Internal Specification

Development of Error Control Module (ECM) for RH850/P1M

(v1.2)

Summary:
This document describes the Detail Design Specification of Error Control Module (ECM).

Reference Manuals				
No.	**Title name**	**Document number**	**Description**	**Path**
1	SC-HEAP_E3 Modeling guideline (Rev. 4.00)	IDF-14-010278-01	This document describes the Guideline for peripheral macro development which is connected to SC-HEAP_E3 simulator (***File***: SC-HEAP_E3_Modeling_Guideline.pdf)	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/02_MCS_Project/From_MCS
2	SC-HEAP_E3 PYTHON I/F function specification (v2.0)	LLWEB-00105192 MSS-SG-12-0062-02	The document describes how to use python interface (***File***: SC-HEAP_E3 Python IF_t.pdf)	
3	M40PF common requirement (Rev1.11)	REQ-SLD-12-010	The common requirement for M40PF models (***File***: REQ-SLD-12010_M40PF_Common.ppt)	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/REQ/2012
4	RH850 P1M/ECM model development: Requirement Specification (Rev1.0)	REQ-SLD-16010	Detail requirement of ECM model (***File***: REQ-SLD-16010_P1M_ECM.pptx)	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/REQ/2016
5	RH850/P1x Group user's manual (Rev.1.20)	-	User's manual for RH850/P1x Group (***File***: r01uh0436ej0120-rh850p1x.pdf)	**Server:** /shsv/sld/ipp/From_RT/P1M_ECM/
6	Difference points between ECM of CC-Cube and P1M	-	Difference points between ECM of P1H-C and P1M (***File***: ECM difference_P1M_P1H-C_161031v1.xlsx)	
7	P1M/ECM target specification	-	Target specification of ECM (***File***: P1M_1chip_TS_V02.41_ECM.pdf)	Redmine #71996
8	P1M/ECM's implemented classes: detail specification	-	Detail specification of ECM's implemented classes (***File***: INT-SLD-16010_refman.pdf)	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/INT/2016/PDF
9	M40-PF Error Control Module "uhiapecm0020" Target Specification (Rev.1.0)	LLWEB-00026493	Target specification of ECM (***File***: uhiapecm0020_target_speicfication_ver1.pdf)	-
10	PFC1A Error Control Module target specifications (Rev.1.2)	-	Target specification of ECM (***File***: PFC1A_ECM_translated.pdf)	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/RVC_documents/20

				1109_M40PF/
11	Internal Specification for ECM improvement Phase 3 (CC-Cube) (v1.18)	INT-SLD-13018	Internal Specification for ECM improvement Phase 3 (CC-Cube) (***File***: INT-SLD-13018.pdf)	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/INT/2013/PDF
12	Confirmation list for P1M/ECM	CFM-SLD-16010_ECM_P1M	Confirmation points for ECM (P1M) (***File***: CFM-SLD-16010_ECM_P1M.xlsx)	**DMS:** Documents/1. General Documents/010_ENG/140_FrontEnd/Project/01_SLD/2_SLD_Project/Model_Documents/01_Project_Document_Management/CFM/2016

Table of Contents

Index of Figures

Index of Tables

1. Model summary

- The Error Control Module (ECM) collects error signals coming from different error sources and monitoring circuits. It also outputs error signals and generates interrupts and internal reset signals.
- This model is developed for RH850/P1M.
- Registers of model can be accessed to read/write via target sockets (of TLM target interface).
- Both loosely time (LT) mode and approximately time (AT) mode are supported.
- This model supports little endian mode as the endian of APB bus interface.

Note: *Hereafter, Error Control Module is simply called "ECM".*

2. Supported features

Table 2.1: Feature of model

Feature	Description		HWM chapter
	Hardware	Model	
Max frequency of APB bus clock (pclk clock port is connected to CLK_HSB clock signal)	80 MHz	Unlimited frequency. There is no setting condition.	*12.2/Table 12.1(ref[5])*
Read/Write registers	Use bus interface	Use TLM target socket	-
Reset	Hardware reset (assert/negate reset signal)	Hardware reset (assert/negate reset signal) Software reset (set by command AssertReset) (*)	-
Safety processing	Error flag set	<-	*32.1/Table 32.1(ref[5])*
	EI level (mask-able) interrupt generation: EI level interrupt generation can be controlled (enabled/disabled) for individual errors	<-	
	FE level (non-mask-able) interrupt generation: FE level interrupt generation can be controlled (enabled/disabled) for individual errors	<-	
	Internal reset (ECMRES) generation: Internal reset (ECMRES) generation can be controlled (enabled/disabled) for individual errors	<-	
	ERROROUT output: Pin output mask can be controlled (enabled/disabled) for individual errors. Output can be toggled in response to a timer input or made at a fixed level	<-	
Error status	The ECM incorporates the ECM master/checker error source status register, which can be used to confirm the error status from the error flag	<-	

Debug, self-diagnosis	Pseudo errors can be generated for debug and self-diagnosis	<-	
	The status of the ERROROUT pin is monitored by a loopback function and reflected to an internal register and can be confirmed by reading the register	<-	
Timeout function	The ECM incorporates a function that generates an ERROROUT output or internal reset (ECMRES) when the count value of the delay timer matches with the delay timer compare register	<-	
Register protection	A write-protection with a special sequence is incorporated to protect registers from inadvertent write access	<-	
Others	The ECM is duplexed. The ECM incorporates the ERROROUT pin. The ERROROUT outputs from the ECM master and ECM checker are constantly compared. If they do not match, an ECM compare error (error source 29) occurs	<-	

Notes:

- The symbol "<-" means that these features are supported as description in the hardware manual (ref[5]/Chapter 32.1/Table 32.1).

- All features described in HWM (ref[5]/Chapter 32.1/Table 32.1) are listed in "Table 2.1/column Hardware" and they are supported in model as description in "Table 2.1/column Model".

- (*) This command is described in Chapter 6.4

3. Block diagram

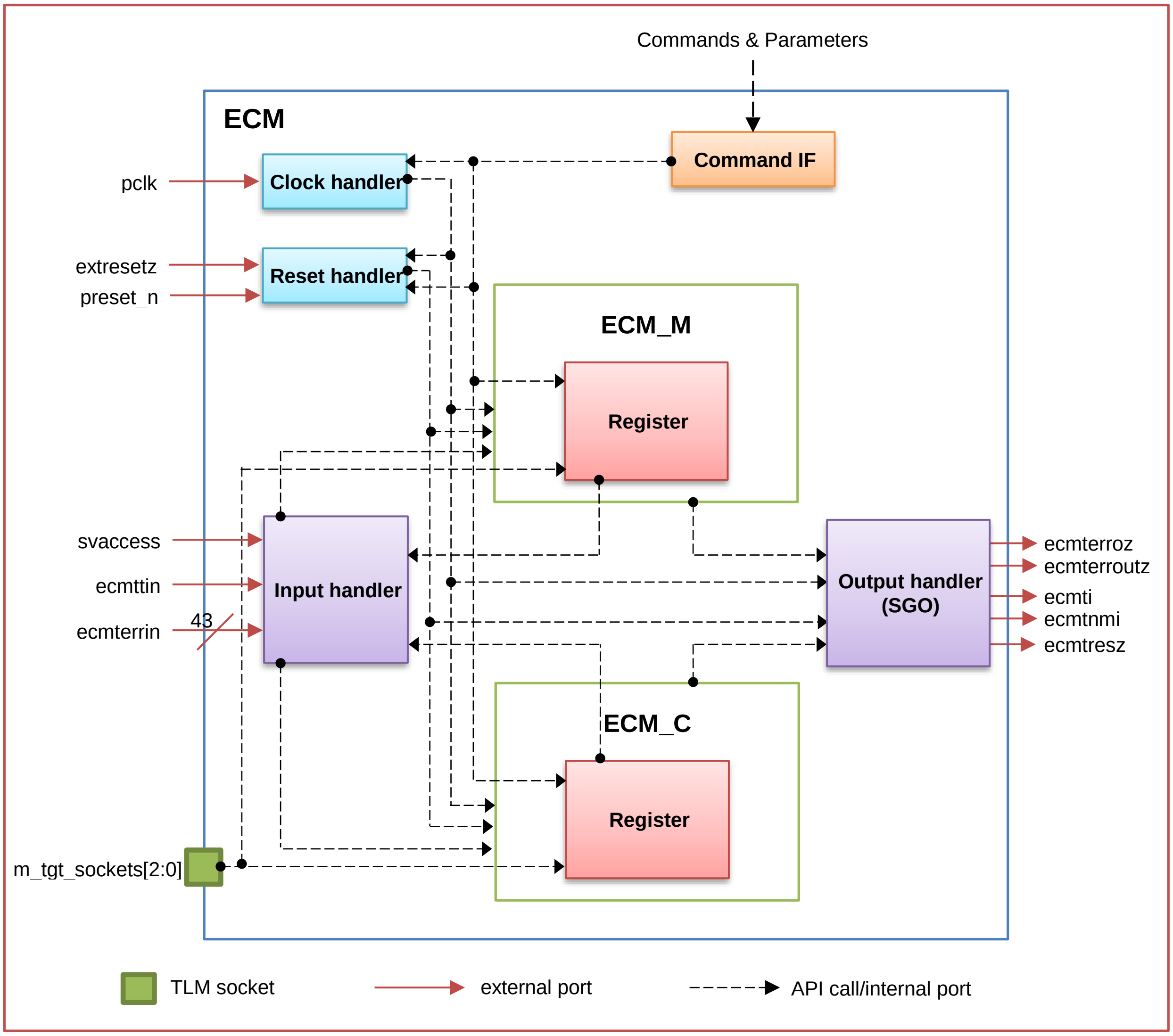


Figure 3.1: General block diagram

Explanation:

- ECM model has 3 target sockets used for read/write accessing to model's registers:
  - ECM's registers are allocated on 3 address areas: Common address area (the base address is indicated by <ECM_base>), Master address area (the base address is <ECMM_base>) and Checker address area (the base address is <ECMC_base>) (1*). (refer to Table 4.1 for more detail about register address of this model)
  - Each target socket is corresponding to an address area of ECM model (m_tgt_sockets[0] is used to access Common address area; m_tgt_sockets[1] is used to access Master address area ; m_tgt_sockets[2] is used to access Checker address area).

  - Data from bus is transferred to this model via TLM target interface (target sockets). Then, the data is arbitrated to transfer to Master block (ECM_M) or Checker block (ECM_C).
- ECM includes 7 blocks:
  - “Clock handler” block receives external input clock (pclk) and provides the clock to other blocks.
  - “Reset Handler” block handles reset signals (preset_n, extresetz) and correlative reset commands.
  - “Command IF” block handles commands and parameters which are input from users. (refer to Chapter 6.4 for the list of commands and parameters)
  - “ECM_M” and “ECM_C” blocks has their own registers. The read/write accessing to registers is handle by themselves. Besides, these blocks control model’s operation (e.g: interrupt issuing, error output issuing, internal reset issuing...).
  - “Input handler” block receives external input signals. After disposing those input signals, this block will transfer signals to “ECM_M” and “ECM_C” blocks. (refer Figure 3.2)
  - After signals are received in the “Input handler” block, “ECM_M” and “ECM_C” blocks will process operation and notify operation result to “Output handler” block. The “Output handler” block will dispose the results (e.g: compare the results) and output signals to other models. (refer Figure 3.3)

***<u>Note</u>**: - (1*) The address area for common registers is used for both master and checker registers. Writing to this common area is performed by master and checker registers simultaneously. Reading from the common area reads the master register's value.*

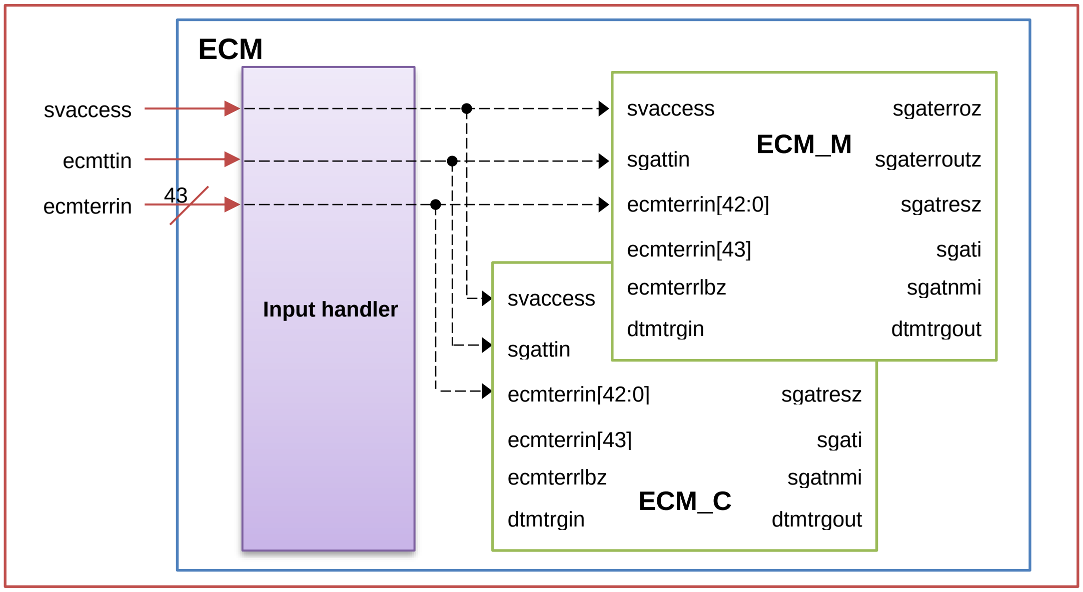


Figure 3.2: Disposing input signals

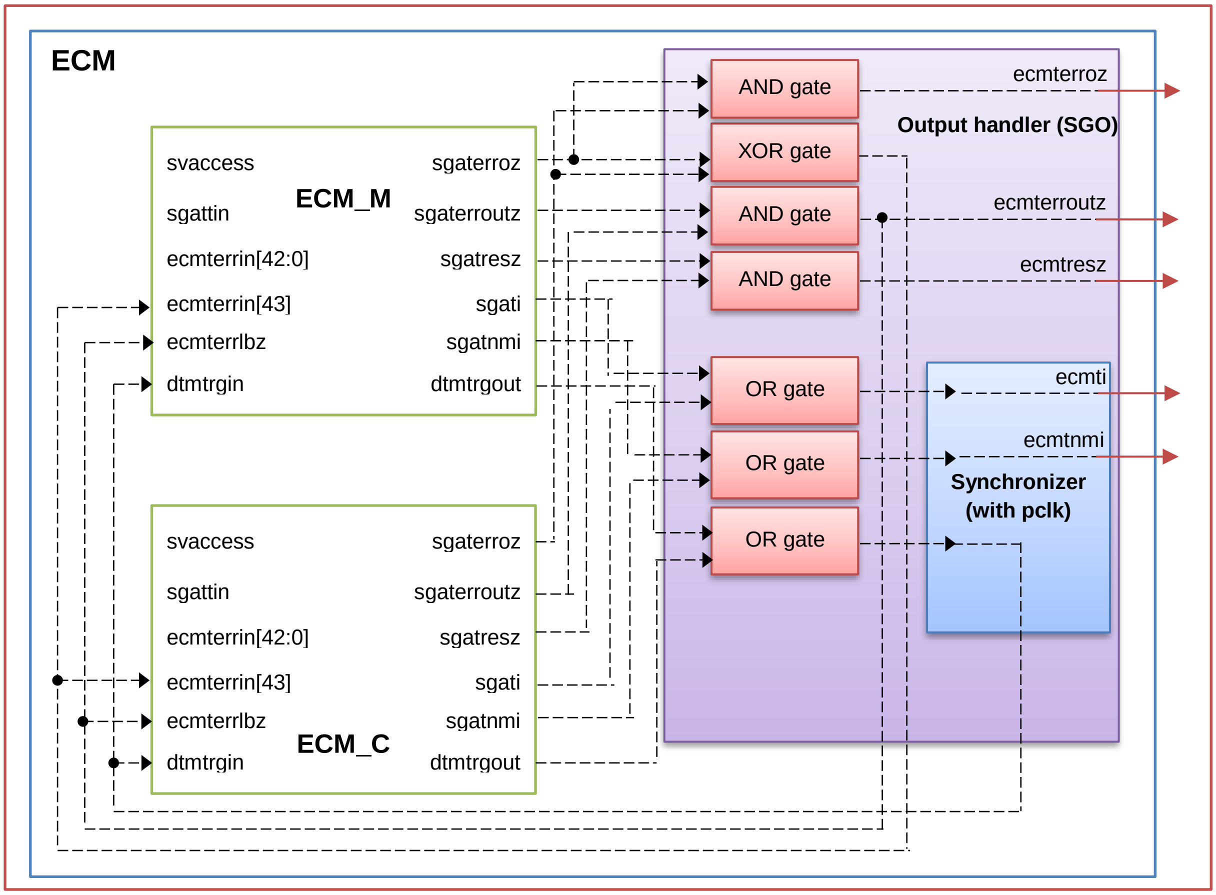


Figure 3.3: Disposing output signals

4. List of implemented registers

Table 4.1: List of implemented registers

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
ECMmESET Error set trigger (4a*)(4b*)	<ECMm_base> + 0x00 (2*)	0x0	1	8	8|16|32	R:0|W1	0	ECMmEST	**Error Set Trigger** + 0x0: Writing 0 is invalid + 0x1: Set the output level from the ERROROUT pin to active level Read value is always 0 The ERROROUT cannot be masked (not depend on setting of ECMEMK0/1 Register)
ECMmECLR Error clear trigger (4a*)(4b*)	<ECMm_base> + 0x04 (2*)	0x0	1	8	8|16|32	R:0|W1	0	ECMmECT	**Error Clear Trigger** + 0x0: Writing 0 is invalid + 0x1: Set the output level from the ERROROUT pin to inactive level Read value is always 0 Clearing of the ERROROUT output is only possible if all errors not masked by ECMEMK0/1, and the ECMmSSE130 bit in the ECMmESSTR1 register are cleared beforehand.
ECMmESSTR0 Error status 0	<ECMm_base> + 0x08 (2*)	0x0	4	-	8|16|32	R	31:26	ECMmSSE031 to ECMmSSE026	**Error Source Status** Corresponds to error sources 31 to 26. + 0x0: Error not occurred + 0x1: Error occurred
							24:14	ECMmSSE024 to ECMmSSE014	**Error Source Status** Corresponds to error sources 24 to 14. + 0x0: Error not occurred + 0x1: Error occurred
							12:4	ECMmSSE012 to ECMmSSE004	**Error Source Status** Corresponds to error sources 12 to 4. + 0x0: Error not occurred + 0x1: Error occurred
							1:0	ECMmSSE001, ECMmSSE000	**Error Source Status** Corresponds to error sources 1 and 0. + 0x0: Error not occurred + 0x1: Error occurred

