selects whether modulation will be centered around the nominal frequency or spread below the nominal frequency.

**Table 4-45. Detailed description.**

| | |
|---|---|
| 0 | Modulation centered around nominal frequency. |
| 1 | Modulation spread below nominal frequency |

**Table 4-46. Attribute MODSEL (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Modulation selection |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |

#### 4.5.5.3.4.4.6 McuModulationRate (McuPllParameter)

Frequency for the triangular wave modulation (Fmod) in Hz.

**Table 4-47. Attribute McuModulationRate (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Modulation Rate |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 10000 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=40000000<br>>=32 |

#### 4.5.5.3.4.4.7 McuPeaktoPeakModulationDepth (McuPllParameter)

Enhanced Modulation depth: +/- 2.0 % (4% peak-to-peak)

**Table 4-48. Attribute McuPeaktoPeakModulationDepth (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Peak To Peak Modulation Depth (%) |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 0 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.4.4.8 McuLossOfClockEnable (McuPllParameter)

Loss-of-Clock Enable. This parameter corresponds to LOCEN. The LOCEN bit determines whether the loss-of-clock function is operational along with backup clock modes, and interrupt and reset functions. Note: Implementation specific Parameter.

**Table 4-49. Attribute McuLossOfClockEnable (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Loss Of Clock Enable |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### *4.5.5.3.4.4.9 McuLossOfLockResetEnable (McuPllParameter)*

Loss-of-Lock Reset Enable. This parameter corresponds to LOLRE. The LOLRE bit determines how the integration module handles a loss-of-lock indication. Note: Implementation specific Parameter.

**Table 4-50. Attribute McuLossOfLockResetEnable (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Loss Of Lock Reset Enable |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### *4.5.5.3.4.4.10 McuLossOfClockResetEnable (McuPllParameter)*

Loss-of-Clock Reset Enable. This parameter corresponds to LOCRE. The LOCRE bit determines how the integration module handles a loss-of-clock condition when LOCEN is equal to 1. LOCRE has no effect when LOCEN is equal to 0. Note: Implementation specific Parameter.

**Table 4-51. Attribute McuLossOfClockResetEnable (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Loss Of Clock Reset Enable |
| Origin | Custom |
| Symbolic Name | false |

*Table continues on the next page...*

**Table 4-51. Attribute McuLossOfClockResetEnable (McuPllParameter) detailed description (continued)**

| Property | Value |
|---|---|
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### *4.5.5.3.4.4.11 McuLossOfLockInterruptRequest (McuPllParameter)*

Loss-of-Lock Interrupt Request. This parameter corresponds to LOLIRQ. The LOLIRQ bit determines how the integration module handles a loss-of-lock indication. Note: Implementation specific Parameter.

**Table 4-52. Attribute McuLossOfLockInterruptRequest (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Loss Of Lock Interrupt Request |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### *4.5.5.3.4.4.12 McuLossOfClockInterruptRequest (McuPllParameter)*

Loss- of-Clock Interrupt Request. This bit corresponds to LOCIRQ. The LOCIRQ bit determines how the integration module handles a lossof-clock condition when LOCEN = 1. LOCIRQ has no effect when LOCEN = 0. Note: Implementation specific Parameter.

**Table 4-53. Attribute McuLossOfClockInterruptRequest (McuPllParameter) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Loss Of Clock Interrupt Request |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |

*Table continues on the next page...*

**Table 4-53. Attribute McuLossOfClockInterruptRequest (McuPllParameter) detailed description (continued)**

| Property | Value |
|---|---|
| Upper Multiplicity | 1 |

#### 4.5.5.3.5 Form McuEMIOSSettings

This container contains the SPECIFIC configuration (parameters) of the EMIOS Channel A. To use it please enable McuOsc. Note: Implementation specific Container

**Is included by form :** Form McuClockReferencePoint

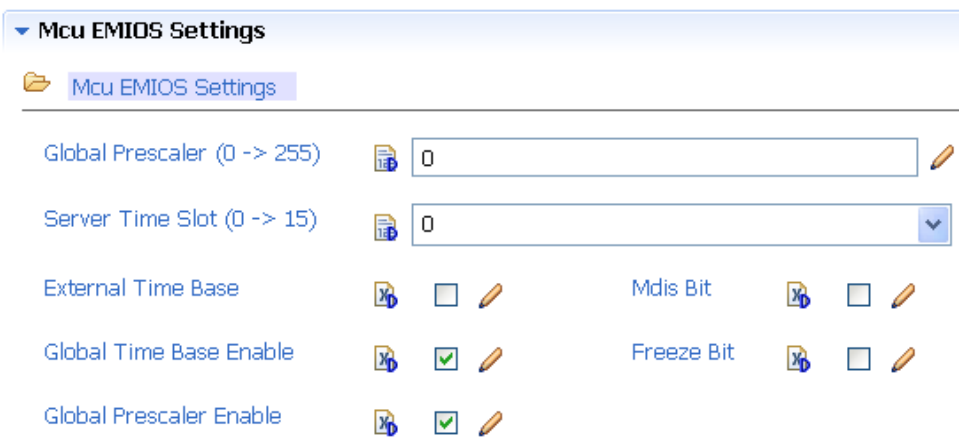



**Figure 4-11. Tresos Plugin snapshot for McuEMIOSSettings form.**

##### 4.5.5.3.5.1 GlobalPrescaler (McuEMIOSSettings)

Global Prescaler Bits (GPRE[0:7] in EMIOS_MCR register).The bits select the clock divider value for the global prescaler.The allowed values are 0 up to 255 (divide ratio 1 up to 256). Note: Implementation specific Parameter .

**Table 4-54. Attribute GlobalPrescaler (McuEMIOSSettings) detailed description**

| Property | Value |
|---|---|
| Label | Global Prescaler |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 0 |
| Lower Multiplicity | 1 |

*Table continues on the next page...*

**Table 4-54. Attribute GlobalPrescaler (McuEMIOSSettings) detailed description (continued)**

| Property | Value |
|---|---|
| Upper Multiplicity | 1 |
| Invalid | Range<br><=255<br>>=0 |

#### 4.5.5.3.5.2 ServerTimeSlot (McuEMIOSSettings)

Server time slot. Selects the address of a specific STAC server to which the STAC client submodule is assigned. See Section 22.4.3 STAC Client Submodule.

**Table 4-55. Detailed description.**

| | |
|---|---|
| 0 | eTPU engine A, TCR1 |
| 1 | Reserved |
| 2 | eTPU engine A, TCR2 |
| 3-15 | Reserved |

**Table 4-56. Attribute ServerTimeSlot (McuEMIOSSettings) detailed description**

| Property | Value |
|---|---|
| Label | Server Time Slot |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 0 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.5.3 ExternalTimeBase (McuEMIOSSettings)

External Time base Bit. The ETB bit selects the time base source that drives counter bus[A].

**Table 4-57. Detailed description.**

| | |
|---|---|
| False | Counter bus[A] assigned to eMIOS Channel |
| True | 1 STAC drives counter bus [A] |

Note: If ETB is set to select STAC as the counter bus[A] source, the GTBE must be set to enable the STAC to counter bus[A]. See the STAC bus configuration register (ETPU_REDCR) section of the eTPU chapter for more information about the STAC.

**Table 4-58. Attribute ExternalTimeBase (McuEMIOSSettings) detailed description**

| Property | Value |
|---|---|
| Label | External Time Base |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.5.4 MdisBit (McuEMIOSSettings)

Check this to put the eMIOS200 in low power mode. The MDIS bit is used to stop the clock of the block, except the access to registers EMIOSMCR, EMIOSOUDIS and EMIOSUCDIS.

**Table 4-59. Attribute MdisBit (McuEMIOSSettings) detailed description**

| Property | Value |
|---|---|
| Label | Mdis Bit |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.5.5 GlobalTimeBaseEnable (McuEMIOSSettings)

Global Time Base Enable Bit (GTBE in EMIOS_MCR register). The GTBE bit is used to export a global time base enable from the module and provide a method to start time bases of several blocks simultaneously. Note: Implementation specific Parameter

**Table 4-60. Attribute GlobalTimeBaseEnable (McuEMIOSSettings) detailed description**

| Property | Value |
|---|---|
| Label | Global Time Base Enable |
| Type | BOOLEAN |

*Table continues on the next page...*

**Table 4-60. Attribute GlobalTimeBaseEnable (McuEMIOSSettings) detailed description (continued)**

| Property | Value |
|---|---|
| Origin | Custom |
| Symbolic Name | false |
| Default | true |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.5.6 FreezeBit (McuEMIOSSettings)

Freeze Bit (FRZ in EMIOS_MCR register).Enable the eMIOS200 to freeze the registers of the unified channels when debug mode is requested at MCU level. Each unified channel must have FREN bit set in order to enter freeze mode. Note: Implementation specific Parameter.

