RX FAMILY
HARDWARE MANUAL GUIDE
(PERIPHERAL FUNCTIONS)

2026/02 REV 2.0
RENESAS ELECTRONICS CORPORATION

R01TU0444EJ0200 rev2.0
Feb. 2026

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev. 5.0-1 October 2020)

PURPOSE OF THIS DOCUMENT

- This manual is an easy-to-understand summary of each functional chapter of the hardware manual.
- The purpose of this document is to provide much more deeper understanding of the functions by using it with the hardware manual.
- For more detailed usage instructions, please refer to the application notes provided on each page.
- This document does not cover all the information described in the manual. For detailed information such as precautions for each function, please check the hardware manual for the product you want to use.

LIST OF PERIPHERAL FUNCTIONS

This document explains the following peripheral functions. The referenced function name is in parentheses.
The peripheral functions described in this material will continue to be added.

Click here for the Hardware Manual Guide Electrical Characteristics

LOW POWER CONSUMPTION

“LOW POWER CONSUMPTION” FUNCTION BY PRODUCT GROUP

- The “Low Power Consumption” function for each product is shown in the table below.

＊The RX26T is included in product group A

Product		Sleep	Deep Sleep	All-Module Clock Stop	Software Standby	Deep Software Standby	Snooze
Product Group A	RX600, 700 series	✓	-	✓	✓	✓	-
Product Group B	RX100, 200 series	✓	✓	-	✓	-	✓

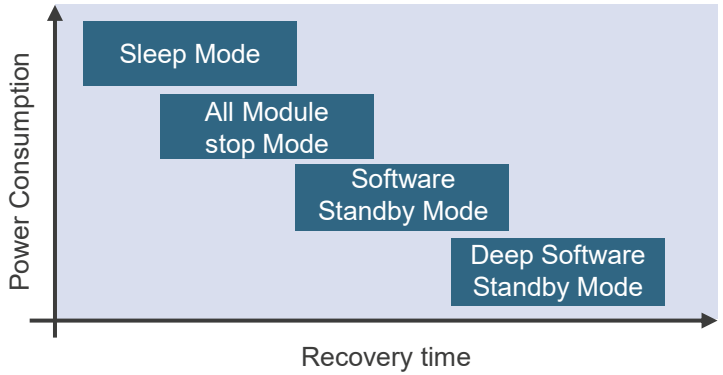
* Some products may not support certain modes, so please refer to the hardware section of the user's manual for each product for details.

OVERVIEW
“LOW POWER CONSUMPTION” FUNCTION OF PRODUCT GROUP A

“Stopped (Retained)” means that internal register values are retained and internal operations are suspended.
“Stopped (Undefined)” means that internal register values are undefined and power is not supplied to the internal circuit.

Low power consumption mode	CPU	Oscillator / Power management				Peripheral modules						Memory		I/O Port
		Main clock	HOCO LOCO PLL	Sub-clock RTC LVD	POR	IWDT	WDT	USB	TMR POE	REMC	Other	SRAM	Standby RAM	
Sleep	Stop (retained)	Selectable	Selectable	Selectable	Operating	Selectable	Stop (retained)	Selectable	Selectable	Selectable	Selectable	Selectable	Selectable	Operating
All-Module Clock Stop	Stop (retained)	Selectable	Selectable	Selectable	Operating	Selectable	Stop (retained)	Stop*2	Selectable	Stop (retained)	Stop (retained)	Stop (retained)	Stop (retained)	Stop (retained)
Software Standby	Stop (retained)	Selectable or Stop*1	Stop	Selectable	Operating	Selectable	Stop (retained)	Stop*2	Stop (retained)	Selectable	Stop (retained)	Stop (retained)	Stop (retained)	Stop (retained)
Deep Software Standby	Stop (不定)	Selectable or Stop*1	Stop	Selectable	Operating	Stop (Undefined)	Power off (Undefined)	Stop*3	Stop (Undefined)	Selectable	Stop (Undefined)	Stop (Undefined)	Stop (retained/Undefined)	Stop (retained)

*1 : Varies depending on the product
*2 : Resume function is enabled
*3 : Resume function can be selected to enable or to disable

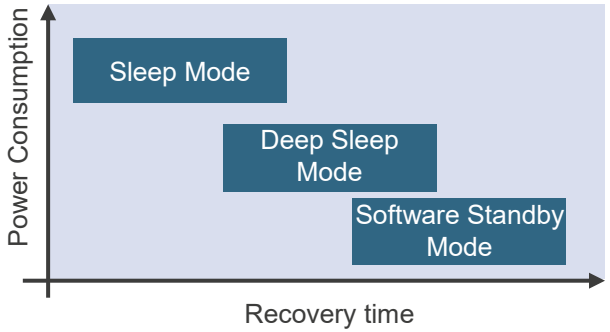


OVERVIEW
“LOW POWER CONSUMPTION” FUNCTION OF PRODUCT GROUP B

“Stopped (Retained)” means that internal register values are retained and internal operations are suspended.

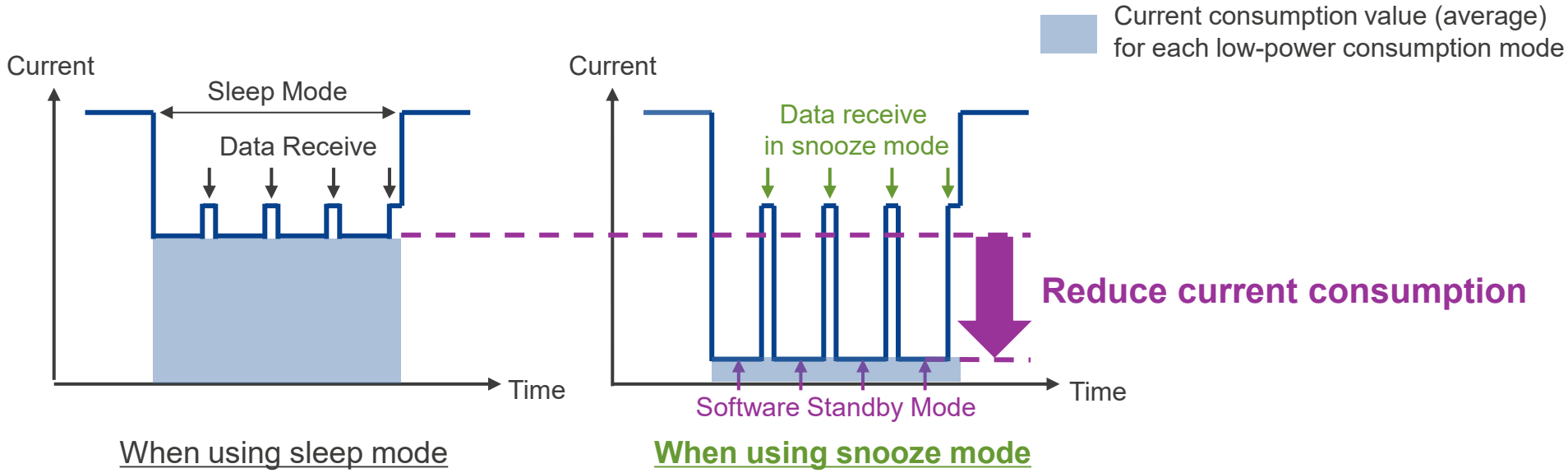
Low power consumption mode	CPU	Oscillator / Power management				Peripheral modules						SRAM	I/O Port
		Main clock	HOCO LOCO PLL	Sub-clock RTC LVD	POR	IWDT	WDT	LPT Comparator B REMC RTCOUT CLKOUT	DMAC	DTC	Other		
Sleep	Stop (retained)	Selectable	Selectable	Selectable	Operating	Selectable	Stop (retained)	Selectable	Selectable	Selectable	Selectable	Selectable	Operating
Deep Sleep	Stop (retained)	Selectable	Selectable	Selectable	Operating	Selectable	Stop (retained)	Selectable	Stop (retained)	Stop (retained)	Selectable	Stop (retained)	Operating
Software Standby	Stop (retained)	Stop	Stop	Selectable	Operating	Selectable	Stop (retained)	Selectable	Stop (retained)	Stop (retained)	Stop (retained)	Stop (retained)	Stop (retained)
Snooze*1	Stop (retained)	Stop (retained)	Selectable	Selectable	Operating	Selectable	Stop (retained)	Selectable	Stop (retained)	Selectable	Selectable	Selectable (retained)	Operating

*1:The “Snooze” function is explained on the next page.



SNOOZE MODE

- During software standby mode, some peripheral functions can be operated temporarily (intermittent operation).
- By performing intermittent operation in software standby and snooze mode, the total current consumption can be reduced from mA to uA* level compared to sleep mode operation.
  * : Based on actual measurements result in APN “RX Family Snooze Mode Usage Examples R01AN5914”.



TIPS FOR REDUCING POWER CONSUMPTION

Introduce tips for reducing power consumption using the RX140 as an example.

① Utilize High-speed / Middle-speed / Low-speed operation mode
Power consumption can be further reduced by lowering the operating frequency of the system when entering sleep or deep sleep mode.

② Stop unused peripheral modules
There is a large difference in current consumption when “all modules are operated” or “all modules are stopped”. Therefore, power consumption can be reduced by stopping unused peripheral modules.

③ I/O Port
The current consumption of the I/O port for which the output is selected can be reduced by the following process.
- If it is connected to a pull-up resistor, set the output level to High, and if it is connected to a pull-down resistor, set the output level to Low.

