

MCU

Module Introduction DOCUMENT

**Project Code:<Code of the project>**

**Document Code: <07ae-BM/PM/HDCV/FSOFT> – <1.4>**

RECORD OF CHANGE

| No | Effective Date | Version | Change Description | Reason | Reviewer | Approver |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 01/07/2017 | 1.0 | Create | N/A |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |

**TABLE OF CONTENTS**

1. Introduction 6

1.1 Purpose 6

1.2 Definitions, Acronyms and Abbreviations 6

1.3 References 7

1.4 Overview 7

2. COMMON package and mechanism 8

2.1 Protocol introduction 8

2.2 Module introduction 8

2.3 Hardware (IPs) introduction 8

2.3.1 IPV\_AFE 11

2.3.2 IPV\_CORTEXM 12

2.3.3 IPV\_CMU 12

2.3.4 IPV\_DFS 13

2.3.5 IPV\_EMIOS 14

2.3.6 IPV\_FIRC 15

2.3.7 IPV\_FXOSC 15

2.3.8 IPV\_FLASHV2 15

2.3.9 IPV\_LPU 16

2.3.10 IPV\_MCV4 16

2.3.11 IPV\_PCC 20

2.3.12 IPV\_PLLDIG 20

2.3.13 IPV\_PMC 21

2.3.14 IPV\_PMCDIG 22

2.3.15 IPV\_PRAM 22

2.3.16 IPV\_RCM 23

2.3.17 IPV\_SCG 24

2.3.18 IPV\_SIM 24

2.3.19 IPV\_SIRC 25

2.3.20 IPV\_SIUL2 25

2.3.21 IPV\_SMC 25

2.3.22 IPV\_SSCM 26

2.3.23 IPV\_STCUV2 26

2.3.24 IPV\_SXOSC 27

2.3.25 IPV\_XOSC 27

2.4 Implementation 27

1.4.1 Mcu\_Init 27

1.4.2 Mcu\_InitRamSection 36

1.4.3 Mcu\_InitClock 39

1.4.4 Mcu\_GetPllStatus 49

1.4.5 Mcu\_DistributePllClock 53

1.4.6 Mcu\_GetResetReason 56

1.4.7 Mcu\_GetResetRawValue 59

1.4.8 Mcu\_PerformReset 62

1.4.9 Mcu\_SetMode 65

1.4.10 Mcu\_GetRamState 68

1.4.11 Mcu\_GetVersionInfo 71

1.4.12 Mcu\_GetMidrStructure 71

1.4.13 Mcu\_GetPeripheralState 74

1.4.14 Mcu\_GetSystemState 77

1.4.15 Mcu\_SscmGetMemConfig 80

1.4.16 Mcu\_SscmGetStatus 82

1.4.17 Mcu\_SscmGetUops 84

1.4.18 Mcu\_GetPowerModeState 86

1.4.19 Mcu\_GetPowerDomainState 87

1.4.20 Mcu\_EmiosConfigureGpren 90

1.4.21 Mcu\_Disable\_CMU 91

2.5 TIPS &Tricks 93

2.6 Q&A 93

3. Appendix 94

# Introduction

## Purpose

Tài liệu training về MCU module cung cấp kiến thức cấu trúc phần cứng cũng như cách thực hiện xây dựng driver theo chuẩn AUTOSAR.

Đối tượng mà tài liệu này hướng đến là những Developer và Validator có kinh nghiệm làm việc với các module khác của team MCAL\_Development\_Standard, có nền tảng hiểu biết về các mảng kiến thức sau đây:

- EUnit, BEART (Tài liệu tham khảo: BEART and EUnit User Manual).

- UML design.

- Kiến thức basic về các nền tảng phần cứng ARM, PPC.

- Kiến thức lập trình C, coding convention (Misra, Compiler warning…).

- Đọc, hiểu các requirement Autosar, Non-Autosar.

- Cấu trúc các file code driver và thư mục test.

- …

Vì MCU có khá nhiều IPVs và quản lý nhiều HW module, vậy nên song song với đọc tài liệu này người đọc nên tập trung Code và UML design tương ứng với mỗi phần được trình bày trong tài liệu để có cái nhìn chi tiết hơn.

## Definitions, Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| **Abbreviations** | **Description** | **Comment** |
| MCU | Micro Controller Unit | It means “not decided yet” |
| UML | Unified Modeling Language | Ngôn ngữ mô hình hóa thống nhất |
| MAF | Measurement Automation Framework | Hệ thống đo lường tự động |
| IPV | IP Vault |  |
| IPW | IP Wrapper |  |
| HLD | High Layer Driver |  |
| AUTOSAR | Automotive Open System Architecture |  |
| MCAL | MicroController Abstraction Layer |  |
| RM | Reference Manual | Specific for each platform |
| UM | User Manual | Specific for each platform |
| IM | Integration Manual | Specific for each platform |

## References

|  |  |
| --- | --- |
| **Document Number** | **Title** |
| 1 | Specification of MCU Driver V3.2.0 R4.0 Rev 3 |
| 2 | Specification of MCU Driver AUTOSAR Release 4.2.2 |
| 3 | Reference Manual of all platforms |
| 4 | UM of all platforms |
| 5 | IM of all platforms |

## Overview

*Tài liệu này giới thiệu về MCU module bao gồm các nội dung về Scope của MCU module trong AUTOSAR standard. Giới thiệu về HW của MCU và cách thức triển khai driver theo chuẩn AUTOSAR theo các feature mà HW cung cấp.*

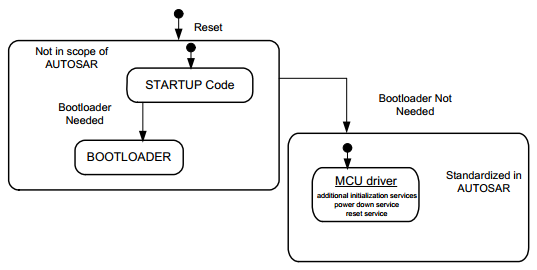
# COMMON package and mechanism

## Protocol introduction

None

## Module introduction

The MCU driver accesses the microcontrollerhardware directly and is located in the Microcontroller Abstraction Layer (MCAL). The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The initialization services allow a flexible and application related MCU initialization in addition to the start-up code (see figure below).



MCU driver Features:

* Initialization of MCU clock, PLL, clock prescalers and MCU clock distribution
* Initialization of RAM sections
* Activation of µC reduced power modes
* Activation of a µC reset
* Provides a service to get the reset reason from hardware

## Hardware (IPs) introduction

MCU has a lot of IPs and each platform will use some of them. You can see that with the following table:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **IPs** | **Cobra55** | **Halo** | **Materhorn** | **Rainier** | **Calypso** | **RaceRunnerUltra** | **Kinetis** | **TreeRunner** |
| **IPV\_AFE** |  |  |  |  |  | **X** |  |  |
| **IPV\_CORTEXM** |  |  |  |  |  |  | **X** |  |
| **IPV\_CMU** | **X** | **X** | **X** | **X** | **X** | **X** |  | **X** |
| **IPV\_DFS** |  |  |  |  |  |  |  | **X** |
| **IPV\_EMIOS** | **X** |  |  | **X** | **X** |  |  |  |
| **IPV\_FIRC** |  | **X** |  |  | **X** |  |  |  |
| **IPV\_FXOSC** |  | **X** |  |  | **X** |  |  | **X** |
| **IPV\_FLASHV2** | **X** | **X** | **X** | **X** | **X** | **X** |  |  |
| **IPV\_LPU** |  |  |  |  | **X** |  |  |  |
| **IPV\_MCV4** | **X** | **X** | **X** | **X** | **X** | **X** |  | **X** |
| **IPV\_PCC** |  |  |  |  |  |  | **X** |  |
| **IPV\_PLLDIG** | **X** | **X** | **X** | **X** | **X** | **X** |  | **X** |
| **IPV\_PMC** | **X** |  |  |  |  |  | **X** | **X** |
| **IPV\_PMCDIG** |  | **X** |  |  | **X** |  |  |  |
| **IPV\_PRAM** | **X** | **X** | **X** | **X** | **X** | **X** |  |  |
| **IPV\_RCM** |  |  |  |  |  |  | **X** |  |
| **IPV\_SCG** |  |  |  |  |  |  | **X** |  |
| **IPV\_SIM** |  |  |  |  |  |  | **X** |  |
| **IPV\_SIRC** |  | **X** |  |  | **X** |  |  |  |
| **IPV\_SIUL2** |  | **X** | **X** | **X** | **X** | **X** |  | **X** |
| **IPV\_SIU** | **X** |  |  |  |  |  |  |  |
| **IPV\_SMC** |  |  |  |  |  |  | **X** |  |
| **IPV\_SSCM** | **X** | **X** | **X** | **X** | **X** | **X** |  |  |
| **IPV\_STCUV2** | **X** | **X** | **X** | **X** | **X** | **X** |  | **X** |
| **IPV\_SXOSC** |  | **X** |  |  | **X** |  |  |  |
| **IPV\_XOSC** |  |  | **X** | **X** |  |  |  |  |

Bảng 1. Thông tin IPV trên mỗi platform.

|  |  |  |
| --- | --- | --- |
| **IPs** | **Chapter name in RM** | **Control by API** |
| **IPV\_AFE** | RADAR Analog Front-End | Mcu\_InitClock |
| **IPV\_CORTEXM** | Cortex-M4 Devices | Mcu\_PerformReset |
| **IPV\_CMU** | Clock Monitor Unit | Mcu\_InitClock |
| **IPV\_DFS** | Digital Frequency Synthesizer | Mcu\_InitClock |
| **IPV\_EMIOS** | Configurable Enhanced Modular IO Subsystem | Mcu\_InitClock  Mcu\_DistributePllClock  Mcu\_EmiosConfigureGpren |
| **IPV\_FIRC** | Fast Internal RC Oscillator | Mcu\_InitClock |
| **IPV\_FXOSC** | Fast OSC Digital Interface | Mcu\_InitClock |
| **IPV\_FLASHV2** | Flash Memory Controller | Mcu\_Init  Mcu\_InitClock |
| **IPV\_LPU** | Low Power Subsystem | Mcu\_SetMode |
| **IPV\_MCV4** | Include:  Clock Generation Module (CGM)  Mode Entry Module (MC\_ME)  Power Control Unit (PCU)  Reset Generation Module (RGM) | Mcu\_Init  Mcu\_InitClock  Mcu\_DistributePllClock  Mcu\_SetMode  MCU\_PerformReset  Mcu\_GetResetReason  Mcu\_GetResetRawValue |
| **IPV\_PCC** | Peripheral Clock Controller | Mcu\_InitClock |
| **IPV\_PLLDIG** | PLL Digital Interface | Mcu\_InitClock  Mcu\_GetPllStatus |
| **IPV\_PMC** | Power Management Controller | Mcu\_Init |
| **IPV\_PMCDIG** | Power Management Controller digital interface | Mcu\_Init |
| **IPV\_PRAM** | RAM Controller | Mcu\_InitClock |
| **IPV\_RCM** | Reset Control Module | Mcu\_Init  Mcu\_GetResetReason  Mcu\_GetResetRawValue |
| **IPV\_SCG** | System Clock Generator | Mcu\_InitClock |
| **IPV\_SIM** | System Integration Module | Mcu\_Init  Mcu\_InitClock |
| **IPV\_SIRC** | Slow Internal Oscillator | Mcu\_InitClock |
| **IPV\_SIUL2** | System Integration Unit Lite2 | Mcu\_GetMidrStructure |
| **IPV\_SIU** | System Integration Unit | Mcu\_Init  Mcu\_InitClock  Mcu\_DistributePllClock  Mcu\_SetMode  MCU\_PerformReset  Mcu\_GetResetReason  Mcu\_GetResetRawValue  Mcu\_GetMidrStructure |
| **IPV\_SMC** | System Mode Controller | Mcu\_Init  Mcu\_SetMode |
| **IPV\_SSCM** | System Status and Configuration Module | Mcu\_SscmGetMemConfig  Mcu\_GetSystemState  Mcu\_SscmGetStatus  Mcu\_SscmGetUops |
| **IPV\_STCUV2** | Self-Test Control Unit | Mcu\_GetRamState |
| **IPV\_SXOSC** | Slow External Oscillator | Mcu\_InitClock |
| **IPV\_XOSC** | OSC Digital Interface | Mcu\_InitClock |

Bảng 2. Thông tin tổng quan của các IPV.

MCU gồm nhiều IPVs, tuy nhiên trong số đó có những IPV đơn giản chỉ quản lí từ 1-2 register. Và nội dung những register này có thể thay đổi trên các platform khác nhau. Một số IPV đơn giản sẽ được nói tổng quan, chi tiết có thể tìm thấy trong RM. Các IPV phức tạp sẽ được trình bày kĩ và cụ thể hơn. Và focus nhiều hơn vào những feature chính mà IPV cung cấp.

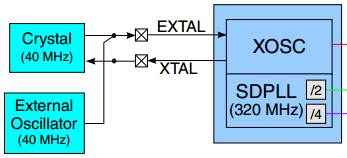
### IPV\_AFE

AFE có khá nhiều thanh ghi, tuy nhiên MCU không control tất cả các thanh ghi của AFE, MCU chỉ control các thanh ghi sau:

* Oscillator Control Register (AFE\_OSCCTRL)
* Oscillator Status Register (AFE\_OSCSTS)
* Oscillator Delay Register (AFE\_OSCDLY)
* SDPLL Control Register 1 (AFE\_PLLCTRL1)
* SDPLL Control Register 2 (AFE\_PLLCTRL2)
* SDPLL Control Register 3 (AFE\_PLLCTRL3)
* SDPLL Control Register 8 (AFE\_PLLCTRL8)
* SDPLL Status Register (AFE\_PLLSTS)

Và cung cấp các Features:

* Generate SDPLL clocks với output 320MHz, 160MHz và 80MHz.
* Provides for bypassing the 40 MHz crystal reference clock (Clock source của XOSC clock có thể được lựa chọn bởi Crystal hoặc Bộ dao động ngoài – External Oscillator. AFE cung cấp AFE\_OSCCTRL[EN\_EXT] bit, cho phép lựa chọn Clock source cho XOSC clock).



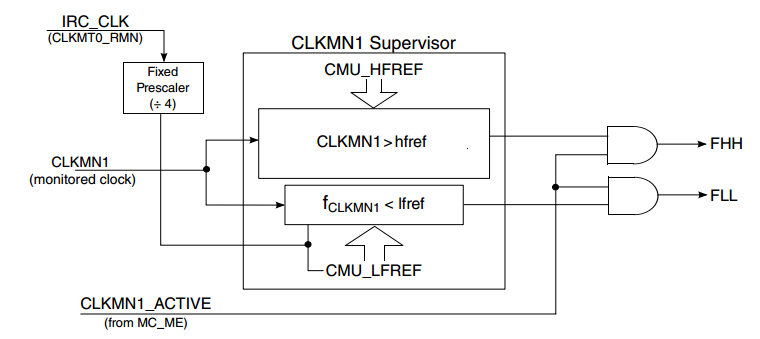
### IPV\_CORTEXM

IPV\_CORTEXM chỉ có trên Kinetis, IPV\_CORTEXM cung cấp Reset service và Cấu hình việc Enable/Disable chip go into deep sleep mode. Các thanh ghi được sử dụng để thực hiện những tính năng này là CM4\_AIRCR(Reset service) và CM4\_SCR (Enable/Disable chip go into deep sleep mode). Các thanh ghi này là các thanh ghi CORE, chúng ta sẽ ko tìm thấy trong RM, thông tin của các thanh ghi này có thể tìm thấy trong các tài liệu của Cortex-M4 Devices (Ex: Click [Here](http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0553a/Cihehdge.html))

### IPV\_CMU

* IPV\_CMU cung cấp chức năng giám sát các giá trị tần số clock. Tùy thuộc vào mỗi platform mà số lượng CMU khác nhau, Mỗi CMU sẽ giám sát 1 hoặc 1 vài nguồn clock.
* Nguyên lý của việc giám sát các giá trị tần số clock: Mỗi CMU sẽ có một Reference Clock Source và hai thông số Low Frequency reference value (CMU\_LFREFR[LFREF] bits), High Frequency reference value (CMU\_HFREFR[HFREF] bits). Ba thông số này sẽ quyết định khoảng giá trị tần số mà CMU này giám sát. Khi giá trị tần số của nguồn clock được giám sát nằm ngoài giải tần số này. Tùy thuộc vào nằm trên dải hay dưới dải, một cờ ngắt sẽ được raised tương ứng (FHHI, FLLI or OLRI). Chi tiết về thông tin của CMU cho từng platform có thể tìm hiểu trong RM của platform đó.

Example:



Hình trên là sơ đồ nguyên lý của CMU\_1 trên RaceRunnerUltra (S32R274). CMU này giám sát CORE0\_CLK clock.

Reference Clock Source: IRC\_CLK – 16MHz

Giả sử muốn CORE0\_CLK trong khoảng [40MHz, 48MHz]. Ta sẽ phải cấu hình:

Low Frequency reference value= 160 và High Frequency reference value= 192. Khi đó, lfref = (LFREF ÷ 16) × (fCLKMT0\_RMN÷ 4) = (160 ÷ 16) × (16÷ 4) = 40MHz

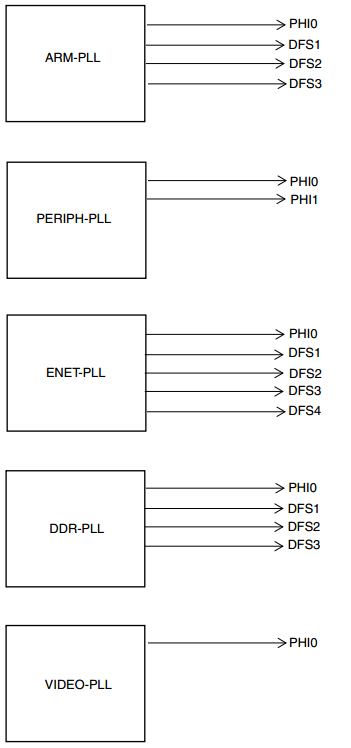
hfref = (HFREF ÷ 16) × (fCLKMT0\_RMN÷ 4) = (192 ÷ 16) × (16÷ 4) = 48MHz

Trong trường hợp CMU\_1 được active, CORE0\_CLK tùy thuộc vào giá trị thực tế mà CMU\_1 sẽ raised cơ ngắt tương ứng:

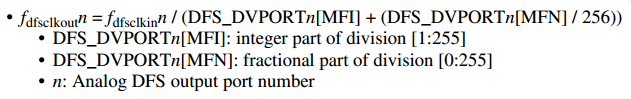
* CORE0\_CLK nằm trong khoảng [40MHz, 48MHz], ko có cờ ngắt nào được raised.
* CORE0\_CLK nhỏ hơn 40MHz, cờ ngắt FLLI raised.
* CORE0\_CLK lớn hơn 48MHz, cờ ngắt FHHI raised.

### IPV\_DFS

IPV\_DFS chỉ có trên TreeRunner, IPV\_DFS được hiểu đơn giản như là một module phân phối clock, DFS tạo ra các nguồn clock đầu ra đa dạng hơn của bộ PLL. Trên hầu hết các platform, bộ PLL chỉ cho tối đa hai đầu ra là PHI0 và PHI1, khi sử dụng DFS module, số đầu ra được đa dạng hóa và có thể tạo ra được các nguồn clock với tần số cao:



Nhìn vào hình trên có thể thấy được mỗi bộ PLL có thể không sử dụng or sử dụng một vài bộ DFS. Mỗi bộ DFS có hai bitfeilds quan trọng là DFS\_DVPORT[MFI] và DFS\_DVPORT[MFN]. Hai thông số này sẽ quyết định tần số đầu ra của một bộ DFS, điều này được thể hiện thông qua công thức:



### IPV\_EMIOS

EMIOS là một IPV dùng chung và MCU chỉ control thanh ghi eMIOS\_MCR. MCU sẽ cung cấp cấu hình các bit MDIS, FRZ, GTBE, ETB trong thanh ghi eMIOS\_MCR khi gọi các API Mcu\_InitClock và Mcu\_DistributePllClock. Ngoài ra MCU còn cung cấp Mcu\_EmiosConfigureGpren API – API này cho phép Enable/Disable the GPREN bit trong eMIOS\_MCR register. Chi tiết các bits được nêu trong RM.

### IPV\_FIRC

This internal RC oscillator provides the 16 MHz clock with high accuracy, low-power, fast stabilization, and fast recovery.

Key features of the 16 MHz FIRC module include:

* Operation from internally generated power supply (3 V or 5 V, with support down to 2.7 V)
* Nominal frequency (Fnom) of 16 MHz
* Frequency accuracy
* Low current consumption
* Ultra fast stabilization, enabling fast recovery in less than 16 cycles

### IPV\_FXOSC

The FXOSC digital interface is used to control the on-chip oscillator (FXOSC) and provide the register interface for the programmable features. Selection of the FXOSC as a source clock is controlled by the Clock Generation Module (MC\_CGM). The FXOSC is enabled and disabled by configuring the Mode Entry Module (MC\_ME).

The programmable features of the FXOSC digital interface are as follows:

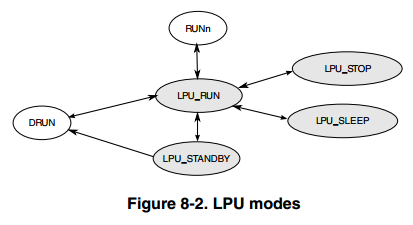
* Oscillator power-down control and status
* Oscillator startup delay
* Oscillator clock available interrupt
* Oscillator bypass mode control
* Output clock division factors ranging from 1, 2, 3…32

### IPV\_FLASHV2

FLASHV2 là một IPV dùng chung và MCU chỉ control các thanh ghi PFLASH\_PFCRx - controls the operation of Port Px of the flash memory controller. Chi tiết nêu trong RM.

### IPV\_LPU

LPU cung cấp chức năng đưa silicon vào trạng thái LPU mode (Low Power Mode). Các mode này bao gồm LPU\_RUN, LPU\_SLEEP, LPU\_STOP và LPU\_STANDBY. Thông tin về Modes of operation của IPV\_LPU được thể hiện trong hình sau:



IPV này chỉ có trên Calypso (both 3M và 6M). IPV này phần nào giống MC\_ME trong IPV\_MCV4. Tham khảo MC\_ME và RM (Low Power Subsystem (LPU) chapter) khi tìm hiểu IPV này.