**Table 4-61. Attribute FreezeBit (McuEMIOSSettings) detailed description**

| Property | Value |
|---|---|
| Label | Freeze Bit |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.5.7 GlobalPrescalerEnable (McuEMIOSSettings)

Global Prescaler Enable Bit (GPREN in EMIOS_MCR register).The GPREN bit enables the prescaler counter.Note: Implementation specific Parameter.

**Table 4-62. Attribute GlobalPrescalerEnable (McuEMIOSSettings) detailed description**

| Property | Value |
|---|---|
| Label | Global Prescaler Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |

*Table continues on the next page...*

**Table 4-62. Attribute GlobalPrescalerEnable (McuEMIOSSettings) detailed description (continued)**

| Property | Value |
|---|---|
| Default | true |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.5.3.6 Form McuFlashBIUCR0

This container contains the SPECIFIC configuration (parameters) of the Mcu Flash BIUCR0 register. Note: implementation specific parameter.

**Is included by form :** Form McuClockReferencePoint

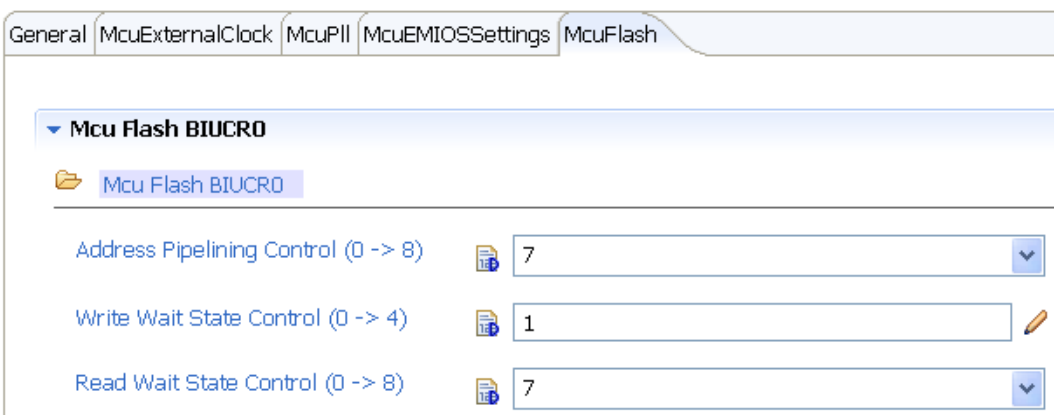



**Figure 4-12. Tresos Plugin snapshot for McuFlashBIUCR0 form.**

#### 4.5.5.3.6.1 AddressPipeliningControl (McuFlashBIUCR0)

Address Pipelining Control This field is used to control the number of cycles between pipelined access requests. It must be set to a value corresponding to the operating frequency of the PFLASH(1). Higher operating frequencies require non-zero settings for this field for proper Flash operation. This field is set to 0b111 by hardware reset.

**Table 4-63. Detailed description.**

| | |
|---|---|
| 000 | Accesses may be pipelined back-to-back |
| 001 | Access requests require one additional hold cycle |
| 010 | Access requests require two additional hold cycles |
| ... | |
| 110 | Access requests require six additional hold cycles |
| 111 | No address pipelining |

Note: Insert 8 to let the tool calculate the recommended value automatically. Note: implementation specific parameter.

**Table 4-64. Attribute AddressPipeliningControl (McuFlashBIUCR0) detailed description**

| Property | Value |
|---|---|
| Label | Address Pipelining Control |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 7 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.6.2 WriteWaitStateControl (McuFlashBIUCR0)

Write Wait State Control This field is used to control the number of wait-states to be added to the best-case Flash array access time for writes. The best-case Flash array access time for writes is two cycles. This field must be set to a value corresponding to the operating frequency of the PFLASH1. Higher operating frequencies require non-zero settings for this field for proper Flash operation. This field is set to 0b11 by hardware reset.

**Table 4-65. Detailed description.**

| | |
|---|---|
| 00 | No additional wait-states are added |
| 01 | One additional wait-state is added |
| 10 | Two additional wait-states are added |
| 11 | Three additional wait-states are added |

Note: Insert 4 to let the tool calculate the recommended value automatically. Note: implementation specific parameter.

**Table 4-66. Attribute WriteWaitStateControl (McuFlashBIUCR0) detailed description**

| Property | Value |
|---|---|
| Label | Write Wait State Control |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 1 |

*Table continues on the next page...*

**Table 4-66. Attribute WriteWaitStateControl (McuFlashBIUCR0) detailed description (continued)**

| Property | Value |
|---|---|
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

#### 4.5.5.3.6.3 ReadWaitStateControl (McuFlashBIUCR0)

Read Wait State Control This field is used to control the number of wait-states to be added to the best-case Flash array access time for reads. The best-case Flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the PFLASH and the actual read access time of the PFLASH1. Higher operating frequencies require non-zero settings for this field for proper Flash operation. This field is set to 0b111 by hardware reset.

**Table 4-67. Detailed description.**

| | |
|---|---|
| 000 | No additional wait-states are added |
| 001 | One additional wait-state is added |
| ... | |
| 111 | Seven additional wait-states are added |

Note: Insert 8 to let the tool calculate the recommended value automatically. Note: implementation specific parameter.

**Table 4-68. Attribute ReadWaitStateControl (McuFlashBIUCR0) detailed description**

| Property | Value |
|---|---|
| Label | Read Wait State Control |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 7 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.5.6 Form McuInterrupt

Note: Implementation specific Container.

**Is included by form :** Form McuModuleConfiguration

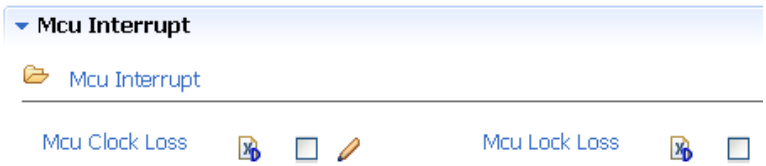



**Figure 4-13. Tresos Plugin snapshot for McuInterrupt form.**

### 4.5.6.1 McuClockLoss (McuInterrupt)

Check if you want to include the ISR on clock loss. Note: Implementation specific Parameter .

**Table 4-69. Attribute McuClockLoss (McuInterrupt) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Clock Loss |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.6.2 McuLockLoss (McuInterrupt)

Check if you want to include the ISR on lock loss. Note: Implementation specific Parameter.

**Table 4-70. Attribute McuLockLoss (McuInterrupt) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Lock Loss |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.7 Form McuEDMA_A_Config

This container contains the SPECIFIC configuration (parameters) of the EDMA Configuration. Implementation specific Container.