Example) Comparison of current consumption
in sleep mode and operation of all peripheral modules (typical)

	High-speed (48MHz)	Middle-speed (8MHz)	Low-speed (32.768KHz)
Current Consumption	4.0 mA	1.5 mA	3.8 **uA**

Example) Comparison of current consumption
of peripheral module operation/stop in 48MHz drive sleep mode (typical)

	All peripheral operation	Stop all peripherals
Current Consumption	4.0 mA	1.4 mA

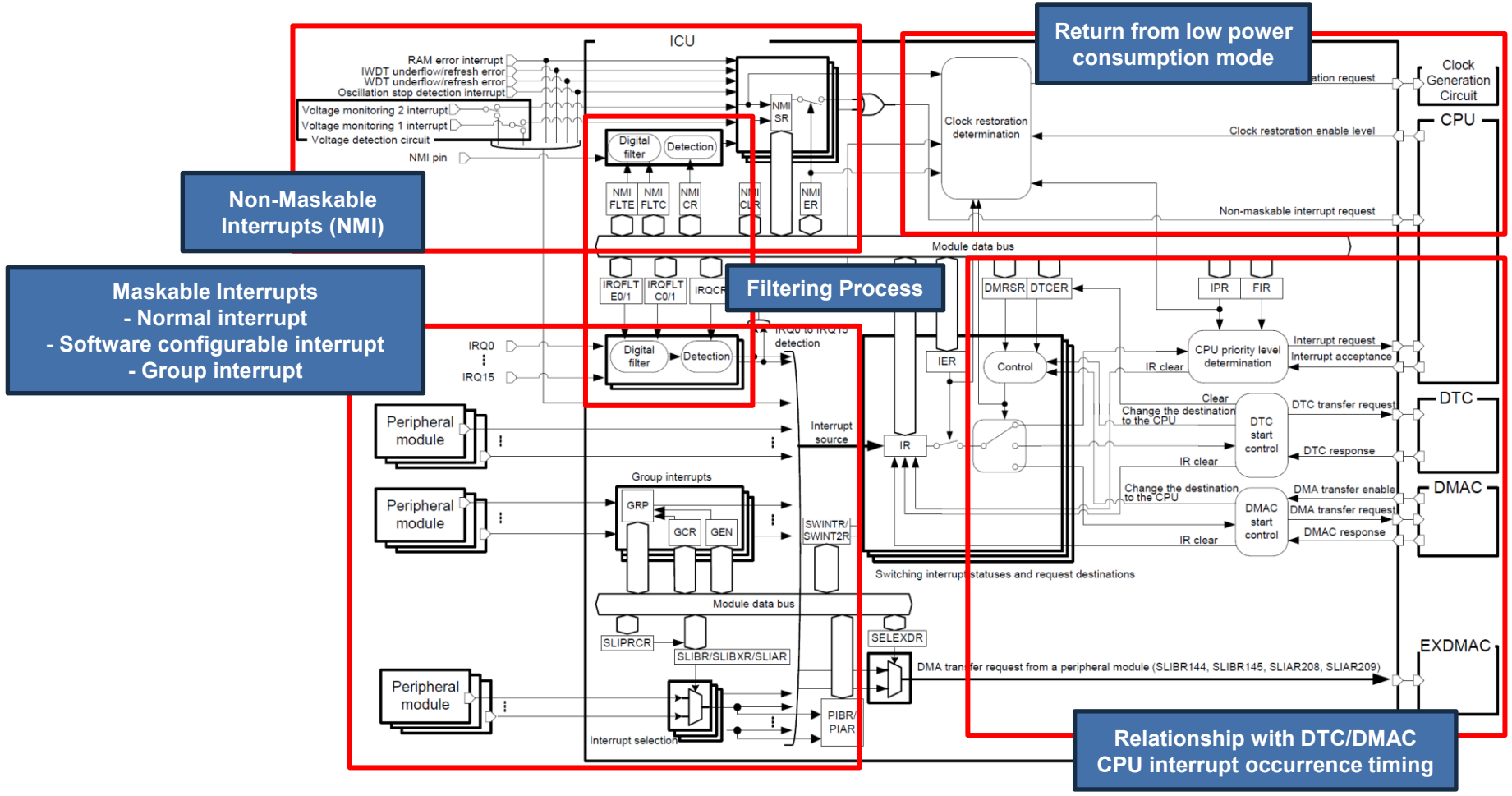
APPLICATION NOTE FOR “LOW POWER CONSUMPTION” FUNCTION

- Please refer to the following APN for details on how to use the “Low Power Consumption” function.
  - RX Family Examples of Transitioning to Low Power Consumption Modes R01AN4347
  - RX Family Snooze Mode Usage Examples R01AN5914

INTERRUPT CONTROLLER

SPECIFICATION SUMMARY OF “INTERRUPT CONTROLLER”

- The configuration of the interrupt controller & its points are listed below and explained in order.



NON-MASKABLE INTERRUPT

- This is an interrupt that occurs when it detects a CPU anomaly and believes that a fatal failure has occurred in the system. In the RX family, all of the following interrupts are defined as non-maskable interrupts.

■ **Types of non-maskable interrupts**

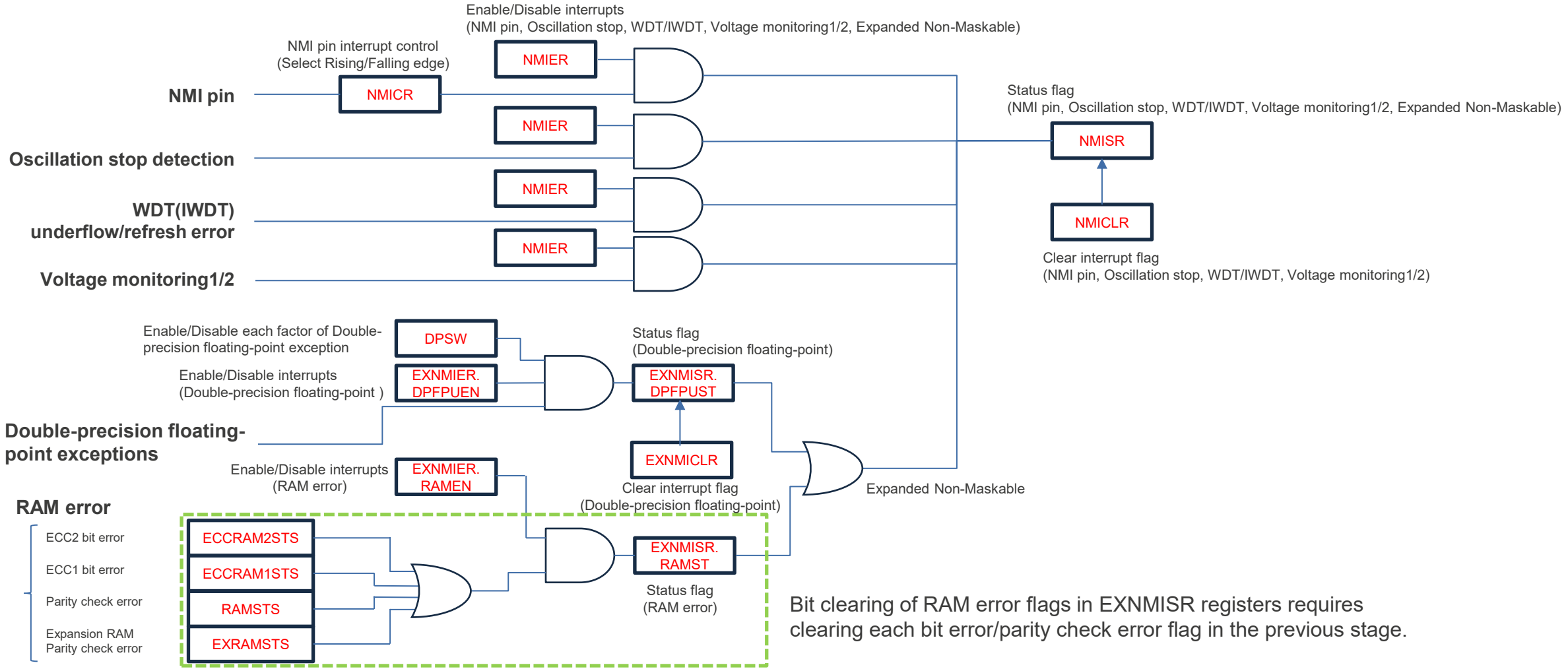
NMI pin	**Oscillation stop detection**	**WDT(IWDT) underflow/refresh error**	**Voltage monitoring 1/2**	**Double-precision floating-point exceptions**	**RAM error**

- All interrupt priority levels for non-maskable interrupts are 15 (maximum).
- Non-maskable interrupts are not affected by PSW I or IPL bits and are always accepted.
- The vector table for non-maskable interrupts is on the exception vector table.
- You cannot start DTC or DMAC with non-maskable interrupts.
- When a non-maskable interrupt occurs, there may be a system abnormality, so please consider a system that does not return to the original processing from the interrupt.
- Except for the NMI pin, it can also be used as a maskable interrupt.
  (The vector table in this case is on the interrupt vector table)

NON-MASKABLE INTERRUPT

NON-MASKABLE INTERRUPT STRUCTURE

The structure of a non-maskable interrupt and the register that need to be set are shown below. For more detail, please refer to the manual.



Bit clearing of RAM error flags in EXNMISR registers requires clearing each bit error/parity check error flag in the previous stage.

MASKABLE INTERRUPT

- This is an interrupt caused by an interrupt factor in the IRQ pin or peripheral function (Request source).
- The interrupt factor is assigned to the interrupt vector table.
- Maskable interrupt causes CPU interrupt or DTC/DMAC/EXDMAC transfer.
- When generating a maskable interrupt, it is necessary to set not only the registers of the interrupt function, but also the registers of the peripheral functions (Request source) and the CPU.
- In addition to unique interrupts, there are also Group interrupts and Software Configurable interrupts.