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
ECMmESSTR1 Error status 1	<ECMm_base> + 0x0C(2*)	0x0	4	-	8|16|32	R	31	ECMmSSE131	**The status of the ERROROUT pin** + 0x0: ERROROUT is low level + 0x1: ERROROUT is high level
							30	ECMmSSE130	**Indicates the ECMmESET write status** + 0x0: No error + 0x1: Error is set by the ECMmESET.ECMmEST bit
							29	ECMmSSE129	**Delay timer overflow status** + 0x0: Delay timer overflow not occurred + 0x1: Delay timer overflow occurred
							10:4	ECMmSSE110 to ECMmSSE104	**Error Source Status** Corresponds to error sources 42 to 36. + 0x0: Error not occurred + 0x1: Error occurred
							2:0	ECMmSSE102 to ECMmSSE100	**Error Source Status** Corresponds to error sources 34 to 32. + 0x0: Error not occurred + 0x1: Error occurred
ECMmPCMD0 Write-Protection Command	<ECMm_base> + 0x10 (2*)	0x0 (5*)	4	32	-	W	7:0	ECMmREG0	**Write protection command** Protection command that enables writing to write-protection target registers
ECMEPCFG Error pulse configuration (4b*)	<ECM_base>	0x0	1	8	8|16|32	R/W	0	ECMSL0	**ERROROUT Pin Output Operation Configuration** Operation setting for the ERROROUT pin + 0x0: Non-dynamic mode + 0x1: Dynamic mode

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
ECMMICFG0 EI level interrupt configuration 0 (4b*)	<ECM_base> + 0x04	0x0	4	32	8|16|32	R/W	31:26	ECMMIE031 to ECMMIE026	**EI Level Interrupt Generation Control** Corresponds to error sources 31 to 26. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
							24:14	ECMMIE024 to ECMMIE014	**EI Level Interrupt Generation Control** Corresponds to error sources 24 to 14. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
							12:4	ECMMIE012 to ECMMIE004	**EI Level Interrupt Generation Control** Corresponds to error sources 12 to 4. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
							1:0	ECMMIE001, ECMMIE000	**EI Level Interrupt Generation Control** Corresponds to error sources 1 and 0. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
ECMMICFG1 EI level interrupt configuration1(4b*)	<ECM_base> + 0x08	0x0	4	32	8|16|32	R/W	10:4	ECMMIE110 to ECMMIE104	**EI Level Interrupt Generation Control** Corresponds to error sources 42 to 36. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
							2:0	ECMMIE102 to ECMMIE100	**EI Level Interrupt Generation Control** Corresponds to error sources 34 to 32. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
ECMNMICFG0 FE level interrupt configuration 0 (4b*)	<ECM_base> + 0x0C	0x0	4	32	8|16|32	R/W	31:26	ECMNMIE031 to ECMNMIE026	**FE Level Interrupt Generation Control** Corresponds to error sources 31 to 26. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
							24:14	ECMNMIE024 to ECMNMIE014	**FE Level Interrupt Generation Control** Corresponds to error sources 24 to 14. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
							12:4	ECMNMIE012 to ECMNMIE004	**FE Level Interrupt Generation Control** Corresponds to error sources 12 to 4. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
							1:0	ECMNMIE001, ECMNMIE000	**FE Level Interrupt Generation Control** Corresponds to error sources 1 and 0. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
ECMNMICFG1 FE level interrupt configuration 1 (4b*)	<ECM_base> + 0x10	0x0	4	32	8|16|32	R/W	10:4	ECMNMIE110 to ECMNMIE104	**FE Level Interrupt Generation Control** Corresponds to error sources 42 to 36. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
							2:0	ECMNMIE102 to ECMNMIE100	**FE Level Interrupt Generation Control** Corresponds to error sources 34 to 32. + 0x0: Interrupt generation disabled + 0x1: Interrupt generation enabled
ECMIRCFG0 Internal reset configuration 0 (4b*)	<ECM_base> + 0x14	0xF	4	32	8|16|32	R/W	31:26	ECMIRE031 to ECMIRE026	**Internal Reset (ECMRES) Generation Control** Corresponds to error sources 31 to 26. + 0x0: internal reset generation disabled + 0x1: internal reset generation enabled
							24:14	ECMIRE024 to ECMIRE014	**Internal Reset (ECMRES) Generation Control** Corresponds to error sources 24 to 14. + 0x0: Internal reset generation disabled + 0x1: Internal reset generation enabled
							12:4	ECMIRE012 to ECMIRE004	**Internal Reset (ECMRES) Generation Control** Corresponds to error sources 12 to 4. + 0x0: Internal reset generation disabled + 0x1: Internal reset generation enabled
							1:0	ECMIRE001, ECMIRE000	**Internal Reset (ECMRES) Generation Control** Corresponds to error sources 1 and 0. + 0x0: Internal reset generation disabled + 0x1: Internal reset generation enabled
ECMIRCFG1 Internal reset configuration 1 (4b*)	<ECM_base> + 0x18	0x0	4	32	8|16|32	R/W	29	ECMIRE129	**Internal Reset (ECMRES) Generation Control** Corresponds to delay timer overflow. + 0x0: internal reset generation disabled + 0x1: Internal reset generation enabled
							10:4	ECMIRE110 to ECMIRE104	**Internal Reset (ECMRES) Generation Control** Corresponds to error sources 42 to 36. + 0x0: Internal reset generation disabled + 0x1: Internal reset generation enabled
							2:0	ECMIRE102 to ECMIRE100	**Internal Reset (ECMRES) Generation Control** Corresponds to error sources 34 to 32. + 0x0: Internal reset generation disabled + 0x1: Internal reset generation enabled

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
ECMEMK0 Error mask 0 (4b*)	<ECM_base> + 0x1C	0x0	4	32	8|16|32	R/W	31:26	ECMEMK031 to ECMEMK026	**ERROROUT Output Mask Control** Corresponds to error sources 31 to 26. + 0x0: ERROROUT output not masked + 0x1: ERROROUT output masked
							24:14	ECMEMK024 to ECMEMK014	**ERROROUT Output Mask Control** Corresponds to error sources 24 to 14. + 0x0: ERROROUT output not masked + 0x1: ERROROUT output masked
							12:4	ECMEMK012 to ECMEMK004	**ERROROUT Output Mask Control** Corresponds to error sources 12 to 4. + 0x0: ERROROUT output not masked + 0x1: ERROROUT output masked
							1:0	ECMEMK001, ECMEMK000	**ERROROUT Output Signal Mask Control** Corresponds to error sources 1 and 0. + 0x0: Error signal output not masked + 0x1: Error signal output masked
ECMEMK1 Error mask 1 (4b*)	<ECM_base> + 0x20	0x0	4	32	8|16|32	R/W	29	ECMEMK129	**ERROROUT Output Mask Control** Corresponds to delay timer overflow. + 0x0: ERROROUT output not masked + 0x1: ERROROUT output masked
							10:4	ECMEMK110 to ECMEMK104	**ERROROUT Output Mask Control** Corresponds to error sources 42 to 36. + 0x0: ERROROUT output not masked + 0x1: ERROROUT output masked
							2:0	ECMEMK102 to ECMEMK100	**ERROROUT Output Mask Control** Corresponds to error sources 34 to 32. + 0x0: ERROROUT output not masked + 0x1: ERROROUT output masked
ECMESSTC0 Error status clear trigger 0 (4b*)	<ECM_base> + 0x24	0x0	4	32	-	W	31:26	ECMCLSSE031 to ECMCLSSE026	**Error Status Clear** Corresponds to error sources 31 to 26. + 0x0: Corresponding error status unchanged + 0x1: Corresponding error status cleared
							24:14	ECMCLSSE024 to ECMCLSSE014	**Error Status Clear** Corresponds to error sources 24 to 14. + 0x0: Corresponding error status unchanged + 0x1: Corresponding error status cleared
							12:4	ECMCLSSE012 to ECMCLSSE004	**Error Status Clear** Corresponds to error sources 12 to 4. + 0x0: Corresponding error status unchanged + 0x1: Corresponding error status cleared
							1:0	ECMCLSSE001 , ECMCLSSE000	**Error Status Clear** Corresponds to error sources 1 and 0. + 0x0: Corresponding error status unchanged + 0x1: Corresponding error status cleared

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
ECMESSTC1 Error status clear trigger 1 (4b*)	<ECM_base> + 0x28	0x0	4	32	-	W	30:29	ECMCLSSE130 , ECMCLSSE129	**Error Status Clear** Corresponds to the write status and delay timer overflow of the ECMmESET register. + 0x0: Corresponding error status unchanged + 0x1: Corresponding error status cleared
							10:4	ECMCLSSE110 to ECMCLSSE104	**Error Status Clear** Corresponds to error sources 42 to 36. + 0x0: Corresponding error status unchanged + 0x1: Corresponding error status cleared
							2:0	ECMCLSSE102 to ECMCLSSE100	**Error Status Clear** Corresponds to error sources 34 to 32. + 0x0: Corresponding error status unchanged + 0x1: Corresponding error status cleared
ECMPCMD1 Write-Protection Command	<ECM_base> + 0x2C	0x0 (5*)	4	32	-	W	7:0	ECMREG1	**Write protection command** Protection command that enables writing to write-protection target registers
ECMPS Write Sequence Status	<ECM_base> + 0x30	0x0	1	-	8|16|32	R	0	ECMPRERR	**Write Sequence Error Monitor** Indicate the status of write sequence of the write-protection target registers + 0x0: A protection error does not occur + 0x1: A protection error does occur
ECMPE0 Pseudo error trigger 0 (4b*)	<ECM_base> + 0x34	0x0	4	32	-	W	31:26	ECMPE031 to ECMPE026	**Pseudo error trigger** Corresponds to error sources 31 to 26. + 0x0: Pseudo error not generated + 0x1: Corresponding pseudo error generated
							24:14	ECMPE024 to ECMPE014	**Pseudo error trigger** Corresponds to error sources 24 to 14. + 0x0: Pseudo error not generated + 0x1: Corresponding pseudo error generated
							12:4	ECMPE012 to ECMPE004	**Pseudo error trigger** Corresponds to error sources 12 to 4. + 0x0: Pseudo error not generated + 0x1: Corresponding pseudo error generated

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
							1:0	ECMPE001, ECMPE000	**Pseudo Error Trigger Clear** Corresponds to error sources 1 and 0. + 0x0: Pseudo error not generated + 0x1: Corresponding pseudo error generated
ECMPE1 Pseudo error trigger 1 (4b*)	<ECM_base> + 0x38	0x0	4	32	-	W	29	ECMPE129	**Pseudo Error Trigger** Corresponds to delay timer overflow. + 0x0: Pseudo error not generated + 0x1: Corresponding pseudo error generated
							10:4	ECMPE110 to ECMPE104	**Pseudo Error Trigger** Corresponds to error sources 42 to 36. + 0x0: Pseudo error not generated + 0x1: Corresponding pseudo error generated
							2:0	ECMPE102 to ECMPE100	**Pseudo Error Trigger** Corresponds to error sources 34 to 32. + 0x0: Pseudo error not generated + 0x1: Corresponding pseudo error generated
ECMDTMCTL Delay timer control (4b*)	<ECM_base> + 0x3C	0x0	1	8	-	W	1	ECMSTP	**Delay Timer Stop** + 0x0: Delay timer is completed or not executed + 0x1: Stop request for delay timer is on execution Writing 1 to this bit initializes the delay timer register, causing the delay timer to stop. Simultaneously, the ECMSTA bit is set to 0. Writing 0 is ignored.
							0	ECMSTA	**Delay Timer Start** Writing 1 to this bit starts delay timer counting upon occurrence of an interrupt. The counting always starts from 0. (Writing 0 to this bit stops the delay timer).
ECMDTMR Delay timer	<ECM_base> + 0x40	0x0	2	-	8|16|32	R	15:0	ECMDTMR	**Delay timer counter value** Delay timer counter value is initialized by setting ECMDTMCTL.ECMSTA from 1 to 0.

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
ECMDTMCMP Delay timer compare (4b*)	<ECM_base> + 0x44	0x0	2	16	8|16|32	R/W	15:0	ECMDTMCMP	**Delay timer compare match value** The writing value is used to compare with delay timer counter value When this value matches with the value of ECMDTMR, the ECMmSSE129 bit is set. Writing data to this register has to be conducted while the delay timer is stopped.
ECMDTMCFG0 Delay timer configuration 0 (4b*)	<ECM_base> + 0x48	0x0	4	32	8|16|32	R/W	31:26	ECMTE031 to ECMTE026	**Delay Timer Start Control** Corresponds to EI level interrupts generated by error sources 31 to 26. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
							24:14	ECMTE024 to ECMTE014	**Delay Timer Start Control** Corresponds to EI level interrupts generated by error sources 24 to 14. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
							12:4	ECMTE012 to ECMTE004	**Delay Timer Start Control** Corresponds to EI level interrupts generated by error sources 24 to 4. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
							1,0	ECMTE001, ECMTE000	**Delay Timer Start Control** Corresponds to EI level interrupts generated by error sources 1 and 0. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
ECMDTMCFG1 Delay timer configuration 1 (4b*)	<ECM_base> + 0x4C	0x0	4	32	8|16|32	R/W	10:4	ECMTE110 to ECMTE104	**Delay Timer Start Control** Corresponds to EI level interrupts generated by error sources 42 to 36. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
							2:0	ECMTE102 to ECMTE100	**Delay Timer Start Control** Corresponds to EI level interrupts generated by error sources 34 to 32. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled

Register name (1*)	Address offset (3*)	Initial value	Size (byte)	Write Access size (bit)	Read Access size (bit)	R/W	Bit position	Bit name (1*)	Explanation
ECMDTMCFG2 (4b*) Delay timer configuration 2	<ECM_base> + 0x50	0x0	4	32	8|16|32	R/W	31:26	ECMTE231 to ECMTE226	**Delay Timer Start Control** Corresponds to FE level interrupts generated by error sources 31 to 26. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
							24:14	ECMTE224 to ECMTE214	**Delay Timer Start Control** Corresponds to FE level interrupts generated by error sources 24 to 14. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
							12:4	ECMTE212 to ECMTE204	**Delay Timer Start Control** Corresponds to FE level interrupts generated by error sources 12 to 4. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
							1, 0	ECMTE201, ECMTE200	**Delay Timer Start Control** Corresponds to FE level interrupts generated by error sources 1 and 0. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
ECMDTMCFG3 (4b*) Delay timer configuration 3	<ECM_base> + 0x54	0x0	4	32	8|16|32	R/W	10:4	ECMTE310 to ECMTE304	**Delay Timer Start Control** Corresponds to FE level interrupts generated by error sources 42 to 36. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled
							2:0	ECMTE302 to ECMTE300	**Delay Timer Start Control** Corresponds to FE level interrupts generated by error sources 34 to 32. + 0x0: Delay timer start disabled + 0x1: Delay timer start enabled

Notes: *- All registers described in HWM (ref[5]/Chapter 32.3 and Chapter 4.3.2.6) are supported in model (except ECMEPCTL register is supported in other model).*

- (1) Register name/Bit name is name used in model.*

- (2) m = M/C (Master/Checker).*

- (3) <ECM_base>, <ECMM_base> and <ECMC_base> are base address of register areas which be accessed by target sockets m_tgt_sockets[0] -> m_tgt_sockets[2] (refer to Chapter 3 and Table 5.1 for more detail about these target sockets). The address area for common registers is used for both master and checker registers. Writing to this common area is performed by master and checker registers simultaneously. Reading from the common area reads the master register's value.*

- (4) Protected register: Writing to this register is protected by a special sequence of instructions by using the protection command register ((a): use ECMmPCMD0 register; (b): use ECMPCMD1 register)*

- (5) The initial value of this register in HWM (ref[5]) is "undefined". In modeling, its value is "0x0".*

5. Port behavior

5.1. List of implemented ports

Table 5.1: List of implemented ports

Port name	I/O	Type	Initial	Active	Synchronous clock	Description
Clock and Reset						
pclk	In	sc_in<sc_dt::uint64>	-	-	-	Delay timer clock/ APB bus clock (unit: Hz)
preset_n	In	sc_in<bool>	-	(1*)	pclk	Delay timer reset/ APB Bus reset
extresetz	In	sc_in<bool>	-	(1*)	pclk	External reset
APB I/F						
m_tgt_sockets[0]	In/ Out	tlm::tlm_target_socket	-	-	pclk	TLM target socket controls accessing to ECM Common area. The based address is indicated by <ECM_base>
m_tgt_sockets[1]	In/ Out	tlm::tlm_target_socket	-	-	pclk	TLM target socket controls accessing to ECM Master area. The based address is indicated by <ECMM_base>
m_tgt_sockets[2]	In/ Out	tlm::tlm_target_socket	-	-	pclk	TLM target socket controls accessing to ECM Checker area. The based address is indicated by <ECMC_base>
Input and Output ports						
ecmterrin [emErrSrcP1M] (2*)	In	sc_in<bool> *	-	High	-	Error source input (3*)
ecmttin	In	sc_in<bool>	-	-	-	Timer input for dynamic mode
svaccess	In	sc_in<bool>	-	High	-	Break
ecmterroz	Out	sc_out<bool>	Low (4*)	Low	Async	Error output compare signal
ecmterroutz	Out	sc_out<bool>	Low (4*)	Low	Async	ERROROUT output
ecmti	Out	sc_out<bool>	Low	High	pclk	Mask-able interrupt output
ecmtnmi	Out	sc_out<bool>	Low	High	pclk	Non-mask-able interrupt output
ecmtresz	Out	sc_out<bool>	High	Low	Async	ECM internal reset request

Notes: - *(1*) Active level of reset signals depend on defining macro IS_RESET_ACTIVE_LOW (refer to Chapter 6.6).*

- *(2*) emErrSrc_P1M = 43*
- *(3*) The error source are listed in ref[6]/"Error Sources" sheet/"P1M" column*

- The port list is implemented according signal description in HWM (ref[7]/ Figure 21-1 Connection image of ECM). In this list, below ports are not supported: "testmode", "scan_enable" and "scanmode". Besides, APB I/F ports used to access ECM's registers are implemented by TLM target sockets.

- (4) The reset level is changed from High to Low at ref[11] v1.7 according to "uhiapecm0020_target_speicfication_ver2.pdf".*

5.2. Clock

- ECM model has 1 clock port ("pclk") used to receive clock frequency signal (Hz) which is used to calculate the time required for transaction (register accessing) and internal behavior processing (e.g: count clock for delay timer, synchronization clock for interrupt and reset signals).
- When clock frequency is zero:
  - Model does not start operation
  - Model stops operation (e.g: delay timer stops counting)
  - No error message dumped. If any input ports which notify the internal process are active, a warning message is dumped.
  - Reset operation is executed immediately when reset port is activated.