**Is included by form :** Form McuModuleConfiguration

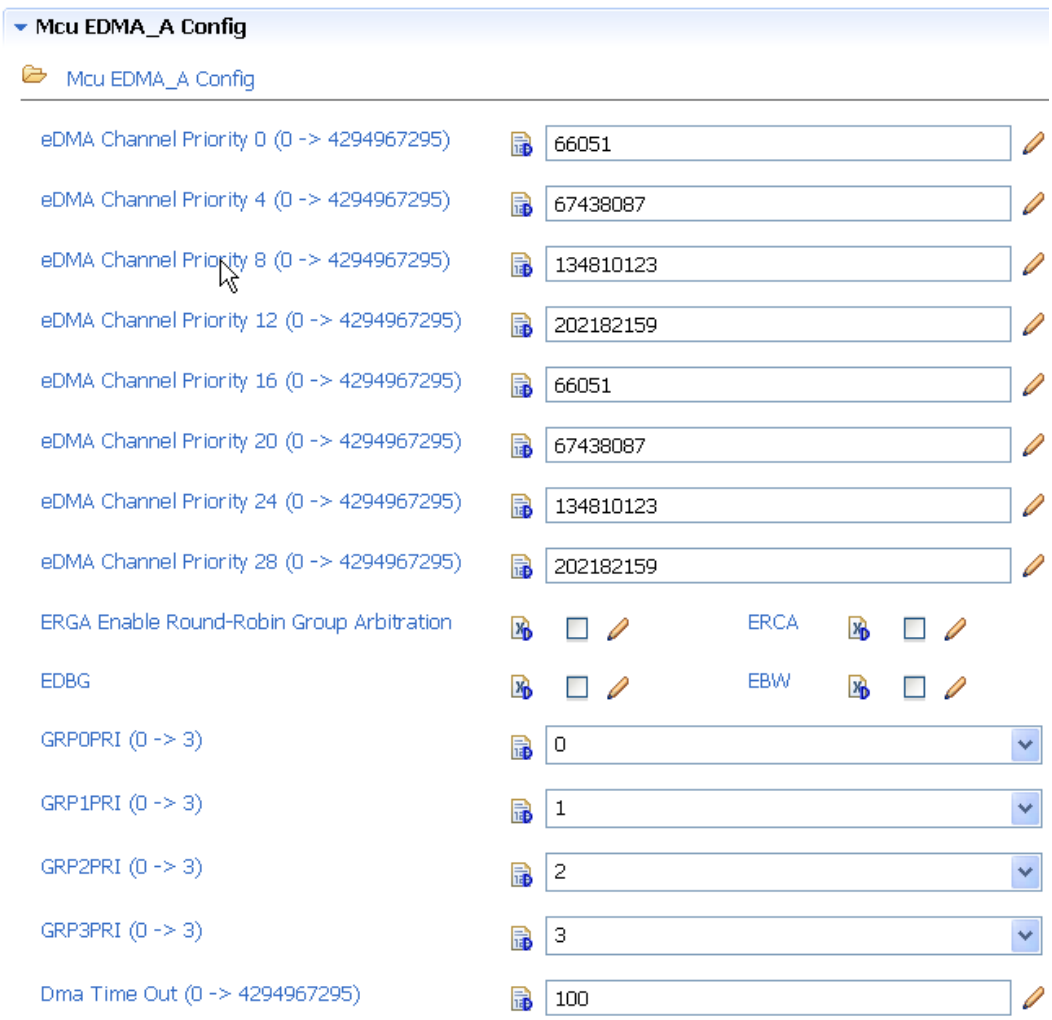



**Figure 4-14. Tresos Plugin snapshot for McuEDMA_A_Config form.**

#### 4.5.7.1 eDMAChannelPriority0 (McuEDMA_A_Config)

eDMA Channel Priority 0,1,2 and 3 register value (DCHPRI0 up to DCHPRI3). First byte corresponds to DCHPRI0 register. Example: 0x80010203UL corresponds to DCHPRI0=0x80 DCHPRI1=0x01 DCHPRI2=0x02 DCHPRI3=0x03

Note: each eDMA channel must have different priority.

DCHPRIn[0] represents ECP bit of (DCHPRIn) register and corresponds to Enable Channel Preemption.

**Table 4-71. Detailed description.**

| | |
|---|---|
| 0 | Channel n cannot be suspended by a higher priority channels service request. |
| 1 | Channel n can be temporarily suspended by the service request of a higher priority channel. |

DCHPRIn[2_3] represents GRPPRI bit of (DCHPRIn) register and corresponds to the priority assigned to the channel within the group of channels in which the channel resides when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. DCHPRIn[4_7] represents CHPRI bit of (DCHPRIn) register corresponds to channel priority. Value ranges from 0 to 15.

Note: Implementation specific Parameter.

**Table 4-72. Attribute eDMAChannelPriority0 (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | eDMA Channel Priority 0 |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 66051 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

## 4.5.7.2 eDMAChannelPriority4 (McuEDMA_A_Config)

eDMA Channel Priority 4,5,6 and 7 register value (DCHPRI4 up to DCHPRI7). First byte corresponds to DCHPRI0 register. Example: 0x80010203UL corresponds to DCHPRI4=0x80 DCHPRI5=0x01 DCHPRI6=0x02 DCHPRI7=0x03

Note: each eDMA channel must have different priority.

DCHPRIn[0] represents ECP bit of (DCHPRIn) register and corresponds to Enable Channel Preemption.

**Table 4-73. Detailed description.**

| | |
|---|---|
| 0 | Channel n cannot be suspended by a higher priority channels service request. |
| 1 | Channel n can be temporarily suspended by the service request of a higher priority channel. |

DCHPRIn[2_3] represents GRPPRI bit of (DCHPRIn) register and corresponds to the priority assigned to the channel within the group of channels in which the channel resides when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. DCHPRIn[4_7] represents CHPRI bit of (DCHPRIn) register corresponds to channel priority. Value ranges from 0 to 15.

Note: Implementation specific Parameter.

**Table 4-74. Attribute eDMAChannelPriority4 (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | eDMA Channel Priority 4 |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 67438087 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

### 4.5.7.3 eDMAChannelPriority8 (McuEDMA_A_Config)

eDMA Channel Priority 8,9,10 and 11 register value (DCHPRI8 up to DCHPRI11). First byte corresponds to DCHPRI0 register. Example: 0x80010203UL corresponds to DCHPRI8=0x80 DCHPRI9=0x01 DCHPRI10=0x02 DCHPRI11=0x03

Note: each eDMA channel must have different priority.

DCHPRIn[0] represents ECP bit of (DCHPRIn) register and corresponds to Enable Channel Preemption.

**Table 4-75. Detailed description.**

| | |
|---|---|
| 0 | Channel n cannot be suspended by a higher priority channels service request. |
| 1 | Channel n can be temporarily suspended by the service request of a higher priority channel. |

DCHPRIn[2_3] represents GRPPRI bit of (DCHPRIn) register and corresponds to the priority assigned to the channel within the group of channels in which the channel resides when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. DCHPRIn[4_7] represents CHPRI bit of (DCHPRIn) register corresponds to channel priority. Value ranges from 0 to 15.

Note: Implementation specific Parameter.

**Table 4-76. Attribute eDMAChannelPriority8 (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | eDMA Channel Priority 8 |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 134810123 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

## 4.5.7.4 eDMAChannelPriority12 (McuEDMA_A_Config)

eDMA Channel Priority 12,13,14 and 15 register value (DCHPRI12 up to DCHPRI5). First byte corresponds to DCHPRI0 register. Example: 0x80010203UL corresponds to DCHPRI12=0x80 DCHPRI13=0x01 DCHPRI14=0x02 DCHPRI15=0x03

Note: each eDMA channel must have different priority.

DCHPRIn[0] represents ECP bit of (DCHPRIn) register and corresponds to Enable Channel Preemption.

**Table 4-77. Detailed description.**

| | |
|---|---|
| 0 | Channel n cannot be suspended by a higher priority channels service request. |
| 1 | Channel n can be temporarily suspended by the service request of a higher priority channel. |

DCHPRIn[2_3] represents GRPPRI bit of (DCHPRIn) register and corresponds to the priority assigned to the channel within the group of channels in which the channel resides when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. DCHPRIn[4_7] represents CHPRI bit of (DCHPRIn) register corresponds to channel priority. Value ranges from 0 to 15.

Note: Implementation specific Parameter.

**Table 4-78. Attribute eDMAChannelPriority12 (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | eDMA Channel Priority 12 |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 202182159 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

## 4.5.7.5 eDMAChannelPriority16 (McuEDMA_A_Config)

eDMA Channel Priority 16,17,18 and 19 register value (DCHPRI16 up to DCHPRI19). First byte corresponds to DCHPRI0 register. Example: 0x80010203UL corresponds to DCHPRI16=0x80 DCHPRI17=0x01 DCHPRI18=0x02 DCHPRI19=0x03

Note: each eDMA channel must have different priority.

DCHPRIn[0] represents ECP bit of (DCHPRIn) register and corresponds to Enable Channel Preemption.

**Table 4-79. Detailed description.**

| | |
|---|---|
| 0 | Channel n cannot be suspended by a higher priority channels service request. |
| 1 | Channel n can be temporarily suspended by the service request of a higher priority channel. |

DCHPRIn[2_3] represents GRPPRI bit of (DCHPRIn) register and corresponds to the priority assigned to the channel within the group of channels in which the channel resides when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. DCHPRIn[4_7] represents CHPRI bit of (DCHPRIn) register corresponds to channel priority. Value ranges from 0 to 15.

Note: Implementation specific Parameter.