MASKABLE INTERRUPT
VECTOR TABLE OF MASKABLE INTERRUPT

The vector table of maskable interrupts is shown below.

Interrupt Request Source	Name	Vector No.(IR)	Interrupt detection method	Start the DMAC	Start the DTC	IER	IPR
. .	. .	. .	. .	. .	. .	. .	. .
CMT0	CMI0	28	Edge	✓	✓	IER03.IEN4	IPR004
. .	. .	. .	. .	. .	. .	. .	. .
RIIC1	RXI1	50	Edge	✓	✓	IER06.IEN2	IPR051
	TXI1	51	Edge	✓	✓	IER06.IEN3	IPR052
. .	. .	. .	. .	. .	. .	. .	. .
SCI0	RXI0	58	Edge	✓	✓	IER07.IEN2	IPR058
	TXI0	59	Edge	✓	✓	IER07.IEN3	IPR059
. .	. .	. .	. .	. .	. .	. .	. .
ICU (Group interrupts)	GROUPBL0	110	Level	✓	✓	IER0D.IEN6	IPR110
	GROUPBL1	111	Level	✓	✓	IER0D.IEN7	IPR111
. .	. .	. .	. .	. .	. .	. .	. .
PERIB (Software Configurable interrupts)	INTB128	128	Edge			IER10.IEN0	IPR128
	INTB129	129	Edge			IER10.IEN1	IPR129
. .	. .	. .	. .	. .	. .	. .	. .

Simple interrupts, Group interrupts and Software Configurable interrupts are distributed into vector table(up to 256).

Simple interrupts : Place interrupts that require immediate response to interrupt requests (e.g., unconditional trap, System(BSC, FCU, IRQ, CMT0, LVD, WDT, DTC/DMAC), Communication(transmit data empty, transmit end) and etc.)
Group interrupts : Interrupts are placed that do not require urgent response to interrupt requests (e.g., Communication(transmission end, receive error), A/D(compare interrupt))
Software Configurable interrupts : Interrupts for functions with multiple units are placed (e.g., Timer, A/D(A/D scan end interrupt))

Each “interrupt detection method” is divided into “Edge” and “Level”. In the case of a level interrupt, please note that if the interrupt factor is not cleared, the interrupt flag (IR bit) will not be cleared to zero. In particular, Group interrupts (BLx, ALx) composed of level interrupt factors will not have the IR flag of Group interrupts to 0 unless all factors are zero.

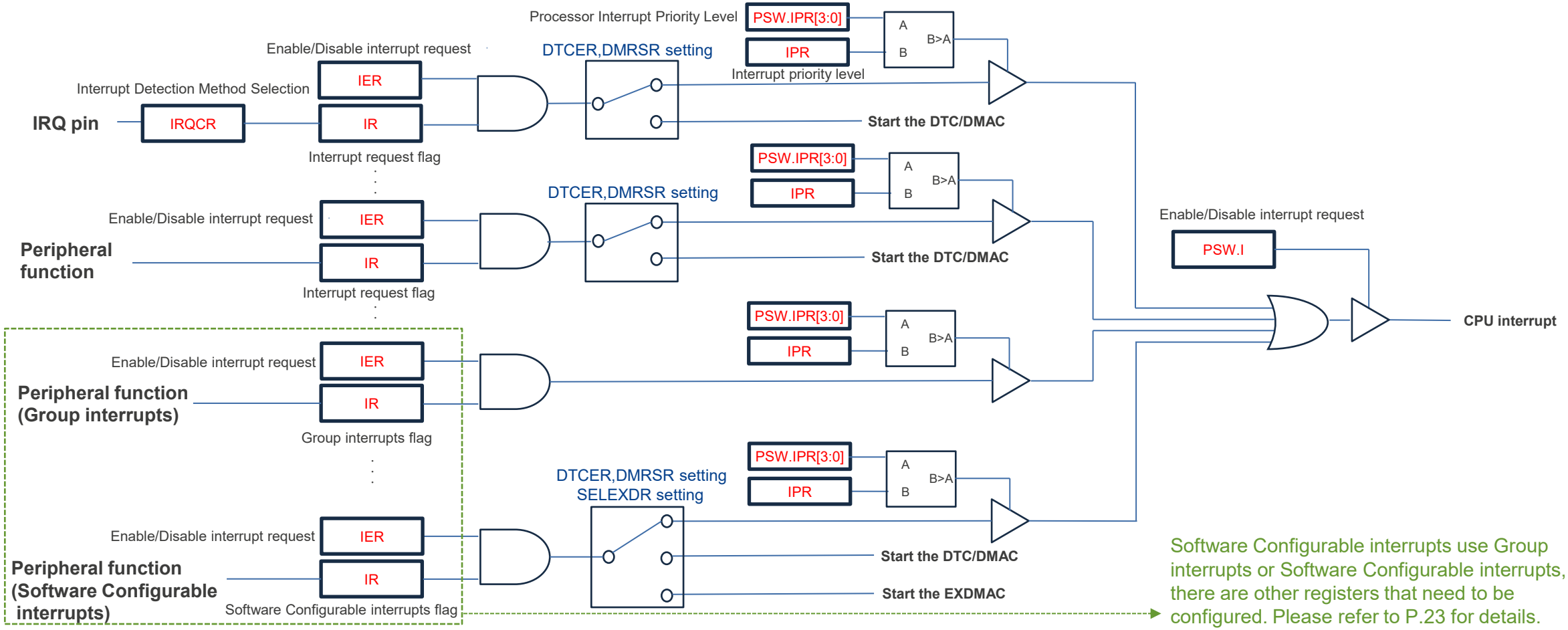
In the case of interrupts that can start DMAC/DTC, it is marked with “✓” in this item. On the other hand, interrupts that can start DMAC/DTC are marked as “×”. DMAC/DTC can only be started with edge detection, and level detection cannot be used.

The CPU interrupt priority order of each interrupt factor can be freely set between 0~15 regardless of the order of the vector table by using IPR registers. If the priority level set in IPR is the same, the vector number will be in order of smallest.

MASKABLE INTERRUPT

MASKABLE INTERRUPT STRUCTURE

- Summarizes various interrupt factors (NMI, IRQ, peripheral functions), CPU interrupts caused by each factor, and the flow of DTC/DMAC/EXDMAC transfers and the required registers.



Software Configurable interrupts use Group interrupts or Software Configurable interrupts, there are other registers that need to be configured. Please refer to P.23 for details.

MASKABLE INTERRUPT
GROUP INTERRUPTS

- In the case of a normal interrupt, a unique vector table number/IR/IER/IPR are provided for each interrupt factor.
- In Group interrupts, interrupt requests (up to 32) from multiple peripheral modules are treated as a group/interrupt request. Vector table numbers, IR, IER and IPR are also available for group interrupts. (Interrupt flag and interrupt request Enable/disable settings for individual factors assigned to Group interrupts are controlled by dedicated registers)
- Group interrupts have group for edge interrupt(IE0, BE0) and group for level interrupt(BL0~2, AL0~1).

Interrupt Request Source	Name	Vector No.(IR)	IER	IPR
CMT0	CMI0	28	IER03.IEN4	IPR004
RIIC1	RXI1	50	IER06.IEN2	IPR051
	TXI1	51	IER06.IEN3	IPR052
SCI0	RXI0	58	IER07.IEN2	IPR058
	TXI0	59	IER0	
ICU	GROUPBL0	110	IER0D.IEN6	IPR110
	GROUPBL1	111	IER0D.IEN7	IPR111
	GROUPAL0	112	IER0E.IEN0	IPR112
	GROUPAL1	113	IER0E.IEN1	IPR113

Group interrupts

Vector table example

Interrupt flags and interrupt request Enable/Disable for individual factors assigned to Group interrupts are controlled in registers not vector tables.

Group	Interrupt Request Source	Name	Interrupt flag	Enable/Disable interrupt request setting
GROUPBL0	SCI0	TEI0	GRPBL0.IS0	GENBL0.EN0
		REI0	GRPBL0.IS1	GENBL0.EN1
	SCI1	TEI1	GRPBL0.IS2	GENBL0.EN2
		RXI1	GRPBL0.IS3	GENBL0.EN3
	QSPI	QSPSSLI	GRPBL0.IS24	GENBL0.EN24
	CAC	FERRI	GRPBL0.IS26	GENBL0.EN26
		MENDI	GRPBL0.IS27	GENBL0.EN27
		OVFI	GRPBL0.IS28	GENBL0.EN28
	DOC	DOPCI	GRPBL0.IS29	GENBL0.EN29

Group interrupts example

MASKABLE INTERRUPT
GROUP INTERRUPTS STRUCTURE

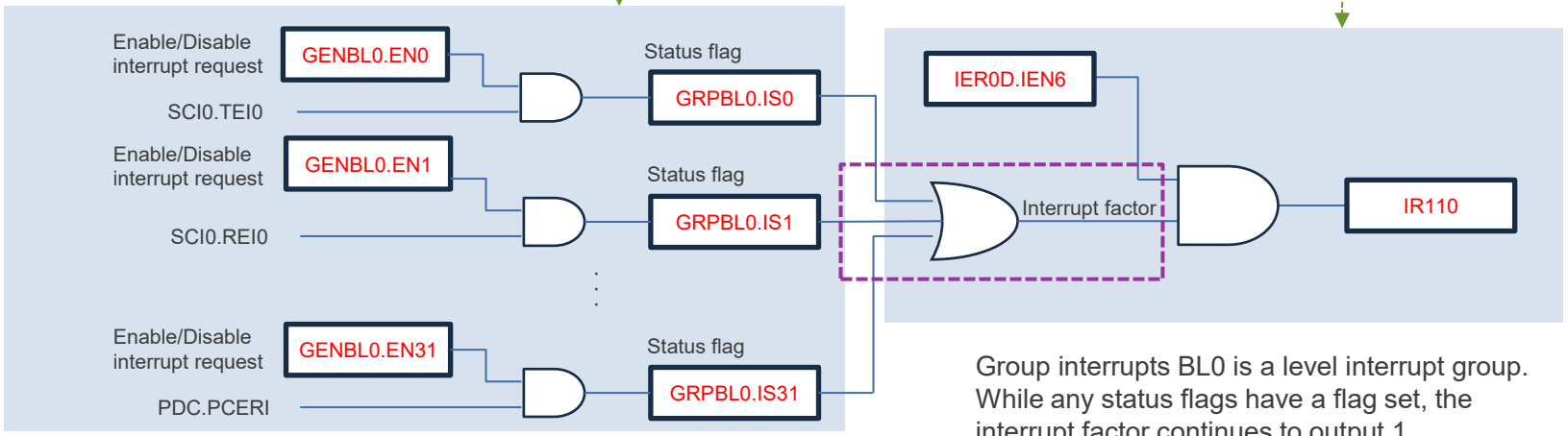
- The Group interrupts structure is shown below.