### IPV\_MCV4

MCV4 thực chất bao gồm 4 IPVs là MC\_CGM, MC\_ME, MC\_RGM và MC\_PCU và cũng là IPV chính và quan trọng của MCU module trên hầu hết các platform. Trong đó MC\_CGM và MC\_ME có sự gắn bó mật thiết với nhau.

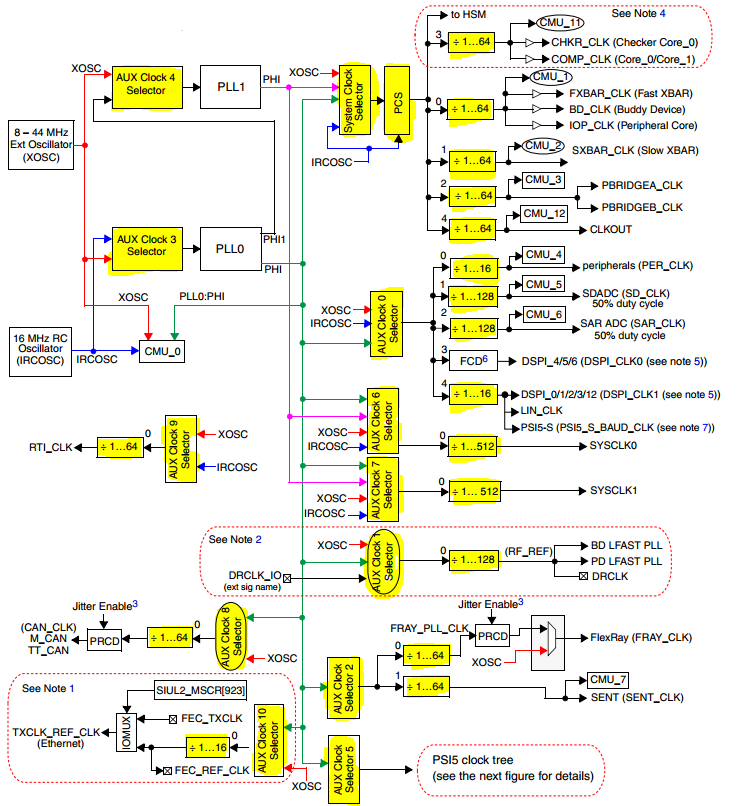
**MC\_CGM (Clock Generation Module).**

The clock generation module (MC\_CGM) generates reference clocks for all the chip blocks. The MC\_CGM selects one of the system clock sources to supply the system clock. The MC\_ME controls the system clock selection (see the MC\_ME for more details). The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC\_CGM memory space.

The MC\_CGM includes the following features:

* generates system and peripheral clocks
* selects and enables/disables the system clock supply from system clock sources according to MC\_ME control
* performs progressive system clock frequency change depending on MC\_ME mode configuration
* contains a set of registers to control clock dividers for divided clock generation
* contains a set of registers to control peripheral clock selection
* contains IPS interface for mapped clock source modules
* supports multiple clock sources and maps their address spaces to its memory map

Dưới đây là Clock Tree của MatterHorn:



MC\_CGM sẽ control tất cả các khối được bôi vàng trên Clock Tree. Bao gồm các bộ MUX (Bộ Select Clock Source cho System Clock, cho các Auxiliary Clock và sau là cho các Peripherals), các bộ Divider cho System Clock và cho các Clock cung cấp đến các Peripherals. Nguyên lý này là tương tự trên MC\_CGM của các platform khác (Chỉ khác nhau về số lượng nguồn Clock, số lượng Auxiliary Clock), ngoài ra số lượng thanh ghi của MC\_CGM tuy nhiều nhưng dễ quản lý vì có sự tương đồng về cụm thanh ghi dùng để control mỗi nhánh trên Clock Tree.

Ví dụ: Mỗi Auxiliary Clock luôn có các thanh ghi:

* MC\_CGM\_ACx\_SC (Auxiliary Clock Select Control Register): là thanh ghi control việc chọn nguồn Clock cho Auxiliary Clock.
* MC\_CGM\_ACx\_Dcy (Auxiliary Clock Divider Configuration Register): control các bộ chia trước khi Clock được cung cấp đến các Peripherals.

**MC\_ME (Mode Entry Module).**

The MC\_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

The MC\_ME includes the following features:

* control of the available modes by the ME\_ME register
* definition of various chip mode configurations by the ME\_<mode>\_MC registers
* control of the actual chip mode by the ME\_MCTL register
* capture of the current mode and various resource status within the contents of the
* MC\_ME\_GS register
* optional generation of various mode transition interrupts
* status bits for each cause of invalid mode transitions
* peripheral clock gating control based on the ME\_RUN\_PC0…7, ME\_LP\_PC0…7,
* nd ME\_PCTLn registers
* additional core clock gating and boot address control based on the ME\_CCTLn
* nd ME\_CADDRn registers
* capture of current peripheral and additional core clock gated/enabled status
* progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa

Chúng ta có thể tìm hiểu các thanh ghi của MC\_ME theo nhóm sau:

* Nhóm thanh ghi điều khiển Core:
* MC\_ME\_CCTLx (COREx Core Control Register): Cung cấp các bits control việc Active/Disable các Mode khi running với Core x tương ứng.
* MC\_ME\_CADDRx (COREx Core Address Register): Register này cung cấp địa chỉ boot cho Core x tương ứng và một bit (RMC) để kiểm soát xem Core này có được thiết lập lại khi thay đổi chế độ kế tiếp hay không.
* MC\_ME\_CS (Core Status Register): Register cung cấp trạng thái của các Core có đang Running hay đang ở trạng thái Disable.
* Nhóm thanh ghi điểu khiển Peripherals:
* MC\_ME\_PCTL (Peripheral Control Register): Mỗi một Peripheral sẽ có một thanh ghi control MC\_ME\_PCTL. Mỗi MC\_ME\_PCTL sẽ cấu hình cho một Peripheral refer tới một ME\_RUN\_PCx (Sẽ được nói phía dưới) và một ME\_LP\_PCx (Sẽ được nói phía dưới).
* ME\_RUN\_PCx và ME\_LP\_PCx (Run Peripheral Configuration Register và Low-Power Peripheral Configuration Register): Mỗi nhóm thanh ghi này sẽ bao gồm 8 thanh ghi (Được đánh số x=0~7). Các thanh ghi này sẽ control việc Enable/Disable các peripherals ở các Mode.

Từ hai nhóm thanh ghi điều khiển Peripherals ta có thể thấy. Các Peripherals sẽ được điều khiển linh hoạt (Enable/Disable) tùy thuộc vào việc Peripheral đó đang refer đến “Run Peripheral Configuration” và “Low-Power Peripheral Configuration” nào, Các “Peripheral Configuration” đó Active các Mode nào. Khi đó, tùy thuộc vào Mode hiện tại mà Peripherals sẽ ở trạng thái Enable hay Disable.

* MC\_ME\_PSx (Peripheral Status Register x): Thanh ghi này cho biết trạng thái của các Peripherals.
* Nhóm thanh ghi cấu hình Mode:
* MC\_ME\_<mode’s name>\_MC: Mỗi Mode sẽ có một thanh ghi. Thanh ghi này control việc ON/OFF các nguồn Clock, lựa chọn System Clock. Các thông số này sẽ được apply khi silicon được đưa vào Mode này.
* MC\_ME\_GS (Global Status Register): Cung cấp trạng thái hiện tại của silicon: Đang ở Mode nào, các Clock nào đang hoạt động hay bị Disable.

**MC\_RGM (Reset Generation Module).**

The reset generation module (MC\_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

The MC\_RGM contains the functionality for the following features:

* 'Destructive' resets management
* 'Functional' resets management
* Sgnalling of reset events after each reset sequence (reset status flags)
* Conversion of reset events to SAFE mode or interrupt request events
* Short reset sequence configuration
* bidirectional reset behavior configuration
* configurable escalation of recurring 'functional' resets to 'destructive' reset
* configurable escalation of recurring 'destructive' resets to keep chip in reset state until next power-on reset

MC\_RGM provide 2 status registers: MC\_RGM\_DES ('Destructive' Event Status Register) and MC\_RGM\_FES ('Functional' Event Status Register). These registers includes read-only status flags to indicate the source of the most recent reset. Note that multiple flags can be set if multiple reset events occur at the same time. The reset state of these bits depends on what caused the MCU to reset.

**MC\_PCU (Power Control Unit).**

MC\_PCU là IPV tương tự PMC (xem PMC và RM để tìm hiểu chi tiết). Trên các platform sử dụng cả IPV\_PMC và MC\_PCU, MC\_PCU không cung cấp bất cứ register nào, nó chỉ đóng vai trò là cầu nối để thiết lập việc control IPV\_PMC.

### IPV\_PCC

PCC là IPV chỉ được sử dụng trên Kinetis. IPV này quản lí các peripherals trên Kinetis. Nó cung cấp cấu hình Enable/Disable mỗi peripheral, chọn nguồn clock và bộ chia clock cho mỗi peripheral. Mỗi thanh ghi trong IPV\_PCC quản lý 1 peripheral.

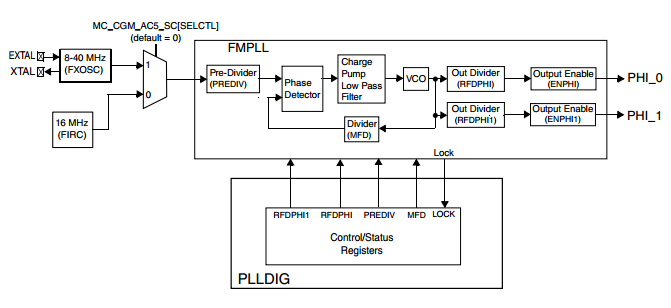
### IPV\_PLLDIG

The PLL system is composed of a single analog PLL block and the digital interface.

The PLLDIG module has the following features:

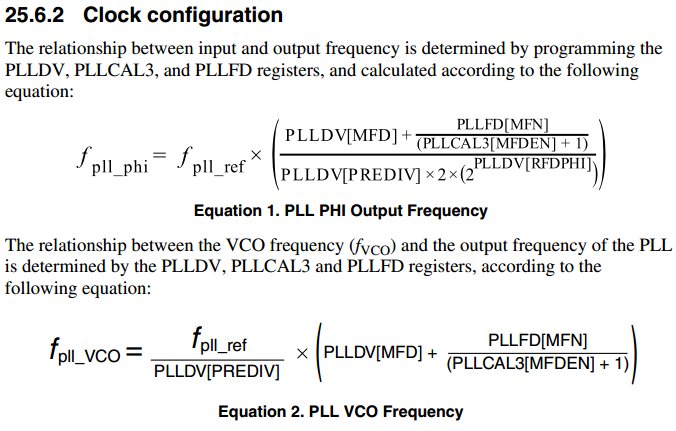
* Reference clock pre-divider for finer frequency synthesis resolution.
* Reduced frequency divider for decreasing the PLL output clock frequency without causing the PLL to lose lock.
* Programmable frequency modulation.
* Lock detect circuitry reports when the PLL has achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
* Loss-of-clock detection for the reference and output clocks.

PLLDIG cung cấp giao diện số với các thông số giúp generate các giá trị clock đầu ra (PLL\_PHI0, PLL\_PHI1, DFS) tùy biến với clock đầu vào bộ PLL cố định.



Hình trên là một sơ đồ ví dụ cho bộ PLL trên Calypso. Các clock đầu vào là FXOSC và FIRC (được lựa chọn thông qua MC\_CGM\_AC5\_SC[SELCTL] bit). Khối PLLDIG chính là khối Digital Interface – cung cấp các thông số Digital, các thông số này sẽ quyết định tần số đầu ra (PHI\_0 và PHI\_1) theo một công thức cụ thể được cung cấp trong RM.

Ví dụ trên Calypso: (Các platform khác có thể được tìm thấy trong RM - PLL Digital Interface (PLLDIG) chapter/ Functional description)



Ở đây có sự xuất hiện của tần số VCO clock, tuy nhiên đây không phải là một nguồn clock được sử dụng trong silicon, nó được sử dụng như một nguồn đồng hồ để theo dõi. VCO clock phải được duy trì trong một giải tần số nào đó (Được quy định trong RM or DataSheet). Khi tần số của VCO clock vượt ra ngoài dải, PLL rơi vào trạng thái mất kiểm soát (Loss of Lock). Và trạng thái này sẽ đc phát hiện bởi cờ ngắt PLLDIG\_PLLSR[LOLF].

### IPV\_PMC

The PMC module provides:

* Configuration and control information to the analog PMC block.
* Configuration and monitoring of the two temperature sensors.
* Host access, via the IPS bus, to all register controls, configurations, and status.
* Self test logic for the LVDs and HVDs.
* Transfer wait state feature supported on IPS bus access.
* Compatible with 3.3V (analog) operation.
* 1.25 V / 1.8 A output voltage switch-mode power supply (SMPS) asynchronous buck regulator.
* Precision LVD (Low-voltage detection) and HVD (high-voltage detection) for: PMC voltage supply (VDDREG) - LVD, VDD core voltage supply - POR/LVD/HVD, VDD ADC voltage supply - LVD/HVD, Flash memory voltage supply - LVD, Oscillators/PLL voltage supply - LVD and VDDIO I/O voltage supply - LVD.
* Two independent voltage references provide fault tolerance: Voltage monitors and ADC references and Core voltage regulator.
* DCF Client interface: An interface with Flash memory for retrieval of trim/control information, during POR for: HVDs/ LVDs Regulators and Band gap filters.
* MC\_RGM Interface: Necessary status and indication information to the MC\_RGM block during power up.
* FCCU (Fault control and correction Unit): Software Triggered fault injection for LVDs, HVDs, temperature sensors, and LVD self test.

### IPV\_PMCDIG

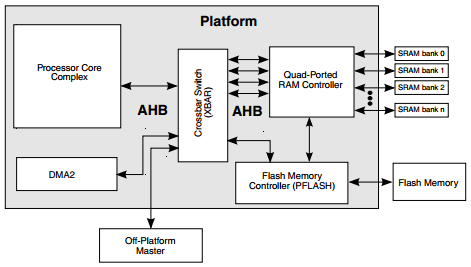
The PMC provides the following main features:

* Compatible with 3.3 V and 5 V operation.
* No specific power sequencing required and all power domains guaranteed safe-state during power-up, power-down.
* Precision voltage monitors: PMC voltage supply - POR (Power on Reset), VDD core voltage supply - POR, HVD (high voltage detector), LVD (low voltage detector), Flash memory voltage supply - LVD and VDD\_HV\_A voltage supply - LVD.
* Bandgap voltage reference for Voltage monitors and regulators and ADC voltage reference channel.

### IPV\_PRAM

This section provides an overview of the quad-ported Platform RAM Controller. The RAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and multiple banks of integrated system RAM, enabling simultaneous SRAM accesses by multiple bus masters.

Internally the quad-ported RAM controller provides an independently-configurable RAM controller for each associated bank of SRAM. One of the internal RAM controllers supports the use of its associated SRAM as overlay SRAM. See the chip-specific information for details.



The figure illustrates the quad-port RAM controller in the context of a typical SoC platform architecture. The RAM controller supports four 64-bit primary AHB interfaces, each to a separate system crossbar switch slave port, a backdoor AHB port to the on-chip flash controller, and a 64-bit RAM array interface to each associated bank of on-chip SRAM.

The primary AHB ports provide a connection to the platform crossbar for direct RAM access from the various system crossbar masters. The backdoor AHB port provides a connection from the flash controller to facilitate calibration overlay access.

The following list summarizes the key features of the RAM controller:

* End-to-End ECC (e2eECC) support
* System bus supports 64-bit data + 8-bit ECC AHB interfaces
* Array interfaces support a 64-bit data + 8-bit ECC interface
* Late-write support via 64-bit data + 8-bit ECC storage buffer to support single-cycle write accesses
* Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
* Read-modify-write operation to support array write size less than a double word

### IPV\_RCM

RCM is only used on Kinetis. Information found here describes the registers of the Reset Control Module (RCM). The RCM implements many of the reset functions for the chip. Click [Here](http://cache.freescale.com/files/32bit/doc/app_note/AN4503.pdf) for further details on using the RCM.

RCM produce System Reset Status Register (RCM\_SRS) - This register includes read-only status flags to indicate the source of the most recent reset. Note that multiple flags can be set if multiple reset events occur at the same time. The reset state of these bits depends on what caused the MCU to reset.

### IPV\_SCG

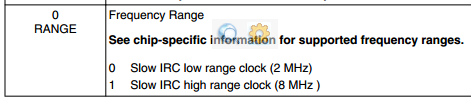
SCG là IPV chỉ được sử dụng trên Kinetis. Cung cấp các registers cấu hình cho các nguồn clock (SOSC, SIRC, FIRC và SPLL) và control System Clock Source và các bộ Divide trên Clock Tree trong mỗi Mode (RUN, VLPR, HRUN). Cụ thể:

* Mỗi nguồn Clock có 3 thanh ghi control:

SCG\_xxxxCSR: Control việc Enable/Disable nguồn clock và một số thông số của nguồn clock (Chi tiết trong RM), ngoài ra còn cung cấp 1 số bít trạng thái của nguồn clock.

SCG\_xxxxDIV: Control bộ chia nguồn clock.

SCG\_xxxCFG: Lựa chọn Frequency Range của nguồn clock (Chi tiết trong hình – ví dụ cho SCG\_SIRCCFG register)



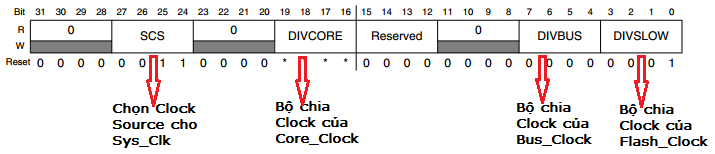
* Chúng ta có 3 registers control Clock trong 3 modes tương ứng:

SCG\_RCCR (RUN Clock Control)

SCG\_VCCR (VLPR Clock Control)

SCG\_HCCR (HRUN Clock Control)

Nội dung của ba thanh ghi trên được mô tả như hình dưới.



* Ngoài ra SCG còn cung cấp các thanh ghi: SCG CLKOUT Configuration Register (SCG\_CLKOUTCNFG) - Lựa chọn nguồn Clock Out – là tín hiệu Clock ra Clock Out Pin; Clock Status Register (SCG\_CSR) – Thanh ghi trạng thái của System Clock.

### IPV\_SIM

SIM is only used on Kinetis. SIM provides system control and chip configuration registers.

Features of the SIM include:

* System clocking configuration
* FlexTimer clock channel and configuration
* LPO clock source selection
* ENET clock control

MCU module quản lí các thanh ghi sau trong SIM module:

* SIM\_CHIPCTL: contains the controls for selecting ADC COCO trigger, trace clock, clock out source, PDB back-to-back mode, and ADC interleave channel.
* SIM\_LPOCLKS: Enable/Disable LPO\_CLK and Select Frequency Range
* SIM\_ADCOPT: Selects pretrigger source for ADC module
* SIM\_FTMOPT0: Selects the external pin used to drive the FTMx module clock.
* SIM\_FTMOPT1: Control FTMx Sync Bit.
* SIM\_MISCTRL0: Enable/Disable Clock of QSPI, Support to configure FTMx OBE CTRL bits
* SIM\_MISCTRL1: Support to generates software trigger to peripherals through TRGMUX
* SIM\_PLATCGC: Enable/Disable the clock to the EIM, ERM, DMA, MPU, MSCM modules.

### IPV\_SIRC

SIRC provides the 128 kHz clock with high accuracy, low power consumption, fast stabilization, and fast recovery.

Key features of the 128 kHz SIRC module are as follows:

* Operating voltage generated internally from VDD\_LV
* Nominal frequency (Fnom) of 128 kHz
* Frequency accuracy
* Low current consumption

### IPV\_SIUL2

SIUL2 là một IPV dùng chung và MCU chỉ control các thanh ghi SIUL2\_MIDR1 và SIUL2\_MIDR2. Hai thanh ghi này cung cấp một số thông tin về HW như: PARTNUM (Cung cấp mã HW, ví dụ Calypso là 5748G…), PKG (Phân biệt các drivatives), Major Mask Revision and Minor Mask Revision (Version của silicon).

### IPV\_SMC

The System Mode Controller (SMC) is responsible for sequencing the system into and out of all low-power Stop and Run modes.

Specifically, it monitors events to trigger transitions between power modes while controlling the power, clocks, and memories of the system to achieve the power consumption and functionality of that mode.

SMC describes all the available low-power modes, the sequence followed to enter/exit each mode, and the functionality available while in each of the modes.

The SMC is able to function during even the deepest low power modes.

Click [Here](http://www.nxp.com/docs/en/application-note/AN4503.pdf) for Kinetis MCUs for further details on using the SMC.

### IPV\_SSCM

SSCM cung cấp 1 vài thanh ghi status. Thông qua các API, MCU rẽ đọc và return này:

* Mcu\_SSCM\_GetSystemState: Return giá trị hai registers là SSCM\_STATUS và SSCM\_MEMCONFIG, những registers này cung cấp các thông tin về mode, security, vle, jtag id, etc…
* Mcu\_SscmGetMemConfig: This function returns the System Memory and ID register from readding SSCM\_MEMCONFIG register.
* Mcu\_SSCM\_GetUops: This function returns information about User Option Bits, HSM Boot Configuration from readding SSCM\_UOPS register.

### IPV\_STCUV2

STCUV2 là IPV dùng chung. MCU module chỉ sử dụng hai registers là STCU2 Off-Line MBIST End Flag Low Register (STCU2\_MBEL) và STCU2 Off-Line MBIST Status Low Register (STCU2\_MBSL):

* The STCU2\_MBEL register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31. The size of the register depends on the number of BISTed RAMs/ROMs.
* The STCU2\_MBSL register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.The size of the register depends on the number of BISTed RAMs/ROMs.

Thông tin hai thanh gi này cung cấp được sử dụng cho Mcu\_GetRamState API - This function returns the actual state of the RAM.

### IPV\_SXOSC

The slow external 32 KHz oscillator (SXOSC) provides a 32 KHz clock with control and status registers accessible for use by applications.