**Table 4-80. Attribute eDMAChannelPriority16 (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | eDMA Channel Priority 16 |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 66051 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

## 4.5.7.6 eDMAChannelPriority20 (McuEDMA_A_Config)

eDMA Channel Priority 20,21,22 and 23 register value (DCHPRI20 up to DCHPRI23). First byte corresponds to DCHPRI0 register. Example: 0x80010203UL corresponds to DCHPRI20=0x80 DCHPRI21=0x01 DCHPRI22=0x02 DCHPRI23=0x03

Note: each eDMA channel must have different priority.

DCHPRIn[0] represents ECP bit of (DCHPRIn) register and corresponds to Enable Channel Preemption.

**Table 4-81. Detailed description.**

| | |
|---|---|
| 0 | Channel n cannot be suspended by a higher priority channels service request. |
| 1 | Channel n can be temporarily suspended by the service request of a higher priority channel. |

DCHPRIn[2_3] represents GRPPRI bit of (DCHPRIn) register and corresponds to the priority assigned to the channel within the group of channels in which the channel resides when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. DCHPRIn[4_7] represents CHPRI bit of (DCHPRIn) register corresponds to channel priority. Value ranges from 0 to 15.

Note: Implementation specific Parameter.

**Table 4-82. Attribute eDMAChannelPriority20 (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | eDMA Channel Priority 20 |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 67438087 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

## 4.5.7.7 eDMAChannelPriority24 (McuEDMA_A_Config)

eDMA Channel Priority 24,25,26 and 27 register value (DCHPRI24 up to DCHPRI27). First byte corresponds to DCHPRI0 register. Example: 0x80010203UL corresponds to DCHPRI24=0x80 DCHPRI25=0x01 DCHPRI26=0x02 DCHPRI27=0x03

Note: each eDMA channel must have different priority.

DCHPRIn[0] represents ECP bit of (DCHPRIn) register and corresponds to Enable Channel Preemption.

**Table 4-83. Detailed description.**

| | |
|---|---|
| 0 | Channel n cannot be suspended by a higher priority channels service request. |
| 1 | Channel n can be temporarily suspended by the service request of a higher priority channel. |

DCHPRIn[2_3] represents GRPPRI bit of (DCHPRIn) register and corresponds to the priority assigned to the channel within the group of channels in which the channel resides when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. DCHPRIn[4_7] represents CHPRI bit of (DCHPRIn) register corresponds to channel priority. Value ranges from 0 to 15.

Note: Implementation specific Parameter.

**Table 4-84. Attribute eDMAChannelPriority24 (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | eDMA Channel Priority 24 |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 134810123 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

## 4.5.7.8 eDMAChannelPriority28 (McuEDMA_A_Config)

eDMA Channel Priority 28,29,30 and 31 register value (DCHPRI28 up to DCHPRI31). First byte corresponds to DCHPRI0 register. Example: 0x80010203UL corresponds to DCHPRI28=0x80 DCHPRI29=0x01 DCHPRI30=0x02 DCHPRI31=0x03

Note: each eDMA channel must have different priority.

DCHPRIn[0] represents ECP bit of (DCHPRIn) register and corresponds to Enable Channel Preemption.

**Table 4-85. Detailed description.**

| | |
|---|---|
| 0 | Channel n cannot be suspended by a higher priority channels service request. |
| 1 | Channel n can be temporarily suspended by the service request of a higher priority channel. |

DCHPRIn[2_3] represents GRPPRI bit of (DCHPRIn) register and corresponds to the priority assigned to the channel within the group of channels in which the channel resides when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. DCHPRIn[4_7] represents CHPRI bit of (DCHPRIn) register corresponds to channel priority. Value ranges from 0 to 15.

Note: Implementation specific Parameter.

**Table 4-86. Attribute eDMAChannelPriority28 (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | eDMA Channel Priority 28 |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 202182159 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

## 4.5.7.9 ERGA (McuEDMA_A_Config)

Enable Round-Robin Group Arbitration.

**Table 4-87. Detailed description.**

| | |
|---|---|
| 0 | Fixed-priority arbitration is used for selection among the groups. |
| 1 | Round-robin arbitration is used for selection among the groups. |

Note: Implementation specific Parameter.

**Table 4-88. Attribute ERGA (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | ERGA Enable Round-Robin Group Arbitration |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.5.7.10 ERCA (McuEDMA_A_Config)

Enable Round-Robin Channel Arbitration.

**Table 4-89. Detailed description.**

| | |
|---|---|
| 0 | Fixed-priority arbitration is used for channel selection. |
| 1 | Round-robin arbitration is used for channel selection |

Note: Implementation specific Parameter.

**Table 4-90. Attribute ERCA (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | ERCA |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.5.7.11 EDBG (McuEDMA_A_Config)

Enable Debug.

**Table 4-91. Detailed description.**

| | |
|---|---|
| 0 | The assertion of the system debug control input is ignored. |

*Table continues on the next page...*

**Table 4-91. Detailed description. (continued)**

| | |
|---|---|
| 1 | The assertion of the system debug control input causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution will resume when either the system debug control input is negated or the EDBG bit is cleared. |

Note: Implementation specific Parameter.

**Table 4-92. Attribute EDBG (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | EDBG |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.7.12 EBW (McuEDMA_A_Config)

Enable Buffered Writes.

**Table 4-93. Detailed description.**

| | |
|---|---|
| 0 | The bufferable write signal (hprot[2]) is not asserted during AMBA AHB writes. |
| 1 | The bufferable write signal (hprot[2]) is asserted on all AMBA AHB writes except for the last write sequence. |

Note: Implementation specific Parameter.

**Table 4-94. Attribute EBW (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | EBW |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.7.13 GRP0PRI (McuEDMA_A_Config)

**Note**

Group 0 priority level when fixed priority group arbitration is enabled.

Implementation specific Parameter.

**Table 4-95. Attribute GRP0PRI (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 0 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=3<br>>=0 |

### 4.5.7.14 GRP1PRI (McuEDMA_A_Config)

**Note**

Group 1 priority level when fixed priority group arbitration is enabled.

Implementation specific Parameter.

**Table 4-96. Attribute GRP1PRI (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 1 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=3<br>>=0 |

### 4.5.7.15 GRP2PRI (McuEDMA_A_Config)

**Note**

Group 2 priority level when fixed priority group arbitration is enabled.

Implementation specific Parameter.

**Table 4-97. Attribute GRP2PRI (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 2 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=3<br>>=0 |

### 4.5.7.16 GRP3PRI (McuEDMA_A_Config)

**Note**

Group 3 priority level when fixed priority group arbitration is enabled.

Implementation specific Parameter.

**Table 4-98. Attribute GRP3PRI (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 3 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=3<br>>=0 |

### 4.5.7.17 DmaTimeOut (McuEDMA_A_Config)

Software Delay needs to wait till current transfers completion. The software delay is used in the API McuDMASetMode(). When the API McuDMASetMode(Disabled) is called then the Current DMA job is checked for completion of the job else the DMA timeout is verified and the DMA operaion is disabled. Note: Implementation specific Parameter.

**Table 4-99. Attribute DmaTimeOut (McuEDMA_A_Config) detailed description**

| Property | Value |
|---|---|
| Label | Dma Time Out |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 100 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

## 4.5.8 Form McuFlashBIUCR

The BIU configuration register is used to specify operation of the dual-Flash controller. Note: Implementation specific Container.

**Is included by form :** Form McuModuleConfiguration

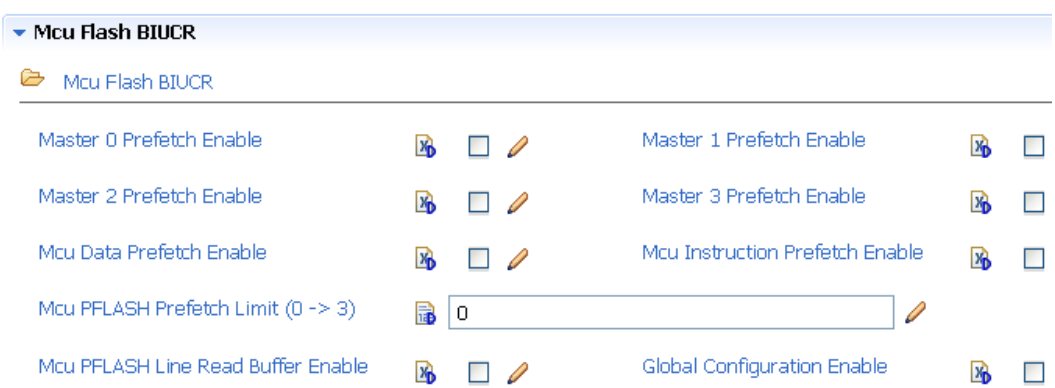



**Figure 4-15. Tresos Plugin snapshot for McuFlashBIUCR form.**

### 4.5.8.1 Master0PrefetchEnable (McuFlashBIUCR)

Prefetching Master ID 0 enabled/disabled. Note: Implementation specific Container.