Group	Interrupt Request Source	Name	Interrupt flag	Enable/Disable interrupt request setting
GROUPBL0	SCI0	TEI0	GRPBL0.IS0	GENBL0.EN0
		REI0	GRPBL0.IS1	GENBL0.EN1
	SCI1	TEI1	GRPBL0.IS2	GENBL0.EN2
		RXI1	GRPBL0.IS3	GENBL0.EN3
	:	:	:	:
	DOC	DOPCI	GRPBL0.IS29	GENBL0.EN29
	PDC	PCFEI	GRPBL0.IS30	GENBL0.EN30
		PCERI	GRPBL0.IS31	GENBL0.EN31

Group interrupts(BL0)

Interrupt Request Source	Name	Vector No.(IR)	IER
ICU	GROUPBL0	110	IER0D.IEN6
	GROUPBL1	111	IER0D.IEN7
	GROUPAL0	112	IER0E.IEN0
	GROUPAL1	113	IER0E.IEN1

Vector Table



Group interrupts Structure

Group interrupts BL0 is a level interrupt group. While any status flags have a flag set, the interrupt factor continues to output 1.

MASKABLE INTERRUPT
NOTES FOR HANDLING ROUTINE IN GROUP INTERRUPTS

- In case of Group interrupts, need review all interrupt flags assigned to a group within the interrupt process and process all flagged interrupts.

Group	Interrupt Request Source	Name	Interrupt flag	Enable/Disable interrupt request setting
GROUPBL0	SCI0	TEI0	GRPBL0.IS0	GENBL0.EN0
		REI0	GRPBL0.IS1	GENBL0.EN1
	SCI1	TEI1	GRPBL0.IS2	GENBL0.EN2
		RXI1	GRPBL0.IS3	GENBL0.EN3
	QSPI	QSPSSLI	GRPBL0.IS24	GENBL0.EN24
	CAC	FERRI	GRPBL0.IS26	GENBL0.EN26
		MENDI	GRPBL0.IS27	GENBL0.EN27
		OVFI	GRPBL0.IS28	GENBL0.EN28
	DOC	DOPCI	GRPBL0.IS29	GENBL0.EN29

Group BL0 interrupt factor

■ Group interrupts handling program example

```
if(GRPBL0.IS0==1){
 TEI0(transmission end) handling

}else if(GRPBL0.IS1 ==1){
 ERI0(receive error) handling

}else if(GRPBL0.IS2 ==1){
 TEI1(transmission end) handling
                .
                .
                .
```

MASKABLE INTERRUPT
SOFTWARE CONFIGURABLE INTERRUPTS

- In Software Configurable interrupts, assign a given interrupt vector number by selecting one of the interrupt factors of multiple peripheral modules.
- IR/IEN/IPR associated with the assigned vector number are used.
- Interrupt factors that are not assigned to Software Configurable interrupts cannot be used as CPU interrupts. However, interrupt status flags are available for each factor and can be used for polling usage.

Interrupt Request Source	Name	Vector No.(IR)	IER	IPR
CMT0	CMI0	28	IER03.IEN4	IRP004
RIIC1	RXI1	50	IER06.IEN2	IPR051
	TXI1	51	IER06.IEN3	IPR052
SCI0	RXI0	58	IER07.IEN2	IPR058
	TXI0	59		
PERIB	INTB128	128	IER10.IEN0	IPR128
	INTB129	129	IER10.IEN1	IPR129
	INTB130	130	IER10.IEN2	IPR130
	INTB131	131	IER10.IEN3	IPR131

Software Configurable interrupts

Vector table example

Optional

Interrupt Request Source	Name	Interrupt status flag	
CMT2	CMI2	PIBR0.PIR0	
CMT3	CMI3	PIBR0.PIR1	Can be used in polling
TMR0	CMIA0	PIBR0.PIR3	
	CMIB0	PIBR0.PIR4	
	OVI0	PIBR0.PIR5	
USB	USBI0	PIBR7.PIR6	Can be used in polling
S12AD	S12ADI	PIBR8.PIR0	
	S12GBDAI	PIBR8.PIR1	Can be used in polling
	S12GCDAI	PIBR8.PIR2	

Example : Software Configurable interrupts B factor list

MASKABLE INTERRUPT

GROUP INTERRUPTS, SOFTWARE CONFIGURABLE INTERRUPTS REGISTER LIST

- Lists the Group interrupts and Software Configurable interrupts registers.

Group interrupts register list

Register Name	Details	Types of Group interrupts and registers						
		IE0	BE0	BL0	BL1	BL2	AL0	AL1
Interrupt Status Flag Register	View the interrupt request status for each grouped interrupt factor	GRPIE0*1	GRPBE0*1	GRPBL0	GRPBL1	GRPBL2	GRPAL0	GRPAL1
Interrupt Request Enable Register	Enable/Disable grouped interrupt factors	GENIE0	GENBE0	GENBL0	GENBL1	GENBL2	GENAL0	GENAL1
Group Interrupt Clear Register	Clear the interrupt status flag register	GCRIE0	GCRBE0	-	-	-	-	-

*1: Clearing the flag is done by the Group interrupts clear register.

Software Configurable interrupts register list

Register Name	Details	Software Configurable interrupts and registers	
		Software Configurable interrupts A	Software Configurable interrupts B
Software Configurable Interrupt Request Register	The register is used for polling of interrupt requests of interrupt sources that is assigned to software configurable interrupt by software.	PIBRn	PIARn
Software Configurable Interrupt Source Select Register	The register is used to assign interrupt vector numbers to interrupt sources assigned to software configurable interrupt.	SLIBXRn SLIBRn	SLIAXRn
Software Configurable Interrupt Source Select Register Write Protect Register	The register protects registers that control assignment of software configurable interrupts from being written to.	SLIPRCR	

MASKABLE INTERRUPT
DTC/DMAC, EXDMAC TRANSFER

- Maskable interrupt factors (edge factors) can start DTC/DMAC/EXDMAC.
- DTC and DMAC are not affected by IPR. EXDMAC is not affected by IPR and IR.
- DTC and DMAC can generate CPU interrupts due to interrupts that became a transfer factor depending on the number of transfers.
  The CPU interrupt factor at this time is affected by IPR.
- DTC/DMAC/EXDMAC can raise individual interrupt depending on the end of the transfer and so on.
  For more information, please refer to the DTC/DMAC/EXDMAC chapters of the User's Manual Hardware section.

MASKABLE INTERRUPT
IR BIT CLEAR TIMING

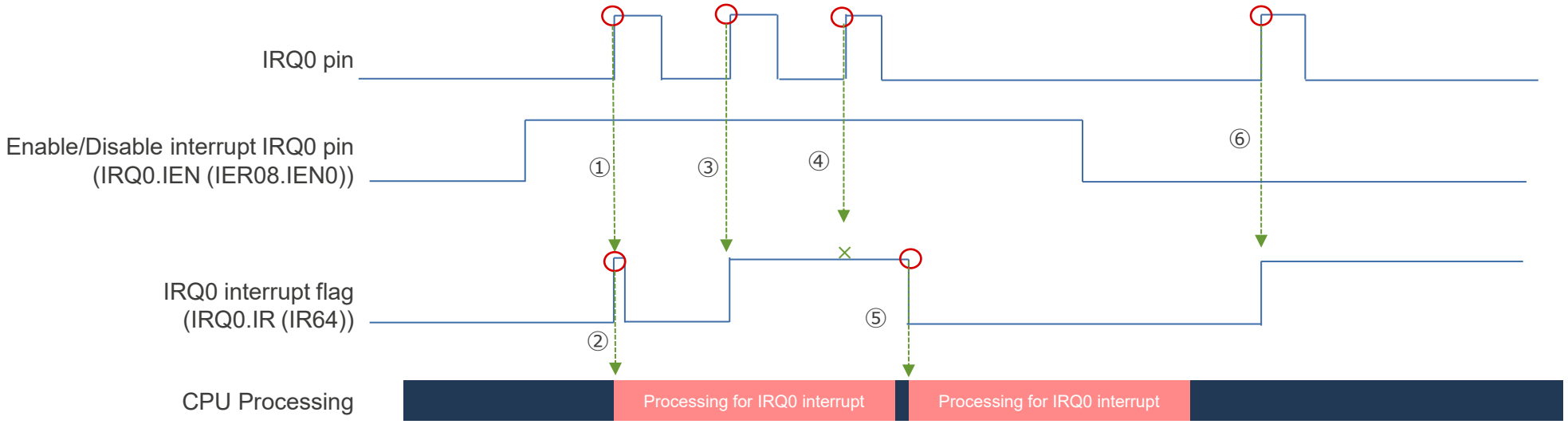
Interrupt		Clear Timing
Edge detection	CPU interrupt	・Cleared when the CPU accepts an interrupt request. ・When 0 is written to IR flag
	DMAC/DTC transfer	・Cleared when the DMAC/DTC starts data transfer. * Prohibit clearing IR flags by CPU
Level detection	CPU interrupt	・Except Group interrupts Clear interrupt request output of peripheral modules ・Group interrupts Enj bit of Interrupt Request Enable Register is “0” or all Isj flag of Group interrupt request register is “0”.