Key features of the 32 KHz SXOSC module include:

* Operation from internally generated power supply (3 V or 5 V)
* Frequency (Fnom) — 32 KHz crystal
* Voltage and Frequency filtering to guarantee clock frequency and stability
* Power Consumption target: 5 µA
* Startup time lower than 2 sec

### IPV\_XOSC

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provide the register interface for the programmable features. Selection of the XOSC as a source clock is controlled by the Clock Generation Module (MC\_CGM). The XOSC is enabled and disabled by configuring the Mode Entry Module (MC\_ME).

The programmable features of the XOSC digital interface are as follows:

* Oscillator power-down control and status
* Oscillator startup delay
* Oscillator clock available interrupt
* Oscillator bypass mode control

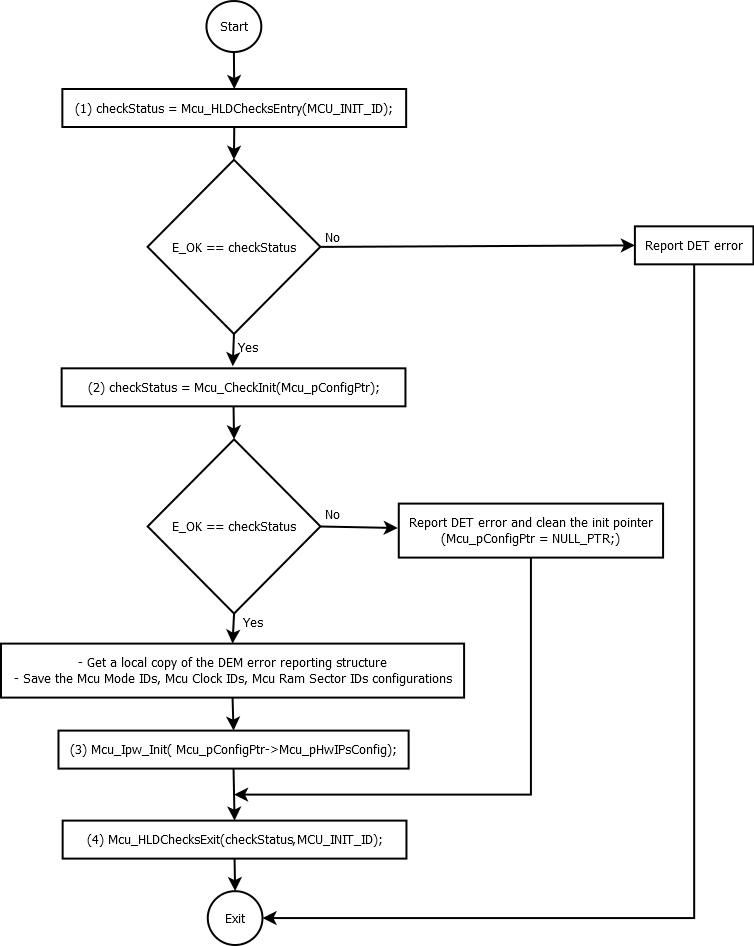
## Implementation

Phần này, trình bày về tất cả các APIs. Với mỗi API chúng ta sẽ đi từ tổng quát (vai trò, chức năng, ... ) đến chi tiết (design, thuật toán,...). Chúng ta có 11 AUTOSAR-APIs (Từ 1.4.1 đến 1.4.11), các APIs còn lại là Non- AUTOSAR-APIs. Những APIs và những Functions phức tạp, ngoài sơ đồ thuật toán sẽ được giải thích thêm (step by step). Những Function đơn giản có thể tìm hiểu thông qua Sơ đồ thuật toán và đọc Code trong Plugins.

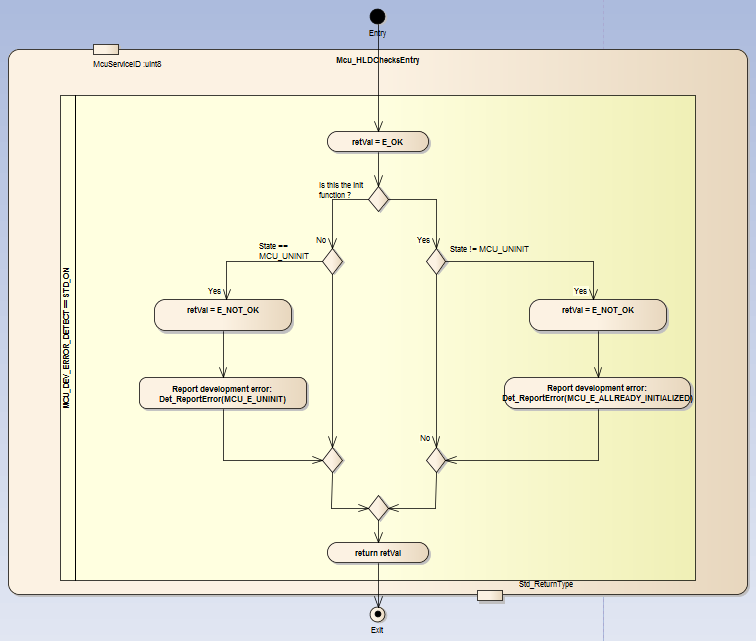
### Mcu\_Init

Overview: 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.

#### 1.4.1.1 HLD Layer

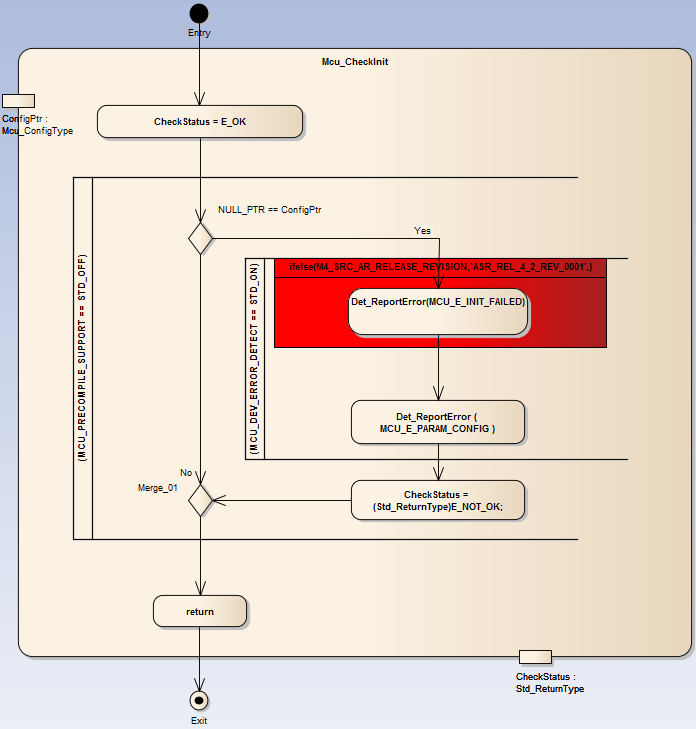


1. Mcu\_HLDChecksEntry Diagrams:



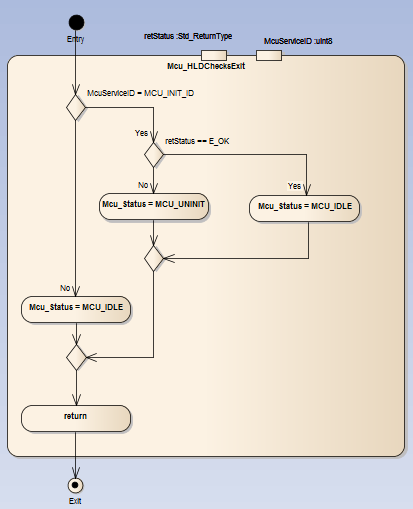
Function này được hiểu đơn giản như sau. Bước đầu tiên thực hiện một API, chúng ta cần phải kiểm tra API đó được thực hiện có hợp lệ không. Mcu\_Init chỉ được thực hiện một lần duy nhất, các API còn lại (Ngoại trừ Mcu\_GetVersionInfo) chỉ được thực hiện sau khi hoàn thành Mcu\_Init. Các trường hợp còn lại là vi phạm và sẽ có một lỗi DET được raised.

1. Mcu\_CheckInit Diagrams:



Function này kiểm tra parameter [in] đã hợp lệ. Khi Post-Build được sử dụng, tham số đầu vào 'pConfigPtr' bắt buộc phải khác với NULL\_PTR.

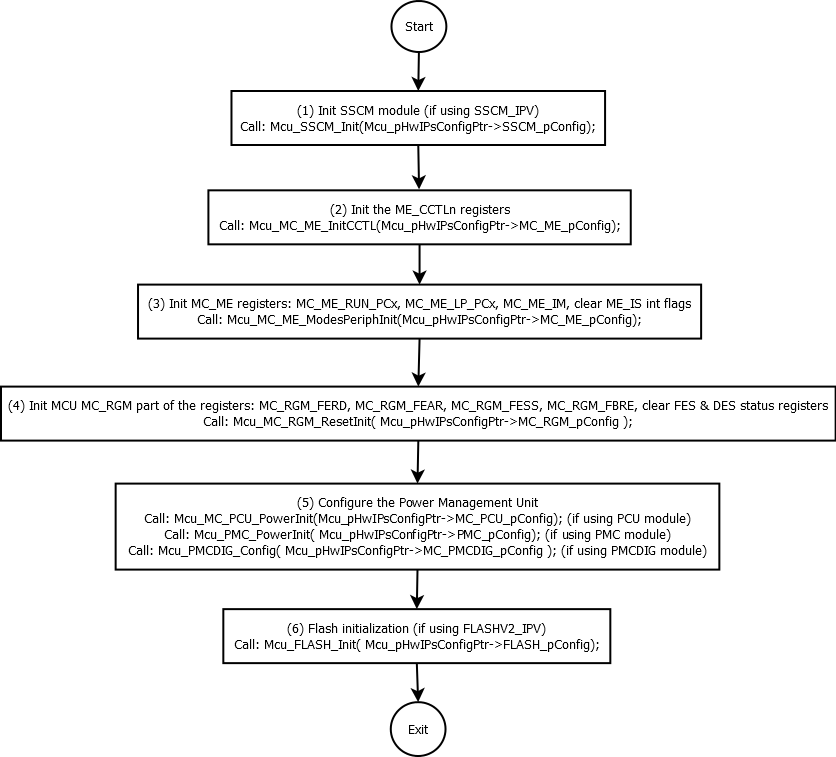
1. Mcu\_Ipw\_Init: See **IPW Layer** section
2. Mcu\_HLDChecksExit Diagrams:



Function này sẽ là một cặp với Mcu\_HLDChecksEntry, sau khi gọi Mcu\_Init lần đầu tiên, trạng thái của biến toàn cục Mcu\_eStatus sẽ là MCU\_IDLE. Trạng thái này sẽ được sử dụng cho việc check các API được gọi sau đó và raised DET Error nếu API này được gọi không hợp lệ.

#### 1.4.1.2 IPW Layer

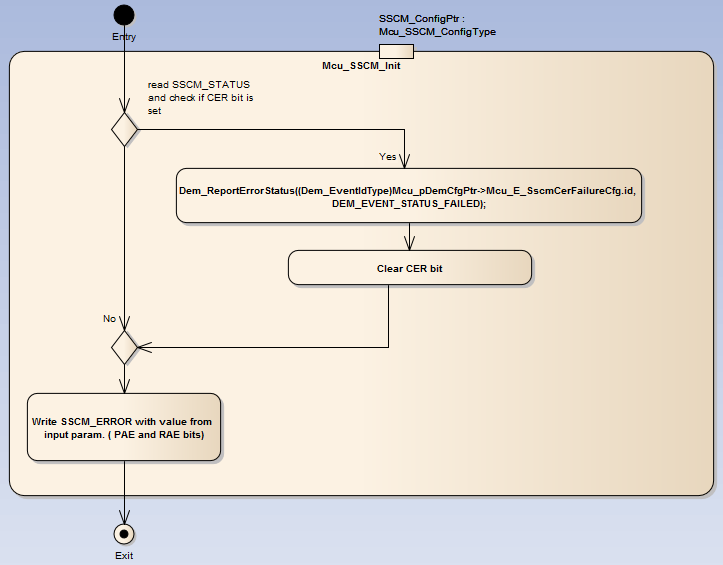
Mcu\_Ipw\_Init Diagram:



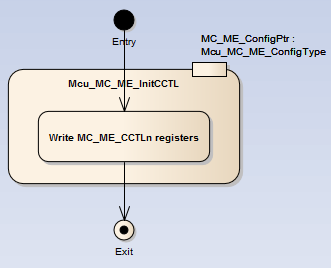
For detail, see **IPV Layer** section

#### 1.4.1.3 IPV Layer

1. Mcu\_SSCM\_Init diagram:



1. Mcu\_MC\_ME\_InitCCTL diagram:



Việc ghi vào các thanh ghi MC\_ME\_CCTLn được ghi lần lượt bằng 1 vòng lặp for. Mỗi thanh ghi được ghi dựa trên 1 phần tử của mảng có kiểu dữ liệu Mcu\_MC\_ME\_CoreConfigType

typedef struct

{

VAR( uint32, MCU\_VAR) u32RegisterAddr; /\* register address, added by hand in the code template \*/

VAR( uint16, MCU\_VAR) u16RegisterData; /\* generated value, as it's already defined \*/

} Mcu\_MC\_ME\_CoreConfigType;

Việc ghi vào mỗi thanh ghi sẽ được thực hiện như sau:

/\* Write u16RegisterData value to u32RegisterAddr \*/

for (u8CoreIdx = 0x00U; u8CoreIdx < MC\_ME\_pConfigPtr->u8NumberOfCoreCtrlRegs; u8CoreIdx++)

{

/\*\* @violates @ref Mcu\_MC\_ME\_c\_REF\_4 Conversion form pointer to integer\*/

/\*\* @violates @ref Mcu\_MC\_ME\_c\_REF\_7 The cast is used to access memory mapped registers.\*/

REG\_WRITE\_LOCK16 \

( \

MC\_ME\_BASEADDR, \

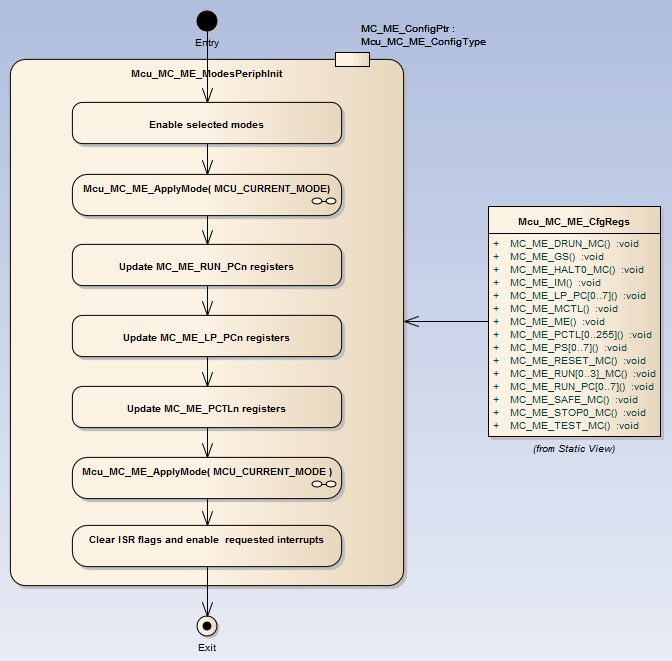
(\*MC\_ME\_pConfigPtr->apCoreControl)[u8CoreIdx].u32RegisterAddr, \

(\*MC\_ME\_pConfigPtr->apCoreControl)[u8CoreIdx].u16RegisterData \

);

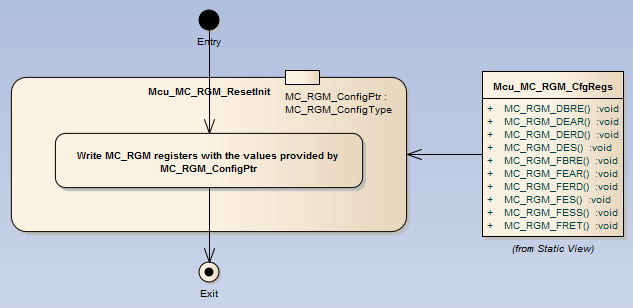
}

1. Mcu\_MC\_ME\_ModesPeriphInit diagram:



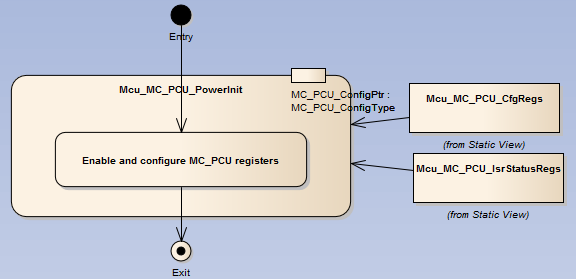
Function này sẽ configure các thanh ghi điều khiển Pheripherals. Các thanh gì này nằm trong MC\_ME module và có địa chỉ liên tiếp nhau. Việc configure các thanh ghi này được thực hiện theo thuật toán tương tư như function Mcu\_MC\_ME\_InitCCTL đã nêu ở trên. Xem thêm “Nhóm thanh ghi điểu khiển Peripherals” trong 2.3.10 IPV\_MCV4 để hiểu hơn về function này.

1. Mcu\_MC\_RGM\_ResetInit diagram:

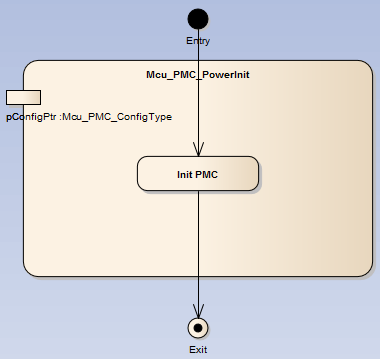


1. Power Management Unit

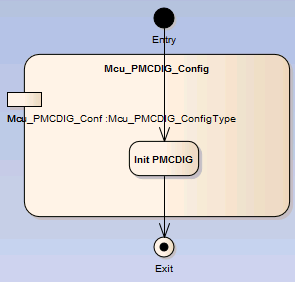
MC\_PCU\_PowerInit diagram:



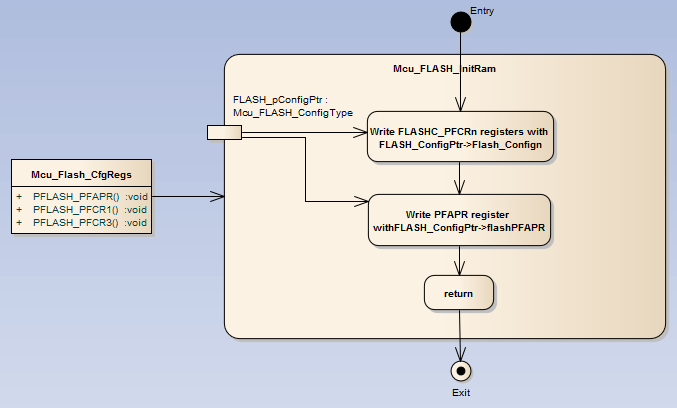
Mcu\_PMC\_PowerInit:



Mcu\_PMCDIG\_Config:



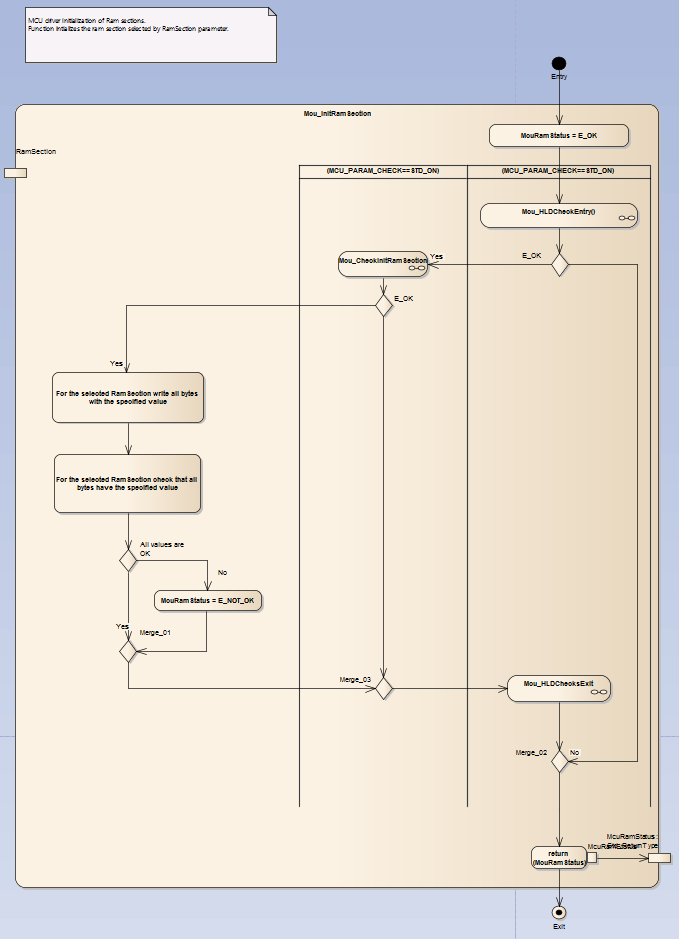
1. Mcu\_FLASH\_Init diagram:



### Mcu\_InitRamSection

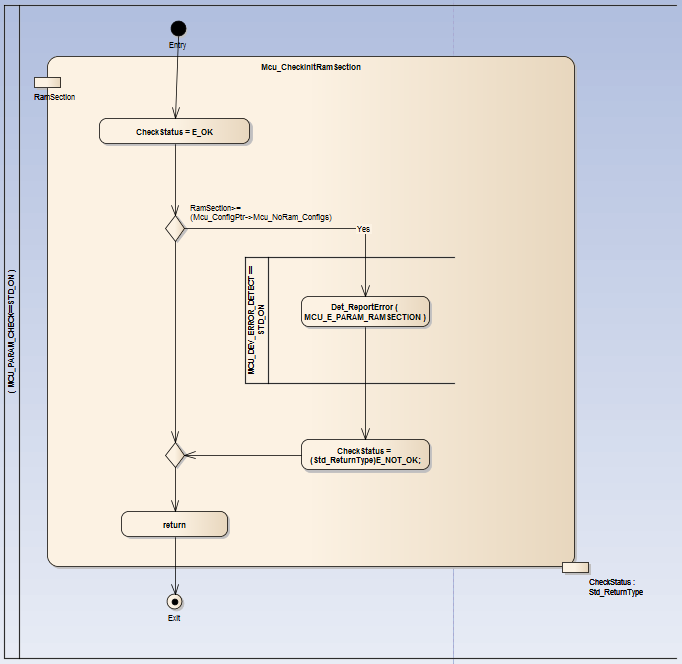
Overview: Function intializes the ram section selected by RamSection parameter. The section base address, size and value to be written are provided from the configuration structure. The function will write the value specified in the configuration structure indexed by RamSection. After the write it will read back the RAM to verify that the requested value was written.