***Note**: - A core dump error may be occurred when access to register (AT mode) due to "pclk" is register access clock.*

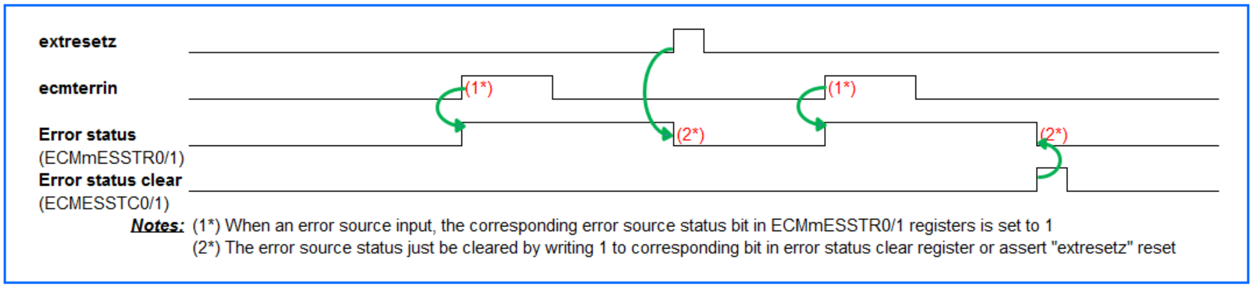
5.3. Reset

- This model has 2 reset ports ("preset_n" and "extresetz") used to reset registers and operation of model. Parameters in Table 6.2 are not effected by reset operation (refer to Chapter 7.3 for detail of reset operation).
  - When "preset_n" port is activated, all registers (except ECMmESSTR0/1) are reset.
  - When "extresetz" port is activated, ECMmESSTR0/1 registers are reset.
  - When reset port is activated ("preset_n" or "extresetz"), all output ports are initialized.
- Both "preset_n" and "extresetz" ports are synchronized with "pclk" clock when "pclk" frequency is different from zero value.
- When "pclk" frequency is zero value, reset operation is executed immediately if reset port is activated.
- Active level of reset port depend on defining the macro IS_RESET_ACTIVE_LOW (refer to Chapter 6.6).

***Note**: - During reset period, accessing to registers may not be allowed.*

5.4. “ecmterrin” input port - Error source input

- ECM model collects error signals coming from different error sources via “ecmterrin” input ports.
- When an error source input is active, the corresponding error source status bit in ECMmESSTR0/1 registers is set. (refer to Chapter 7.5)



Notes: (1*) When an error source input, the corresponding error source status bit in ECMmESSTR0/1 registers is set to 1
(2*) The error source status just be cleared by writing 1 to corresponding bit in error status clear register or assert "extresetz" reset

Figure 5.1: Error status updating

5.5. “ecmttin” input port - Timer input for dynamic mode

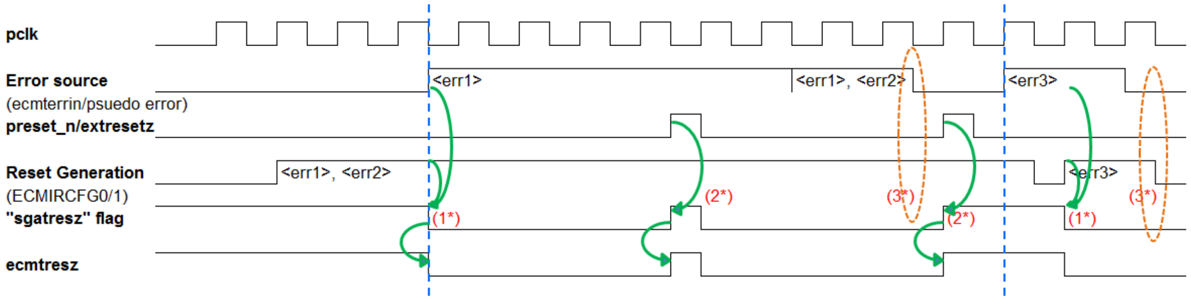
- The “ecmttin” input port receives timer input signal used for dynamic mode of operation.
- When the “ecmterroutz” output port is configured for dynamic mode (by setting ECMEPCFG.ECMSL0 = 1) and error status is no error, the “ecmterroutz” output signal is issued depend on value of “ecmttin” input port. (refer to Chapters 5.10, 7.8)

5.6. “svaccess” input port - Emulation break

- The “svaccess” input port receives break signal used for emulation break during “Write sequence”. (refer to Chapter 7.9)
- If an emulation break occurs during write procedure to write-protected registers, the register protection is suspended until normal operation is resumed. Even if any register of ECM module is accessed during the break, the write sequence is not suspended and the ECMPS.ECMPRERR bit is not set to 1. (ref[5]/Chapter 4.3.1.4)

5.7. “ecmtresz” output port - Internal reset request

- The “ecmtresz” output port is used to issue ECM internal reset request signal (ECMRES reset source).
- The internal reset request output signal (“ecmtresz”) is issued accordingly with active level is result of **AND** operator between internal reset flags (“sgatresz”) from Master and Checker sides: “ecmtresz” active level = “sgatresz (M)” & “sgatresz (C)”
- The “sgatresz” flag from each side (Master and Checker) is set to 0 in case an error source available (Error input source actives or Pseudo error actives or delay timer overflow occurs) and reset generation is enabled by set 1 to corresponding bit in ECMIRCFG0/1. (refer to Chapter 7.7)
- Once this reset request is asserted, this value is kept until reset port (“extresetz” or “preset_n”) is asserted.



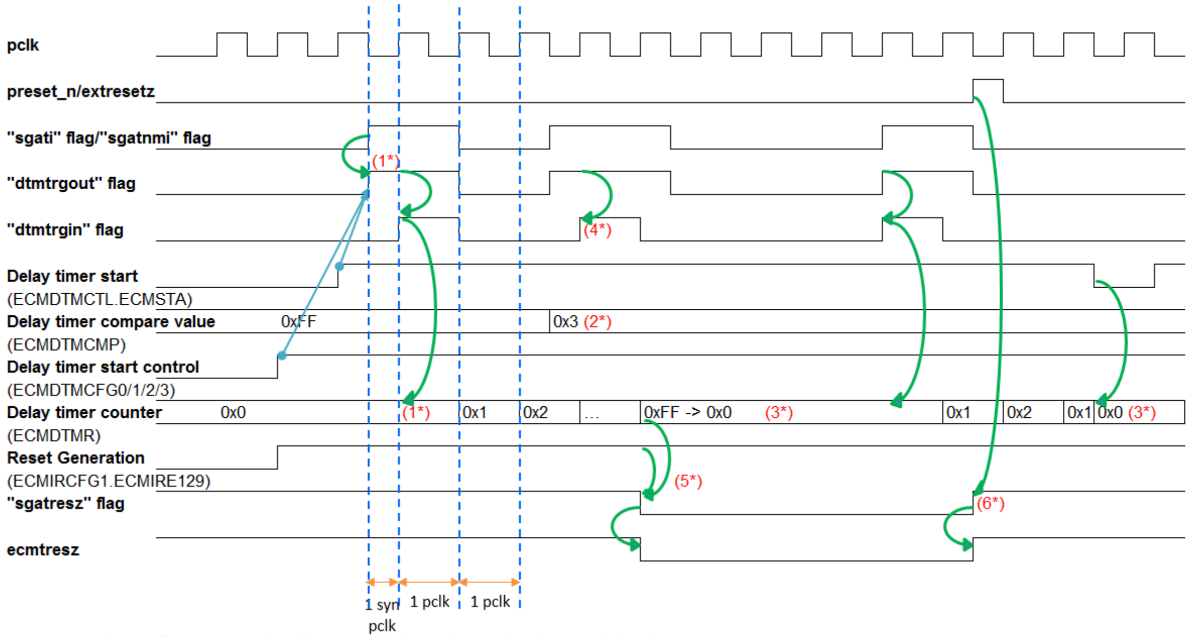
Notes: Reset output signal "ecmtresz" is issued according "sgatresz" flag from each side

(1*) "sgatresz" flag is set to 0 (active) in case at least 1 reset generation condition is reached
(error source available and corresponding reset generation is enabled)

(2*) Once this reset request "ecmtresz" is asserted, this value is kept until reset port is asserted
After reset port is negated, if reset generation condition still be reached, this reset request "ecmtresz" is asserted again

(3*) "sgatresz" flag is NOT set to 1 in case that all reset generation condition is NOT reached

Figure 5.2: Internal reset issuing (1/2)



Notes: Reset output signal "ecmtresz" is issued according "sgatresz" flag from each side

(1*) The delay timer incorporated in the ECM can be started simultaneously with the occurrence of an interrupt request
("dtmtrgout" flag from each side is sent to Synchronizer, after that, "dtmtrgin" flag is set (with 1 pulse of pclk) to start delay timer)
The counting of the delay timer always starts from 0

(2*) Writing data to ECMDTMCMP register has to be conducted while the delay timer is stopped

(3*) Delay timer counter stop counting after matches value with ECMDTMCMP and reset to 0
When delay timer stop (by writing 0 to ECMDTMCTL.ECMSTP bit),delay timer stops counting and initialize counter value to 0x0

(4*) During delay timer is running, if another error occurs, the counter value is not reset but the timer continues to run

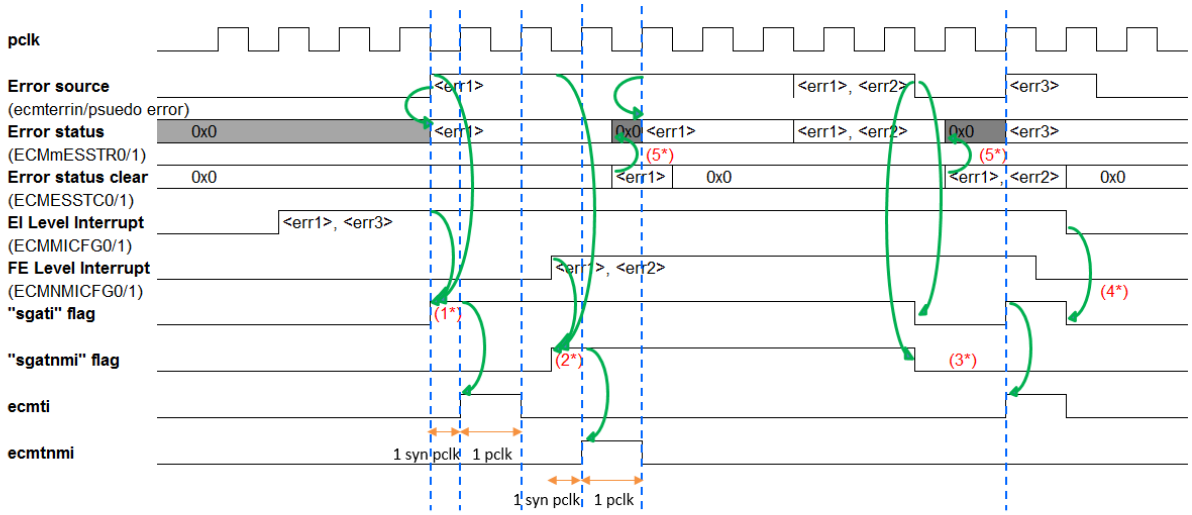
(5*) "sgatresz" flag is set to 0 (active) in case at least 1 reset generation condition is reached
(error source available and corresponding reset generation is enabled)

(6*) Once this reset request "ecmtresz" is asserted, this value is kept until reset port is asserted

Figure 5.3: Internal reset issuing (2/2)

5.8. “ecmti” and “ecmtnmi” output ports - Interrupt output

- ➢ The “ecmti” and “ecmtnmi” output ports are used to issue ECM interrupt output signals.



Notes: Interrupt output signal "ecmti"/"ecmtnmi" is issued with 1 pulse of "pclk" according "sgati"/"sgatnmi" flag from each side
(1*), (2*) "sgati"/"sgatnmi" flag just be set to 1 in case there is at least 1 of interrupt generation condition is reached
(error source available and corresponding interrupt generation is enabled)
(3*), (4*) If there is no interrupt generation condition is reached, "sgati"/"sgatnmi" flag is set to 0
(5*) When users write value to error status clear register (ECMESSTC0/1), the corresponding error status bit is cleared
+ If error source is still active, the error status is set at next rising edge of pclk clock

Figure 5.4: Interrupt issuing

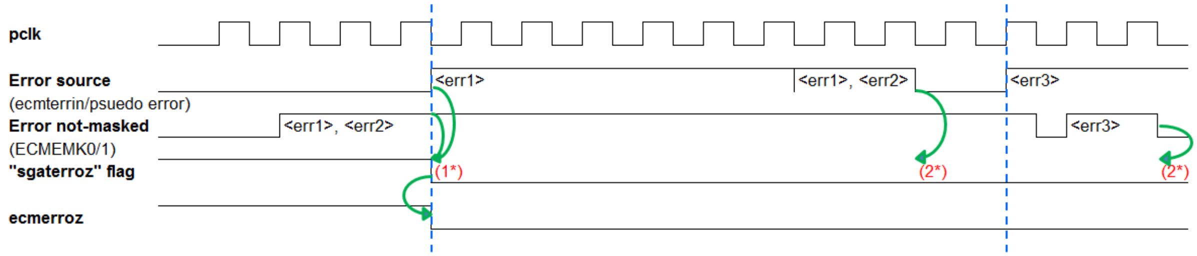
- ➢ These ports are synchronized with “pclk” clock. The output signals issued from these ports are signals with 1 pulse of “pclk” clock.
- ➢ The interrupt output signals (“ecmti” and “ecmtnmi”) are issued accordingly with active level is result of **OR** operator between interrupt flags (“sgati”/“sgatnmi”) from Master and Checker sides.
  - ○ “ecmti” active level = “sgati (M)” | “sgati (C)”
  - ○ “ecmtnmi” active level = “sgatnmi (M)” | “sgatnmi (C)”
- ➢ The “sgati”/“sgatnmi” flag from each side (Master and Checker) is set to 1 in case there is at least 1 of interrupt generation condition is reached (*) (refer to Chapter 7.6)
- ➢ Besides, if there is no interrupt generation condition is reached, “sgati”/“sgatnmi” flag is set to 0.

Note*: - (*) “The interrupt generation condition” = “error source available (Error input source actives or Pseudo error actives)” and “interrupt generation is enabled by set 1 to corresponding bit in ECMMICFG0/1 and/or ECMNMICFG0/1 register(s)”.*

5.9. “ecmterroz” output port - Error output compare

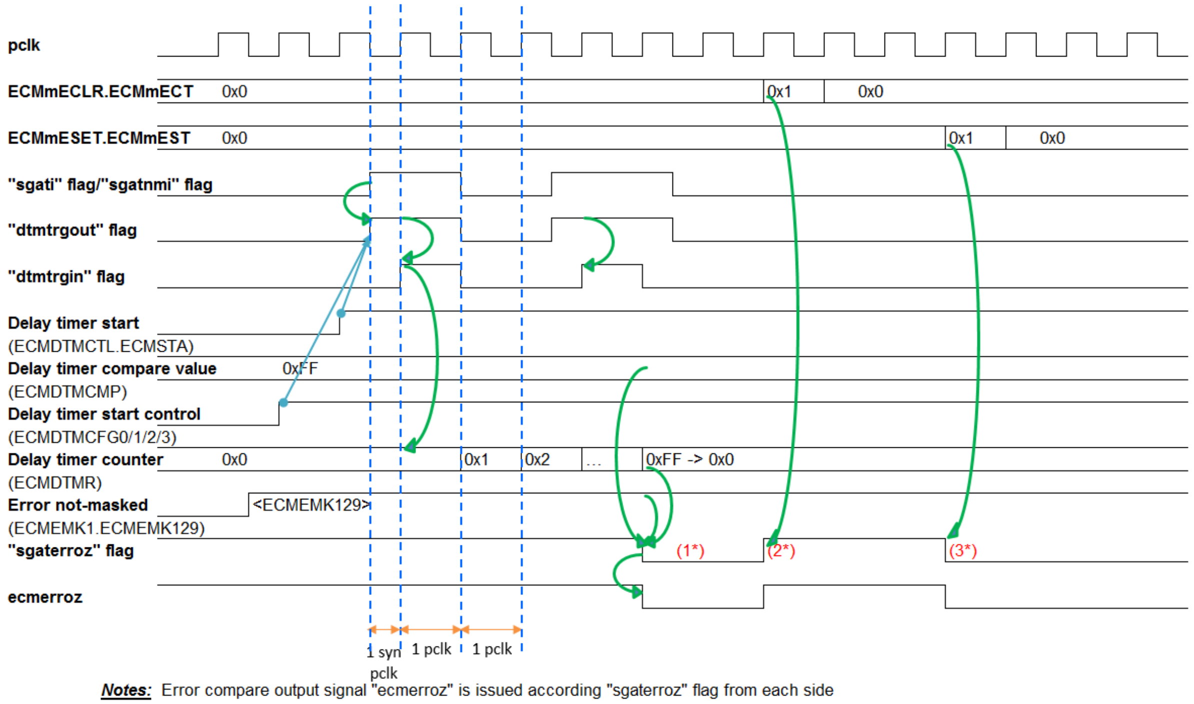
- ➢ The “ecmterroz” output port is used to issue Error output compare signal.
- ➢ The error output compare signal (“ecmterroz”) is issued accordingly with active level is result of **AND** operator between error output flags (“sgaterroz”) from Master and Checker sides: “ecmterroz” active level = “sgaterroz (M)” & “sgaterroz (C)”.

- The “sgaterroz” flag from each side (Master and Checker) is set to 0 (active level) in case an error source available (Error input source actives or Pseudo error actives or delay timer overflow occurs) and corresponding bit of error source in ECMEMK0/1 is 0.
- The “sgaterroz” flag can be set/clear directly by using “port” command (refer to Chapter 6.4) or write 1 into ECMmESET/ECMmECLR reigister.
- The timer input signal (“ecmttin”) does not affect even if the ECM is set for dynamic mode. (refer to Chapter 7.8)



Notes: Error compare output signal "ecmerroz" is issued according "sgaterroz" flag from each side
(1*) The error output signal indicates no-masked errors (setting by ECMEMK0/1) in synchronization with errors as they occur
"sgaterroz" flag is set to 0 (active) when error source available and corresponding bit in ECMEMK0/1 = 0
(2*) "sgaterroz" flag is NOT set to 1 (inactive) when error source is not available or corresponding bit in ECMEMK0/1 = 1

Figure 5.5: Error compare output issuing (1/2)

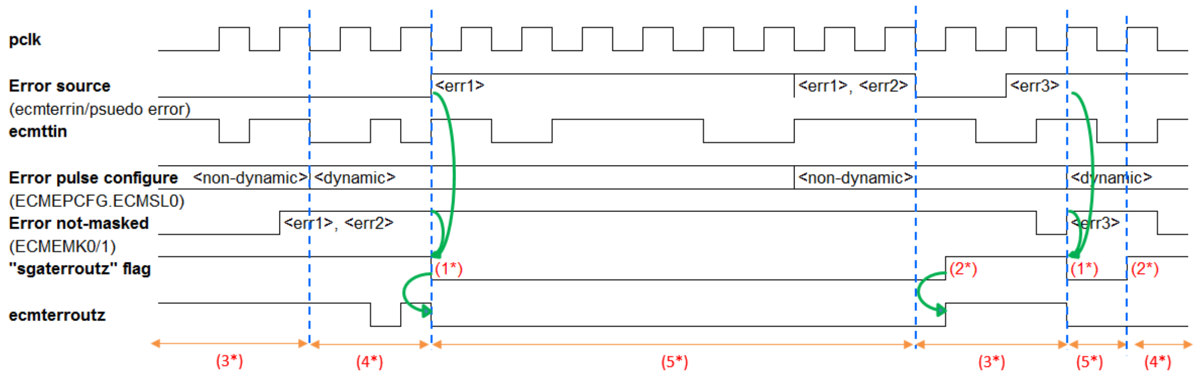


Notes: Error compare output signal "ecmerroz" is issued according "sgaterroz" flag from each side
(1*) The error output signal indicates no-masked errors (setting by ECMEMK0/1) in synchronization with errors as they occur
"sgaterroz" flag is set to 0 (active) when timer overflow and ECMEMK1.ECMEMK129 = 0
(2*) "sgaterroz" flag is set to 1 (inactive) when clear error trigger is progress by set 1 to ECMmECLR register
(3*) "sgaterroz" flag is set to 0 when clear error trigger is progress by set 1 to ECMmESET register

Figure 5.6: Error compare output issuing (2/2)

5.10. “ecmterroutz” output port - ERROROUT output

- The “ecmterroutz” output port is used to issue ERROROUT output signal.
- When error occurs or in non-dynamic mode, the ERROROUT output signal (“ecmterroutz”) is issued accordingly with active level is result of **AND** operator between error output flags (“sgaterroutz”) from Master and Checker sides: “ecmterroutz” active level = “sgaterroutz (M)” & “sgaterroutz (C)”.
- When there is no error occurs in dynamic mode, the “ecmterroutz” is affected by timer input port (“ecmttin”): “ecmterroutz” = ~(“ecmttin”) (refer to Chapter 7.8)
- Same as “sgaterroz”, the “sgaterroutz” flag from each side (Master and Checker) is set to 0 (active level) in case an error source available (Error input source actives or Pseudo error actives or delay timer overflow occurs) and corresponding bit of error source in ECMEMK0/1 is 0. The “sgaterroutz” flag is not changed from 0 to 1 if error source released or all bits in ECMEMK0 and ECMEMK1 registers are changed to 1.
- Besides, “sgaterroutz” flag can be set/clear directly by using “port” command (refer to Chapter 6.4) or write 1 into ECMmESET/ECMmECLR reigister.