MASKABLE INTERRUPT
FAST INTERRUPT

- The fast interrupt is an interrupt that the CPU can respond to fast. Only one interrupt source can be assigned to the fast interrupt.
- The priority level of the fast interrupt is 15 (highest) regardless of the IPRn.IPR[3:0] bit setting (n = interrupt vector number). Also, the fast interrupt has higher priority than the other interrupt sources of which the priority level is 15. Note that the fast interrupt cannot be accepted when the PSW.IPL[3:0] bits are 1111b (priority level 15).
- To assign an interrupt source to the fast interrupt, set the FIR.FVCT[7:0] bits to select the vector number of the interrupt source, and set the FIR.FIEN bit to 1 (fast interrupt is enabled).

EXAMPLE : IRQ PIN / EDGE INTERRUPT

- This is an example of an edge interrupt operation using the IRQ pin.

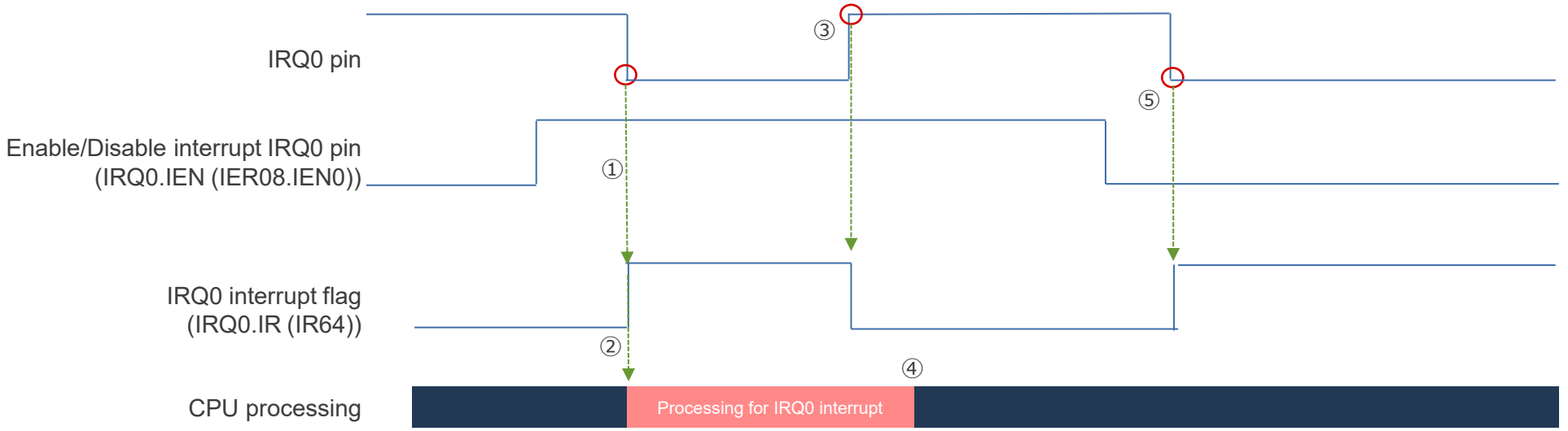


Example : IRQ Interrupt by Rising Edge Interrupt

① The rising edge of the IRQ pin is detected. IRQ0.IR is set.
② The CPU processing for interrupt by IRQ0 begins. IRQ0.IR bits are cleared automatically.
③ During interrupt processing, the rising edge detection of the IRQ pin occurs again. IRQ0.IR is set (pending).
④ The rising edge of the IRQ pin is detected again. However, this request is discarded because the IRQ0.IR bit is already “Pending”.
⑤ After the processing for IRQ0 interrupt is completed, the IRQ0 interrupt occurs again due to the pending IRQ0.IR bit.
⑥ The rising edge of the IRQ pin is detected. IRQ0.IR is set, but IRQ0. Since the IEN bit is 0, CPU interrupts do not occur.

EXAMPLE : IRQ PIN / LEVEL INTERRUPT

- This is an example of a level interrupt operation using the IRQ pin.



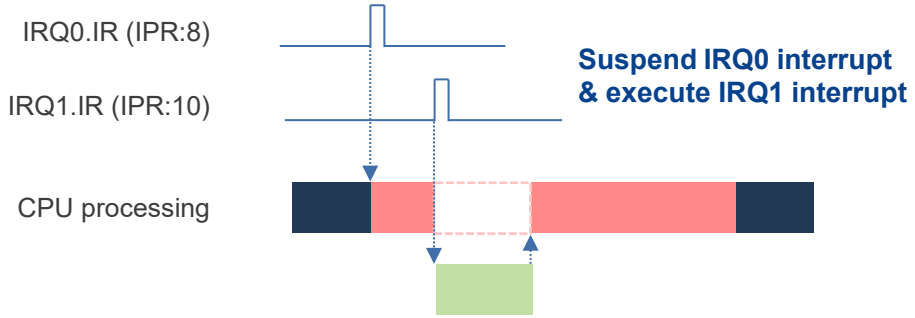
Example : IRQ Interrupt by low level Interrupt

① The low level of the IRQ pin is detected. IRQ0.IR is set.
② CPU processing for IRQ0 interrupt is started (Unlike edge interrupt, IRQ0.IR bit is not cleared).
③ IRQ pin changed to high level. Along with this, IRQ0.IR will also be cleared.
④ After confirming that the IRQ0.64 bit is cleared, the processing for IRQ0 interrupt is completed.
⑤ The low level of the IRQ0 pin is detected. IRQ0.IR bit is set, but IRQ0. Interrupt processing does not occur because the IEN bit is 0

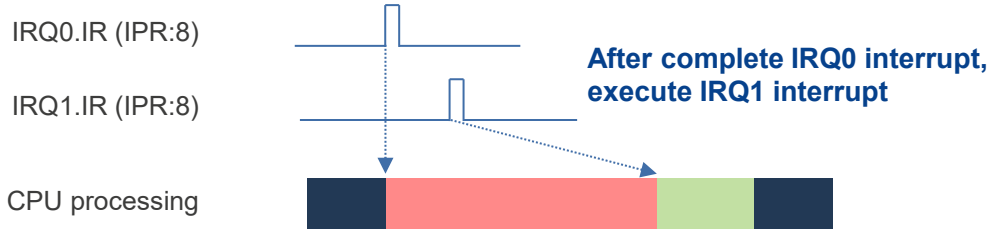
EXAMPLE :
IRQ PIN / MULTIPLE INTERRUPTS, SIMULTANEOUS INTERRUPTS OCCUR

- If you set the PSW.I bit to 1 at the beginning of the interrupt process, you can allow “Multiple interrupts”. When you allow multiple interrupts, the system accepts interrupts when a priority level interrupt request occurs that is higher than the interrupt priority level that is being processed. The following is an example of IRQ pin interrupt behavior when multiple interrupts are allowed.
  The higher the interrupt that you set in IPR, the higher the priority level. If the IPR values are the same, they are processed in order of the smallest vector number.

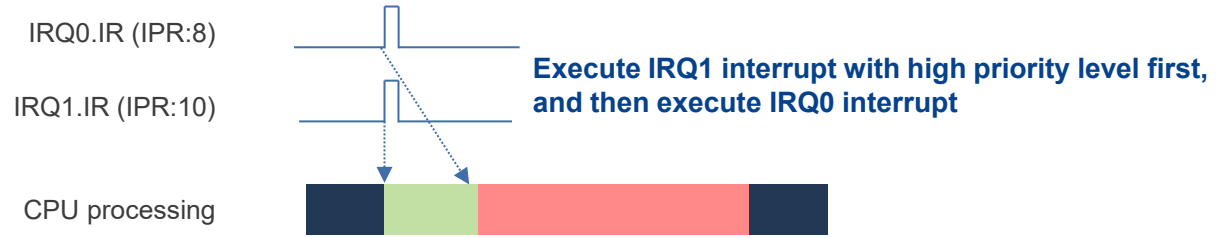
(1) If a higher priority level interrupt occurs during processing for interrupt.



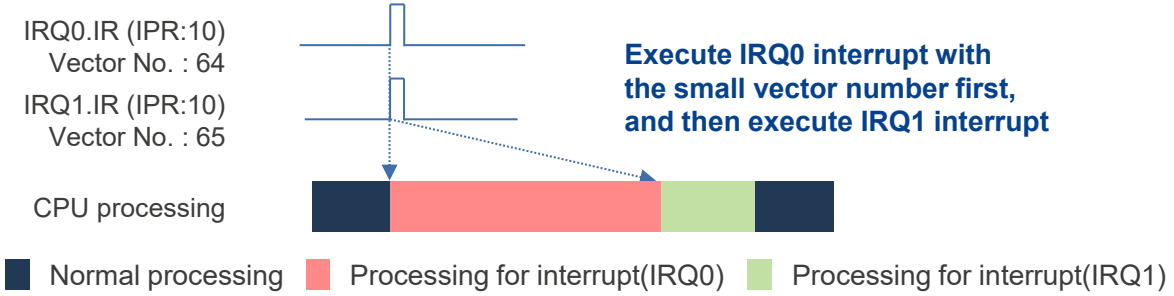
(2) If an interrupt of the same priority level occurs during processing for interrupt.