#### 1.4.2.1 HLD Layer



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. The function will write the value specified in the configuration structure indexed by RamSection. After the write it will read back the RAM to verify that the requested value was written.

1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_CheckInitRamSection diagram:



Function này kiểm tra parameter [in] của Mcu\_InitRamSection API. parameter [in] của API này là ID của RAM configuration (ID này là ID của phần tử cần được configured), và function này check việc ID đó có out of range và nội dung configuration của ID này có khác NULL hay không? Vi phạm 1 trong hai điều này, Function sẽ lập tức trả về E\_NOT\_OK và raise DET Error.

1. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.2.2 IPW Layer

Doesn’t use

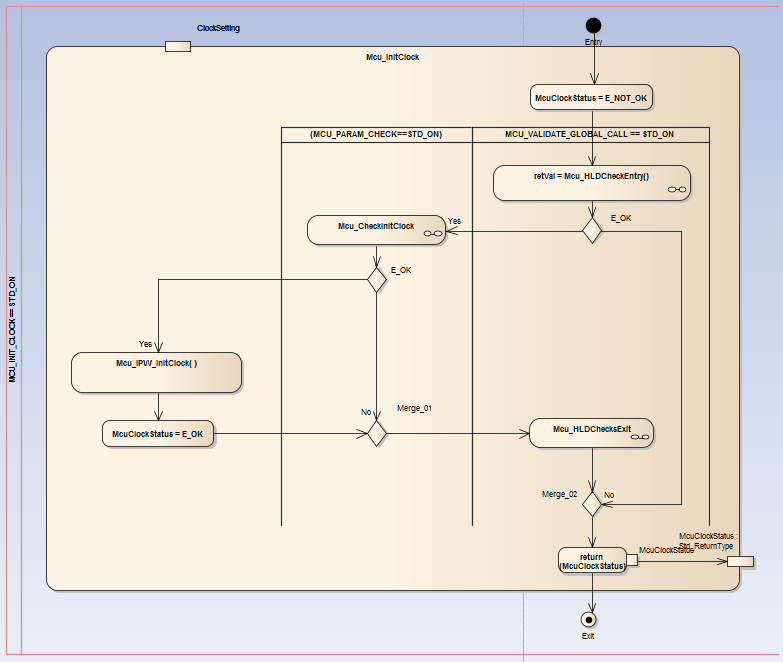
#### 1.4.2.2 IPW Layer

Doesn’t use

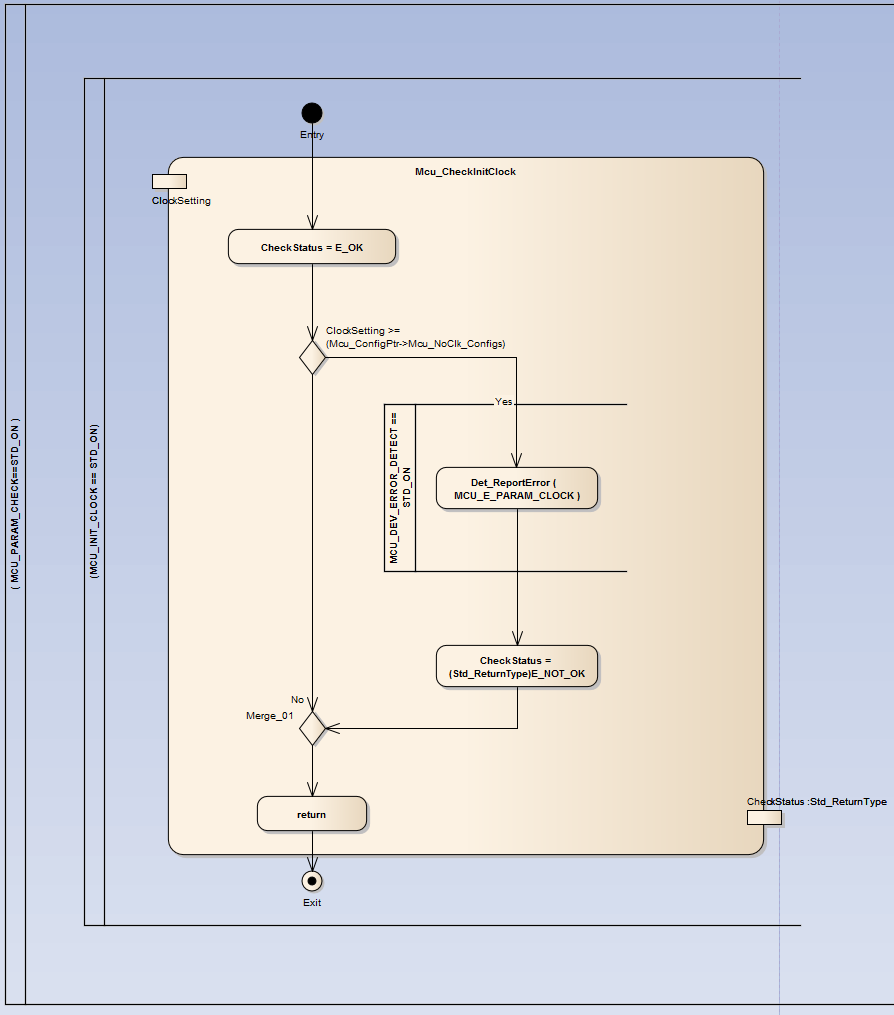
### Mcu\_InitClock

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

#### 1.4.3.1HLD Layer



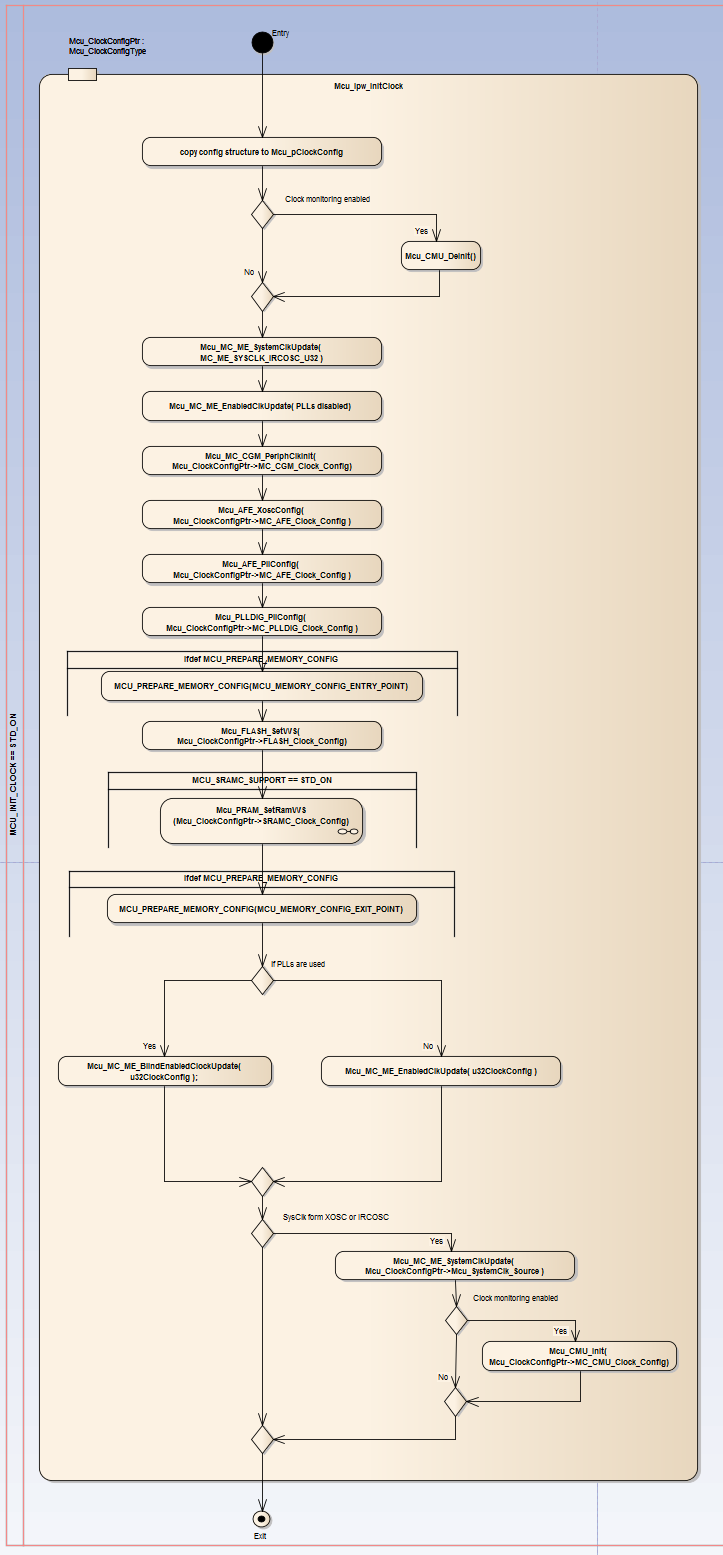
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_CheckInitClock diagram:



Function này tương tự Mcu\_CheckInitRamSection.

1. Mcu\_Ipw\_InitClock: See **IPW Layer** section
2. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.3.2 IPW Layer



Step1: Vô hiệu hóa CMU module trước khi cấu hình Clock Tree

Step2: Chuyển System Clock sang FIRC clock (Mục đích của bước này là giải thoát các Clock khác để thực hiện việc Re-configure các nguồn Clock này)

Step3: Get trạng thái các nguồn Clock, Nguồn Clock nào được Re-configure sẽ được Disable trước khi Re-configure, Nguồn Clock nào không được Re-configure sẽ được giữ nguyên trạng thái như trước đó.

Step4: Thực hiện configure các nguồn Clock được request to configure.

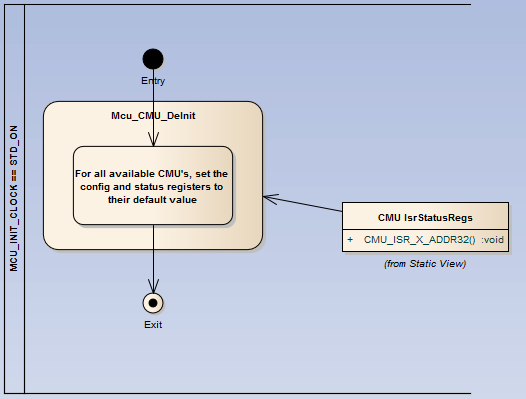
Step5: Thực hiện configure các thông số RAM và FLASH ảnh hưởng đến Clock.

Step6: Apply cấu hình Clock đã hoành thành configure với trạng thái Mode hiện tại.

For detail, see **IPV Layer** section

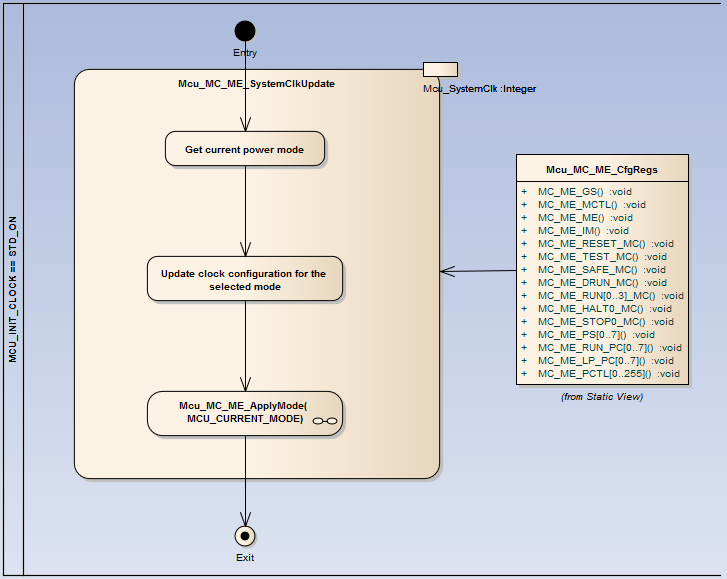
#### 1.4.3.3 IPV Layer

1. Mcu\_CMU\_DeInit diagram:



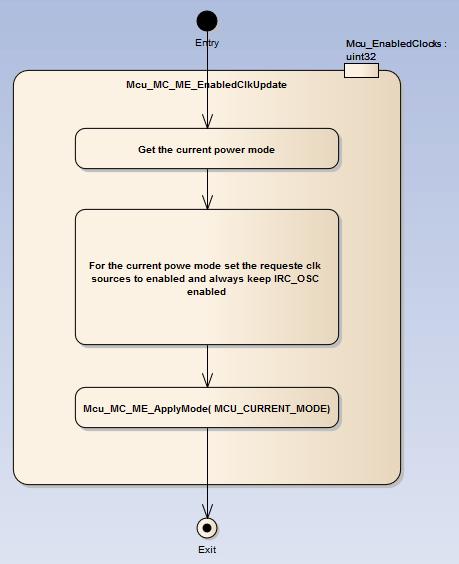
Function này sẽ clear CMU\_CSR[CME] bit của tất cả các CMU để Disable tất cả các CMU. Việc này được thực hiện trước khi thực hiện configure các nguồn Clock nhằm tránh việc CMU interrupt được bật không chính xác trong quá trình re-configure các nguồn Clock.

1. Mcu\_MC\_ME\_SystemClkUpdate diagram:



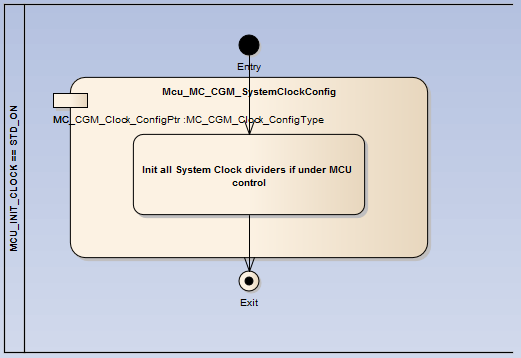
Function này get thông tin Mode hiện tại bằng cách đọc MC\_ME\_GS[S\_CURRENT\_MODE] bitfield. Thông tin này sẽ phục vụ việc ghi cấu hình System Clock vào thanh ghi control của Mode tương ứng (SYSCLK bitfield). Sau khi cấu hình System Clock được ghi vào Mode hiện tại, bước tiếp theo là apply Mode đó với cấu hình đã được updated.

1. Mcu\_MC\_ME\_EnabledClkUpdate diagram:



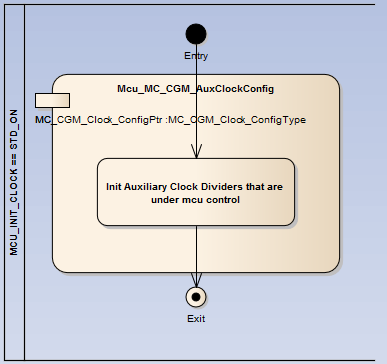
Function này get thông tin Mode hiện tại bằng cách đọc MC\_ME\_GS[S\_CURRENT\_MODE] bitfield. Thông tin này sẽ phục vụ việc ghi cấu hình Clock (Control việc Enable/Disable các nguồn Clocks) vào thanh ghi control của Mode tương ứng. Sau khi cấu hình Clock được ghi vào Mode hiện tại, bước tiếp theo là apply Mode đó với cấu hình đã được updated.

1. Mcu\_MC\_CGM\_SystemClockConfig diagram:



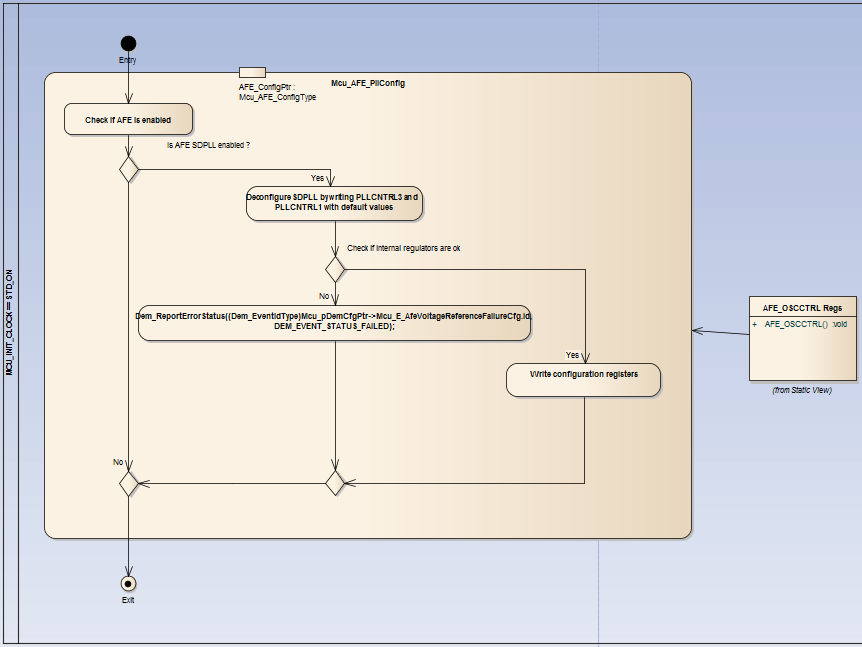
Function này configure lần lượt các thanh ghi System Clock Divider Configuration Register (CGM\_SC\_DCx), các thanh ghi thuộc MC\_ME module này sẽ control các bộ chia của các nhánh của System Clock.

1. Mcu\_MC\_CGM\_AuxClockConfig diagram:

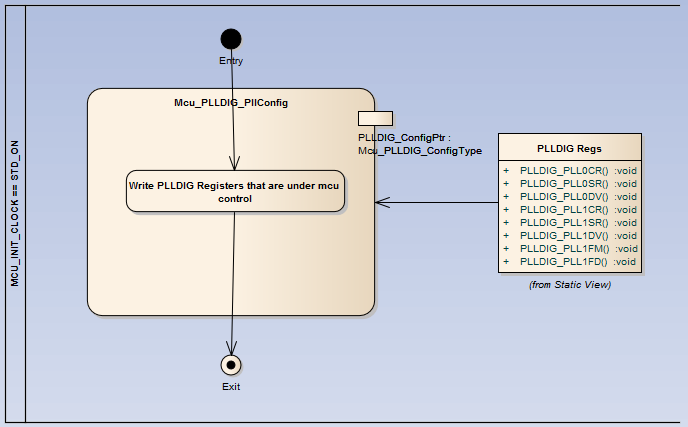


Function này sẽ configure các thanh ghi của các Auxiliary Clock. Xem “**MC\_CGM (Clock Generation Module)”** trong phần 2.3.10 IPV\_MCV4.

1. Mcu\_AFE\_PllConfig diagram:

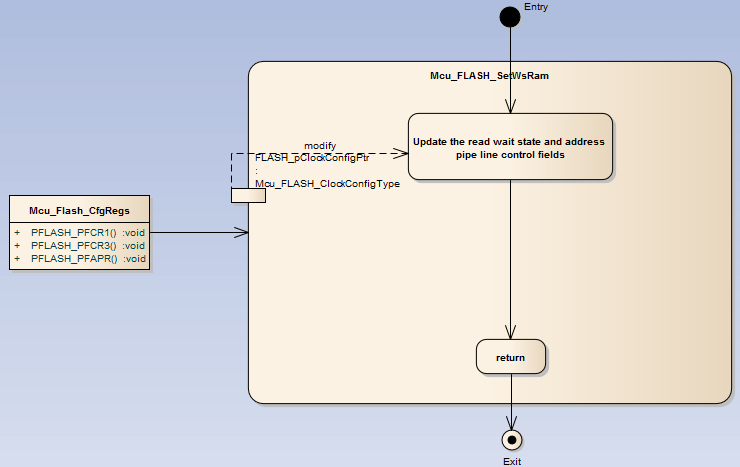


1. Mcu\_PLLDIG\_PllConfig diagram:

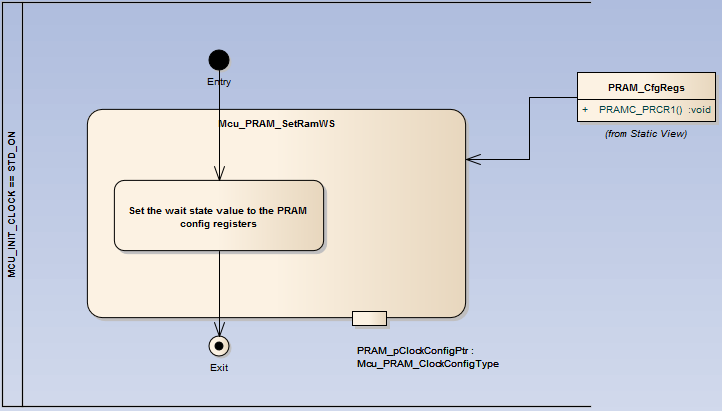


Function này configure các thông số đầu vào của PLL module để tạo ra nguồn Clock đầu ra với giá trị như mong muốn. Xem phần 2.3.12 IPV\_PLLDIG để hiểu về IPV này.