Notes: Error compare output signal "ecmerroz" is issued according "sgaterroz" flag from each side

(1*) The error output signal indicates no-masked errors (setting by ECMEMK0/1) in synchronization with errors as they occur "sgaterroutz" flag is set to 0 (active) when error source available and corresponding bit in ECMEMK0/1 = 0

(2*) "sgaterroutz" flag is set to 1 (inactive) when clear error trigger is progress by set 1 to ECMmECLR register

(3*) In non-dynamic mode, ERROROUT output signal "ecmterroutz" is issued according "sgaterroutz" flag from each side

(4*) When the error flag (“sgaterroutz”) = 1, "ecmterroutz" will be issued depend on “ecmttin” input if this signal is in dynamic mode

(5*) When error flag = 0, "ecmterroutz" is issued according "sgaterroutz" flag from each side

Figure 5.7: ERROROUT output issuing

6. Direction for users

6.1. File structures

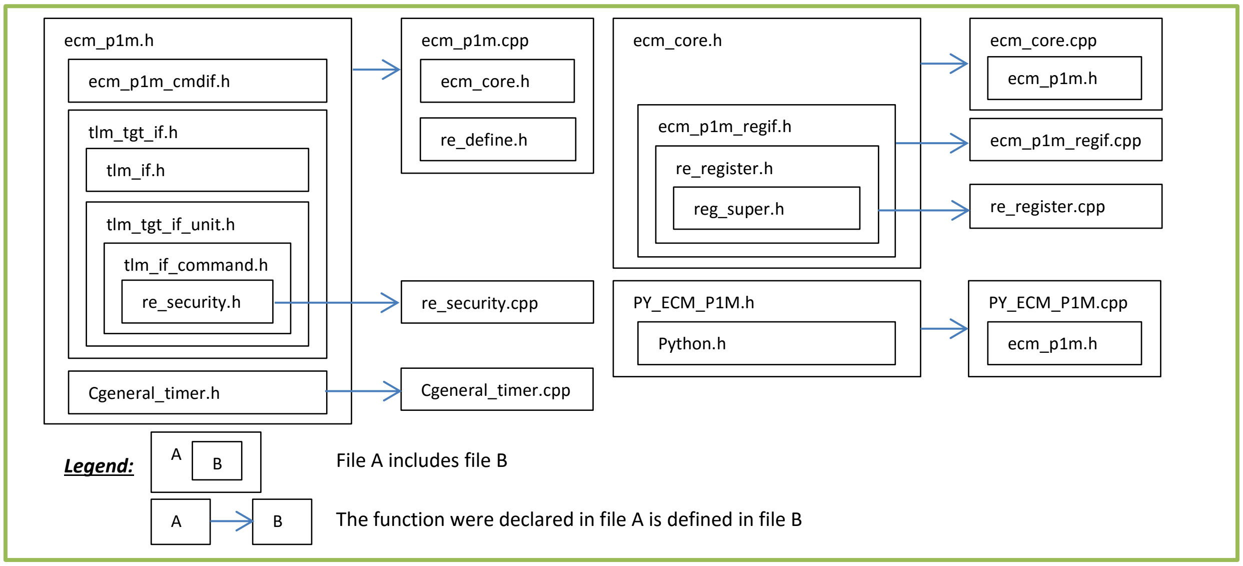


Figure 6.1: File structure

Table 6.1: File description

File name	CVS tag	Implementation	Description
ecm_p1m.h	-	Developed	Header file of ECM wrapper
ecm_p1m.cpp	-	Developed	Implementation file of ECM wrapper
ecm_core.h	-	Developed	Header file of ECM function
ecm_core.cpp	-	Developed	Implementation file of ECM function
ecm_p1m_regif.h	-	Generated (1*)	Header file of ECM register interface
ecm_p1m_regif.cpp	-	Generated (1*)	Implementation file of ECM register interface
ecm_p1m_cmdif.h	-	Generated (1*)	Implementation file of command interface and re_printf function
PY_ECM_P1M.h	-	Generated (1*)	Header file of ECM Python interface
PY_ECM_P1M.cpp	-	Generated (1*)	Implementation file of ECM Python interface
Python.h	-	Reused	Header file of python library
re_register.h	v2016_09_21	Reused	Header file of the re_register class
re_register.cpp		Reused	Implement the attributes and the operations of common register class
reg_super.h		Reused	General class for models to access to the memory array
tlm_tgt_if.h	v2016_08_11_b_frm_v2014_04_02	Reused	Header file of the tlm_tgt_if class
tlm_if.h		Reused	Header file of the tlm_if class
tlm_tgt_if_unit.h		Reused	Header file of the tlm_tgt_if_unit class
tlm_if_command.h		Reused	Header file of the tlm_if_command class
re_security.h	v100419	Reused	Additional file of tlm_ini_if class and tlm_tgt_if class
re_security.cpp		Reused	
re_define.h	v1.2 (2012/01/30)	Reused	Define common define macro, enum and so on
Cgeneral_timer.h	v2012_05_17	Reused	Header file of Cgeneral_timer
Cgeneral_timer.cpp		Reused	Implementation file of Cgeneral_timer

Note: *- (1*) Files ecm_core_regif.h/.cpp are generated from Register IF Generator (v2014_12_01). Files ecm_p1m_cmdif.h and PY_ECM_P1M.h/.cpp are generated from Command IF Generator (v2015_02_12). After that they are modified for suitableness of the model.*

6.2. Input/Output file

There is no input or output file.

6.3. How to connect Verification Environment

There are 3 basic steps to connect ECM model to a verification environment.

- Step 1: Declare an instance of ECM class “Cecm_p1m_wp”.
- Step 2: Bind the TLM target sockets m_tgt_sockets[0] -> m_tgt_sockets[2].
- Step 3: Bind the input/output ports (refer to Table 5.1 for list of implemented ports).

6.4. Commands and parameters

Table 6.2: List of parameters

Category	Parameter	Default	Description
command	MessageLevel <fatal|error|warning|info>	fatal|error	Select debug message level (“fatal”, “error”, “warning” and “info”) (1*) One or more than levels can be connected by vertical bar (Example “fatal|error”)
reg	MessageLevel <fatal|error|warning|info>		
reg	<register_name> MessageLevel <fatal|error|warning|info>		
reg	DumpRegisterRW <true/false>	false	Enable/disable dumping access register (2*) + false: Not dump register access information + true: Dump register access information
command	DumpInterrupt <enable>	false	Enable/disable interrupt information display when an interrupt is sent + false: Not dump interrupt information + true: Dump interrupt information
command	EnableTransInfo <enable>	false	Enable/disable error input information (error source name) display when an error input port is updated + false: Not dump error input information + true: Dump error input information

Table 6.3: List of commands

Category	Command	Description
command	AssertReset <rst_name> <start_time> <period>	Assert and negate reset signal + <rst_name>: name of reset signal + <start-time>: the time until asserting reset signal from current time. The unit is “ns” + <period>: the time from asserting reset signal to de-assert it. The unit is “ns”
command	SetCLKfreq <clk_name> <clk_freq> [<unit>]	Set clock frequency + <clk_name>: name of clock signal + < clk_freq>: clock frequency + <unit>: frequency unit (“Hz”, “KHz”, “MHz” or “GHz”). If this argument is not specified, frequency unit is “Hz” as default
command	GetCLKfreq <clk_name>	Get clock frequency + <clk_name>: name of clock signal
reg	[<master/checker>] <reg_name> force <value> (3*)	Force register with setting value + <master/checker>: Select Master or Checker side

		("master" or "checker"). If this argument is not specified, both Master and Checker sides are selected. + <reg_name>: name of register + <value>: value which is set to register
reg	[<master/checker>] <reg_name> release	Release register from force value + <master/checker>: Select Master or Checker side ("master" or "checker"). If this argument is not specified, both Master and Checker sides are selected. + <reg_name>: name of register
reg	[<master/checker>] <reg_name> <value> (3*)	Write a value to register + <master/checker>: Select Master or Checker side ("master" or "checker"). If this argument is not specified, both Master and Checker sides are selected. + <reg_name>: name of register + <value>: value which is set to register
reg	[<master/checker>] <reg_name>	Read value from register + <master/checker>: Select Master or Checker side ("master" or "checker"). If this argument is not specified, both Master and Checker sides are selected. + <reg_name>: name of register
reg	[<master/checker>]	Dump register names of model + <master/checker>: Select Master or Checker side ("master" or "checker"). If this argument is not specified, All name of registers from both Master and Checker sides are dumped.
tgt	set_param <term> <value>	Set simulation information about access to target + <term>: m_bus_clk | m_bus_gnt | m_bus_rgnt | m_buf_size| m_wr_latency | m_rd_latency | m_phase_mode | m_p_log_file | m_wr_log | m_rd_log | m_msg_out_lvl | m_wr_req_latency | m_rd_req_latency | m_fw_req_phase | m_info_displayed + <value>: Please see tlm_common_class spec sheet
tgt	get_param <term>	Get simulation information about access to target
tgt	init_param	Initialize simulation information
command	help	Dump the direction how to use parameters and commands + <master/checker>: Select Master or Checker side ("master" or "checker"). If this argument is not specified, Master side is selected.
reg	[<master/checker>] help	
tgt	help	
command	port [<master/checker>] <port_name> [<value>] (4*)	Set/Get value to a specified internal port. If "value" is not specified, return current port value + <master/checker>: Select Master or Checker side ("master" or "checker"). If this argument is not specified, Master side is selected. + <port_name>: • "sgatresz", "sgati", "sgatnmi", "sgaterroz", "sgaterroutz" + <value>: written value (true or false)
command	DumpStatInfo	Dump ECM output information (error output, interrupt, reset)

Notes:

- *(1*) The setting value MessageLevel is not effected when REGIF_SC_REPORT macro is defined.*
- *(2*) The message belong to dumping register information is not only effected by setting of MessageLevel parameter but also DumpRegisterRW parameter.*
- *(3*) The writing to a register by calling this command is not call the callback function of this register. Its means, the value of register just be updated without processing to any operation of model.*

- *(4*) When this command is used, the error output ports (ecmterroz/ecmterroutz) are not masked (same as writing 1 to ECMmESET/ECMmECLR Register). Its mean setting of ECMEMK0/1 Register does not effect.*

<u>How to use</u>: Below example describes how to use commands/parameters

- Python interface (setting in .py file).

```
#########MessageLevel#########
SCHEAP.ECM_MessageLevel(“RH850.ecm_p1m”, “”)                        #Get message level
SCHEAP.ECM_MessageLevel(“RH850.ecm_p1m”,“info|error|warning|fatal”)#Set message level
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“MessageLevel”)                      #Get message level
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“MessageLevel info|error”)           #Set message level
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“ECMDTMCFG0 MessageLevel”)           #Get message level
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“ECMDTMCFG0 MessageLevel info|error”)#Set message level
#########DumpRegisterRW########
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“DumpRegisterRW”)                    #Get setting value
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“DumpRegisterRW true”)               #Set value
#########DumpInterrupt#######
SCHEAP.ECM_DumpInterrupt(“RH850.ecm_p1m”,“”)                        #Get setting value
SCHEAP.ECM_DumpInterrupt(“RH850.ecm_p1m”,“true”)                    #Set value
#########EnableTransInfo#######
SCHEAP.ECM_EnableTransInfo(“RH850.ecm_p1m”,“”)                      #Get setting value
SCHEAP.ECM_EnableTransInfo(“RH850.ecm_p1m”,“true”)                  #Set value
#########AssertReset##########
SCHEAP.ECM_AssertReset(“RH850.ecm_p1m”,“preset_n 100 20”)
#########SetCLKfreq###########
SCHEAP.ECM_SetCLKfreq(“RH850.ecm_p1m”,“pclk 250 Hz”)
SCHEAP.ECM_SetCLKfreq(“RH850.ecm_p1m”,“pclk 50”)
#########GetCLKfreq###########
SCHEAP.ECM_GetCLKfreq(“RH850.ecm_p1m”,“pclk”)
#########ForceRegister########
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“ECMDTMCFG0 force 0xFF”)           # All side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“master ECMDTMCFG0 force 0xFF”)    #Master side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“checker ECMDTMCFG0 force 0xFF”)   #Checker side
#########ReleaseRegister######
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“ECMDTMCFG0 release”)              # All side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“master ECMDTMCFG0 release”)       #Master side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“checker ECMDTMCFG0 release”)      #Checker side
#########WriteRegister########
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“ECMDTMCFG0 0xFF”)                 # All side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“master ECMDTMCFG0 0xFF”)          #Master side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“checker ECMDTMCFG0 0xFF”)         #Checker side
#########ReadRegister#########
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“ECMDTMCFG0”)                      # All side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“master ECMDTMCFG0”)               #Master side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“checker ECMDTMCFG0”)              #Checker side
#########ListRegister#########
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“”)                                #All sides
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“master”)                          #Master side
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“checker”)                         #Checker side
#########set_param############
SCHEAP.ECM_tgt(“RH850.ecm_p1m”,“set_param m_wr_latency=100,SC_NS”)
#########get_param############
SCHEAP.ECM_tgt(“RH850.ecm_p1m”,“get_param m_bus_clk”)
#########init_param############
SCHEAP.ECM_tgt(“RH850.ecm_p1m”,“init_param”)
#########Help#################
SCHEAP.ECM_help(“RH850.ecm_p1m”)                       #Model command help message
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“help”)                 #Register I/F help message
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“master help”)
SCHEAP.ECM_reg(“RH850.ecm_p1m”,“checker help”)
SCHEAP.ECM_tgt(“RH850.ecm_p1m”,“help”)                  #Target I/F help message
#########Port############
SCHEAP.ECM_port(“RH850.ecm_p1m”,“sgatresz true”)  #Set “sgatresz (M & C)” flag = true
SCHEAP.ECM_port(“RH850.ecm_p1m”,“checker sgatresz”) #Get value “sgatresz (C)” flag
#########DumpStatInfo################
SCHEAP.ECM_DumpStatInfo(“RH850.ecm_p1m”)
```

Figure 6.2: An example of python interface usage

- Command interface (setting in .cmd file).
  - Instance name of model is necessary to put in front of the each keyword.
  - If the target is model, it is necessary to put the keyword "command" in front of each command.
  - If the target is register I/F, it is necessary to put the keyword "reg" in front of each command.
  - And if the target is TLM target I/F, it is necessary to put the keyword "tgt" in front of each command.
  - If a command is required to be broadcast to all targets, the keyword should be empty “”.

```
#Instance     keyword command/parameter
#########MessageLevel#########
reslx.ecm_p1m         MessageLevel                          #Get general message
level
reslx.ecm_p1m command MessageLevel                          #Get message level
reslx.ecm_p1m command MessageLevel info|error|warning|fatal #Set message level
reslx.ecm_p1m reg     MessageLevel                          #Get message level
reslx.ecm_p1m reg     MessageLevel fatal|error|warning|info #Set message level
reslx.ecm_p1m reg     ECMDTMCFG0 MessageLevel               #Get message level
reslx.ecm_p1m reg     ECMDTMCFG0 MessageLevel info|error    #Set message level
#########DumpRegisterRW#######
reslx.ecm_p1m reg     DumpRegisterRW                        #Get setting value
reslx.ecm_p1m reg     DumpRegisterRW true                   #Set value
#########DumpInterrupt#######
reslx.ecm_p1m command DumpInterrupt                         #Get setting value
reslx.ecm_p1m command DumpInterrupt true                    #Set value
#########EnableTransInfo#######
reslx.ecm_p1m command EnableTransInfo                       #Get setting value
reslx.ecm_p1m command EnableTransInfo true                  #Set value
#########AssertReset##########
reslx.ecm_p1m command AssertReset PRESETn 100 20
#########SetCLKfreq###########
reslx.ecm_p1m command SetCLKfreq pclk 250 Hz
reslx.ecm_p1m command SetCLKfreq pclk 50
#########GetCLKfreq###########
reslx.ecm_p1m command GetCLKfreq pclk
#########ForceRegister########
reslx.ecm_p1m reg     ECMDTMCFG0 force 0xFF
reslx.ecm_p1m reg     master ECMDTMCFG0 force 0xFF
reslx.ecm_p1m reg     checker ECMDTMCFG0 force 0xFF
#########ReleaseRegister######
reslx.ecm_p1m reg     ECMDTMCFG0 release
reslx.ecm_p1m reg     master ECMDTMCFG0 release
reslx.ecm_p1m reg     checker ECMDTMCFG0 release
#########WriteRegister########
reslx.ecm_p1m reg     ECMDTMCFG0 0xFF
reslx.ecm_p1m reg     master ECMDTMCFG0 0xFF
reslx.ecm_p1m reg     checker ECMDTMCFG0 0xFF
#########ReadRegister#########
reslx.ecm_p1m reg     ECMDTMCFG0
reslx.ecm_p1m reg     master ECMDTMCFG0
reslx.ecm_p1m reg     checker ECMDTMCFG0
#########ListRegister#########
reslx.ecm_p1m reg
reslx.ecm_p1m reg master
reslx.ecm_p1m reg checker
#########set_param############
reslx.ecm_p1m tgt     set_param m_wr_latency=100,SC_NS
#########get_param############
reslx.ecm_p1m tgt     get_param m_bus_clk
#########init_param############
reslx.ecm_p1m tgt     init_param
#########Help#################
reslx.ecm_p1m         help                            #General help message
reslx.ecm_p1m command help                            #Model command help message
reslx.ecm_p1m reg     help                            #Register I/F help message
reslx.ecm_p1m reg     master help
reslx.ecm_p1m reg     checker help
reslx.ecm_p1m tgt     help                            #Target I/F help message
#########Port############
reslx.ecm_p1m command port sgatresz 1               #Set “sgatresz (M & C)” flag = 1
reslx.ecm_p1m command port checker sgatresz         #Get value “sgatresz (C)” flag
#########DumpStatInfo############
reslx.ecm_p1m command DumpStatInfo
```