③ If interrupts with different priority levels occur at the same time

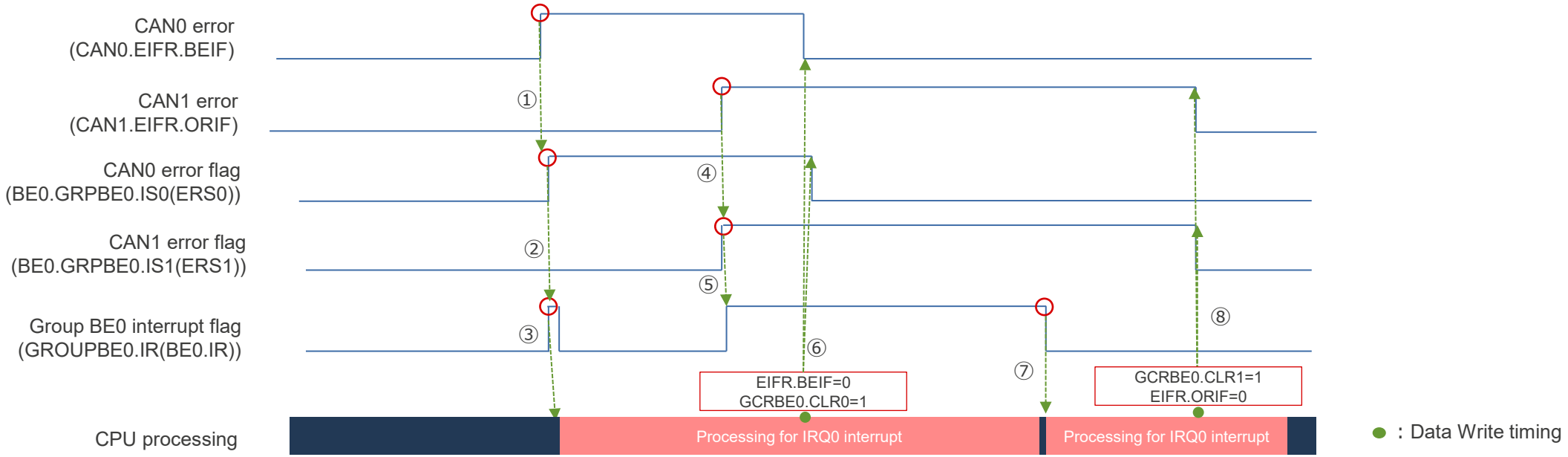


(4) If interrupts of the same priority level occur at the same time



EXAMPLE : GROUP INTERRUPT(EDGE INTERRUPT)

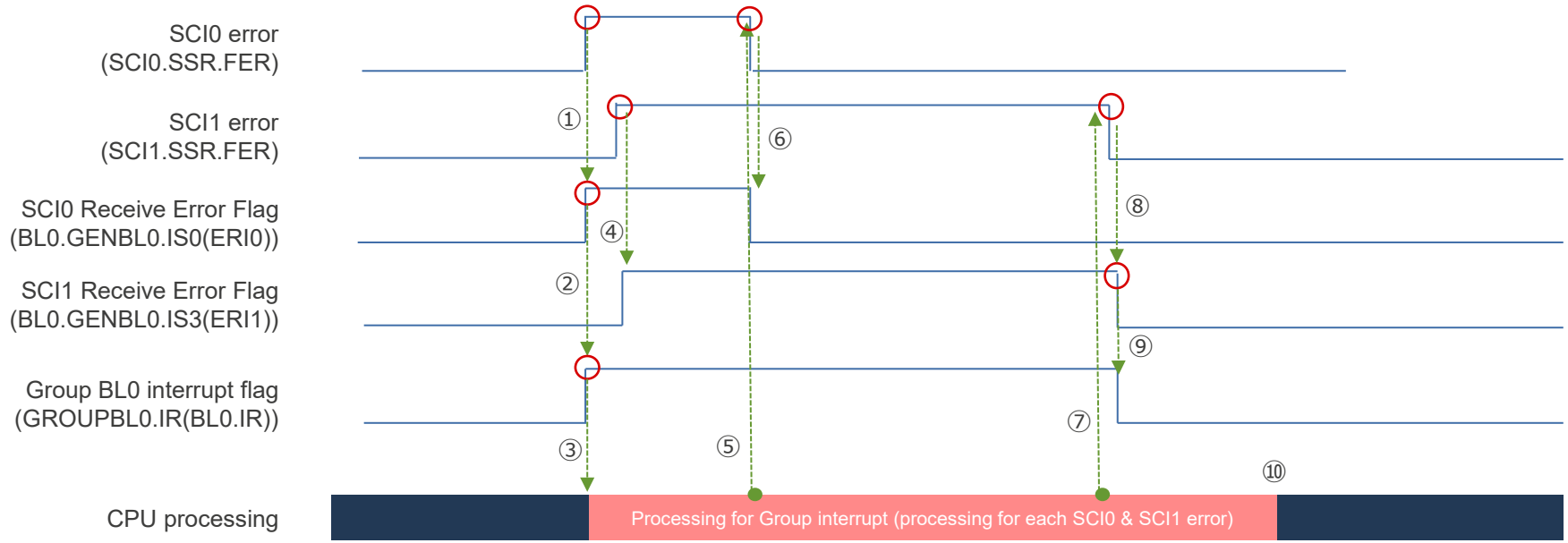
- This is an example of Group interrupt(Edge interrupt).



① Detects CAN0 errors. BE0.GRPBE0.IS0 is set.
② GROUP.IR is set by ①.
③ Processing for Group interrupt begun. GROUP.IR bit is cleared automatically.
④ CAN1 error also occurred during processing for Group interrupt. BE0.GRPBE0.IS1 is set.
⑤ The GROUP.IR is set (pending) according to ④.
⑥ During the processing for Group interrupt, CAN0.EIFR.BEIF and BE0.GROUPBE0.IS0 are cleared.
⑦ After the processing for Group interrupt is completed, group interrupt occurs again due to the pending IR bit.
⑧ During the processing for Group interrupt, CAN1.EIFR.ORIF and BE0.GROUPBE0.IS1 are cleared.

EXAMPLE : GROUP INTERRUPT(LEVEL INTERRUPT)

- This is an example of Group interrupt(Level interrupt).



● : Data Write timing

① Detect SCI0 framing error. BL0.GENBL0.IS0 is set.
② According to ①, GROUPBL0.IR is set.
③ Start the processing for group interrupt. GROUPBL0.IR bit is cleared automatically.
④ SCI framing errors also occurred during processing for group interrupt. BL0.GENBL0.IS3 is set.
⑤ During the processing for group interrupt, SCI0.SSR.FER is cleared.
⑥ According to ⑤, BL0.GENBL0.IS0 is cleared.
⑦ During the processing for group interrupt, SCI1.SSR.FER is cleared.
⑧ According to ⑦, BL0.GENBL0.IS3 is cleared.
⑨ All ISx flags assigned to the group interrupt have been cleared, so GROUPBL0.IR bit are cleared.
⑩ After confirming that the GROUPBL0.IR bit has been cleared, processing for group interrupt is completed.

FILTERING

- The NMI and IRQ input pin can be filtered to remove noise.
- The digital filter samples signals input to pins using the sampling clock for the digital filter, and passes the input signal only when three consecutive sampled signals are the same level.
- Noise filtering is not available during Software Standby Mode, Deep Software Standby Mode, or Snooze Mode.

	Digital Filter Enable/Disable Setting	Digital Filter Sampling Clock Setting
NMI input pin	NMIFLTE	NMIFLTC
IRQ input pin	IRQFLTEn	IRQFLTCn

RETURNING FROM LOW-POWER CONSUMPTION MODE

- Interrupt can be used to returning from low-power consumption mode.
  For details, please refer to “Low Power Consumption” chapter in the hardware manual.

- When returning from low-power consumption mode due to a non-maskable interrupt or a maskable interrupt, there are precautions to be taken when setting each register of the interrupt function. For more information, please refer to Interrupt Controller & Exiting Low Power Consumption State chapter in the Hardware Manual.

HANDLING ROUTINE FOR THIS INTERRUPT AND NOTICE

- The following is handling routine for this interrupt and Notice.

Fast interrupt and normal interrupt have different save destination for PC/PSW.
The save destination for fast interrupts is a dedicated register.
The interrupt processing cycle is faster for fast interrupt.