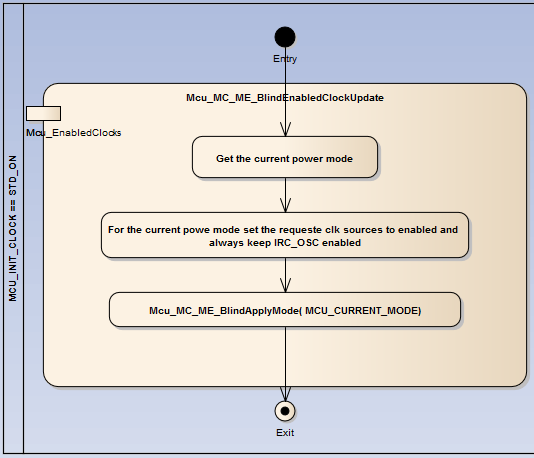
1. Mcu\_FLASH\_SetWS diagram:



1. Mcu\_PRAM\_SetRamWS diagram:

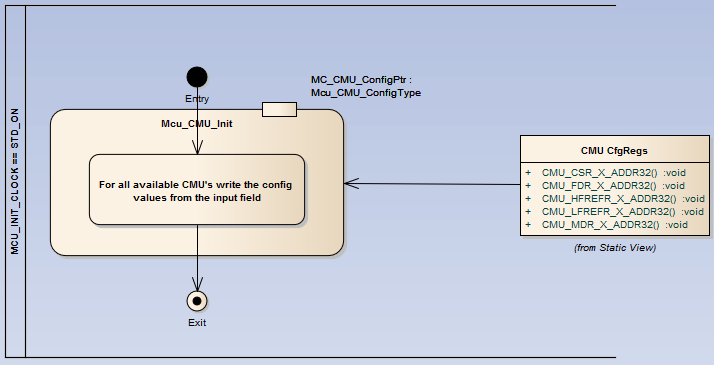


1. Mcu\_MC\_ME\_BlindEnabledClockUpdate diagram:



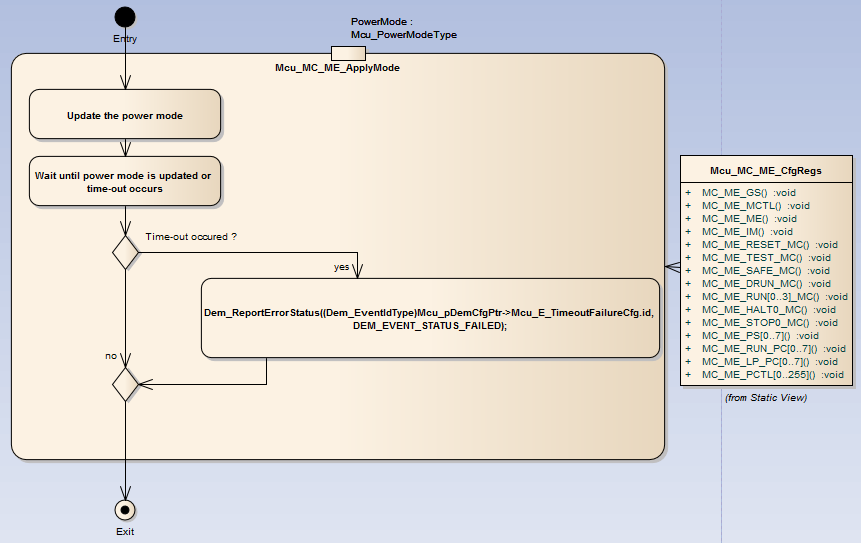
Tương tự Mcu\_MC\_ME\_EnabledClkUpdate function. Điểm khác nhau giữ hai function này nằm ở step3 (khác biệt giữa Mcu\_MC\_ME\_ApplyMode và Mcu\_MC\_ME\_BlindApplyMode ). Sự khác biệt của hai function này nằm ở điểm duy nhất là Mcu\_MC\_ME\_ApplyMode Check quá trình apply Mode có thành công không, còn Mcu\_MC\_ME\_BlindApplyMode chỉ thực hiện apply Mode và không cung cấp bất kì mã lỗi.

1. Mcu\_CMU\_Init diagram:



Sau khi hoàn tất việc configure các nguồn Clock, CMU sẽ được init bằng cách gi lần lượt vào các thanh ghi cấu hình của CMU. Xem 2.3.3 IPV\_CMU đã trình bày về HW này.

1. Mcu\_MC\_ME\_ApplyMode diagram:

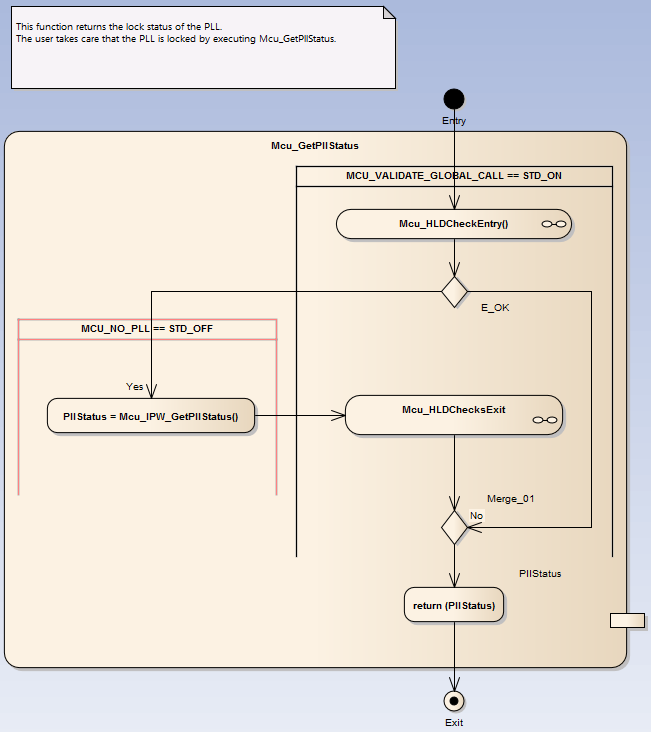


Việc apply Mode sẽ được thực hiện bằng cách ghi hai lần vào thanh ghi Mode Control Register (MC\_ME\_MCTL). Lần 1 sẽ ghi giá trị (TARGET\_MODE | KEY), lần 2 sẽ ghi (TARGET\_MODE | INVERTED KEY). Sau khi ghi, function check MC\_ME\_GS[S\_MTRANS] bit để biết được việc Apply có thành công hay không.

### Mcu\_GetPllStatus

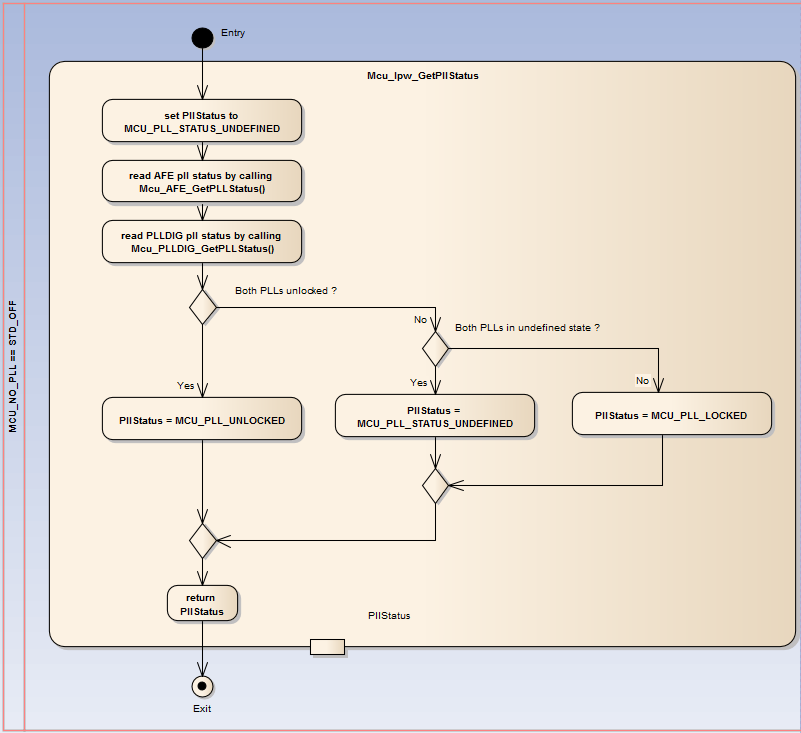
Overview: This function provides the lock status of the PLL (MCU\_PLL\_STATUS\_UNDEFINED / MCU\_PLL\_LOCKED / MCU\_PLL\_UNLOCKED)

#### 1.4.4.1 HLD Layer



1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_GetPllStatus: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.4.2 IPW Layer

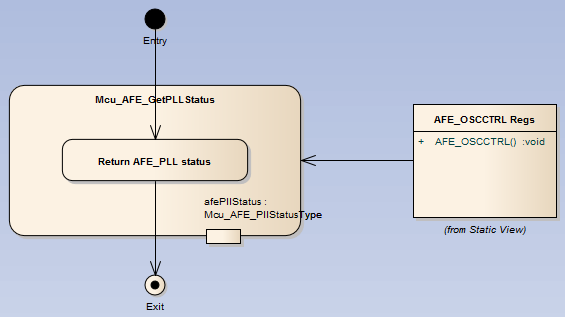


Function này sẽ đọc trạng thái của tất cả các PLL, khi có ít nhất một PLL Clock được LOCKED, function sẽ trả về MCU\_PLL\_LOCKED, các trường hợp còn lại sẽ tả về MCU\_PLL\_UNLOCKED

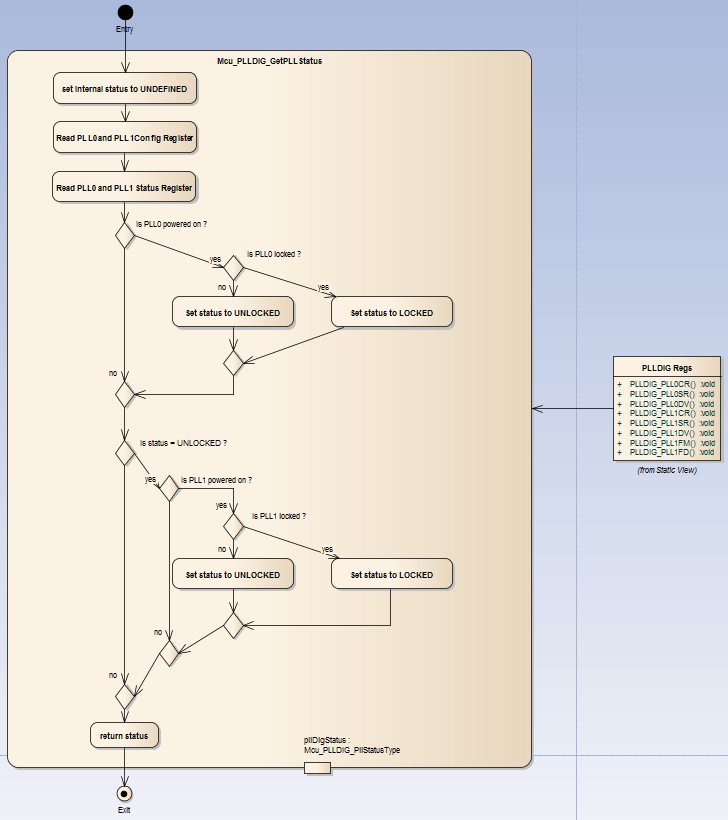
For detail, see **IPV Layer** section

#### 1.4.4.3 IPV Layer

1. Mcu\_AFE\_GetPLLStatus



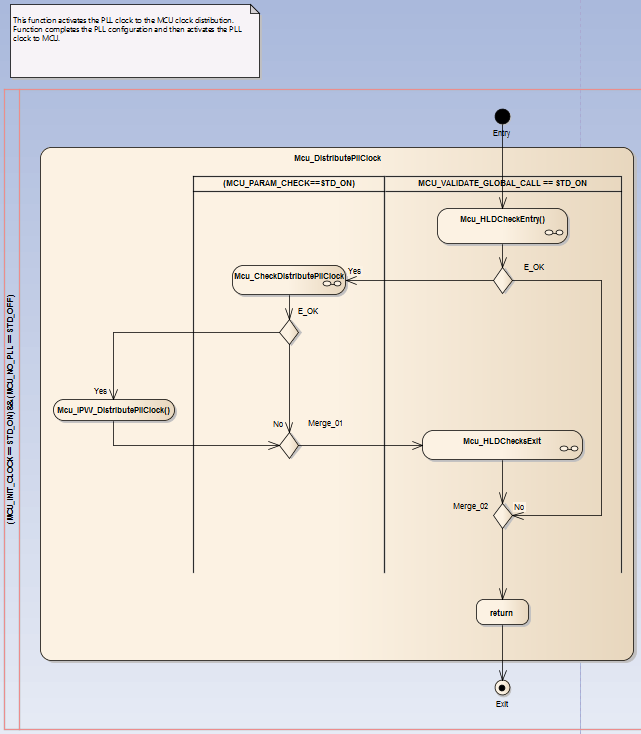
1. Mcu\_PLLDIG\_GetPLLStatus



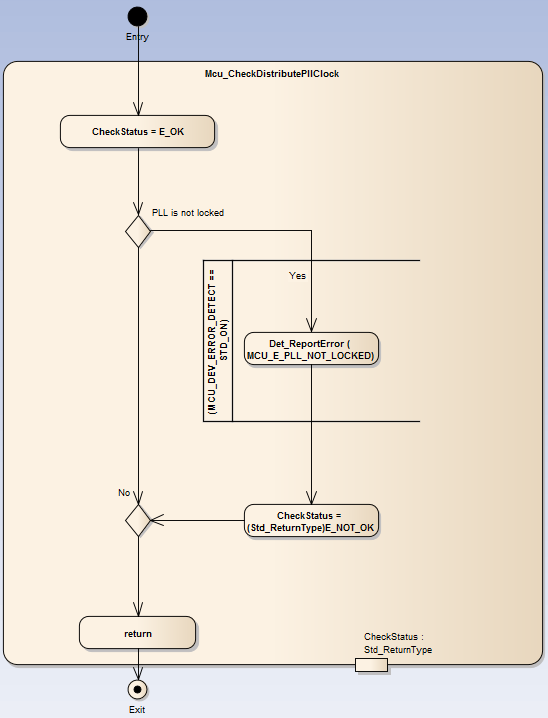
### Mcu\_DistributePllClock

Overview: Function completes the PLL configuration and then activates the PLL clock to MCU. If the MCU\_NO\_PLL is TRUE the Mcu\_DistributePllClock has to be disabled. The function will not distribute the PLL clock if the driver state does not allow it, or the PLL is not stable.

#### 1.4.5.1 HLD Layer



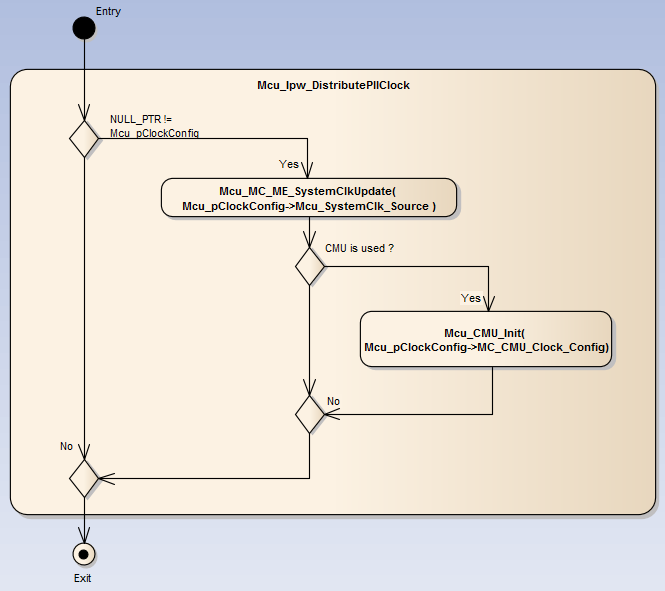
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_CheckDistributePllClock diagram:



Function này kiểm tra trạng thái của PLL, nếu PLL locked function sẽ trả về E\_OK và tiếp tục thự hiện việc Distribute PLL. Nếu PLL unlocked, function sẽ trả về E\_NOT\_OK và raised DET error.

1. Mcu\_Ipw\_DistributePllClock: See IPW Layer section
2. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.5.2 IPW Layer



Mcu\_Ipw\_DistributePllClock sẽ gọi Mcu\_MC\_ME\_SystemClkUpdate sau đó init CMU. Để hiểu được điều này trước tiên chúng ta cần nói rõ hơn về mối liên hệ giữa Mcu\_DistributePllClock và Mcu\_InitClock. Trong tài liệu AUTOSAR specification của MCU module, chúng ta có requirement MCU138 của Mcu\_InitClock (The function Mcu\_InitClock shall start the PLL lock procedure (if PLL shall be initialized) and shall return without waiting until the PLL is locked). Chính vì vậy, trong Mcu\_InitClock, MCU check nếu System Clock khác PLL, MCU sẽ gọi Mcu\_MC\_ME\_SystemClkUpdate để update System Clock. Nếu System Clock là PLL, MCU sẽ hoàn thành việc init Clock mà không thực hiện gì thêm. Trong trường hợp này System Clock sẽ được update khi gọi Mcu\_DistributePllClock API sau khi đã kiểm tra PLL đã được locked.

For detail, see **IPV Layer** section

#### 1.4.5.3 IPV Layer

1. Mcu\_MC\_ME\_SystemClkUpdate diagram: See 1.4.3.3
2. Mcu\_CMU\_Init diagram: See 1.4.3.3

### Mcu\_GetResetReason

Overview: This routine returns the Reset reason that is read from the hardware.

Trên mỗi platform, MCU sẽ định nghĩa một enum type - nơi bao gồm tất cả các Reset Reason trên platform đó. Ví dụ trên RaceRunnerUltra:

typedef enum

{

/\* 'Destructive' Event Status Register (MC\_RGM\_DES) \*/

MCU\_POWER\_ON\_RESET =0x00U, /\*\*< @brief Power on event. RGM\_DES[F\_POR]. \*/

MCU\_SOFT\_DEST\_RESET, /\*\*< @brief Software destructive reset. RGM\_DES[F\_SOFT\_DESC]. \*/

MCU\_FFRR\_RESET, /\*\*< @brief FCCU failure to react reset. RGM\_DES[F\_FFRR]. \*/

MCU\_STCU\_SUF\_RESET, /\*\*< @brief Destructive reset event based on STCU off-line self test and STCU CF (SUF) RGM\_DES[F\_FFRR]. \*/

MCU\_SSSR\_RESET, /\*\*< @brief Flag for SSCM Secure reset RGM\_DES[F\_SSSR]. \*/

MCU\_EDR\_RESET, /\*\*< @brief Functional reset escalation. RGM\_DES[F\_EDR]. \*/

MCU\_LVD\_CORE\_RESET, /\*\*< @brief LVD on core voltage. \*/

MCU\_HVD\_CORE\_RESET, /\*\*< @brief HVD on core voltage. \*/

MCU\_LVD\_ADC\_RESET, /\*\*< @brief LVD on ADC voltage. \*/

MCU\_HVD\_ADC\_RESET, /\*\*< @brief HVD on ADC voltage. \*/

MCU\_LVD\_FLASH\_RESET, /\*\*< @brief LVD on Flash voltage. \*/

MCU\_LVD\_PLL\_RESET, /\*\*< @brief LVD on PLL voltage. \*/

MCU\_LVD\_IO\_RESET, /\*\*< @brief LVD on I/O voltage. \*/

MCU\_LVD\_PMC\_RESET, /\*\*< @brief LVD on PMC voltage. \*/

MCU\_AFE\_LVD0\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_AFE\_LVD2\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_AFE\_LVD3\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_AFE\_LVD4\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_AFE\_LVD5\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_TSR\_DEST\_RESET, /\*\*< @brief Temperature sensor destructive reset. RGM\_DES[F\_TSR\_DEST]. \*/

MCU\_AFE\_LVD7\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_AFE\_LVD8\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_AFE\_LVD6\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_AFE\_LVD9\_RESET, /\*\*< @brief Flag for AFE\_LVD 'destructive' reset. \*/

MCU\_LVD\_MIPI\_RESET, /\*\*< @brief LVD on MIPI voltage. \*/

/\* 'Functional' Event Status Register (MC\_RGM\_FES) \*/

MCU\_EXR\_RESET, /\*\*< @brief Flag for external reset. RGM\_FES[F\_EXR]. \*/

MCU\_ST\_DONE\_RESET, /\*\*< @brief Self-test completed event. RGM\_FES[F\_ST\_DONE]. \*/

MCU\_SOFT\_FUNC\_RESET, /\*\*< @brief Software destructive event. RGM\_FES[F\_SOFT\_FUNC]. \*/

MCU\_FCCU\_HARD\_RESET, /\*\*< @brief FCCU hard reaction request event. RGM\_FES[F\_FCCU\_HARD]. \*/

MCU\_FCCU\_SOFT\_RESET, /\*\*< @brief FCCU soft reaction request event. RGM\_FES[F\_FCCU\_SOFT]. \*/

MCU\_JTAG\_FUNC\_RESET, /\*\*< @brief JTAG initiated reset event. RGM\_FES[F\_JTAG]. \*/

MCU\_TSR\_FUNC\_RESET, /\*\*< @brief Temperature sensor functional reset. RGM\_FES[F\_TSR\_FUNC]. \*/

MCU\_NO\_RESET\_REASON, /\*\*< @brief No reset reason found \*/

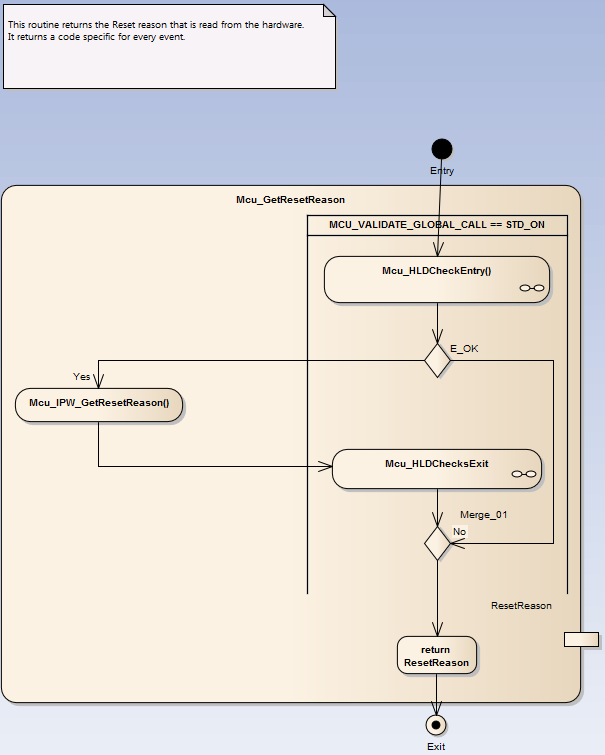
MCU\_MULTIPLE\_RESET\_REASON, /\*\*< @brief More than one reset events are logged except "Power on event" \*/

MCU\_RESET\_UNDEFINED /\*\*< @brief Undefined reset source. \*/

} Mcu\_ResetType;

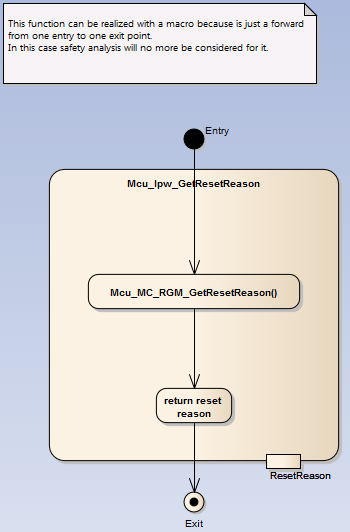
Các Reset Reason này sẽ được mô tả trong thanh gi trạng thái Reset (Reset Event Status Register) trên mỗi platform. Việc Get được thông tin về nguyên nhân Reset cũng đựa trên việc đọc thanh ghi trạng thái này.