**Table 4-100. Attribute Master0PrefetchEnable (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Master 0 Prefetch Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.8.2 Master1PrefetchEnable (McuFlashBIUCR)

Prefetching Master ID 1 enabled/disabled. Note: Implementation specific Container.

**Table 4-101. Attribute Master1PrefetchEnable (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Master 1 Prefetch Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.8.3 Master2PrefetchEnable (McuFlashBIUCR)

Prefetching Master ID 2 enabled/disabled. Note: Implementation specific Container.

**Table 4-102. Attribute Master2PrefetchEnable (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Master 2 Prefetch Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.8.4 Master3PrefetchEnable (McuFlashBIUCR)

This field is not implemented and has no effect.

**Table 4-103. Attribute Master3PrefetchEnable (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Master 3 Prefetch Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.8.5 McuDataPrefetchEnable (McuFlashBIUCR)

Data Prefetch Enable/Disable.

**Table 4-104. Detailed description.**

| | |
|---|---|
| Enable: | Prefetching may be triggered by any data read access |
| Disable: | No prefetching is triggered by a data read access |

Note: Implementation specific Container.

**Table 4-105. Attribute McuDataPrefetchEnable (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Data Prefetch Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.8.6 McuInstructionPrefetchEnable (McuFlashBIUCR)

Instruction Prefetch Enable.

**Table 4-106. Detailed description.**

| | |
|---|---|
| Enable: | Prefetching may be triggered by any instruction read access |
| Disable: | No prefetching is triggered by an instruction read access |

Note: Implementation specific Container.

**Table 4-107. Attribute McuInstructionPrefetchEnable (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Instruction Prefetch Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.8.7 McuPFLASHPrefetchLimit (McuFlashBIUCR)

PFLASH Prefetch Limit. This field controls the prefetch algorithm used by the PFLASH prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset.

**Table 4-108. Detailed description.**

| | |
|---|---|
| 00 | No prefetching or buffering is performed. |
| 01 | The referenced line is prefetched on a buffer miss, that is, prefetch on miss. |
| 1x | The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. |

Note: Implementation specific Container.

**Table 4-109. Attribute McuPFLASHPrefetchLimit (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Mcu PFLASH Prefetch Limit |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 0 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.8.8 McuPFLASHLineReadBuffersEnable (McuFlashBIUCR)

PFLASH Line Read Buffers Enable.

**Table 4-110. Detailed description.**

| | |
|---|---|
| Disable: | The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. |
| Enable: | The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled. |

Note: Implementation specific Container.

**Table 4-111. Attribute McuPFLASHLineReadBuffersEnable (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Mcu PFLASH Line Read Buffer Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.8.9 GlobalConfigurationEnable (McuFlashBIUCR)

This bit determines if specific fields contained in BIUCR apply globally to the PFLASH configuration or apply only to the Bank0 configuration. In particular, this applies to BIUCR[DPFEN, IPFEN, PFLIM, BFEN]. This bit is cleared by reset.

**Table 4-112. Detailed description.**

| | |
|---|---|
| 0 | Use the contents of BIUCR to configure Bank0 operation, PFCR3 to configure bank1 operation |
| 1 | Use the contents of BIUCR to configure both Bank0 and Bank1 operation |

**Table 4-113. Attribute GlobalConfigurationEnable (McuFlashBIUCR) detailed description**

| Property | Value |
|---|---|
| Label | Global Configuration Enable |
| Type | BOOLEAN |
| Origin | Custom |
| Symbolic Name | false |
| Default | false |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.5.9 Form McuFlashBIUAPR

Bus Interface Unit Access Protection Register. Note: Implementation specific Container.

**Is included by form :** Form McuModuleConfiguration

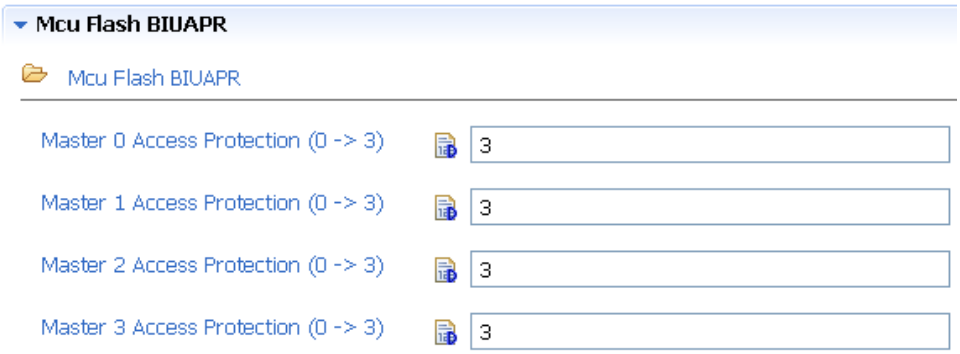



**Figure 4-16. Tresos Plugin snapshot for McuFlashBIUAPR form.**

### 4.5.9.1 Master0AccessProtection (McuFlashBIUAPR)

Master 0 Access Protection These fields are used to control whether read and write accesses to the Flash are allowed based on the master ID of a requesting master (processor core).

**Table 4-114. Detailed description.**

| | |
|---|---|
| 00 | No accesses may be performed by this master. |
| 01 | Only read accesses may be performed by this master. |
| 10 | Only write accesses may be performed by this master. |
| 11 | Both read and write accesses may be performed by this master. |

Note: Implementation specific Container.

**Table 4-115. Attribute Master0AccessProtection (McuFlashBIUAPR) detailed description**

| Property | Value |
|---|---|
| Label | Master 0 Access Protection |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 3 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.9.2 Master1AccessProtection (McuFlashBIUAPR)

Master 1 Access Protection These fields are used to control whether read and write accesses to the Flash are allowed based on the master ID of a requesting master (Nexus Development Interface).

**Table 4-116. Detailed description.**

| | |
|---|---|
| 00 | No accesses may be performed by this master. |
| 01 | Only read accesses may be performed by this master. |
| 10 | Only write accesses may be performed by this master. |
| 11 | Both read and write accesses may be performed by this master. |

Note: Implementation specific Container.

**Table 4-117. Attribute Master1AccessProtection (McuFlashBIUAPR) detailed description**

| Property | Value |
|---|---|
| Label | Master 1 Access Protection |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 3 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.9.3 Master2AccessProtection (McuFlashBIUAPR)

Master 2 Access Protection These fields are used to control whether read and write accesses to the Flash are allowed based on the master ID of a requesting master (DMA).

**Table 4-118. Detailed description.**

| | |
|---|---|
| 00 | No accesses may be performed by this master. |
| 01 | Only read accesses may be performed by this master. |
| 10 | Only write accesses may be performed by this master. |
| 11 | Both read and write accesses may be performed by this master. |

Note: Implementation specific Container.

**Table 4-119. Attribute Master2AccessProtection (McuFlashBIUAPR) detailed description**

| Property | Value |
|---|---|
| Label | Master 2 Access Protection |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 3 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.9.4 Master3AccessProtection (McuFlashBIUAPR)

This field is not implemented; it has no effect.

**Table 4-120. Attribute Master3AccessProtection (McuFlashBIUAPR) detailed description**

| Property | Value |
|---|---|
| Label | Master 3 Access Protection |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 3 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.5.10 Form McuFlashBIUCR2

Bus Interface Unit Configuration Register 2 (BIUCR2). Note: Implementation specific Container.