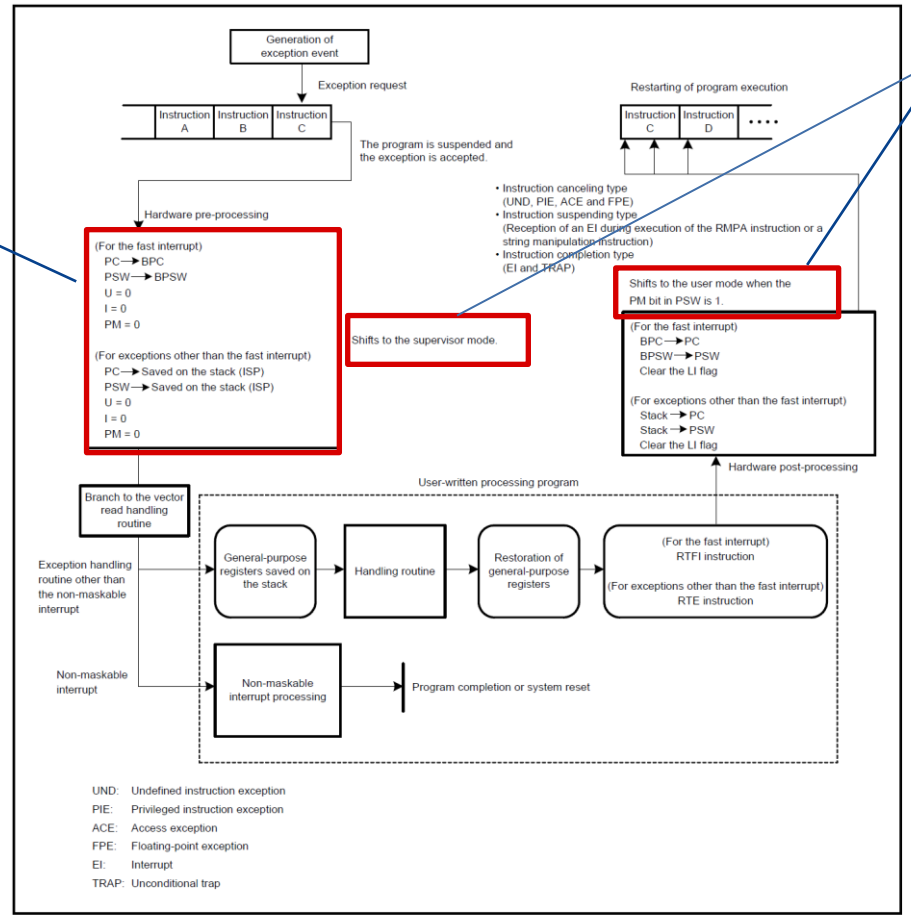


Figure 14.2 Outline of Exception Handling Procedure

If PSW.PM bit is 1, mode is **switched to supervisor mode** before processing interrupt. After processing for interrupt is completed, mode is **switched to user mode**.

APPLICATION NOTE FOR INTERRUPT CONTROLLER

Please refer to the following Application note for detailed instructions on how to use the Interrupt Controller.

- RX Family Using Multiple Interrupts R01AN1954
- RX Family Using the Exception Vector Table and Software Configurable Interrupts R01AN2178

EXTERNAL BUS CONTROLLER

This chapter is created with reference to RX72M. However, it can be used as a reference for all RX family products.

BUS CONTROLLER CONFIGURATION

- The following is an example of bus controller configurations(RX72M case).

CPU bus
- Instruction bus + Operand bus
- Connect to Memory bus 1~3 and Internal main bus (Each memory is directly accessible)
- Parallel operation is possible for buses with different requests for instruction fetching and operand access.

Internal peripheral bus
- Internal peripheral bus 1~6
- Connect to Internal main bus 1 & 2
- The order of priority : Choose toggle or fixed on each Internal peripheral bus 1, Internal peripheral bus 2/3, Internal peripheral bus 4/5, Internal peripheral bus 6.
  - When the priority order is fixed : Internal peripheral bus 2 > Internal peripheral bus 1

External bus
- Arbitrates requests for bus mastership on the external address space and external bus controller registers (CSC and SDRAMC) from internal main bus 1, internal main bus 2, and EXDMAC.
-The order of priority : Choose toggle or fixed
  - When the priority order is fixed : EXDMAC > Internal main bus 2 > Internal main bus 1
  - When the priority order is toggle : Toggle Internal main bus 1 and others. Internal main bus 2 and EXDMAC are fixed to EXDMAC > Internal main bus 2.

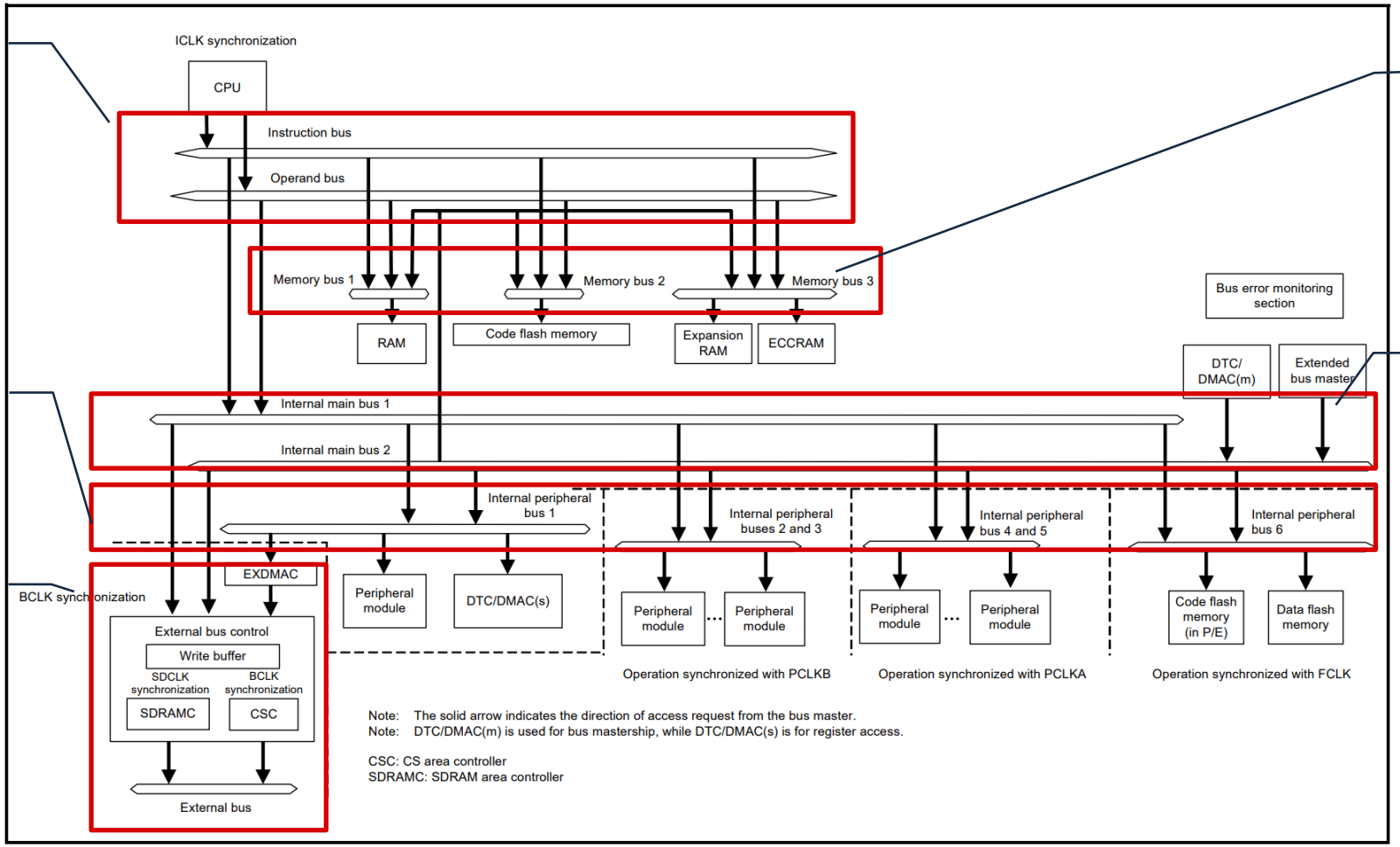


Figure 16.1 Bus Configuration

Memory bus
- Memory bus 1~3
- Connect to CPU bus, Internal main bus 2
- The order of priority : Choose toggle or fixed
  - When the priority order is fixed : Internal main bus > CPU bus

Internal main bus
- Internal main bus 1/2
  - Internal main bus 1 : For CPU
  - Internal main bus 2 : For DTC, DMAC, Extended bus master*1
- The order of priority : Choose toggle or fixed
  - When the priority order of Internal main bus 1 is fixed
    Operand > Instruction fetch
  - When the priority order of Internal main bus 2 is fixed
    Extended bus master > DMAC > DTC*2
- If the CPU and another bus master are requesting access to different buses, the respective bus-access operations can proceed simultaneously.*3

*1 : Extended bus master : EDMAC, GLCDC, DRW2D
The order of priority in case of conflicting bus requests can be set in EBMARCR.
*2 : DMAC & DTC ch have fixed priorities
DMAC0>DMAC1>DMAC2>DMAC3>DMAC4>DMAC5>DMAC6>DMAC7>DTC

*3 :
- If the XCHG instruction is executed, it will not accept any non-CPU bus access until the bus access is complete.
- Requests for bus access from masters other than the DTC are not accepted during reading and writing-back of transfer control information for the DTC.