#### 1.4.6.1 HLD Layer



1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_GetResetReason: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

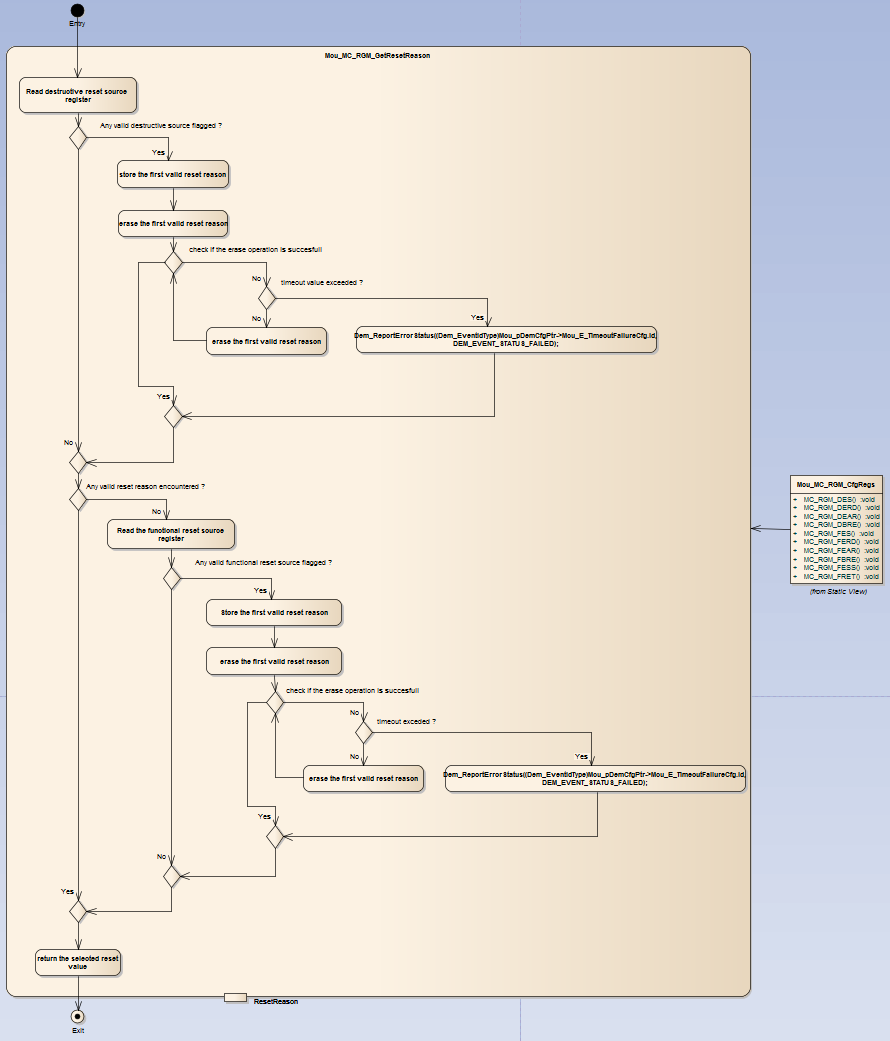
#### 1.4.6.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.6.3 IPV Layer

1. Mcu\_MC\_RGM\_GetResetReason diagram:

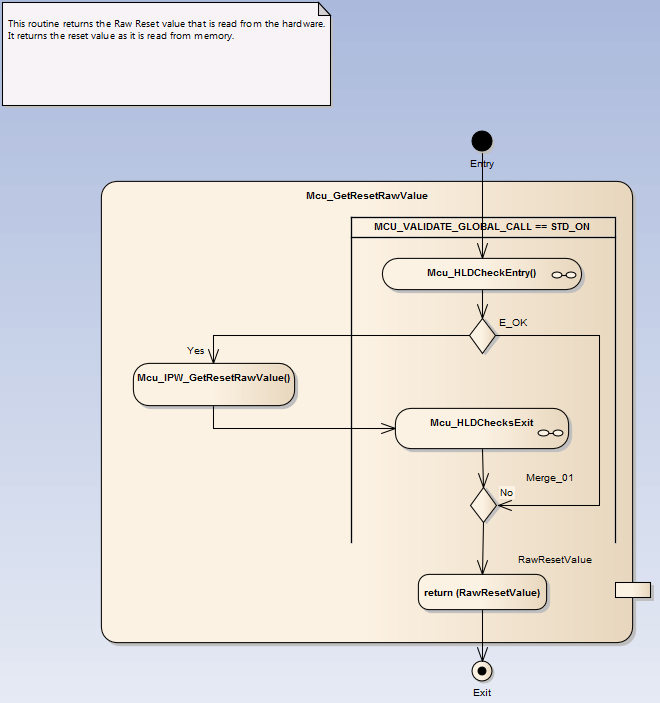


### Mcu\_GetResetRawValue

Overview: This routine returns the Raw Reset value that is read from the hardware.

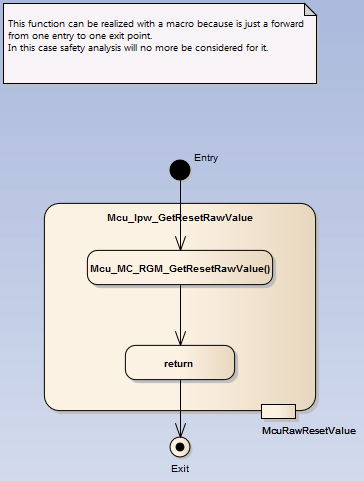
Tương tự như Mcu\_GetResetReason, chỉ khác ở điểm, Mcu\_GetResetReason sẽ trả về nguyên nhân Reset dưới dạng tường minh hơn (Là 1 trong các nguyên nhân trong enum type - Mcu\_ResetType), Mcu\_GetResetRawValue sẽ trả về giá trị của thanh ghi trạng thái Reset. Việc có 2 APIs cùng mục đích cung cấp thông tin về nguyên nhân Reset như đã trình bày không phải là dư thừa. Thực tế hai APIs này sẽ bổ trợ thông tin cho nhau để đưa ra thông tin chính xác về nguyên nhân Reset trong trường hợp có nhiều nguyên nhân Reset đồng thời. Khi đó Mcu\_GetResetReason sẽ chỉ trả về MCU\_MULTIPLE\_RESET\_REASON và chúng ta sẽ không thực sự biết có những nguyên nhân nào trong trường hợp này. Khi đó giá trị trả về của Mcu\_GetResetRawValue sẽ thực sự hữu ích.

#### 1.4.7.1 HLD Layer



1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_GetResetRawValue: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

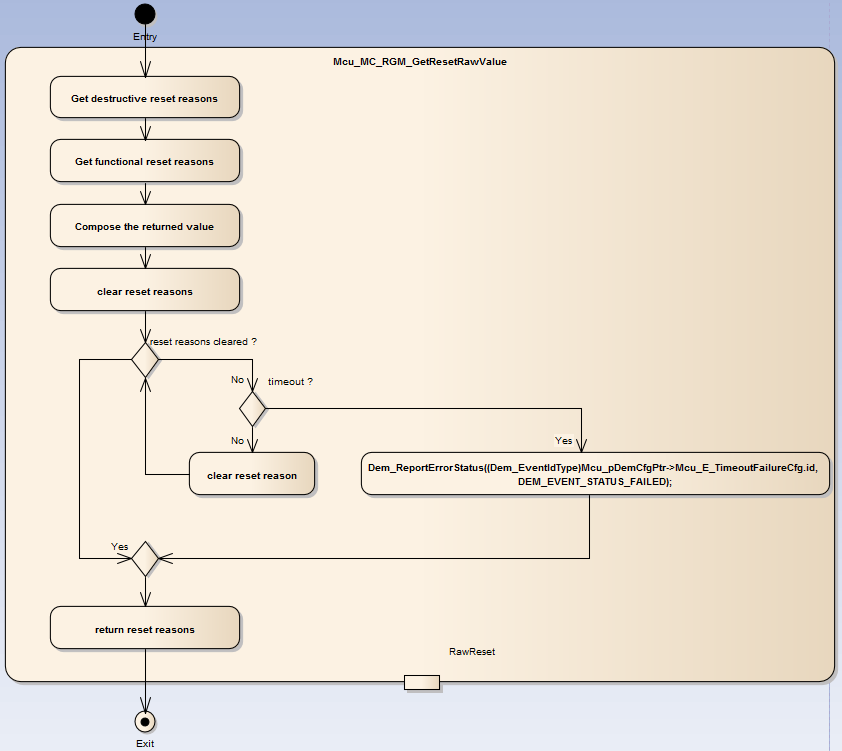
#### 1.4.7.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.7.3 IPV Layer

1. Mcu\_MC\_RGM\_ GetResetRawValue diagram:



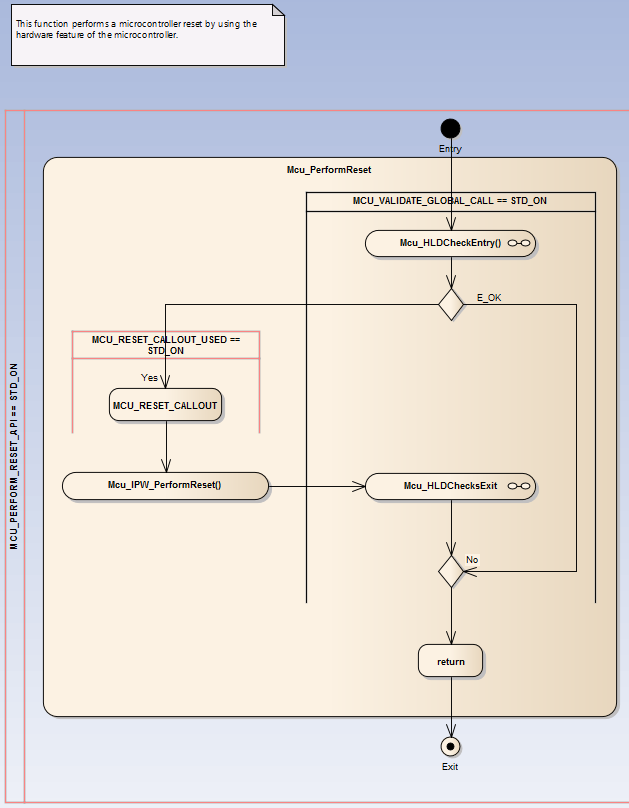
### Mcu\_PerformReset

Overview: This function performs a microcontroller reset by using the hardware feature of the microcontroller. In case the function returns, the user must reset the platform using an alternate reset mechanism.

Với IPV\_MCV4, Mcu\_PerformReset thực chất là chúng ta Apply một mode vời target là RESET Mode (Xem 1.4.9 Mcu\_SetMode). Trên Kinetis, Mcu\_PerformReset được control bởi thanh ghi Core sử dụng IPV\_CORTEXM (Xem 1.3.2 IPV\_CORTEXM).

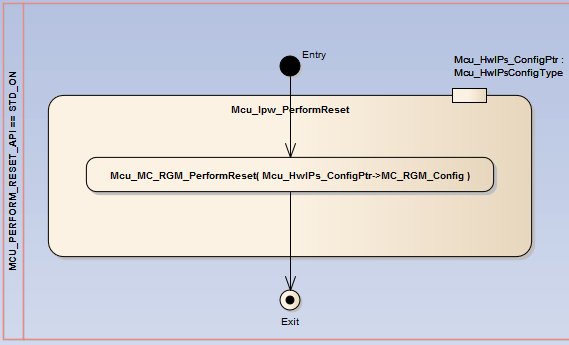
Dưới đây là Mcu\_PerformReset trên RaceRunnerUltra sử dụng IPV\_MCV4

#### 1.4.8.1 HLD Layer



1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_PerformReset: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

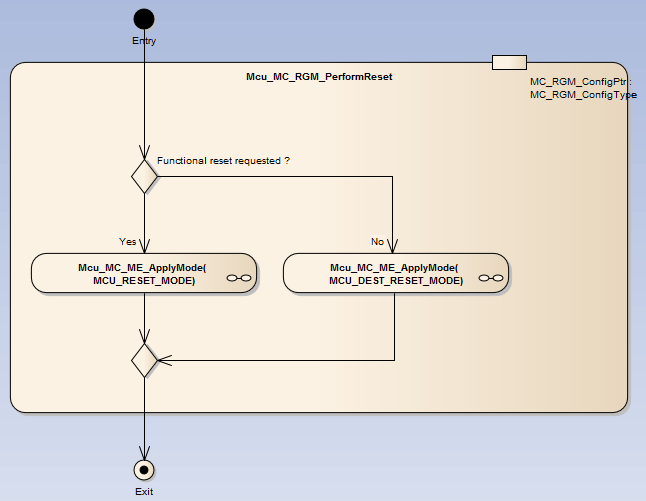
#### 1.4.8.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.8.3 IPV Layer

1. Mcu\_MC\_RGM\_PerformReset diagram:

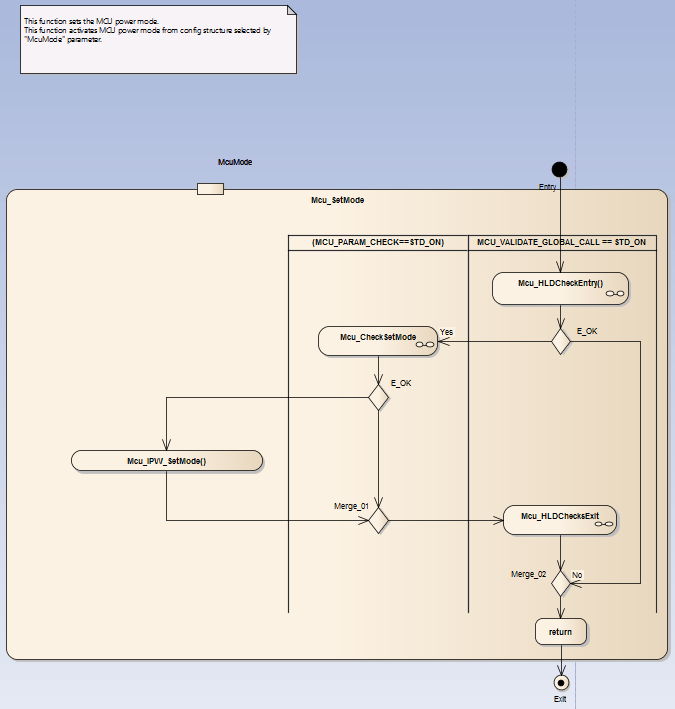


1. Mcu\_MC\_ME\_ApplyMode diagram: See 1.4.3.3

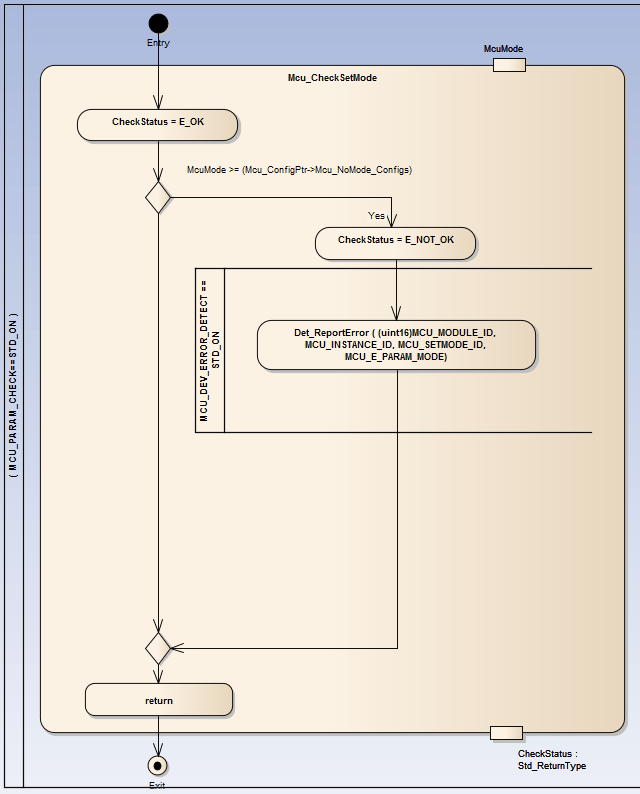
### Mcu\_SetMode

Overview: This function activates MCU power mode from config structure selected by McuMode parameter. If the driver state is invalid or McuMode is not in range the function will skip changing the mcu mode.

#### 1.4.9.1 HLD Layer



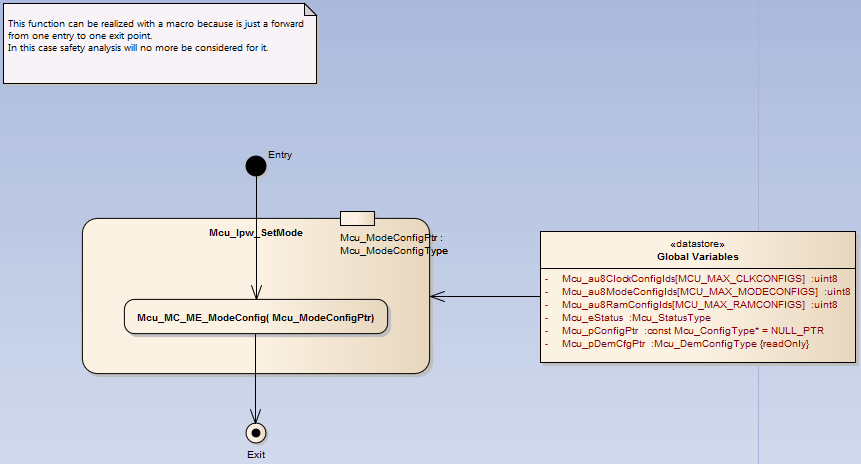
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_CheckSetMode diagram:



Function này tương tự Mcu\_CheckInitRamSection.

1. Mcu\_Ipw\_SetMode: See IPW Layer section
2. Mcu\_HLDChecksExit diagram: See 1.4.1.1

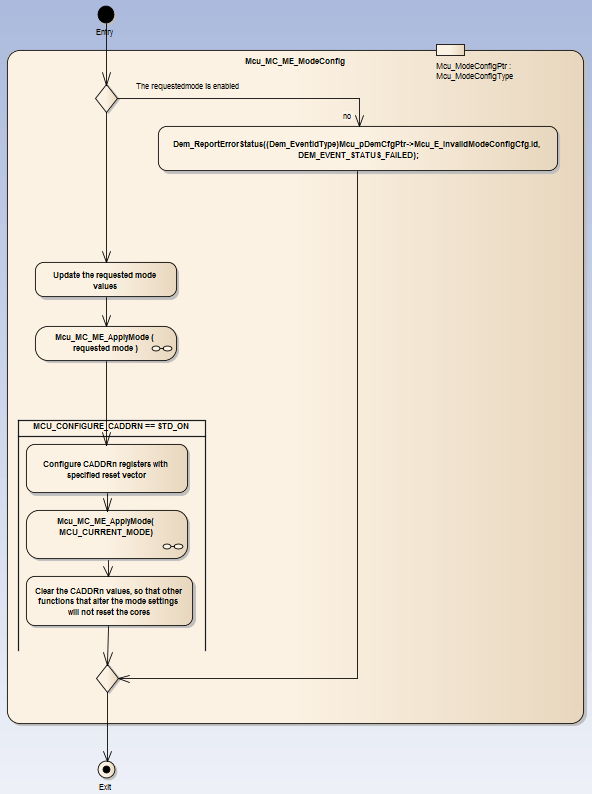
#### 1.4.9.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.9.3 IPV Layer

1. Mcu\_MC\_ME\_ModeConfig diagram:



Step1: Check target Mode đã được Enable chưa thông qua thanh ghi MC\_ME\_ME (Thanh ghi MC\_ME\_ME control việc Enable Mode, Một Mode được Enable mới cho phép Set Mode đó). Nếu target Mode hợp lệ sẽ tiến hành Set Mode, ngược lại sẽ raise DEM error.

Step2: Configure MC\_ME\_CADDRx registers

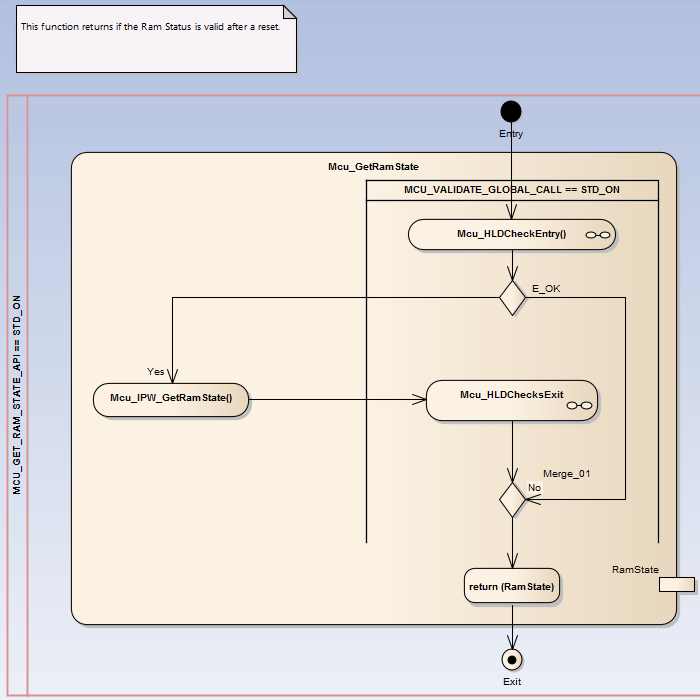
Step3: Configure System Clock, Enable/Disable các nguồn Clock sau đó Apply Mode.