Figure 6.3: An example of command interface usage

6.5. Message style

6.5.1. Help messages

6.5.1.1. Model command help messages

Table 6.4: Dumping model command help message description

Condition	This message is dumped out when model command “help” is called.
Output	This message's kind is printed to standard output (console).

```
--- command ---
help                                          Show direction
MessageLevel   <fatal|error|warning|info>     Select debug message level (Default: fatal,error)
DumpInterrupt <enable>                        Enable/disable interrupt information display when an
interrupt is sent (Default: false)
EnableTransInfo <enable>                      Enable/disable error input information display when an
error input port is updated (Default:false)
AssertReset <rst_name> <start_time> <period>  Assert and negate reset signal
SetCLKfreq <clk_name> <clk_freq> [<unit>]     Set clock frequency
GetCLKfreq <clk_name>                         Get clock frequency
DumpStatInfo                                  Dump ECM output information (error output, interrupt,
reset)
port <port_name> [<value>]                    Set/Get value to a specified internal port
```

6.5.1.2. Register I/F help messages

Table 6.5: Dumping register help message description

Condition	This message is dumped out when register I/F “help” is called.
Output	This message's kind is printed to standard output (console).

```
--- reg ---
reg MessageLevel <fatal|error|warning|info> Select debug message level (Default: fatal|error)
reg DumpRegisterRW <true/false>             Select dump register access information (Default: false)
reg <register_name> MessageLevel <fatal|error|warning|info>
                                            Select debug message level for register (Default: fatal|error)
reg <register_name> force <value>           Force register with setting value
reg <register_name> release                 Release register from force value
reg <register_name> <value>                 Write a value into register
reg <register_name>                         Read value of register
reg help                                    Show a direction
```

6.5.1.3. TLM Target I/F help messages

Table 6.6: Dumping target help message description

Condition	This message is dumped out when TLM target I/F “help” is called.
Output	This message's kind is printed to standard output (console).

```
<model’s instance> has the following target I/F commands.
Command                    Description
-----------------------------------------------------------------------------------------
set_param <term> <value>    : Set simulation information about access to target.
        <term>  : m_bus_clk | m_bus_gnt | m_bus_rgnt | m_buf_size
                  | m_wr_latency | m_rd_latency | m_phase_mode
                  | m_p_log_file | m_wr_log | m_rd_log | m_msg_out_lvl
                  | m_wr_req_latency | m_rd_req_latency | m_fw_req_phase
                  | m_info_displayed
        <value> : Please see tlm_common_class spec sheet.
get_param <term>            : Get simulation information about access to target.
init_param                  : Initialize simulation information.
```

6.5.2. Register RW messages

Table 6.7: Dumping Register RW message description

Condition	This message is outputted when register of model is accessed and parameter DumpRegisterRW is set "true".
Output	This message's kind is printed to standard output (console).
Format: Info: <hier_instance_name>: [<time>ps] REG [<reg_name>] R Size = <size> Addr = <reg_address> Data = <reg_value> Info: <hier_instance_name>: [<time>ps] REG [<reg_name>] W Size = <size> Addr = <reg_address> Data = <reg_value> : <old_value> => <new_value>	
Tag name	**Description**
hier_instance_name	Hierarchy instance name of model is being used.
time	Simulation time
reg_name	Name of accessed register.
size	Access size.
reg_address	Register's address.
reg_value	Register's value.
old_value	Register's value before writing.
new_value	Register's value after writing.

6.5.3. Error and debugging messages

6.5.3.1. Error and debugging messages style

Table 6.8: Error and debugging message description

Condition	This message's kind is output when error occurs or some important events occur. It's depended on setting of parameter MessageLevel. Detailed conditions are described in the "Description" column of Table 6.9.
Output	This message's kind is printed to standard output (console).
Format: <severity>: <hier_instance_name>: [<time><unit>] <Message content>	
Tag name	**Description**
severity	Kind of message's severity.
hier_instance_name	Hierarchy instance name of model is being used.
time	Simulation time.
unit	Simulation time's unit.
Message content	Message content (message list is described in Table 6.9).

6.5.3.2. List of error and debugging messages

Table 6.9: Error and debug messages

No.	Level	Type	Message	Description
1	error	user	Invalid access address <address>	Users access the model's register with invalid address.
2	error	user	Invalid access address <address> with access size <size> bytes	Users access the model's register with invalid address and size.
3	error	user	Invalid access size: <size> bytes	Users access the model's register with invalid size.
4	error	user	Writing access size to <register_name> at address <address> is wrong: <size> byte(s).	Users write the value to register with unsupported size.
5	error	user	Reading access size to <register_name> at address <address> is wrong: <size> byte(s).	Users read the value in register with unsupported size.
6	error	user	command name "<input_command>" is invalid (*)	Users call command with invalid command.

7	error	user	<command name> has too much arguments (*)	Users call command with number of argument is wrong.
8	error	user	<command name> command needs an argument [true/false] (*)	Users call command with invalid arguments.
9	error	user	Register name <register_name> is invalid (*)	Users call command of register I/F with register name is wrong. The register names are list in Table 4.1.
10	error	user	Invalid force value (*)	Users call force command of register I/F with invalid arguments.
11	error	user	Invalid write value (*)	Users call release command of register I/F with invalid arguments.
12	error	user	Wrong command : (<input_command> (*)	Users call command of register I/F which is unsupported.
13	error	user	wrong argument: <argument (s)> (<input_command>) : Type <model_name> help (*)	Users call command with invalid arguments.
14	error	user	wrong argument (<input_command>) : Type <model_name> help (*)	
15	error	user	wrong number of arguments (<input_command>) : Type <model_name> help (*)	Users call command with invalid number arguments.
16	-	user	Wrong number of argument for <command name> command.	Users call command (via python I/F) with invalid number arguments.
17	warning	user	<register_name> is blocked writing from Bus I/F.	Users try to write to forced register.
18	warning	user	Cannot launch call-back function during reset period	Users read the register during reset period.
19	warning	user	Cannot write 1 to reserved bit	Users write value to reserved bit of register.
20	warning	user	Cannot write during reset period	Users write value to register during reset operation.
21	warning	user	Should read all bit in a register	Users read the register with read access size if smaller than register size.
22	warning	user	Clock name (<clock_name>) is invalid.	Users call SetCLKfreq or GetCLKfreq command with clock name is wrong.
23	warning	user	Frequency unit (<unit>) is wrong; frequency unit (Hz) is set as default.	Users call SetCLKfreq with frequency unit is wrong. The frequency unit must be Hz, KHz, MHz, GHz.
24	warning	user	The <value> period is less than 1 unit time of system.	Users issue clock period is less than 1 unit time of system.
25	warning	user	Reset name (<reset_name>) is invalid.	Users call AssertReset command with reset name is wrong.
26	warning	user	The AssertReset command of <reset_name> is called in the reset operation (<reset_name>) of the model. So this calling is ignored.	Users call AssertReset command during reset operation.
27	warning	internal	The error output signals from Master and Checker are different.	“sgaterroz” flags sent from Master and Checker sides are different. This is the cause of “ECM compare error” occurs inside model.
28	warning	user	Write to write protection register at address <address> fails by the write	Users write to a different register in same module during write procedure to

			access to other address <address>.	a write-protected register. (refer to Chapter 7.9)
29	warning	user	Write to write protection register at address <address> fails at step <step>.	Users write to a write-protected register with wrong sequence. (refer to Chapter 7.9)
30	warning	user	Clearing of the ERROROUT output is not possible. All errors not masked by ECMEMK0/1, or the ECMmSSE130 bit in the ECMmESSTR1 register are not cleared beforehand.	Users write 1 to ECMmECLR register or call “port” command to clear error output ports (“ecmterroz”, “ecmterroutz”) when all errors not masked by ECMEMK0/1, or the ECMmSSE130 bit in the ECMmESSTR1 register are not cleared beforehand.
31	warning	user	Do not set the delay timer for clock monitor upper limit/lower limit errors (errors no. 4 to 11).	Users write to ECMDTMCFG0/2 register to enable delay timer for error source no.4 to 11.
32	warning	user	Port name (<port_name>) is invalid.	Users call “port” command with port name is wrong.
33	warning	user	The delay timer clock is changed while timer is running.	Users change clock used for timer during timer running.
34	info	user	Error status is cleared while error input source still active.	Users write to ECMESSTC0/1 register to clear error flag(s) in ECMmESSTR0/1 register when error input source still active.
35	info	user	<clock_name> frequency is <frequency> <unit>.	Users call GetCLKFreq with valid clock name.
36	info	user	<clock_name> frequency is zero.	Users do an action when clock frequency is zero value.
37	info	user	The model will be reset (<reset_name>) for <period_time> ns after <delay_time> ns.	Users call AssertReset command with start reset time and reset period.
38	info	internal	The model is reset by AssertReset command of <reset_name>.	Reset by AssertReset command is active.
39	info	user	The reset signal of <reset_name> is asserted.	Users activate reset signal.
40	info	user	The reset signal of <reset_name> is negated.	Users deactivate reset signal.
41	info	internal	Reset period of <reset_name> is over.	Reset period of reset which is set by AssertReset is over.
42	info	internal	Initialize <register_name> (<value>)	Registers are initialized by reset signal(s)
43	info	internal	INT [<interrupt_name>] Assert. INT [<interrupt_name>] Negate.	Interrupt information is dumped when parameter DumpInterrupt is set true and interrupt signal is changed.
44	info	internal	Error input [<error_name>] Assert. Error input [<error_name>] Negate.	Error input information is dumped when parameter EnableTransInfo is set true and error input source is changed.
45	info	user	ECM output information: + ecmterroz: <value> + ecmterroutz: <value> + ecmtresz: <value> + ecmti: <value> + ecmtnmi: <value>	Users call DumpStatInfo command
46	info	user	<port_name> port value is <value>.	Users call “port” command without

				setting value. Port value is read.
47	info	user	Set dummy value <value> to <port_name> port.	Users call “port” command with setting value. Port value is written.
48	info	internal	A request to start delay timer is called while the delay timer is running.	Another error by which the delay timer is started occurs while the delay timer is running.
49	info	internal	Delay timer compare value is zero.	An error by which the delay timer is started occurs and ECMDTMCMP register value is 0.
50	info	internal	Delay timer starts counting.	An error by which the delay timer is started occurs.
51	info	internal	Delay timer stops counting.	The counter value of the delay timer matches with the value of the ECM delay timer compare register (ECMDTMCMP).
52	warning	user	Cannot write data to ECM delay timer compare register (ECMDTMCMP) while the delay timer is not stopped.	Users update value of ECMDTMCMP register while the delay timer is not stopped.

Note:

- (*) The dumping message is not depend on setting value of MessageLevel parameter. This message is returned to Command Handler (commandHandler.h)/Python Handler (PY_ECM_P1M.cpp). Command/Python Handler will decide the message content will be dumped.

6.6. Defined macro and template

- There is no template in model.
- There are three macros IS_RESET_ACTIVE_LOW, REGIF_SC_REPORT, and REGIF_NOT_USE_SYSTEMC in the model
    - When users define the macro IS_RESET_ACTIVE_LOW, reset active Low level. If not, reset actives High level.
    - If users define the macro REGIF_SC_REPORT, the SC_REPORT function is used. Otherwise, the “printf” function is used. This macro should be not defined if users defined REGIF_NOT_USE_SYSTEMC.
    - Users can define macro REGIF_NOT_USE_SYSTEMC to remove SystemC part.

7. Flow diagram

7.1. Sequence flow

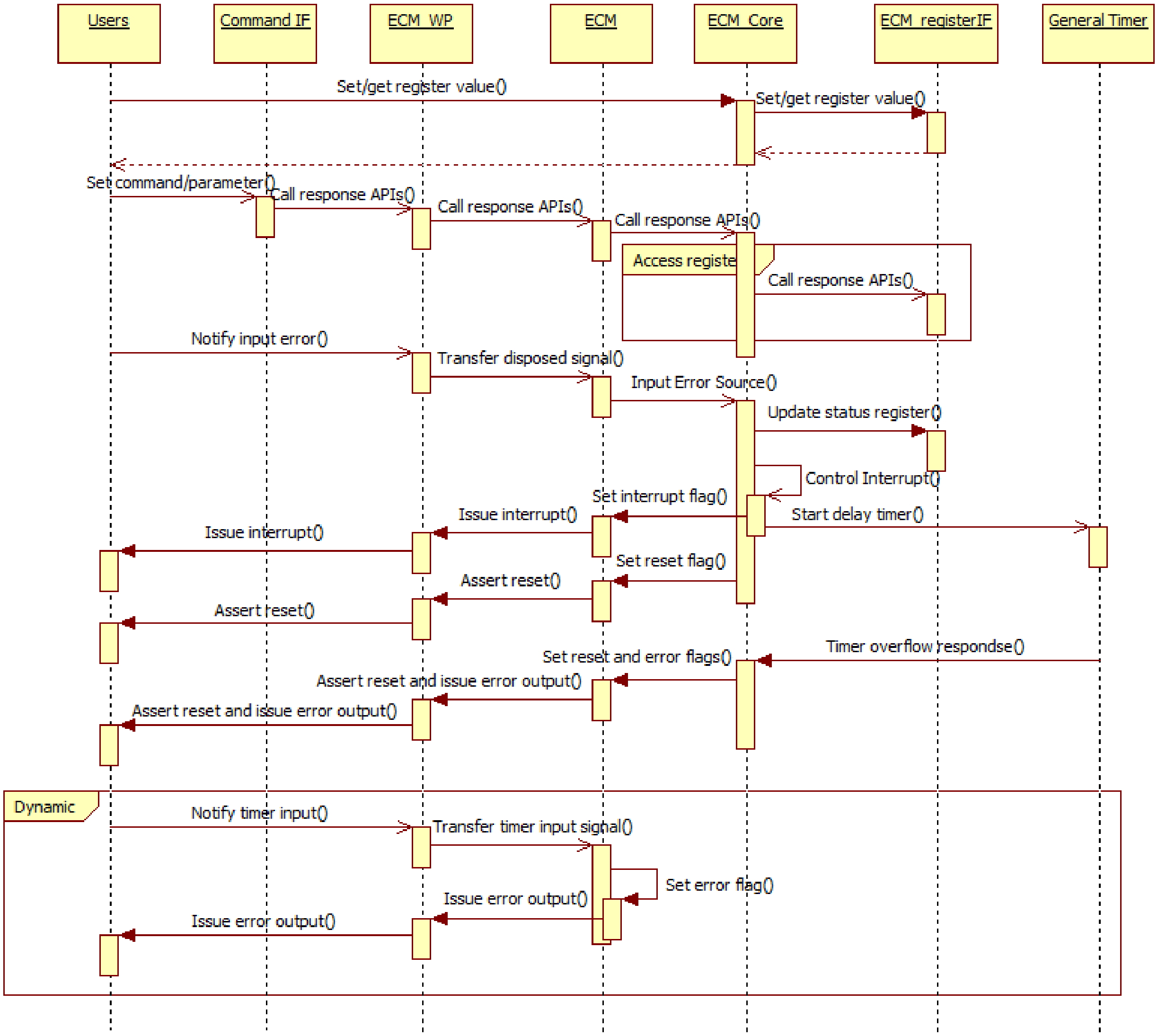


Figure 7.1: Sequence flow

Explanation:

- ➢ Users interact with parameters/commands of the model via Command I/F. (The commands and parameters are described in Table 6.2 and Table 6.3).
- ➢ Users access the registers of model via the TLM target sockets m_tgt_sockets[2:0].
- ➢ When model receives an error input, the Input Error Source is called to set error status to ECMmESSTR0/1 register (refer to Chapter 7.5). After that, the Control Interrupt is called to issue internal interrupt, internal reset and error output signals according corresponding error source. At this time, the delay timer is started for a wait time to issue

internal reset and error output signals according delay timer overflow (refer to Chapter 7.6).

- In case error pulse signal is selected in dynamic mode, if there is no error occurred, “ecmterroutz” is output according “ecmttin” input signal.