**Is included by form :** Form McuModuleConfiguration

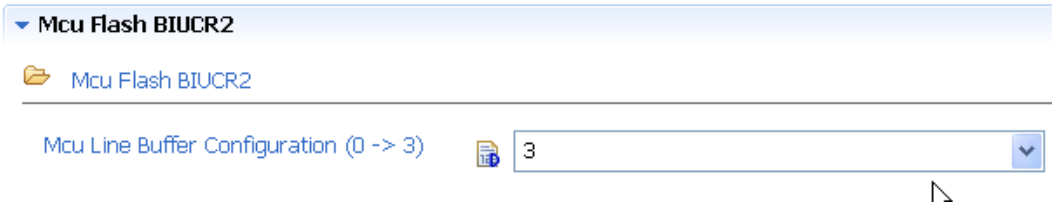



**Figure 4-17. Tresos Plugin snapshot for McuFlashBIUCR2 form.**

### 4.5.10.1 McuLineBufferConfiguration (McuFlashBIUCR2)

This field controls the configuration of all the line buffers in the PFLASH controller. This field globally controls the configuration of the four line buffers in the PFLASH controller. The buffers can be organized as a pool of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, it is allocated to the least-recently-used buffer within the group and the just-fetched entry then marked as most-recently-used. If the flash access is for the next-sequential line, the buffer is not marked as most-recently-used until the given address produces a buffer hit.

**Table 4-121. Detailed description.**

| | |
|---|---|
| 00 | All four buffers are available for any flash access, that is, there is no partitioning based on the access type. |
| 01 | Reserved |
| 10 | The buffers are partitioned into two groups with buffers 0 and 1 allocated for instruction fetches and buffers 2 and 3 for data accesses. |
| 11 | The buffers are partitioned into two groups with buffers 0,1,2 allocated for instruction fetches and buffer 3 for data accesses. |

**Table 4-122. Attribute McuLineBufferConfiguration (McuFlashBIUCR2) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Line Buffer Configuration |
| Type | INTEGER |
| Origin | Custom |
| Symbolic Name | false |
| Default | 3 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

## 4.5.11 Form McuResetSource

This parameter selects the type of the reset performed.

**Is included by form :** Form McuModuleConfiguration

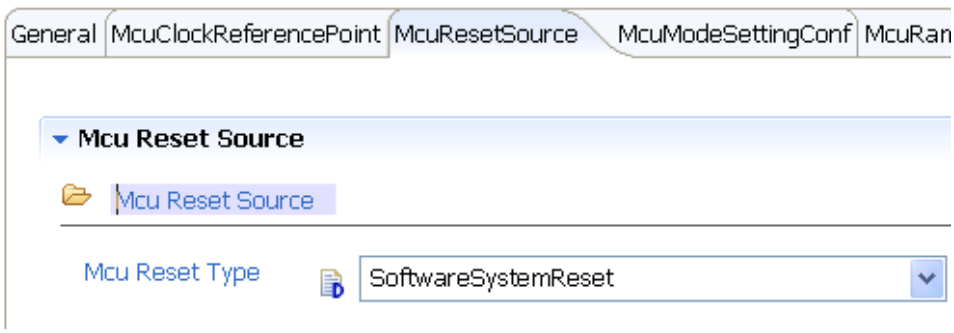



**Figure 4-18. Tresos Plugin snapshot for McuResetSource form.**

### 4.5.11.1 McuResetType (McuResetSource)

**Table 4-123. Attribute McuResetType (McuResetSource) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Reset Type |
| Type | ENUMERATION |
| Origin | Custom |
| Symbolic Name | false |
| Default | SoftwareSystemReset |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Range | SoftwareSystemReset<br>SoftwareExternalReset<br>CheckstopReset |

## 4.5.12 Form McuModeSettingConf

This container contains the configuration (parameters) for the Mode setting of the MCU. Please see MCU035 for more information on the MCU mode settings.

**Is included by form :** Form McuModuleConfiguration

**Included forms :**

- Form HaltRegister

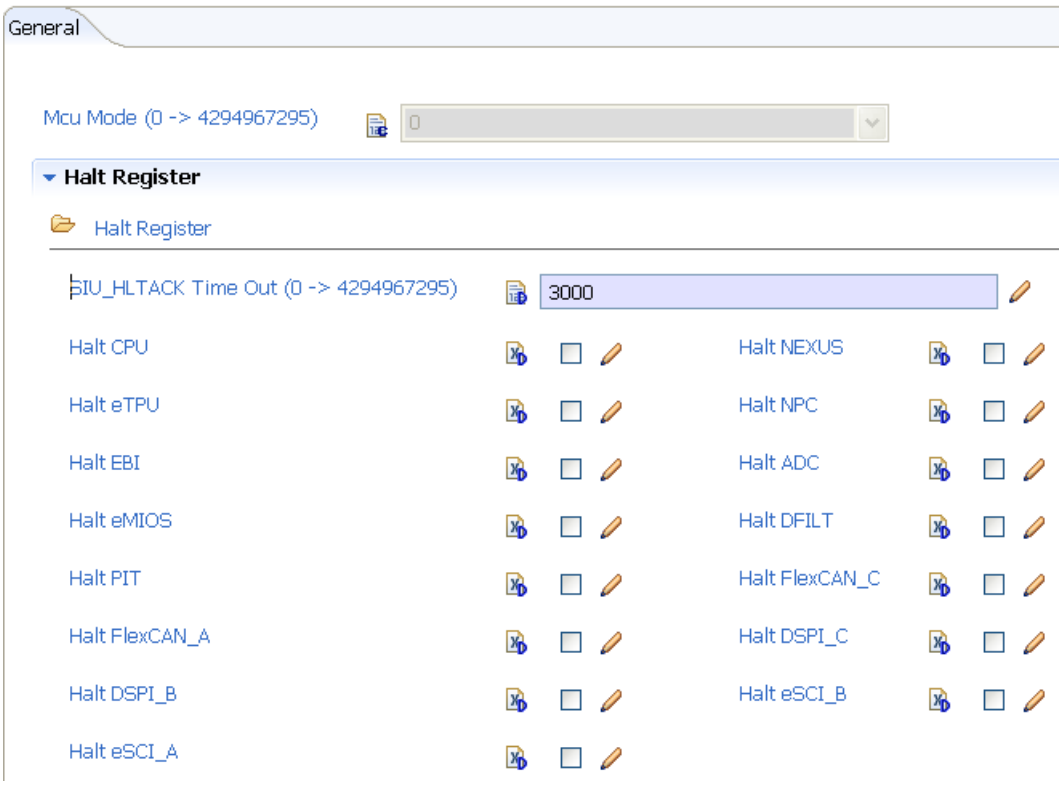



**Figure 4-19. Tresos Plugin snapshot for McuModeSettingConf form.**

### 4.5.12.1 McuMode (McuModeSettingConf)

This parameter shall represent the number of MCU modes available for the MCU. Note: This parameter is not used by the current Implementation

**Table 4-124. Attribute McuMode (McuModeSettingConf) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Mode |
| Type | INTEGER |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | true |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.12.2 Form HaltRegister

This container contains the SPECIFIC configuration (parameters) of the HaltRegister Configuration. Note:Implementation specific Container.</html>

**Is included by form :** Form McuModeSettingConf

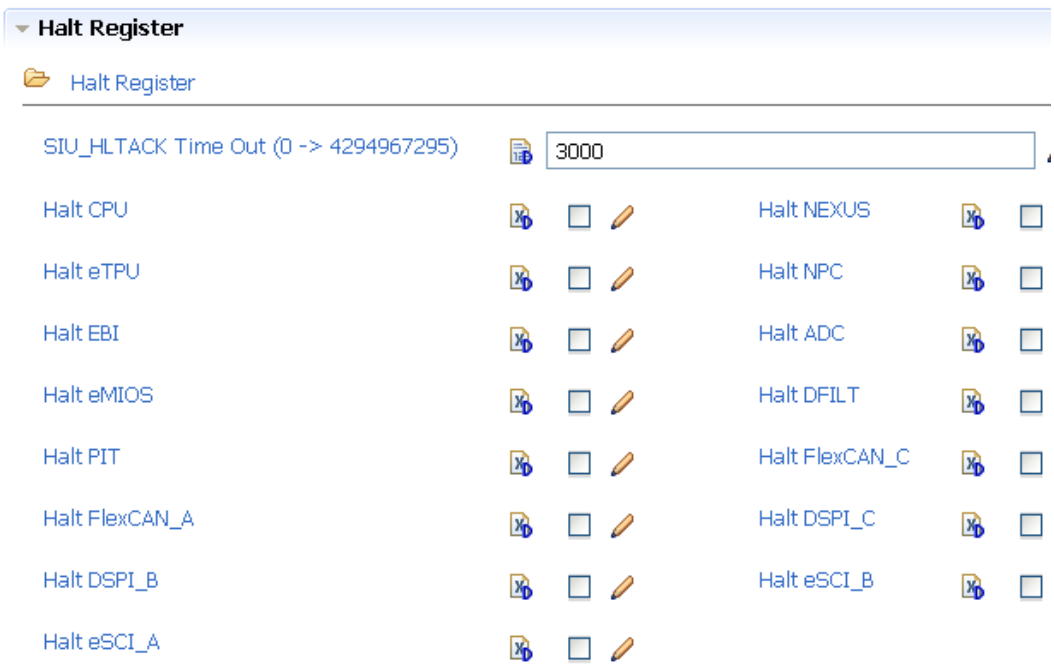



**Figure 4-20. Tresos Plugin snapshot for HaltRegister form.**

#### 4.5.12.2.1 SIU_HLTACKTimeOut (HaltRegister)

Timeout value to wait in Mcu_InitHalt function until the clock to peripheral modules is successfully halted. The unit of the timeout value is number of passes through wait loop. Note: Implementation specific Parameter.