1. Mcu\_MC\_ME\_ApplyMode diagram: See 1.4.3.3

### Mcu\_GetRamState

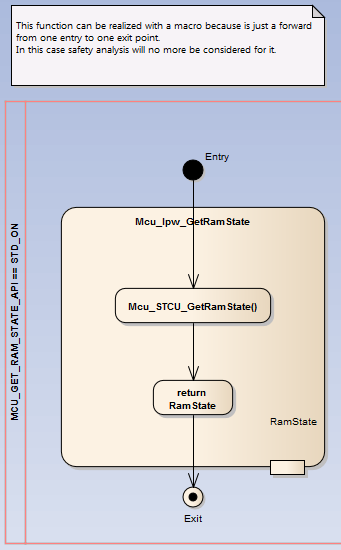
Overview: This function returns if the Ram Status is valid after a reset. The report is get from STCU as a result of MBIST (Memory Built-In Self Tests).

#### 1.4.10.1 HLD Layer



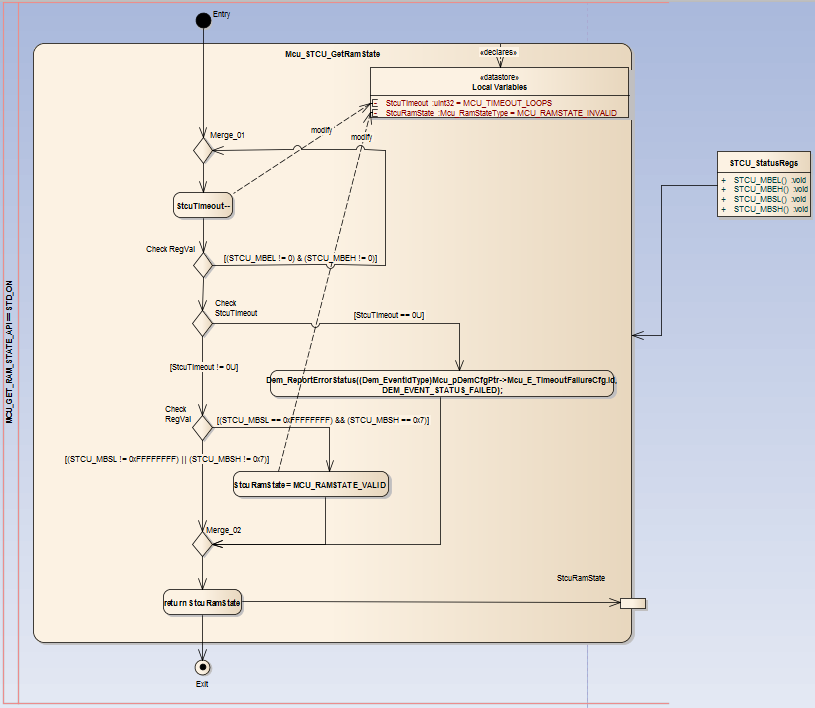
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_GetRamState: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.10.2 IPW Layer



For detail, see **IPV Layer** section

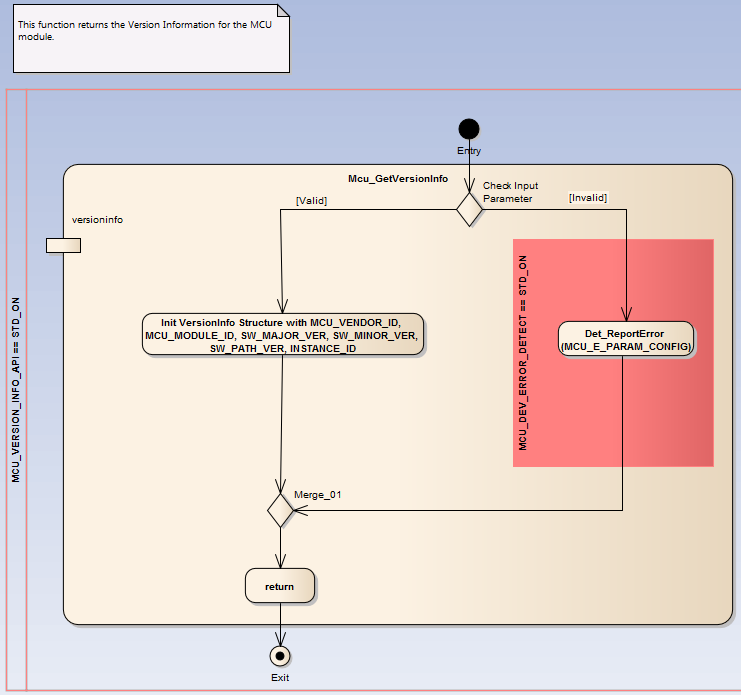
#### 1.4.10.3 IPV Layer



### Mcu\_GetVersionInfo

Overview: This function returns the vendor id, module id, major, minor and patch version.

#### 1.4.11.1 HLD Layer



#### 1.4.11.2 IPW Layer

Doesn’t use

#### 1.4.11.3 IPV Layer

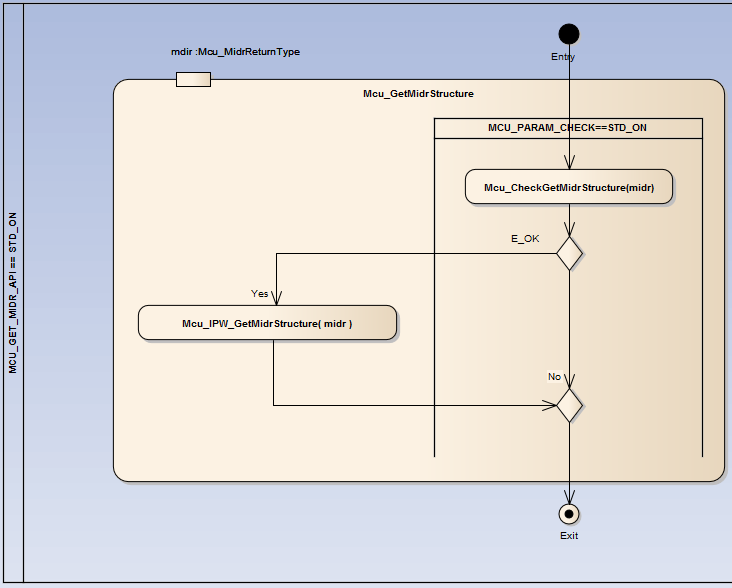
Doesn’t use

### Mcu\_GetMidrStructure

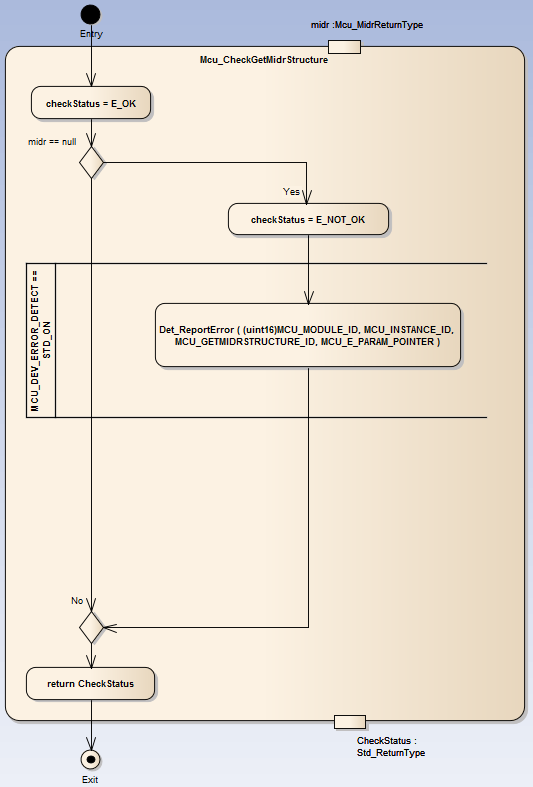
Overview: This function returns the platform dependent Mcu\_MidrReturnType structure witch contains the MIDRn registers.

MIDRn registers cung cấp một số thông tin về HW như: PARTNUM (Cung cấp mã HW, ví dụ Calypso là 5748G…), PKG (Phân biệt các drivatives), Major Mask Revision and Minor Mask Revision (Version của silicon)

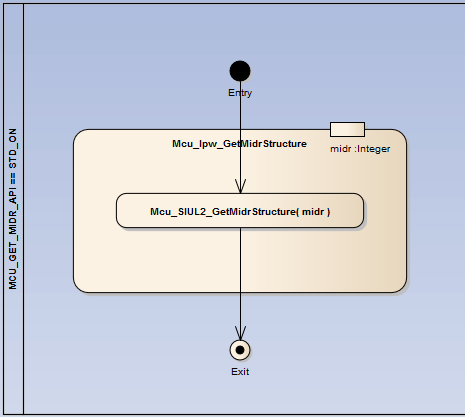
#### 1.4.12.1 HLD Layer



1. Mcu\_CheckGetMidrStructure diagram:

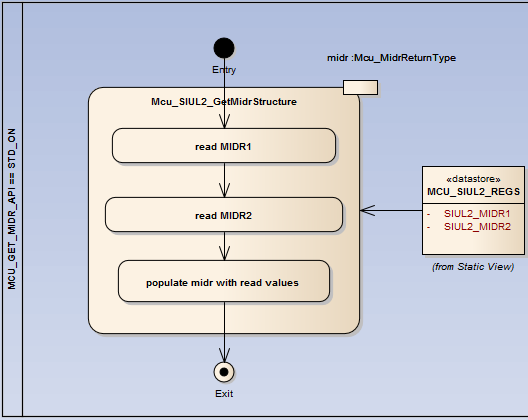


#### 1.4.12.2 IPW Layer



For detail, see **IPV Layer** section

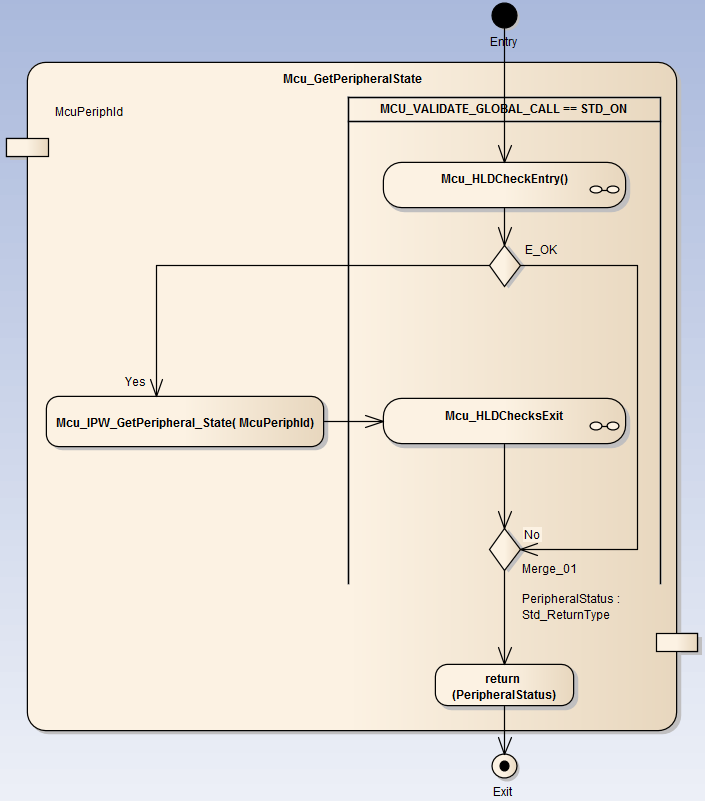
#### 1.4.12.3 IPV Layer



### Mcu\_GetPeripheralState

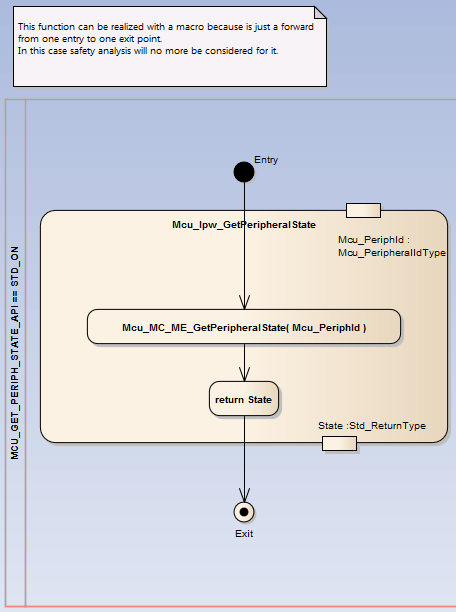
Overview: This function returns one peripheral is frozen or active from MC\_ME HW IP. By reading MC\_ME\_PSx registers (Peripheral Status Register).

#### 1.4.13.1 HLD Layer



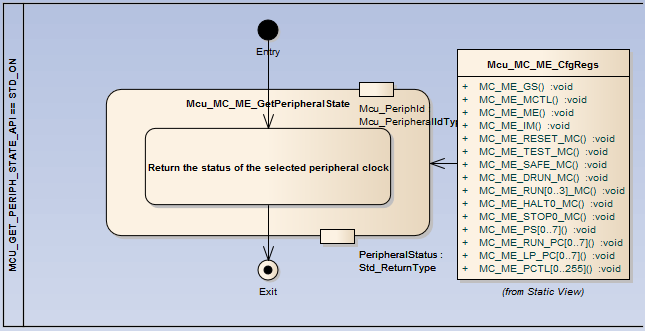
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_GetPeripheralState: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.13.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.13.3 IPV Layer

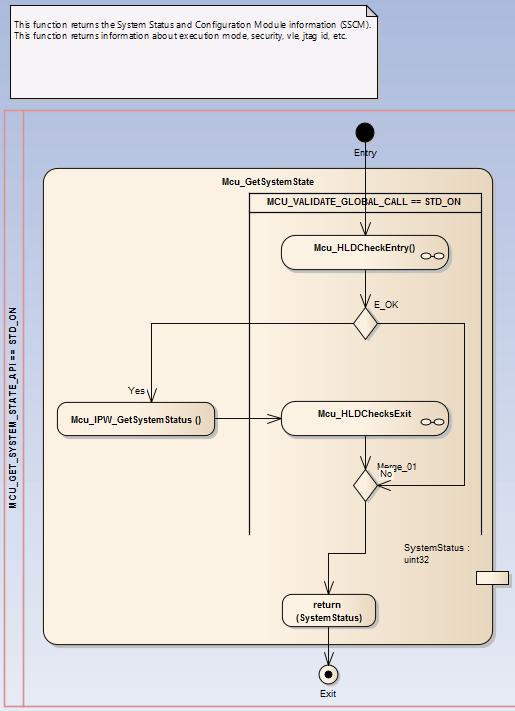


### Mcu\_GetSystemState

Overview: This function returns information about execution mode, security, vle, jtag id.

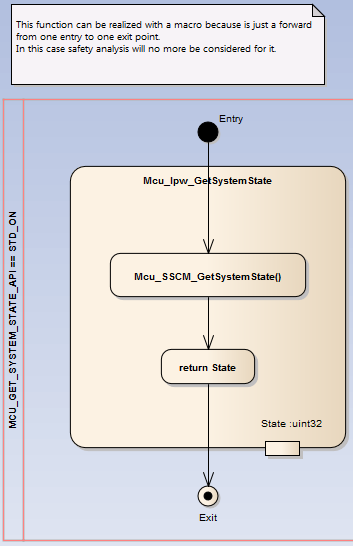
Return giá trị hai registers là SSCM\_STATUS và SSCM\_MEMCONFIG, những registers này cung cấp các thông tin về mode, security, vle, jtag id, etc…

#### 1.4.14.1 HLD Layer



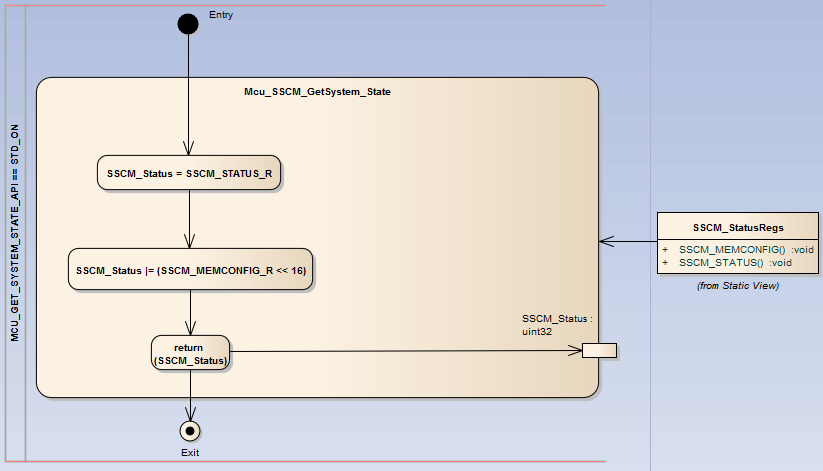
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_GetSystemState: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.14.2 IPW Layer



For detail, see **IPV Layer** section

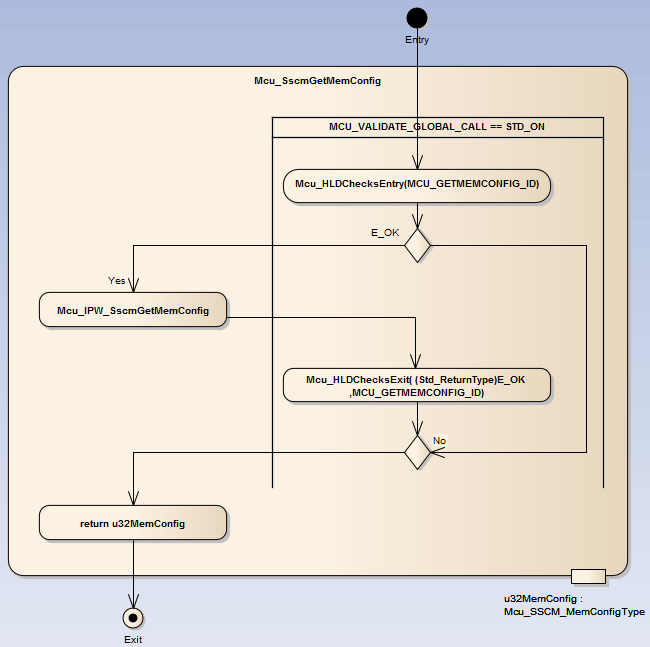
#### 1.4.14.3 IPV Layer



### Mcu\_SscmGetMemConfig

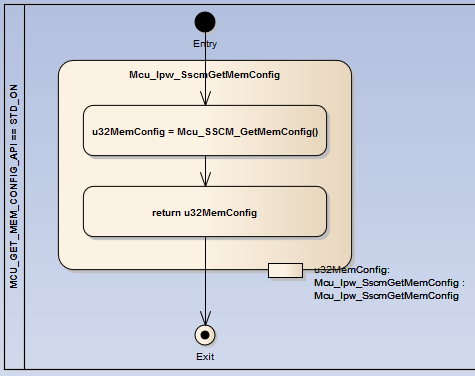
Overview: This function returns the value of the SSCM\_MEMCONFIG register.

#### 1.4.15.1 HLD Layer



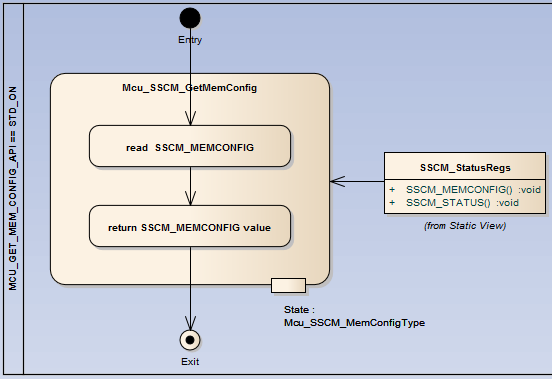
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_SscmGetMemConfig: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.15.2 IPW Layer



For detail, see **IPV Layer** section

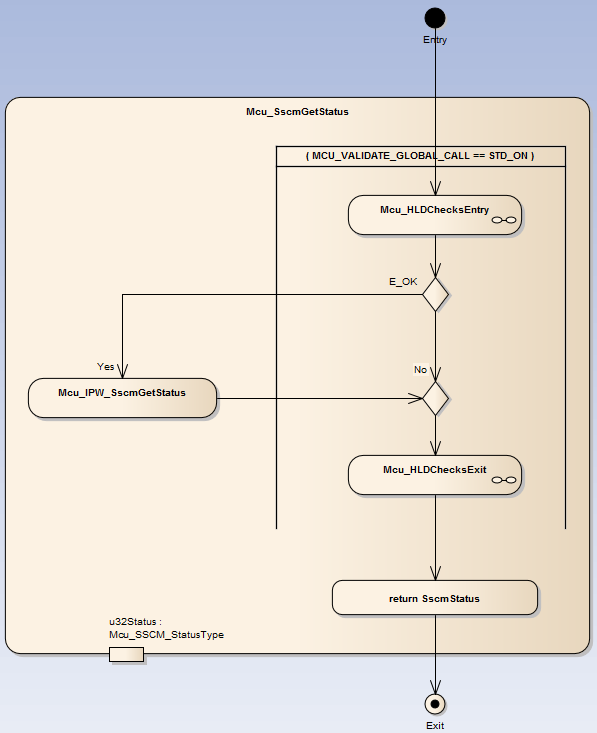
#### 1.4.15.3 IPV Layer



### Mcu\_SscmGetStatus

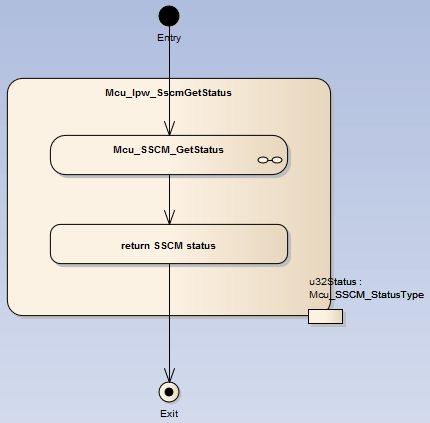
Overview: Return information about execution mode, security, vle. By reading SSCM\_STATUS register.