7.2. State diagram

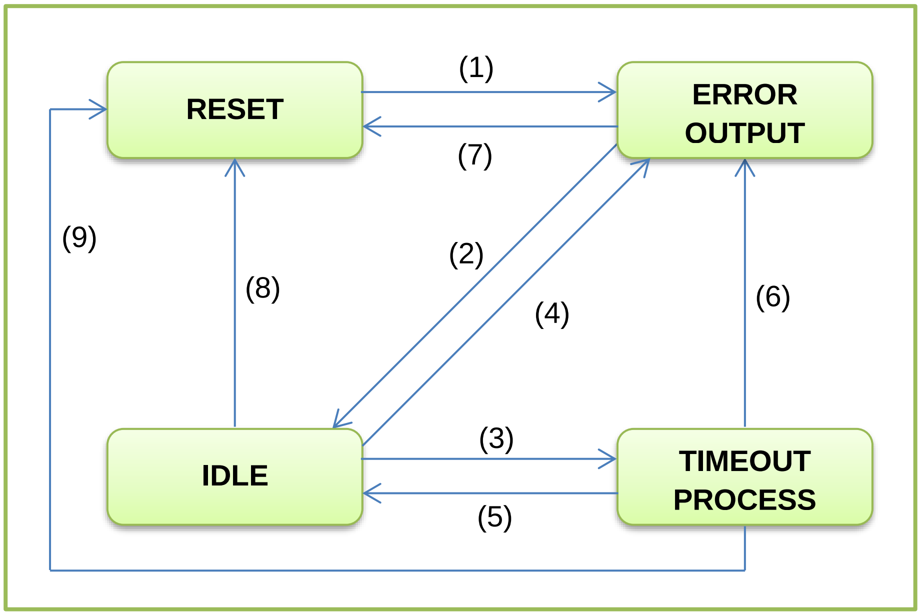


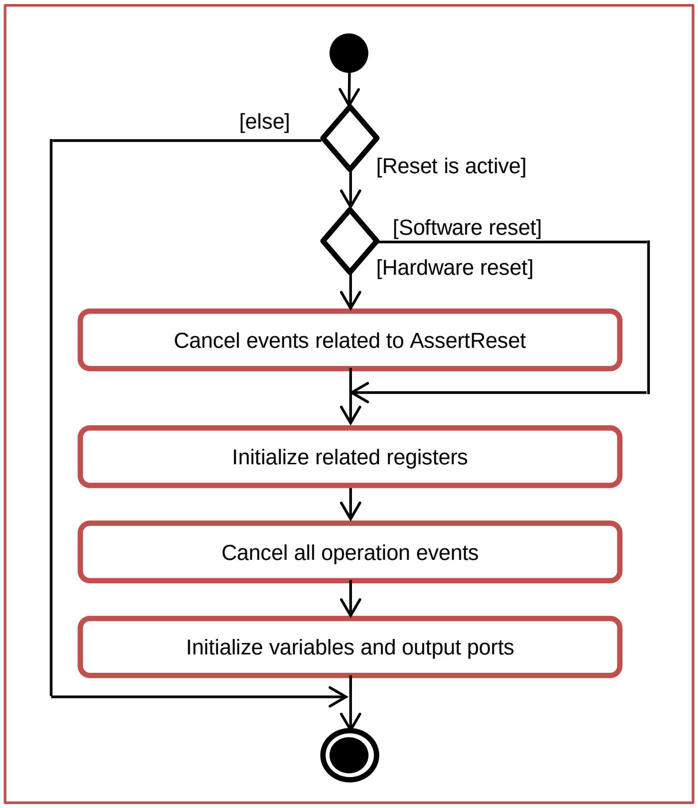
Figure 7.2: State diagram

Explanation:

- RESET: Transition of RESET state to ERROR OUTPUT state is described as below:
  - RESET → ERROR OUTPUT: The state of the model is changed after reset (both “preset_n” and “extresetz”) is negated. The error output ports (“ecmterroz” and “ecmterroutz”) are reset to active level (“Low”). (1)
- ERROR OUTPUT: Transition of this state to other states is described as below:
  - ERROR OUTPUT → RESET: If the reset is asserted (by hardware reset signal or user reset command), the state of the model is changed to RESET state. (7)
  - ERROR OUTPUT → IDLE: If users clear error output port by writing 1 to ECMmECLR register, the state is changed to IDLE state. (2)
- IDLE: Transition of this state to other states is described as below:
  - IDLE → RESET: If the reset is asserted (by hardware reset signal or user reset command), the state of the model is changed to RESET state. (8)
  - IDLE → TIMEOUT PROCESS: The state of the model is changed to TIMEOUT PROCESS state when an error source inputs/or a pseudo error is set. In this state, a delay timer is started after the interrupt signals are issued for waiting time before transfer to ERROR OUTPUT state. (3)

  - IDLE → ERROR OUTPUT: If users set error output port by writing 1 to ECMmESET register, the state is changed to ERROR OUTPUT state. In this case, the interrupt signals are not issued and delay timer is not started accordingly. (4)
- TIMEOUT PROCESS: Transition of this state to other states is described as below:
  - TIMEOUT PROCESS → RESET: If the reset is asserted (by hardware reset signal or user reset command), the state of the model is changed to RESET state. (9)
  - TIMEOUT PROCESS → IDLE: During waiting time after delay timer started, if delay timer is stopped by set ECMDTMCTL.ECMSTP = 1 before delay timer overflows, the state is changed to IDLE state. (5)
  - TIMEOUT PROCESS → ERROR OUTPUT: When delay timer overflows, the state is changed to ERROR OUTPUT state. (6)

7.3. Reset flow



[else]
[Reset is active]
[Software reset]
[Hardware reset]
Cancel events related to AssertReset
Initialize related registers
Cancel all operation events
Initialize variables and output ports

Figure 7.3: Reset flow

Explanation:

- The ECM model has 2 hardware resets and 2 corresponding software resets of “preset_n” and “extresetz”.
- Users can reset model by software reset via Command IF or by reset signal via reset ports. The Figure 7.4 shows the relationship between software reset and hardware reset.
- If the reset is active (“preset_n” or “extresetz” is asserted), the model operates as following:

- ○ Cancel events related to the AssertReset if reset signal is received.
- ○ Initialize related registers:
  - ✓ All ECM's registers except ECMmESSTR0/1 are reset by "preset_n".
  - ✓ ECMmESSTR0/1 registers are reset by only "extresetz".
- ○ Cancel all operation events
- ○ Initialize all internal variables and output ports.

Note:

- Related registers cannot be accessed (read/write) during reset period.

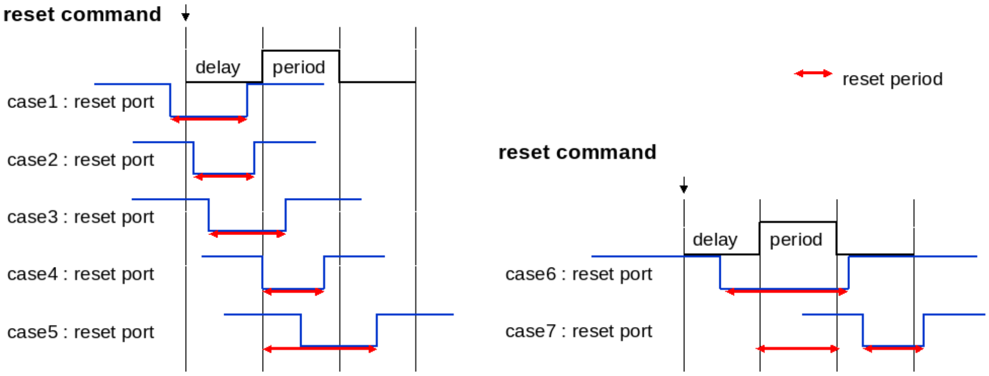


Figure 7.4: Relationship between software reset and hardware reset

7.4. Command/parameter configuration operation flow

- Users set/call the parameter/command of model via the Command I/F. Setting/calling operation is described as Figure 7.5 -> Figure 7.9.
- The function of the parameters and commands is described in Table 6.2 and Table 6.3.

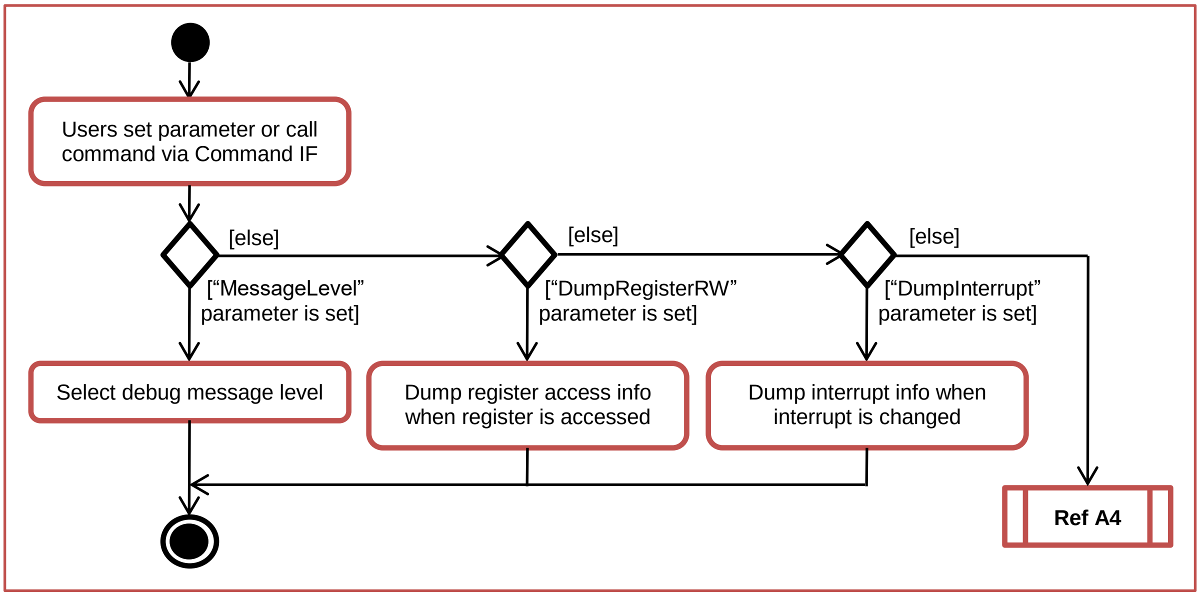

Users set parameter or call command via Command IF
[else]
[“MessageLevel” parameter is set]
[else]
[“DumpRegisterRW” parameter is set]
[else]
[“DumpInterrupt” parameter is set]
Select debug message level
Dump register access info when register is accessed
Dump interrupt info when interrupt is changed
Ref A4

Figure 7.5: Command/parameter configuration operation flow (1/5)

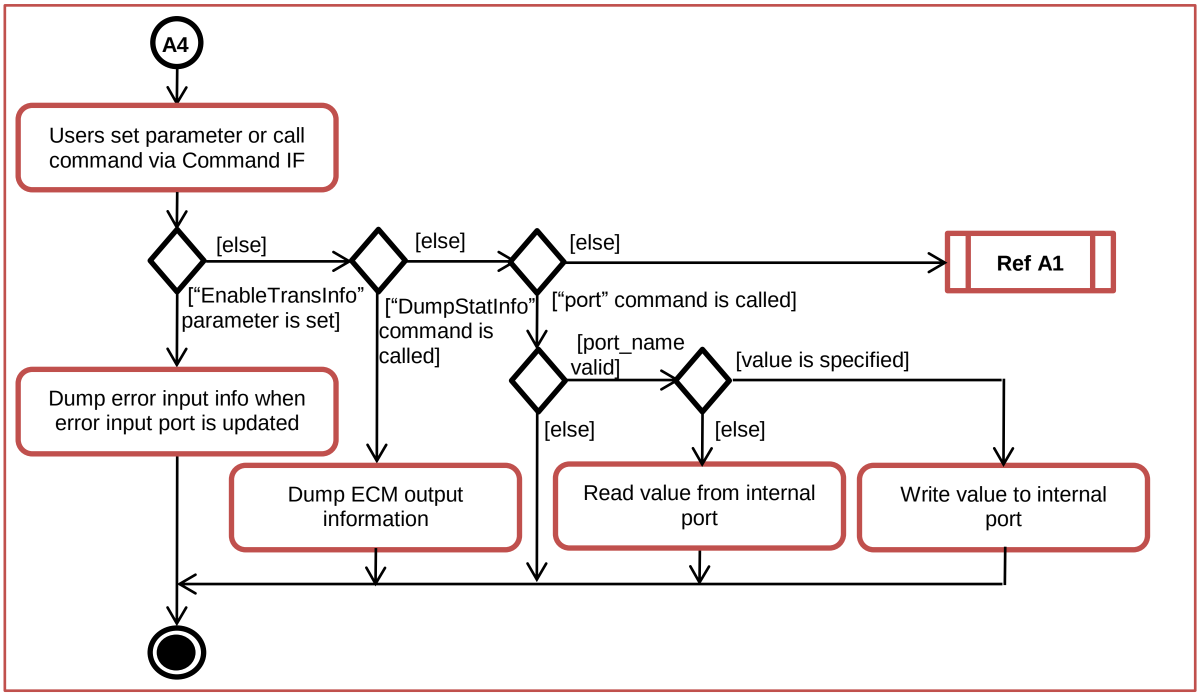
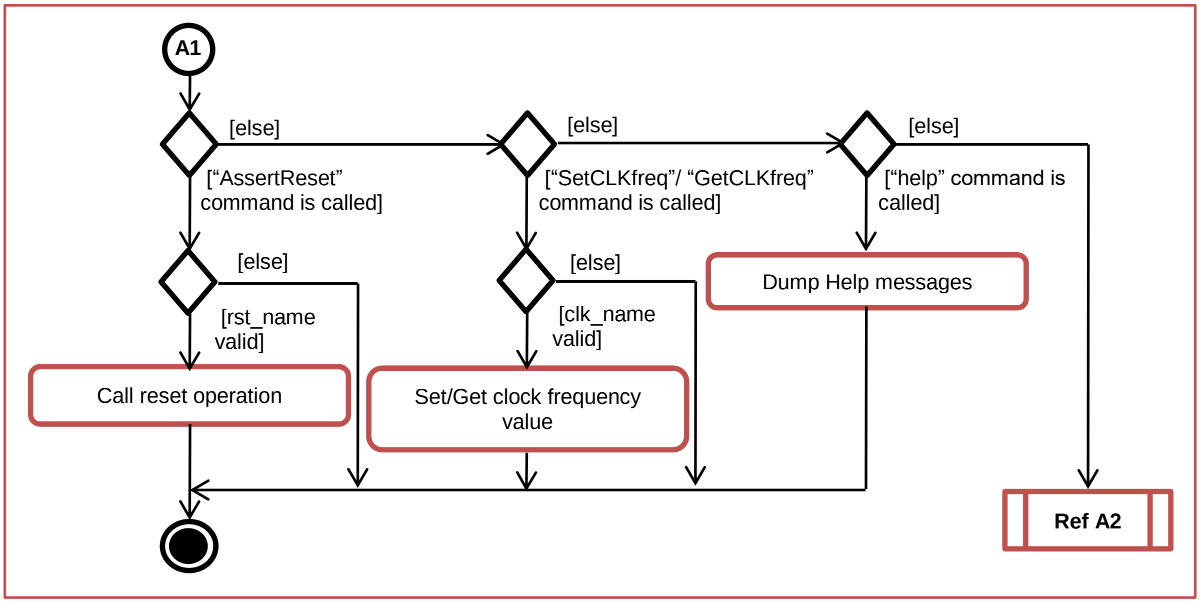
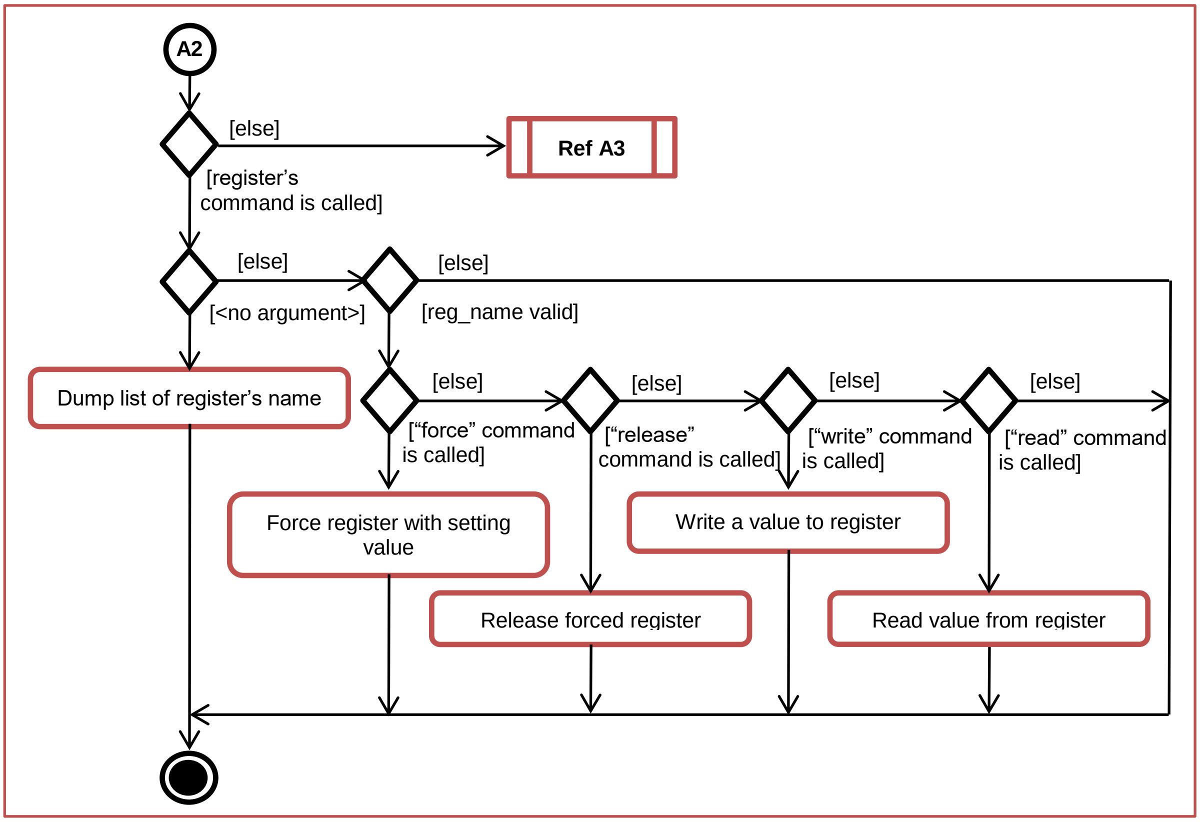

A4
Users set parameter or call command via Command IF
[else]
[“EnableTransInfo” parameter is set]
[else]
[“DumpStatInfo” command is called]
[else]
Ref A1
[“port” command is called]
[port_name valid]
[value is specified]
[else]
[else]
Dump error input info when error input port is updated
Dump ECM output information
Read value from internal port
Write value to internal port

Figure 7.6: Command/parameter configuration operation flow (2/5)



A1
[else]
["AssertReset" command is called]
[else]
[rst_name valid]
Call reset operation
[else]
["SetCLKfreq"/ "GetCLKfreq" command is called]
[else]
[clk_name valid]
Set/Get clock frequency value
[else]
["help" command is called]
Dump Help messages
Ref A2

Figure 7.7: Command/parameter configuration operation flow (3/5)



A2
[else]
Ref A3
[register's command is called]
[else]
[<no argument>]
Dump list of register's name
[else]
[reg_name valid]
[else]
["force" command is called]
Force register with setting value
[else]
["release" command is called]
Release forced register
[else]
["write" command is called]
Write a value to register
[else]
["read" command is called]
Read value from register

Figure 7.8: Command/parameter configuration operation flow (4/5)

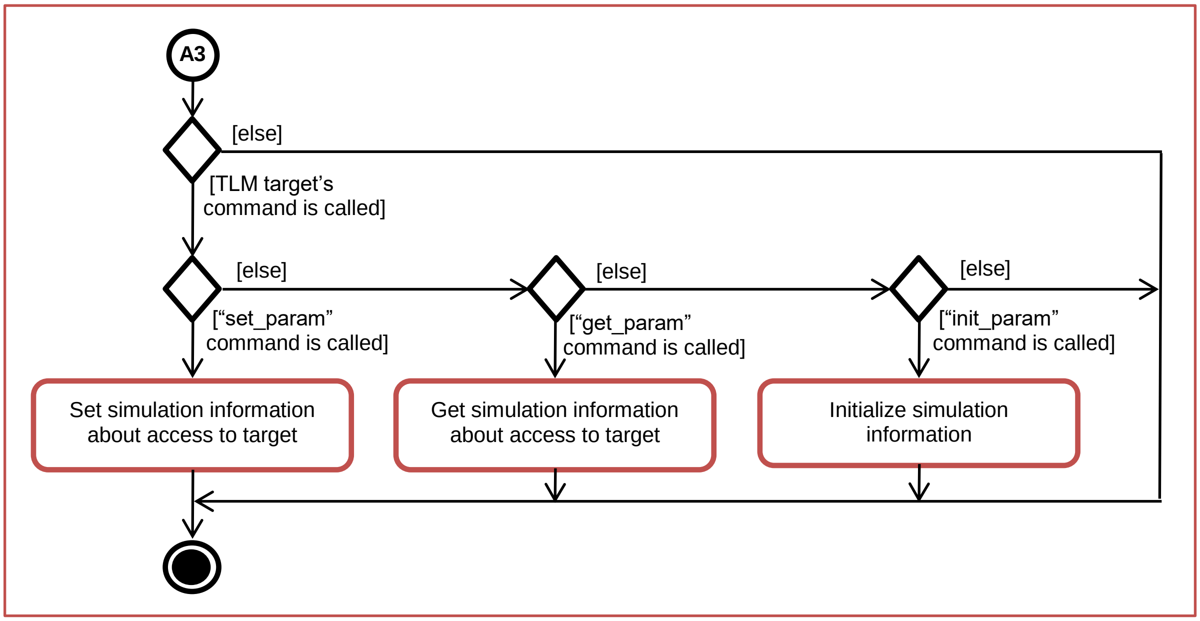

A3
[else]
[TLM target's command is called]
[else]
["set_param" command is called]
[else]
["get_param" command is called]
[else]
["init_param" command is called]
Set simulation information about access to target
Get simulation information about access to target
Initialize simulation information