**Table 4-125. Attribute SIU_HLTACKTimeOut (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | SIU_HLTACK Time Out |
| Default | 3000 |

#### 4.5.12.2.2 HaltCPU (HaltRegister)

**Table 4-126. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-127. Attribute HaltCPU (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt CPU |
| Default | false |

#### 4.5.12.2.3 HaltNDEDI (HaltRegister)

**Table 4-128. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-129. Attribute HaltNDEDI (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt NEXUS |
| Default | false |

#### 4.5.12.2.4 HalteTPU (HaltRegister)

**Table 4-130. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-131. Attribute HalteTPU (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt eTPU |
| Default | false |

#### 4.5.12.2.5 HaltNPC (HaltRegister)

**Table 4-132. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-133. Attribute HaltNPC (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt NPC |
| Default | false |

#### 4.5.12.2.6 HaltEBI (HaltRegister)

**Table 4-134. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-135. Attribute HaltEBI (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt EBI |
| Default | false |

#### 4.5.12.2.7 HaltADC (HaltRegister)

**Table 4-136. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-137. Attribute HaltADC (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt ADC |
| Default | false |

#### 4.5.12.2.8 HalteMIOS (HaltRegister)

**Table 4-138. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-139. Attribute HalteMIOS (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt eMIOS |
| Default | false |

#### 4.5.12.2.9 HaltDFILT (HaltRegister)

**Table 4-140. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-141. Attribute HaltDFILT (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt DFILT |
| Default | false |

#### 4.5.12.2.10 HaltPIT (HaltRegister)

**Table 4-142. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-143. Attribute HaltPIT (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt PIT |
| Default | false |

#### 4.5.12.2.11 HaltFlexCAN_C (HaltRegister)

**Table 4-144. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-145. Attribute HaltFlexCAN_C (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt FlexCAN_C |
| Default | false |

#### 4.5.12.2.12 HaltFlexCAN_A (HaltRegister)

**Table 4-146. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-147. Attribute HaltFlexCAN_A (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt FlexCAN_A |
| Default | false |

#### 4.5.12.2.13 HaltDSPI_C (HaltRegister)

**Table 4-148. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-149. Attribute HaltDSPI_C (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt DSPI_C |
| Default | false |

#### 4.5.12.2.14 HaltDSPI_B (HaltRegister)

**Table 4-150. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-151. Attribute HaltDSPI_B (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt DSPI_B |
| Default | false |

#### 4.5.12.2.15 HalteSCI_B (HaltRegister)

**Table 4-152. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-153. Attribute HalteSCI_B (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt eSCI_B |
| Default | false |

#### 4.5.12.2.16 HalteSCI_A (HaltRegister)

**Table 4-154. Detailed description.**

| | |
|---|---|
| 0 | Normal operation |
| 1 | Stop mode request |

(bit in SIU_HLT register). Note: Implementation specific Parameter.

**Table 4-155. Attribute HalteSCI_A (HaltRegister) detailed description**

| Property | Value |
|---|---|
| Label | Halt eSCI_A |
| Default | false |

## 4.5.13 Form McuRamSectorSettingConf

This container contains the configuration (parameters) for the RAM Sector setting. Please see MCU030 for more information on RAM sec-tor settings.

**Is included by form :** Form McuModuleConfiguration

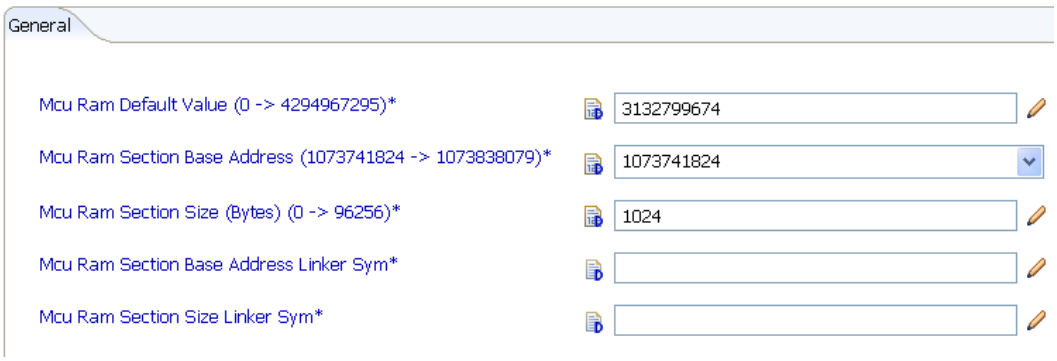



**Figure 4-21. Tresos Plugin snapshot for McuRamSectorSettingConf form.**

### 4.5.13.1 McuRamDefaultValue (McuRamSectorSettingConf)

This parameter shall represent the Data pre-setting to be initialized

**Table 4-156. Attribute McuRamDefaultValue (McuRamSectorSettingConf) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Ram Default Value |
| Type | INTEGER |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 3132799674 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=4294967295<br>>=0 |

### 4.5.13.2 McuRamSectionBaseAddress (McuRamSectorSettingConf)

This parameter represents the RAM section base address. The address must be aligned to 4 bytes. Table for ranges of different derivatives

**Table 4-157. Detailed description.**

| | |
|---|---|
| 1.5MB Flash Memory Device | 0x4000_0000 to 0x4001_77FF |

**Table 4-158. Attribute McuRamSectionBaseAddress (McuRamSectorSettingConf) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Ram Section Base Address |
| Type | INTEGER |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 1073741824 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | Range<br><=1073838079<br>>=1073741824 |

### 4.5.13.3 McuRamSectionSize (McuRamSectorSettingConf)

This parameter represents the RAM section size in bytes. The size must be multiple of 4. Table for size of different derivatives

**Table 4-159. Detailed description.**

| | |
|---|---|
| 1.5MB Flash Memory Device | 96256(94K)Bytes |

**Table 4-160. Attribute McuRamSectionSize (McuRamSectorSettingConf) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Ram Section Size (Bytes) |
| Type | INTEGER |
| Origin | AUTOSAR_ECUC |
| Symbolic Name | false |
| Default | 1024 |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |
| Invalid | XPath Range<br><=96256<br>>=0 |

### 4.5.13.4 McuRamSectionBaseAddrLinkerSym (McuRamSectorSettingConf)

This parameter represents the RAM section base address. The address must be aligned to 4 bytes. Note:If this parameter is empty, then the integer values from McuRamSectionBaseAddress will be used.Note: Implementation specific Parameter.

**Table 4-161. Attribute McuRamSectionBaseAddrLinkerSym (McuRamSectorSettingConf) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Ram Section Base Address Linker Sym |
| Type | STRING |
| Origin | Custom |
| Symbolic Name | false |
| Default | |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

### 4.5.13.5 McuRamSectionSizeLinkerSym (McuRamSectorSettingConf)

This parameter represents the RAM section size in bytes. The size must be multiple of 4. Note:If this parameter is empty, then the integer values from McuRamSectionSize will be used.Note: Implementation specific Parameter.

**Table 4-162. Attribute McuRamSectionSizeLinkerSym (McuRamSectorSettingConf) detailed description**

| Property | Value |
|---|---|
| Label | Mcu Ram Section Size Linker Sym |
| Type | STRING |
| Origin | Custom |
| Symbolic Name | false |
| Default | |
| Lower Multiplicity | 1 |
| Upper Multiplicity | 1 |

# 4.6 Frequency-Modulated Phase-Locked Loop (FMPLL) parameters configuration

This chapter provides further details about the configuration of the parameters of the FMPLL module implemented in the device.

The FMPLL enables the generation of high speed system clocks from a common 4-20 MHz input clock. Further, the FMPLL supports programmable frequency modulation of the system clock. The FMPLL multiplication factor, input clock divider and output clock divider ratio are all software configurable.