#### 1.4.16.1 HLD Layer



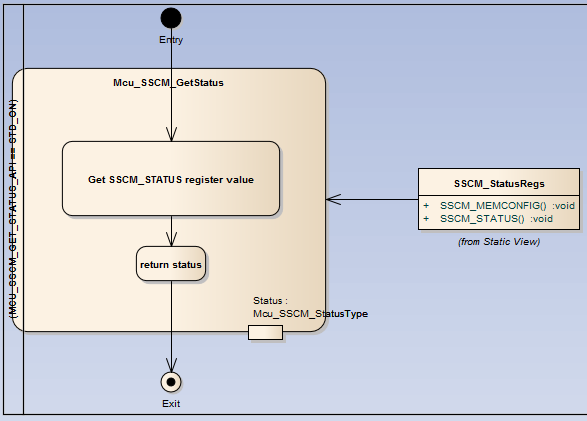
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_SscmGetStatus: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.16.2 IPW Layer



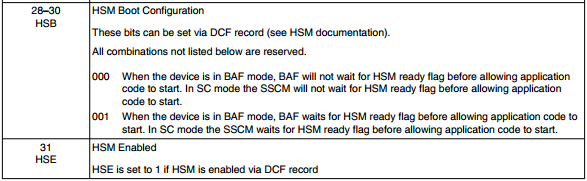
For detail, see **IPV Layer** section

#### 1.4.16.3 IPV Layer

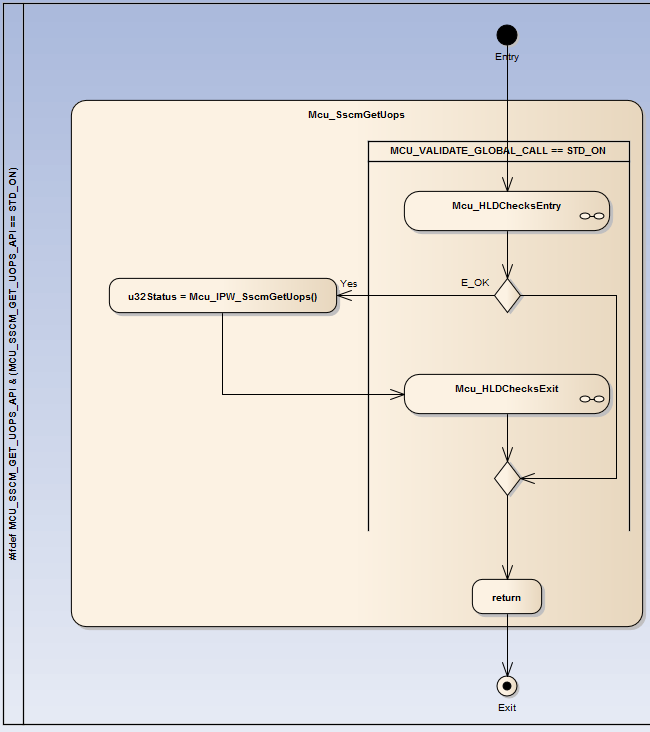


### Mcu\_SscmGetUops

Overview: Return information about User Option Bits, HSM Boot Configuration from readding SSCM\_UOPS register.

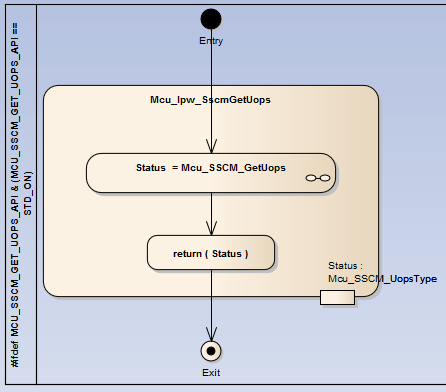


#### 1.4.17.1 HLD Layer



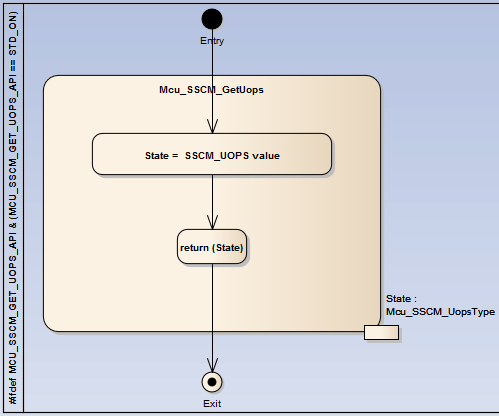
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_SscmGetUops: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.17.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.17.3 IPV Layer

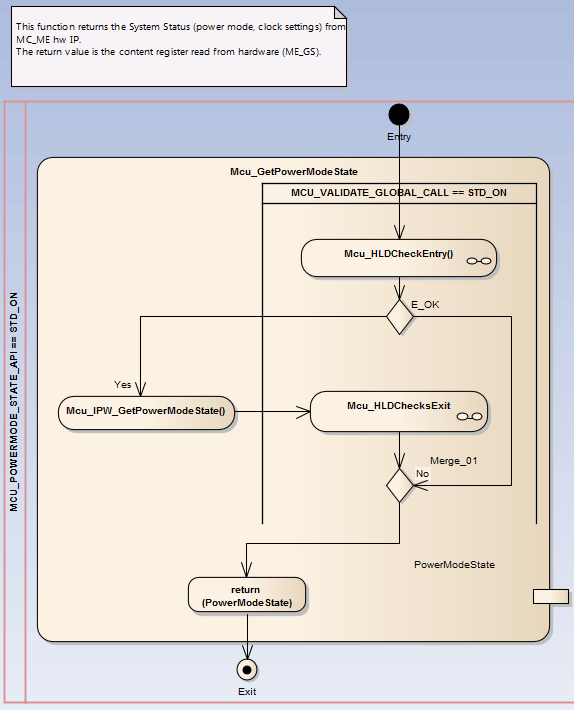


### Mcu\_GetPowerModeState

Overview: The return value is the content register read from hardware (ME\_GS). The return value shall evidentiate the current running mode, clock settings and activation, etc.

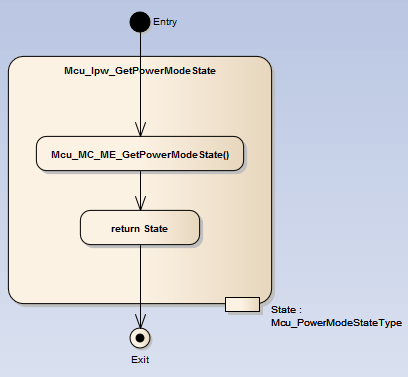
Function đọc MC\_ME\_GS[S\_CURRENT\_MODE] bitfield và giá trị của Mode hiện tại.

#### 1.4.18.1 HLD Layer



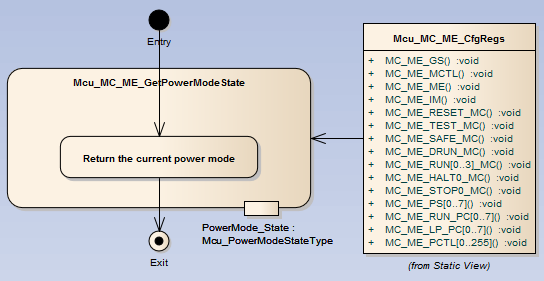
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_GetPowerModeState: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.18.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.18.3 IPV Layer



### Mcu\_GetPowerDomainState

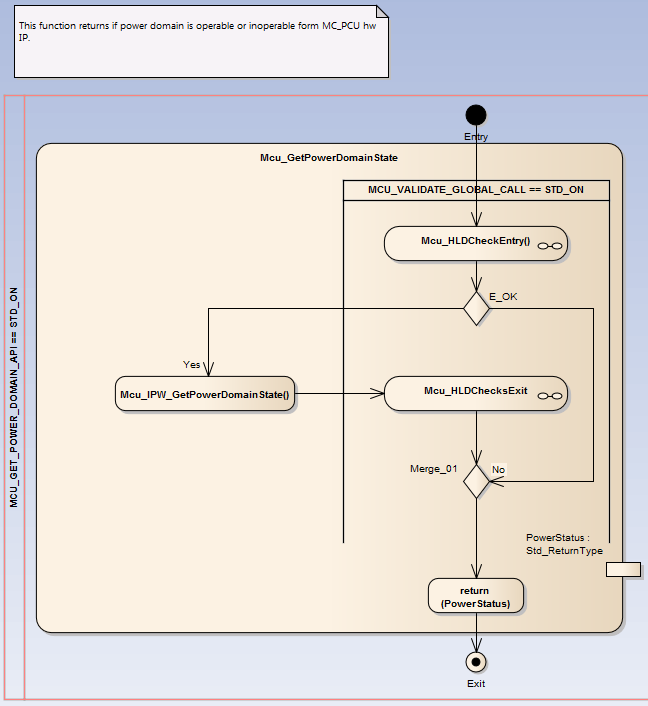
Overview: This function returns if power domain is operable or inoperable form MC\_PCU hw IP.

Function này sẽ đọc MC\_PCU\_PSTAT[PD0] bit.



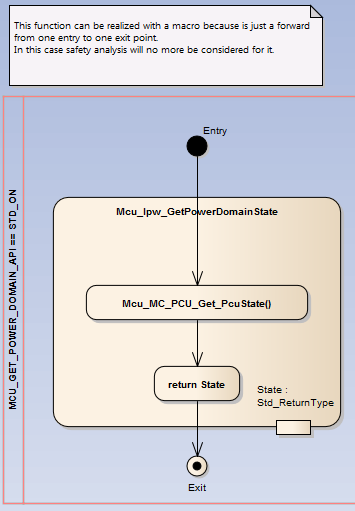
Nếu trạng thái là “Power domain is operable”, function sẽ trả về E\_OK, và ngược lại.

#### 1.4.19.1 HLD Layer



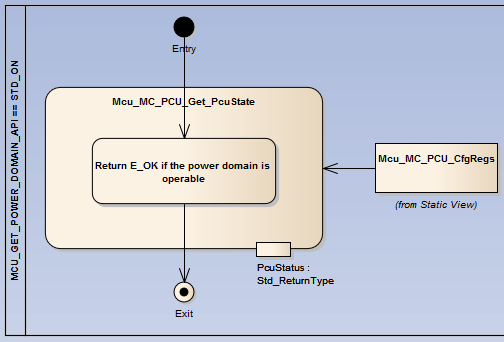
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_GetPowerDomainState: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.19.2 IPW Layer



For detail, see **IPV Layer** section

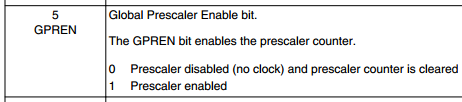
#### 1.4.19.3 IPV Layer



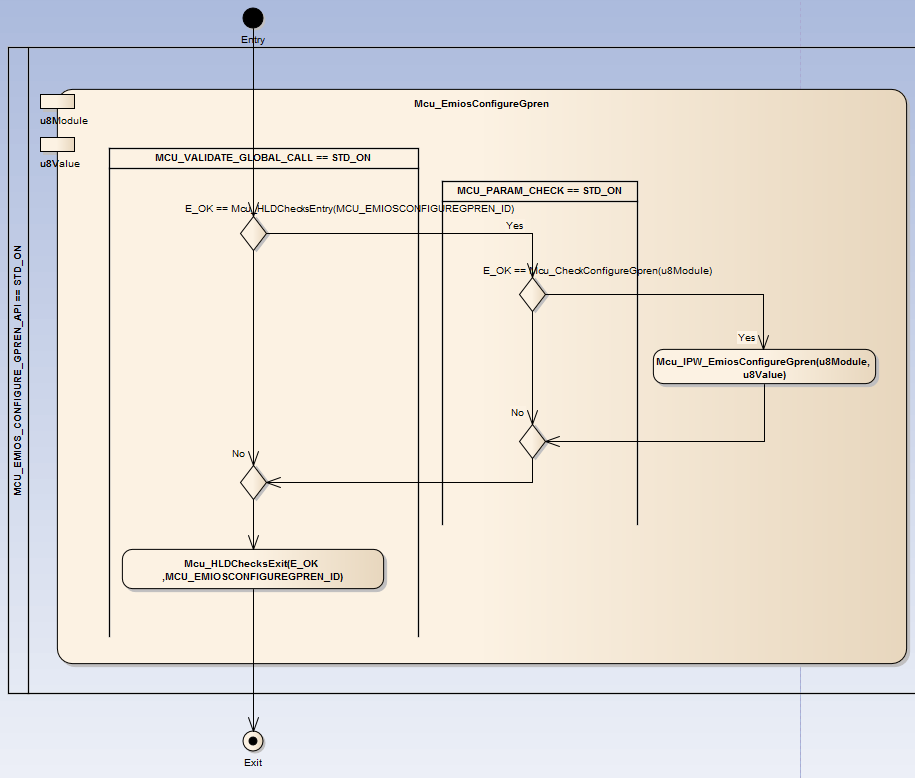
### Mcu\_EmiosConfigureGpren

Overview: This function enables or disables the GPREN bit of the EMIOSMCR register of an addressed eMIOS instance.

Chức năng này cho phép / vô hiệu hóa bit GPREN trong EMIOS\_MCR register:

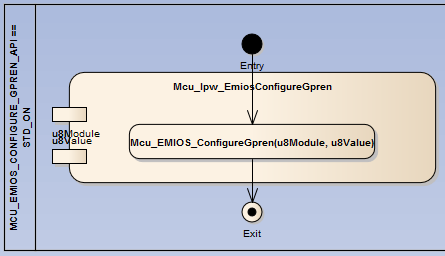


#### 1.4.20.1 HLD Layer



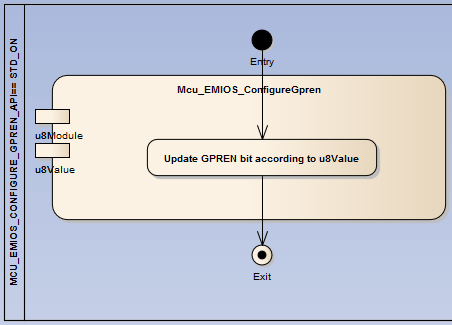
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_IPW\_EmiosConfigureGpren: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.20.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.20.3 IPV Layer

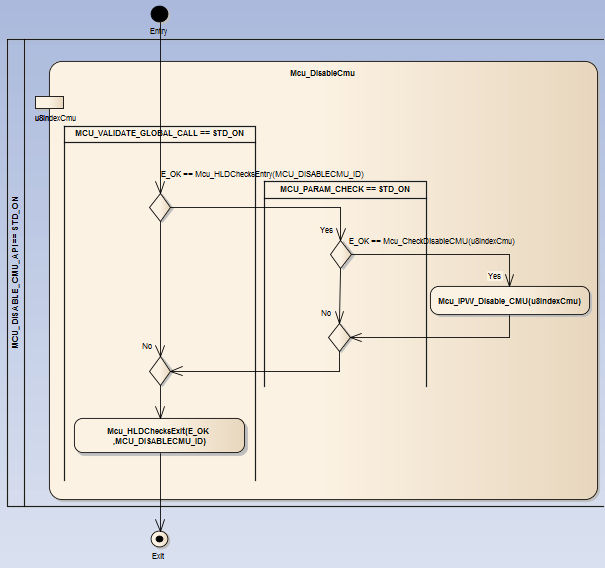


### Mcu\_Disable\_CMU

Overview: This function disables the selected clock monitoring unit.

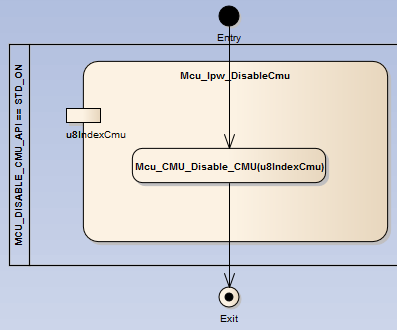
Tương tự Mcu\_CMU\_DeInit, Function Mcu\_Disable\_CMU sẽ clear CMU\_CSR[CME] bit để Disable CMU, tuy nhiên điểm khác ở đây là Mcu\_Disable\_CMU chỉ Disable một CMU (Index của CMU đó là param[in])

#### 1.4.21.1 HLD Layer



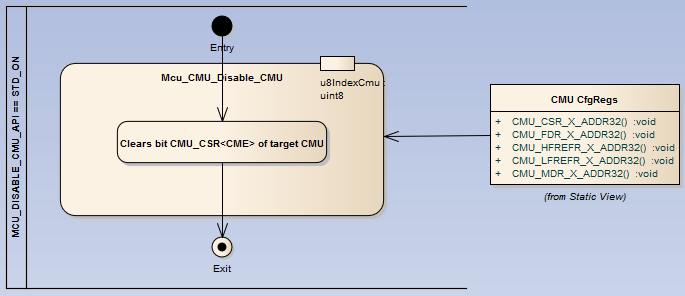
1. Mcu\_HLDChecksEntry diagram: See 1.4.1.1
2. Mcu\_Ipw\_DisableCmu: See IPW Layer section
3. Mcu\_HLDChecksExit diagram: See 1.4.1.1

#### 1.4.21.2 IPW Layer



For detail, see **IPV Layer** section

#### 1.4.21.3 IPV Layer

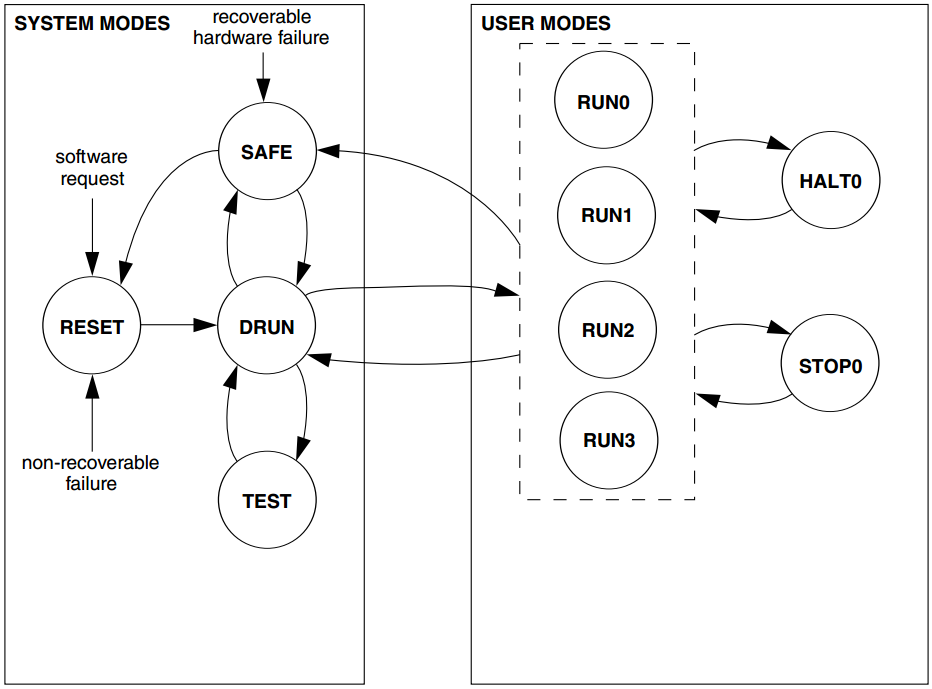


## TIPS &Tricks

### 2.5.1 Take silicon to Low Power Mode and Wake up MCU from Low Power Mode.

Khi MCU ở trong trạng thái Low Power Mode, MCU cần một nguồn wake up để đánh thức MCU rời khỏi trạng thái Low Power Mode khi cần thiết. Thường có hai cách wake up MCU trong trường hợp này là sử dụng Interrupt Even hoặc dùng Wake Up Pin (External Sources). Chúng ta focus vào phần Interrupt Even. Thông thường MCU sẽ được wake up khi interrupt hoặc wakeup event được active. Vấn đề này được mô tả rõ ràng trong RM của mỗi platforms. Tuy nhiên có hai lưu ý khi take MCU to Low Power Mode và wake up MCU:

* MCU không thể take vào Low Power Mode từ một Mode bất kì. Việc take MCU vào Low Power Mode phải tuân thủ Mode Diagram trên mỗi platform. Ví dụ sau trên RaceRunnerUltra:



Nhìn vào Mode Diagram ta có thể thấy MCU chỉ có thể vào được trạng thái Low Power Mode (HALT0 and STOP0) khi mà trạng thái trước đó là RUNx. Và chúng ta không thể đưa MCU vào Low Power Mode từ DRUN or SAFE…

* Ngoài việc chuẩn bị nguồn Wake Up cho MCU trước khi vào Low Power Mode, chúng ta cần chú ý duy trì nguồn Wake Up đó. Ví dụ: Khi sử dụng Timer Interrupt để Wake Up, Timer cần được start trước khi take silicon to Low Power Mode và đảm bảo về mặt thời gian sao cho Interrupt chỉ xảy ra sau khi MCU đã ở trạng thái Low Power Mode, thêm vào đó cần đảm bảo rằng Timer được phép hoạt động trong Low Power Mode bằng cách duy trì clock và cấu hình Timer được phép Active trong Low Power Mode.

### 2.5.2 Peripheral issue on ARM silicon.

Trên các dòng silicon ARM sử dụng IPV\_MCV4 xảy ra issue như sau: Không thể InitClock thành công khi các Peripherals đang ở trạng thái Active. Nguyên nhân là khi các Peripherals đang ở trạng thái Active và đang sử dụng 1 nguồn Clock nào đó và nguồn Clock đó bị Disable trong Mcu\_InitClock, khi đó silicon không thể Apply cấu hình Clock với nguồn Clock của Peripheral đang sử dụng bị thay đổi. Để khắc phục trường hợp này. Sequence sau được suggest sử dụng khi gọi các API của MCU:

Step1: Mcu\_Init

Step2: Mcu\_SetMode /\* To gate all clock peripherals \*/

Step3: Mcu\_InitClock /\* Re-initialize the clock tree \*/

Step4: Check Pll status and wait untill it is locked /\* If needed \*/

Step5: Mcu\_DistributePllClock /\*If needed \*/

Step6: Mcu\_SetMode /\* To switch the Mcu to normal operational mode \*/

## Q&A

# Appendix