Figure 7.9: Command/parameter configuration operation flow (5/5)

7.5. Input error source flow

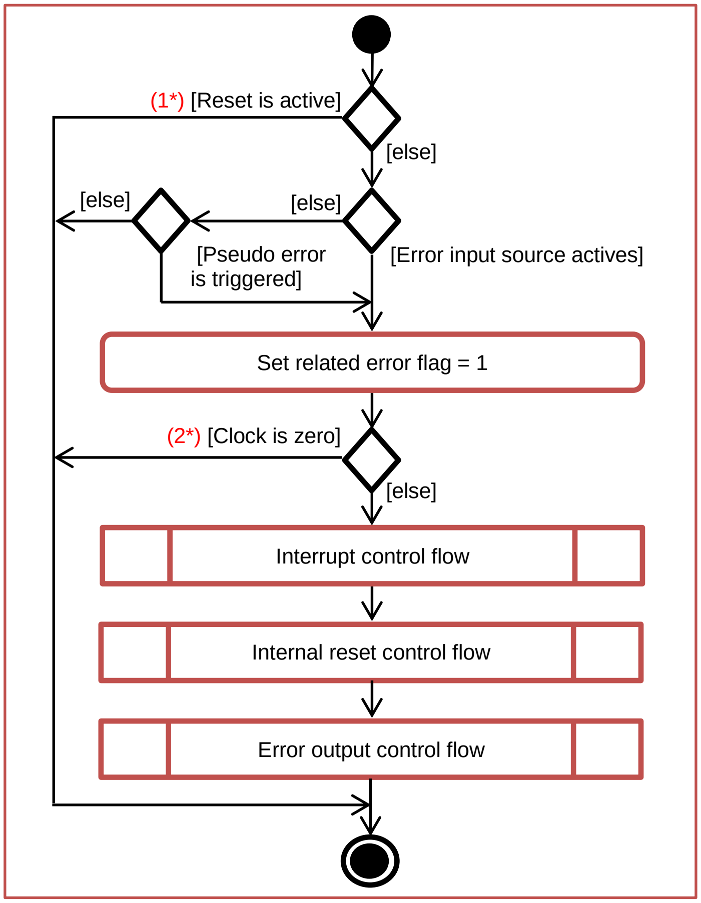

(1*) [Reset is active]
[else]
[else]
[else]
[Pseudo error is triggered]
[Error input source actives]
Set related error flag = 1
(2*) [Clock is zero]
[else]
Interrupt control flow
Internal reset control flow
Error output control flow

Figure 7.10: Input error source flow

Explanation:

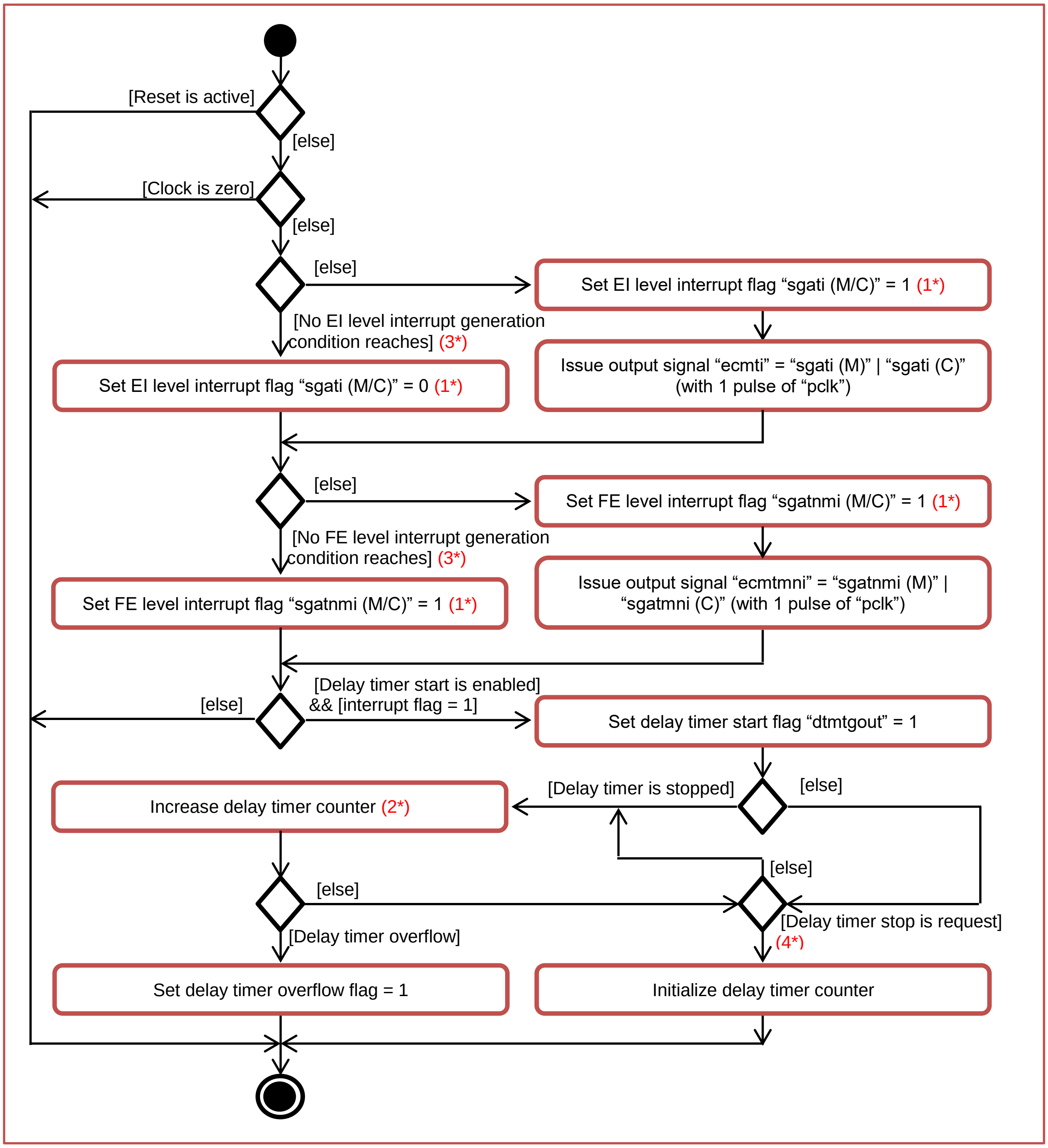
- When an error input source is active by receiving a signal via "ecmterrin" ports, the operation in response to this error source is as below:
  - The related error flag (corresponding bit in ECMmESSTR0/1 register) is set to 1 (without dependence on setting of ECMEMK0/1 register).
  - After that, the interrupt signal(s) ("ecmti" and/or "ecmtnmi") can be output according setting of ECMMICFG0/1 and ECMNMICFG0/1 registers (refer to Chapter 7.6).
  - Besides, the internal reset signal ("ecmtresz") and error output signals ("ecmterroz" and "ecmterroutz") can be output at this time (refer to Chapters 7.7 and 7.8)
- If a pseudo error is triggered by writing 1 to a bit in ECMPE0/1 register, the operation in response to the generation of a pseudo error is identical to that in response to a real error source.
- Besides, the error input source can be active by a loop-back error occurred in error output process (error "ECM compare error" (error No.29)). (refer to Chapter 7.8)

  Notes:

  - (1) When reset operation is active ("preset_n" or "extresetz" is asserted), the receiving input signal is suspended.*

- (2) When clock is zero ("pclk" = 0), model does not operate/stops operation. So, the receiving input signal is suspended too. However, error causes can be captured without clock and the error status can be set. (ref[7]/Chapter 21.1)*

7.6. Interrupt control flow



[Reset is active]
[else]
[Clock is zero]
[else]
[else]
Set EI level interrupt flag "sgati (M/C)" = 1 (1*)
[No EI level interrupt generation condition reaches] (3*)
Issue output signal "ecmti" = "sgati (M)" | "sgati (C)" (with 1 pulse of "pclk")
Set EI level interrupt flag "sgati (M/C)" = 0 (1*)
[else]
Set FE level interrupt flag "sgatnmi (M/C)" = 1 (1*)
[No FE level interrupt generation condition reaches] (3*)
Issue output signal "ecmtmni" = "sgatnmi (M)" | "sgatmni (C)" (with 1 pulse of "pclk")
Set FE level interrupt flag "sgatnmi (M/C)" = 1 (1*)
[Delay timer start is enabled] && [interrupt flag = 1]
[else]
Set delay timer start flag "dtmtgout" = 1
[Delay timer is stopped]
[else]
Increase delay timer counter (2*)
[else]
[else]
[Delay timer stop is request] (4*)
[Delay timer overflow]
Set delay timer overflow flag = 1
Initialize delay timer counter

Figure 7.11: Interrupt control flow

Explanation:

- After an error flag is set to 1 (in both Master and Checker sides), the interrupt flag(s) of each side (Master/Checker) will be set to 1 if the corresponding interrupt generation is enabled by setting ECMMICFG0/1 and ECMNMICFG0/1 registers.
  - If the corresponding EI level interrupt generation is enabled by setting ECMMICFG0/1 register, the "sgati" flag is set accordingly.
  - As the same way, "sgatnmi" flag is set if FE level interrupt generation is enabled.
  - According this change, the interrupt flag values from Master and Checker sides will be concentrated by **OR** operator, the interrupt signal(s) ("ecmti" and/or "ecmnmi") will be issued in 1 pulse of "pclk" with the active level is result of this **OR** operator. (ref[5]/Figure 32.2)
- When the interrupt signal asserts, if corresponding bit in ECMDTMCFG0/1/2/3 register = 1, ECMDTMCTL.ECMSTA = 1 and delay timer is stopped (delay timer is not counting), the delay timer start flag ("dtmtgout") of each side will be set to 1 and the delay timer is started to count up with count clock is "pclk".
  - During delay timer running, if users request to stop delay timer by set ECMDTMCTL.ECMSTP = 1, the delay timer counter is initialized and stop counting (ECMDTMCTL.ECMSTA = 0, ECMDTMR = 0).
  - Otherwise, delay timer continues count up until the count value matches value set in ECMDTMCMP register. At this overflow time, delay timer overflow flag is set to 1 (ECMmESSTR1. ECMmSSE129 = 1).
- Besides, if there is no interrupt generation condition reached, the interrupt flag ("sgati"/ "sgatnmi") is set to 0.

  Notes:

  - (1) M/C = Master/Checker. According error flag is set in ECMmESSTR0/1 register, the corresponding interrupt flag is set.*

  - (2) When delay timer is running, users can read ECMDTMR register for the delay timer counter value.*

  - (3) "The interrupt generation condition" = "error status bit in ECMmESSTR0/1 registers is set to 1" and "corresponding interrupt generation is enabled by set 1 to a bit in ECMMICFG0/1 and/or ECMNMICFG0/1 register(s)".*

  - (4) After delay timer already started, setting to ECMDTMCFG0/1/2/3 registers is not effected. Its mean writing data to this register has to be conducted while the delay timer is stopped.*

7.7. Internal reset control flow

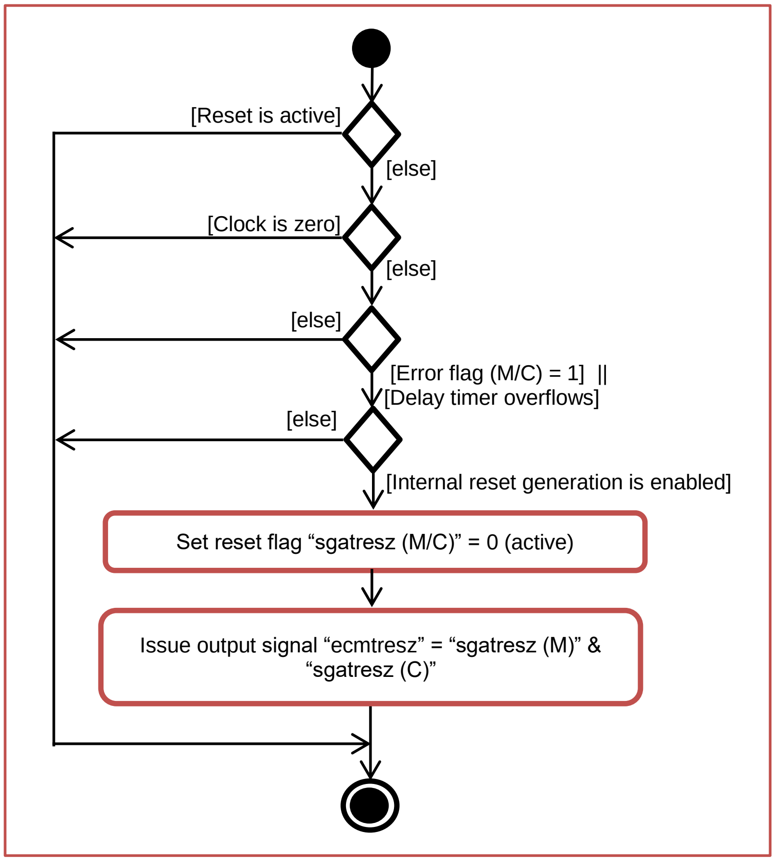

[Reset is active]
[else]
[Clock is zero]
[else]
[else]
[Error flag (M/C) = 1] ||
[Delay timer overflows]
[else]
[Internal reset generation is enabled]
Set reset flag “sgatresz (M/C)” = 0 (active)
Issue output signal “ecmtresz” = “sgatresz (M)” & “sgatresz (C)”

Figure 7.12: Internal reset control flow

Explanation:

- If corresponding bit in ECMIRCFG0/1 register = 1, the reset flag (“sgatresz”) (Master/Checker side) is set to 0 (active level) when one of below cases occurs:
  - The counter value of the delay timer matches with the value of the ECM delay timer compare register (delay timer overflows: ECMmESSTR1. ECMmSSE129 is set to 1).
  - An error flag is set to 1 in ECMmESSTR0/1 register.
- The reset flag “sgatresz” (Master/Checker side) is just set from 0 to 1 by asserting “preset_n”/“extresetz” signal.
- Besides, the reset flag values from Master and Checker sides will be concentrated by **AND** operator, the reset signal (“ecmtresz”) will be issued with the active level is result of this **AND** operator. (ref[5]/Figure 32.2)

7.8. Error output control flow

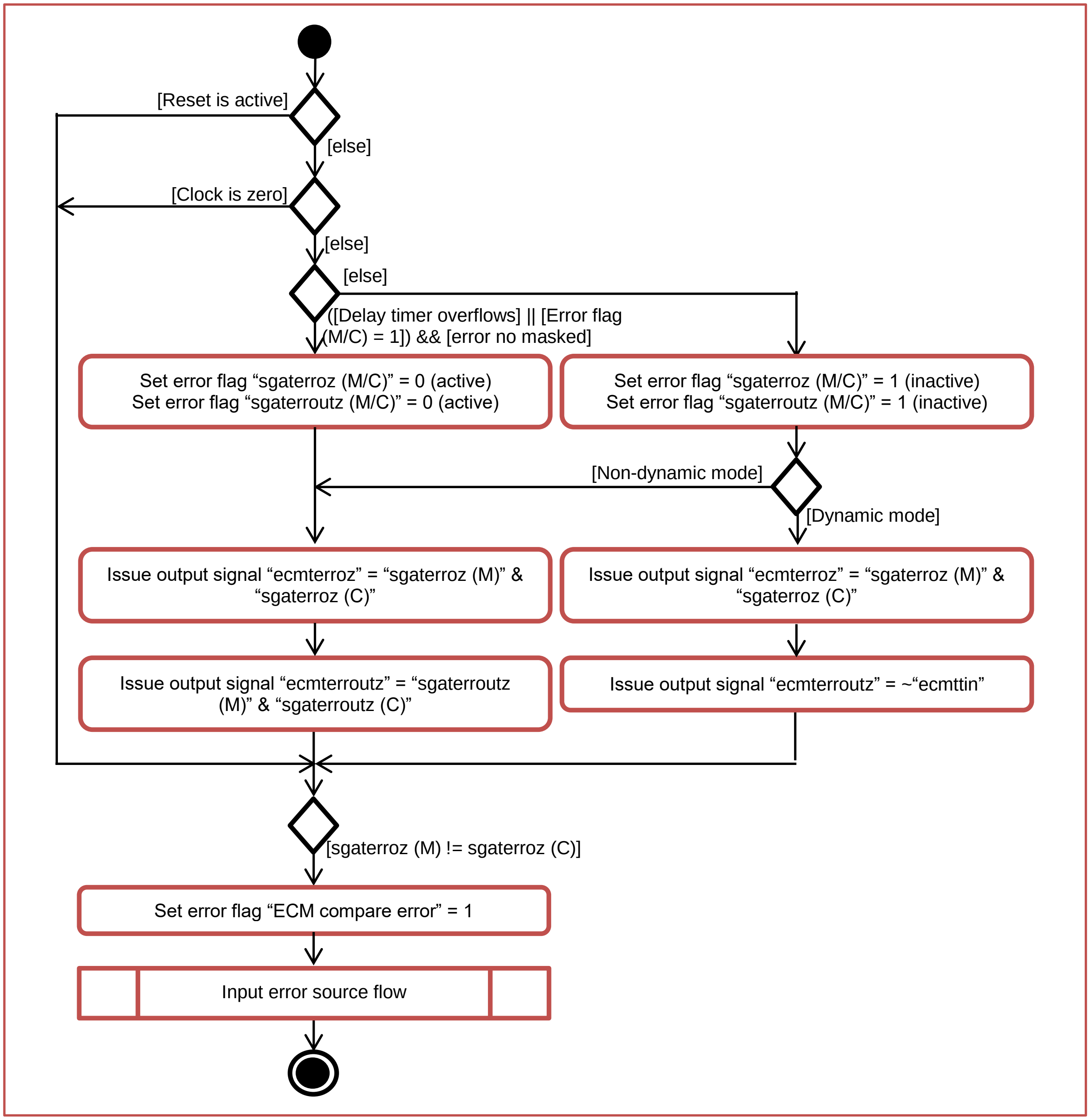

[Reset is active]
[else]
[Clock is zero]
[else]
[else]
([Delay timer overflows] || [Error flag (M/C) = 1]) && [error no masked]
Set error flag “sgaterroz (M/C)” = 0 (active)
Set error flag “sgaterroutz (M/C)” = 0 (active)
Set error flag “sgaterroz (M/C)” = 1 (inactive)
Set error flag “sgaterroutz (M/C)” = 1 (inactive)
[Non-dynamic mode]
[Dynamic mode]
Issue output signal “ecmterroz” = “sgaterroz (M)” & “sgaterroz (C)”
Issue output signal “ecmterroz” = “sgaterroz (M)” & “sgaterroz (C)”
Issue output signal “ecmterroutz” = “sgaterroutz (M)” & “sgaterroutz (C)”
Issue output signal “ecmterroutz” = ~“ecmttin”
[sgaterroz (M) != sgaterroz (C)]
Set error flag “ECM compare error” = 1
Input error source flow

Figure 7.13: Error output control flow

Explanation:

- If delay timer overflow flag is set to 1 or there is an error flag = 1, the error flags (“sgaterroz” and “sgaterroutz”) (Master/Checker side) are set to 0 (active level) when corresponding bit in ECMEMK0/1 is 0 (error output not masked).
- Otherwise, the error flags are set to 1 (inactive level).
- Besides, the error flag values from Master and Checker sides will be concentrated by **AND** operator (ref[5]/Figure 32.2):

- The error compare signal ("ecmterroz") will be issued with the active level is result of **AND** operator between "sgaterroz" flags from both Master and Checker sides.
- When "sgaterroz" flag from Master side is different with another one from Checker side, the error "ECM compare error" (error No.29) will be occurred and a warning message is dumped.
- For the ERROROUT output signal ("ecmterroutz"), when the error flag ("sgaterroutz") = 1, it will be issued depend on "ecmttin" input if this signal is in dynamic mode. Otherwise, this signal will be issued with the active level is result of **AND** operator between "sgaterroutz" flag from both Master and Checker sides.
- The status of "ecmterroutz" will be stored in ECMmESSTR1.ECMmSSE131.