The parameters can be configured in the plugin as shown in the figure:

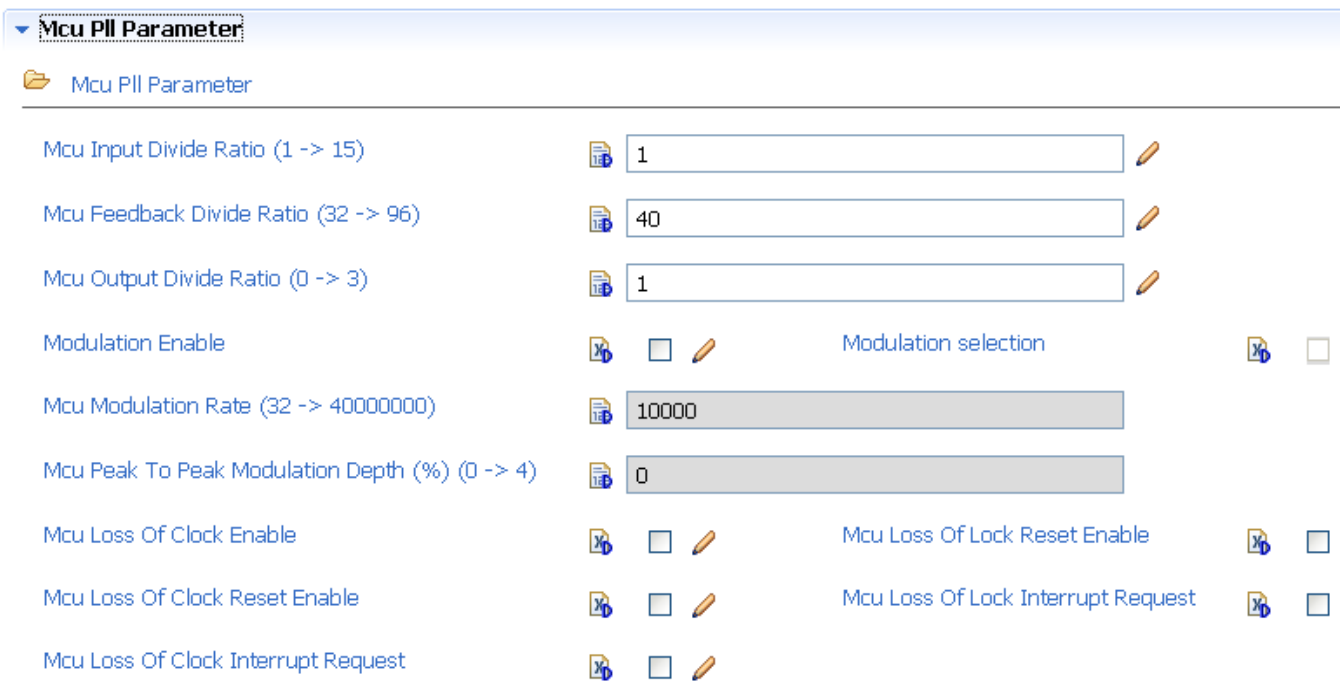



**Figure 4-22. Tresos Plugin snapshot of FMPLL configuration settings form**

When FMPLL is in lock state, the FMPLL output clock is derived by the reference clock through the following relation:

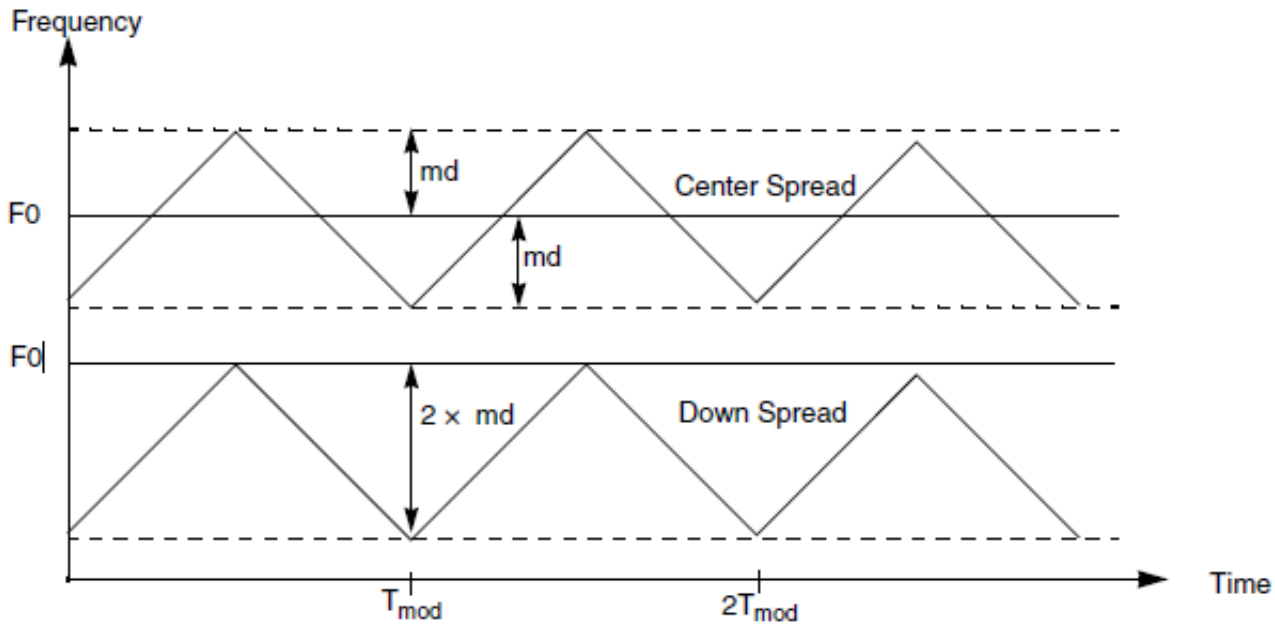
$$f_{sys} = f_{ref} \times \frac{EMFD}{(EPREDIV + 1) \times 2^{(ERFD + 1)}}$$

where *fsys* is the PLL clock frequency, *Fref* is the PLL reference clock frequency, *EMFD* is Mcu Feedback Divide Ratio, *EPREDIV* is Mcu Input Divide Ratio and *ERFD* is the Mcu Output Divide Ratio. **NOTE:** the inner VCO (Voltage Controlled Oscillator) output frequency, whose value is given by *fsys x 2^(ERFD+1)*, should reside in the range 256 MHz to 512 MHz.

The table below shows some examples of parameter configuration, according to different values of reference clock frequency:

| **PLL Reference Frequency** (MHz) | **PLL0 Frequency** (MHz) | **Output divide ratio** (0, 1, 2, 3) | **Input divide ratio** (1..15) | **Feedback divide ratio** (32..96) | **VCO frequency** (MHz) |
|---|---|---|---|---|---|
| 8 | 32 | 2 | 1 | 32 | 256 |
| 8 | 64 | 2 | 1 | 64 | 512 |
| 8 | 80 | 1 | 1 | 40 | 320 |
| 16 | 32 | 2 | 2 | 32 | 256 |
| 16 | 64 | 2 | 2 | 64 | 512 |
| 16 | 80 | 1 | 2 | 40 | 320 |

The user may enable the frequency modulation of the PLL by checking the *Modulation Enables*witch. When the frequency modulation is switched on, the carrier wave represented by the PLL output, varies its instanteneous frequency according to the modulation waveform which is always a triangle wave whose parameters are programmable. In particular the desired level of modulation is achieved through three parameters: the *modulation rate*, the *peak to peak modulation depth* and the *modulation selection*. The *modulation rate* and *depth* represent respectively the frequency and the peak modulation depth in percentage of the triangle waveform. The *module selection* specifies if the frequency of the carrier spreads around the original unmodulated PLL frequency F0 (*Center spread*) or keeps below F0 (*down spread*) that becomes the maximum value (see figure below). In the first case the *modulation depth* corresponds to the peak value of the triangle wave measured with respect to the mean value that is equal to F0. Instead with reference to the *Down spread*, the *modulation depth* represents half the peak-to-peak amplitude of the modulation signal whose maximum value becomes F0.



### *How to Reach Us:*

**Home Page:**
www.freescale.com

**Web Support:**
http://www.freescale.com/support

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com





Document Number: UM14MCUASR3.0R2.0.0
Rev. 1.1