7.9. Write to write-protected register flow

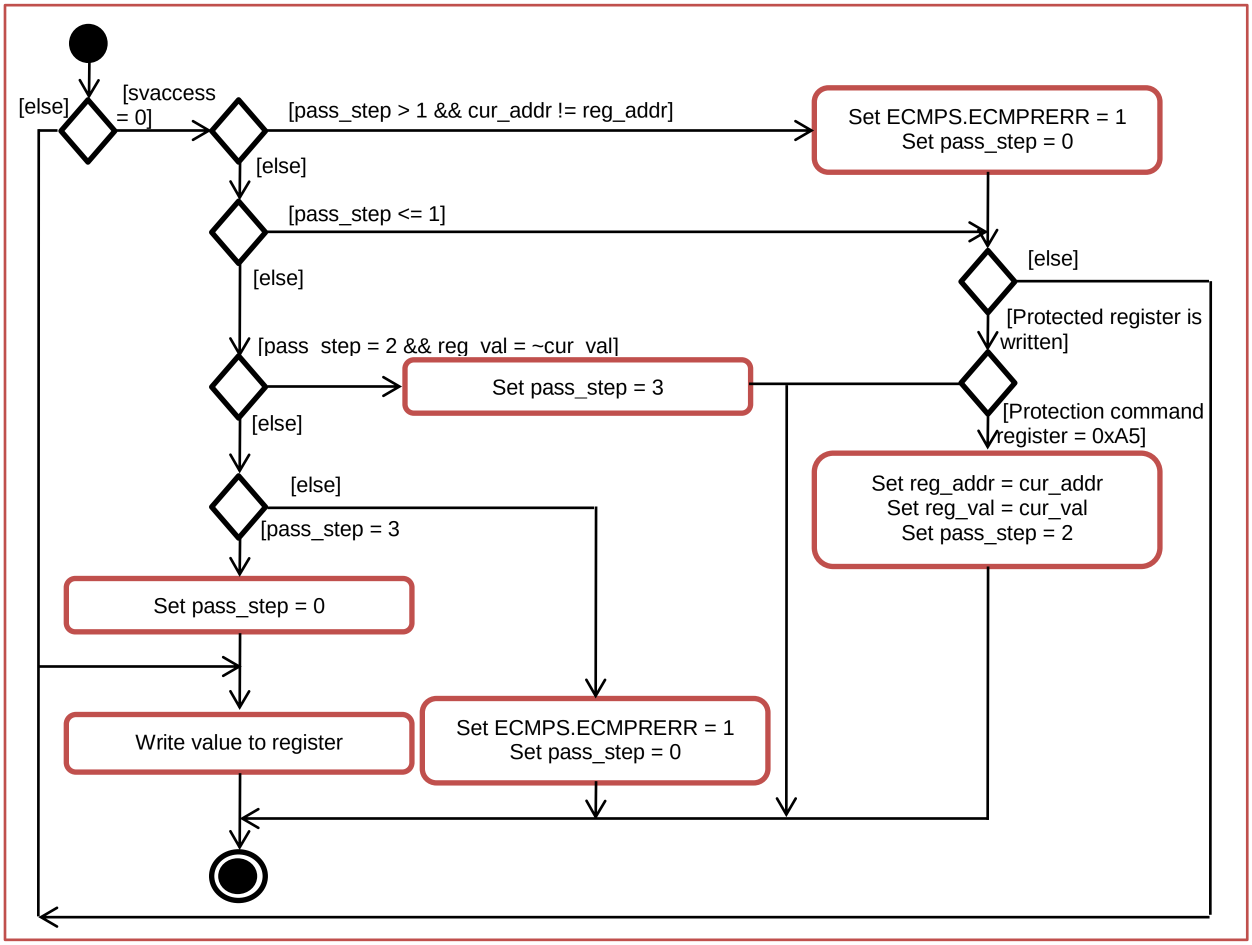


Figure 7.14: Write to write-protected register flow (without break)

Explanation:

- Almost registers in ECM model are protected registers. (refer to Table 4.1 for list of protected registers)
- The settings of protected registers are protected by requiring a special procedure (1*).

- When users write 0xA5 into protection command register (according 1st step in write procedure), pass_step is set to 1.
- According 2nd step in write procedure, if users write to a protected register when protection command register = 0xA5, pass_step is set to 2, reg_addr will store address of this register and reg_val is used to store written value. In the case protection command register != 0xA5, write procedure should be restart at beginning.
- If users write to a different register in same module listed in Table 4.1 (3*), when pass_step = 2 or 3, the write procedure fails, ECMPS.ECMPRERR is set to 1 and write procedure should be restart at beginning.
- At pass_step = 2, users should write inverse value of the desired value to current protected register. If not, the write procedure fails. After finish this step (its means finish 3th step in write procedure), pass_step is set to 3.
- At pass_step = 3, users should write again desired value to current protected register. If not, the write procedure fails. After finish this step (its means finish 4th step in write procedure), register is written completely (the write value is updated to register). Besides, ECMPS.ECMPRERR is set to 0.

➢ If break is occurred ("svaccess" is asserted) while the sequence from 1 to 4, the register protection is suspended until normal operation is resumed. (2*)

- At this time, users can write any dummy value into protected registers and write procedure is not failed (ECMPS.ECMPRERR keeps value 0).
- After "svaccess" is released, the write protection procedure is recover.

Notes:

- *(1*) Write procedure without break:*

+ Step 1: Write the fixed value 0xA5 to the protection command register.
+ Step 2: Write the desired value to the protected register.
+ Step 3: Write the inverse of the desired value to the protected register.
+ Step 4: Write the desired value to the protected register.
+ Step 5: Confirm write value.

Example: Write 0xAA into ECMMICFG1 register (with current value = 0x0)

+ Step 1: Write the fixed value 0xA5 to the ECMPCMD1 => ECMMICFG1 = 0x0
+ Step 2: Write the 0xAA to the ECMMICFG1 => ECMMICFG1 = 0x0
+ Step 3: Write the 0xFFFFFF55 to the ECMMICFG1 => ECMMICFG1 = 0x0
+ Step 4: Write the 0xAA to the ECMMICFG1 => ECMMICFG1 = 0xA2
+ Step 5: Confirm write value ECMMICFG1 = 0xA2

- *(2*) Write procedure with break:*

+ Step 1: Write the fixed value 0xA5 to the protection command register.
+ Step 1break: Write DUMMY value to the protected register.
+ Step 2: Write the desired value to the protected register.
+ Step 2break: Write DUMMY value to the protected register.
+ Step 3: Write the inverse of the desired value to the protected register.
+ Step 3break: Write DUMMY value to the protected register.
+ Step 4: Write the desired value to the protected register.
+ Step 4break: Write DUMMY value to the protected register.
+ Step 5: Confirm write value.

Example: Write 0xAA into ECMMICFG1 register (with current value = 0x0)

+ Step 1: Write the fixed value 0xA5 to the ECMPCMD1 => ECMMICFG1 = 0x0

(“svaccess” is asserted)

+ Step 1break:

Write 0x5555 to the ECMMICFG1 => ECMMICFG1 = 0x555

Write 0x3 to the ECMMICFG1 => ECMMICFG1 = 0x3

(“svaccess” is released)

+ Step 2: Write the 0xAA to the ECMMICFG1 => ECMMICFG1 = 0x3

+ Step 3: Write the 0xFFFFFF55 to the ECMMICFG1 => ECMMICFG1 = 0x3

(“svaccess” is asserted)

+ Step 3break:

Write 0x5555 to the ECMMICFG1 => ECMMICFG1 = 0x555

Write 0x3 to the ECMMICFG1 => ECMMICFG1 = 0x3

(“svaccess” is released)

+ Step 4: Write the 0xAA to the ECMMICFG1 => ECMMICFG1 = 0xA2

+ Step 5: Confirm write value ECMMICFG1 = 0xA2

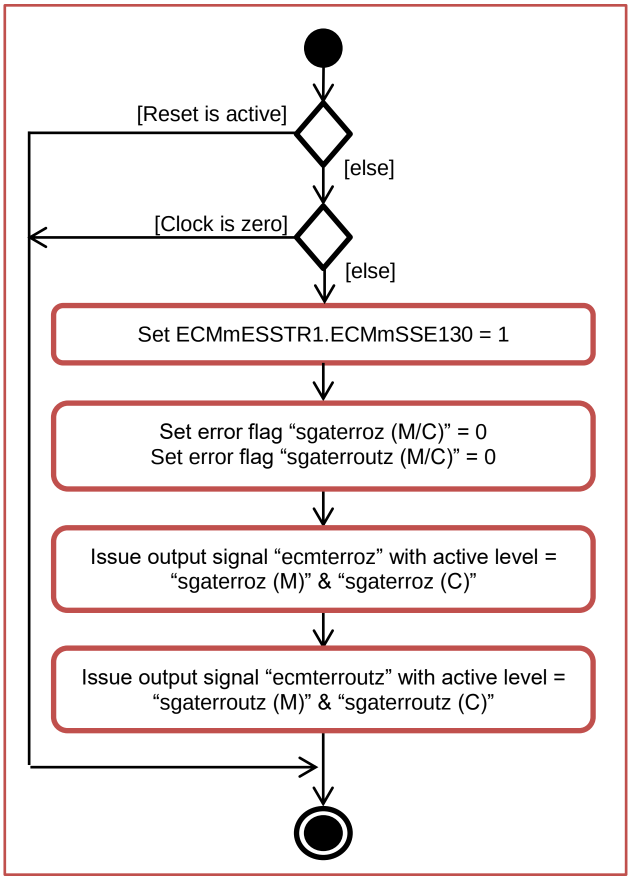
- *(3*) A different register in the same module is a register in the same module category as that of protection command registers, sequence status registers, and write protection target registers:*

+ When a protected register in <ECM_base> area is accessed, different registers in same module are other registers in <ECM_base>, <ECMM_base> and <ECMC_base> areas.

+ When a protected register in <ECMM_base> area is accessed, different registers in same module are other registers in <ECM_base> and <ECMM_base> areas.

+ When a protected register in <ECMC_base> area is accessed, different registers in same module are other registers in <ECM_base> and <ECMC_base> areas.

7.10. Trigger set error flow



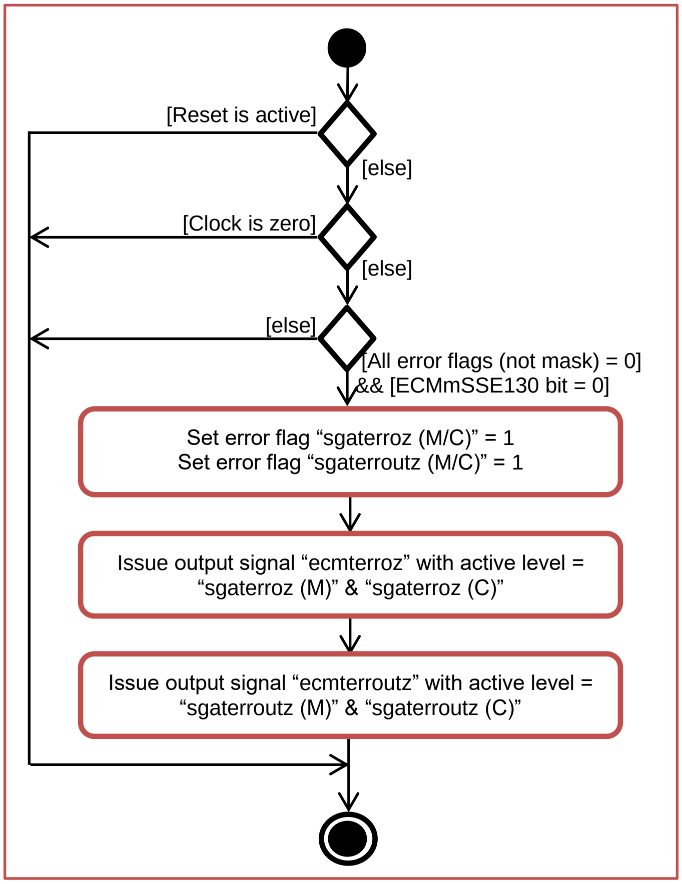
[Reset is active]
[else]
[Clock is zero]
[else]
Set ECMmESSTR1.ECMmSSE130 = 1
Set error flag “sgaterroz (M/C)” = 0
Set error flag “sgaterroutz (M/C)” = 0
Issue output signal “ecmterroz” with active level = “sgaterroz (M)” & “sgaterroz (C)”
Issue output signal “ecmterroutz” with active level = “sgaterroutz (M)” & “sgaterroutz (C)”

Figure 7.15: Trigger set error flow

Explanation:

- When users write 1 to ECMmESET register to trigger to set an error output
  - ECMmESSTR1. ECMmSSE130 is set to 1.
  - When set error request is triggered, the error flags “sgaterroz” and “sgaterroutz” are set to 0.
  - After that, output signals “ecmterroz” and “ecmterroutz” can be output accordingly.
  - Set an error output via the ECMmESET register, the error flag for “ECM compare error” will be set (ECMmESSTR0.ECMmSSE029 = 1) due to “sgaterroz” flags from Master and Checker sides are different at this time.

7.11. Trigger clear error flow



[Reset is active]
[else]
[Clock is zero]
[else]
[else]
[All error flags (not mask) = 0]
&& [ECMmSSE130 bit = 0]
Set error flag “sgaterroz (M/C)” = 1
Set error flag “sgaterroutz (M/C)” = 1
Issue output signal “ecmterroz” with active level = “sgaterroz (M)” & “sgaterroz (C)”
Issue output signal “ecmterroutz” with active level = “sgaterroutz (M)” & “sgaterroutz (C)”

Figure 7.16: Trigger clear error flow

Explanation:

- When users write 1 to ECMmECLR register to trigger to clear an error output
  - The clearing is only possible if all error flags (for not masked error) set in ECMmESSTR0/1 registers and ECMmESSTR1.ECMmSSE130 bit are cleared beforehand.
  - When clear error request is triggered, the error flags “sgaterroz” and “sgaterroutz” are set to 1.
  - After that, output signals “ecmterroz” and “ecmterroutz” can be output accordingly.
  - Clear an error output via the ECMmECLR register, the error flag for “ECM compare error” will be set (ECMmESSTR0.ECMmSSE029 = 1) due to “sgaterroz” flags from Master and Checker sides are different at this time.

8. Functions description

The detail specification of implemented classes is described in ref[8].

9. Limitation

- None

Revision History

Version	Modified points	Agreement	Approver	Checker	Author
1.0	- Created new	-	Yen Nguyen 12/23/2016	Yen Nguyen 12/23/2016	Ngan Tran 12/12/2016
1.1	- Added Figure 3.2 and Figure 3.3 for internal control blocks - Updated Table 4.1 to change read access size (from “-” into “8|16|32”) and change R/W permission (from “W” into “R:0|W1”) of ECMmESET, ECMmECLR - Updated Chapter 5.3 to add “When reset port is activated (“preset_n” or “extresetz”), all output ports are initialized” - Updated Chapter 5.7 to modify “reset generation condition” - Updated Chapter 5.8 and Figure 5.4 to correct “interrupt generation condition” - Updated Chapter 5.9, 5.10 to modify “error output generation condition” - Modified Figure 6.1 to move “re_define.h” from ecm_p1m.h into ecm_p1m.cpp; updated the note to change Register IF Generator (v2015_04_06) into (v2014_12_01) according latest version - Updated Table 6.2, Table 6.3, Figure 6.2, Figure 6.3, Table 6.4, Chapter 7.4 to add new supported commands/parameters; added the notes (3*) for “reg force/write” command and (4*) for “port” command - Updated Table 6.9 to add new messages. - Modified Chapter 7.1/Explanation to add 1 more condition to issue internal reset and error output signals - Modified Chapter 7.5/Explanation - Updated Chapter 7.8 to add feature for “ECM compare error” and modify “error output generation condition” - Updated Chapter 7.9 to add Note (3*) - Modified Chapter 7.10, 7.11 to correct action set error flag for “ECM compare error”	-	Yen Nguyen 01/25/2017	Yen Nguyen 01/25/2017	Ngan Tran 12/26/2016

Version	Modified points	Agreement	Approver	Checker	Author
1.2	- Added the note in Chapter 5.3 “During reset period, accessing to registers may not be allowed” - Updated Figure 5.3/(3*) to changed delay timer counter value when overflow occurs from “compare match value” into “0” - Added ““ecmterroutz” = ~(“ecmttin”)” in Chapter 5.10 - Modified Chapters 5.7, 5.9, 5.10, 7.7, 7.8, 7.10, 7.11 (figures and descriptions) to change active level of internal reset flag and error flags (“sgatresz”, “sgaterroz” and “sgaterroutz”) from 1 into 0 - Updated description in Chapter 7.5 to add “without dependence on setting of ECMEMK0/1 register” - Updated Table 6.9 to modify messages No.50, 51, and to add message No.52	A.Imoto 02/14/2017	Yen Nguyen 02/07/2017	Yen Nguyen 02/07/2017	Ngan Tran 02/06/2017