OPERATION SPECIFICATIONS OF WRITE BUFFER FUNCTION FOR INTERNAL PERIPHERAL BUS AND PARALLEL

- The internal peripheral bus has a write buffer function, and in the case of write access, the next operation can be executed without waiting for the end of operation (Write to the actual register). In addition, if each bus master accesses different access destinations, it can operate in parallel.
- Operation Specifications of Write Buffer
  - ① In the case of Sequential access (If the first and second access destinations are the same irrespective of request source(bus master)) : **Sequential execution is available without waiting for register write**
  - ② In the case of Sequential access by the same requester (If only one of the first and second access destinations is internal memory) :
    - ②-1 : If the requester is a CPU : **Parallel Operation is available**
    - ②-2 : If the requester is DTC, DMAC, or Extended Bus master : **Sequential execution is available without waiting for register write**
  - ③ In the case of Sequential access by the same requester (If both the first and second access destinations are peripheral bus (However, in the case of peripheral bus with different numbers)) : **Waiting until the register write**
- Operation Specifications of Parallel Operation
  - ④ In the case of Sequential access by different requester (If the first and second access destinations are different.) : **Parallel Operation is available**

No	1st access		2nd access		Result
	Source	Access destination	Source	Access destination	
1	CPU	Internal peripheral bus1	CPU	Internal peripheral bus1	① Sequential execution(Not wait register write)
2	DTC	Memory bus	DMAC	Memory bus	① Sequential execution(Not wait register write)
3	CPU	Internal peripheral bus1	CPU	Memory bus	②-1 Parallel Operation (By pipeline)
4	DTC	Internal peripheral bus1	DTC	Memory bus	②-2 Sequential execution (Because the bus master of the internal peripheral bus 2 cannot operate in parallel)
5	CPU	Internal peripheral bus1	CPU	Internal peripheral bus2	③ Waiting until the register write end
6	DMAC	Internal peripheral bus1	Extended Bus Master	Internal peripheral bus2	③ Waiting until the register write end
7	CPU	Internal peripheral bus1	Extended Bus Master	Memory bus	④ Parallel Operation
8	DTC	Internal peripheral bus2	CPU	Internal peripheral bus1	④ Parallel Operation

Example for Internal peripheral bus write buffer and parallel operation

① Sequential execution
(Not wait register write, execute next bus access)

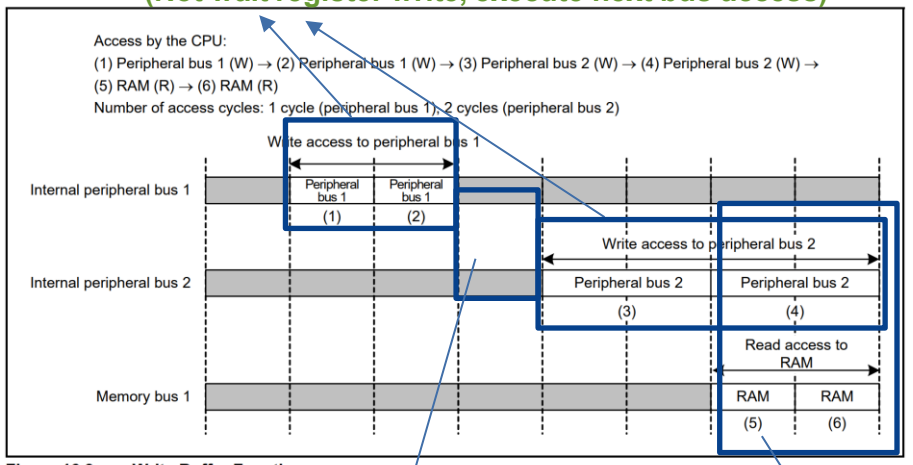
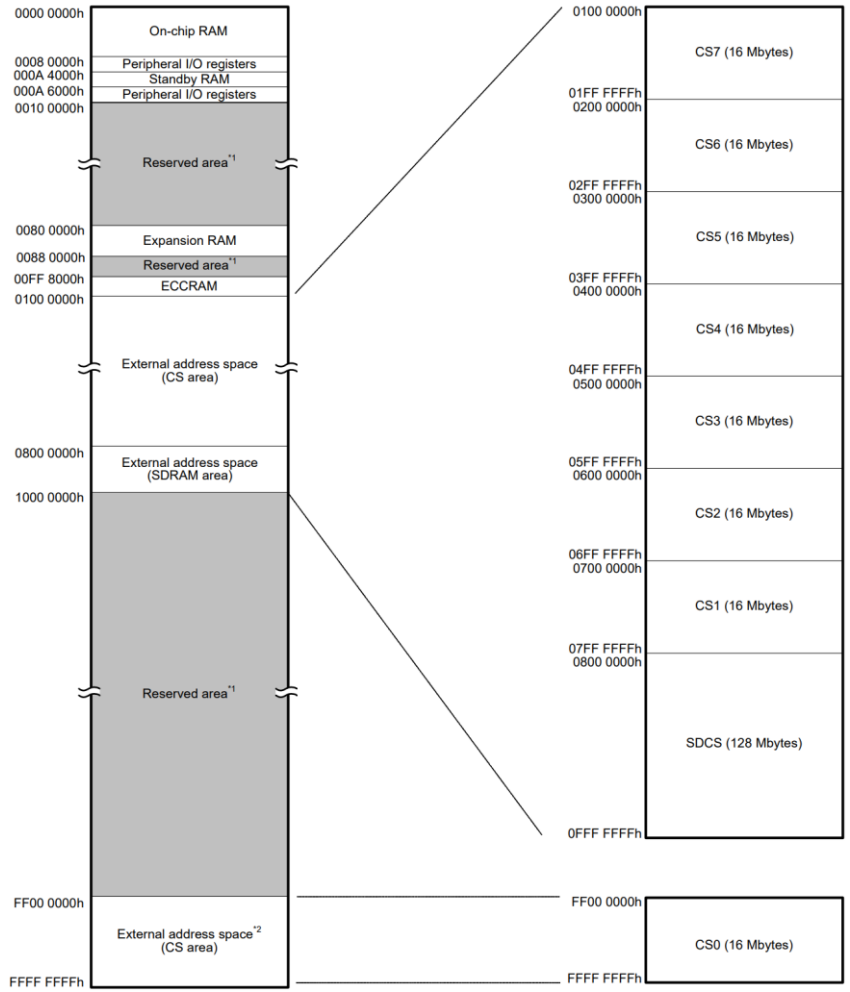


Figure 16.3 Write Buffer Function

③ Waiting the register write **②-1 Parallel Operation**

Internal peripheral bus write buffer control

EXTERNAL BUS



Correspondence between External Address Spaces and CS Areas
(In On-Chip ROM Disabled Extended Mode)

The external address space is divided into several CS areas and SDRAM area.
Bus connection can be made by setting the access method and Wait according to the specifications of the other device, and controlling the output waveform.

■ CS areas

For each space, you can set the following individually:

- Access Methods (Separate bus or Address/data multiplexed bus)
- Bus width
- Recovery cycles, Cycle wait, External/Internal wait control
- Write access mode (Single write strobe mode/byte strobe mode)
- endian mode
- Page Read Access/Page Write Access

The CS0 area and built-in ROM area are exclusive.

■ SDRAM area

Connectable to SDRAM I/F, some products support this function.

MODE AND PIN LIST FOR EXTERNAL BUS

- The following is a list of the modes of the external bus and the available pins.

* A0/D0~A15/D15 are dedicated to address/data multiplexing.
It cannot be used as an address on a separate bus.

Pin	CS areas				SDRAM area	No Access State
	Separate bus		Address/data multiplexed bus			
	Byte strobe	Single write strobe	Byte strobe	Single write strobe		
A23~A0	✓	✓			✓	
D31～D0	✓	✓			✓	high-impedance
A0/D0~A15/D15*			✓	✓		
BC0#~BC3#		✓		✓		High
CS0#~CS7#	✓	✓	✓	✓		High
RD#	✓	✓	✓	✓		High
WR0#/WR#	✓ (Use as WR0#)	✓ (Use as WR#)	✓ (Use as WR0#)	✓ (Use as WR#)		High
WR1#~WR3#	✓		✓			High
ALE			✓	✓		Low
WAIT#	✓	✓	✓	✓		
BCLK	✓	✓	✓	✓		
SDCLK					✓	
CKE					✓	
SDCS#					✓	High
RAS#					✓	High
CAS#					✓	High
WE#					✓	High
DQM0~DQM3					✓	

MODE AND REGISTERS FOR EXTERNAL BUS

- The following is a list of the modes of the external bus and the registers.

Register name and settings		CS area	SDRAM area	Bus error	Priority control
CSnCR	Operation Enable, Bus width, Endian mode, Access method (Separate or Multiplex)	✓			
CSnREC	Recovery Cycle (Read, Write)	✓			
CSRECEN	Recovery Cycle Insertion Enable	✓			
CSnMOD	Write Access Mode Select, External Wait Enable, Page Read/Write Access Enable	✓			
CSnWCR1,2	Wait Control(Normal Write/Read, Page Write/Read, CS Extension Cycle(Read/Write), Write Data Output Extension/Output Wait, Address Cycle, Assert(RD,WD,CS))	✓			
SDCCR	SDRAM Operation Enable		✓		
SDCMOD	SDRAM area Endian Mode		✓		
SDAMOD	SDRAM Continuous Access Enable		✓		
SDSELF	SDRAM Self-Refresh Enable		✓		
SDRFCR	Auto-Refresh related		✓		
SDRFEN	Auto-Refresh Operation Enable		✓		
SDICR	SDRAM Initialization Sequence		✓		
SDIR	SDRAM Initialization Auto-Refresh Interval, Count, Initialization Precharge Cycle Count		✓		
SDADR	SDRAM Address Multiplex Select		✓		
SDTR	SDRAM Column Latency, Write Recovery Interval, Row Precharge Interval, Row Column Latency, Row Active Interval		✓		
SDMOD	SDRAM Mode Register Setting		✓		
SDSR	SDRAM access status		✓		
BERCLR	Bus Error Status Clear			✓	
BEREN	Bus Error Monitoring Enable			✓	
BERSR1,2	Bus Error Status			✓	
BUSPRI	Bus Priority Control				✓
EBMAPCR	Extended Bus Master Priority Control				✓

MODE AND REGISTERS FOR EXTERNAL BUS(DETAIL INFO. FOR CS)

- The following is the register details that require each mode and configuration of the external bus in CS areas. W : Write, R : Read

Register and settings				Separate	Address/data multiplex	Page access	Recovery
Items	Register	Bit	Details				
Settings related to mode	CSnCR	EXENB	Operation Enable	✓	✓	-	-
		BSIZE[1:0]	External Bus Width Select	✓	✓	-	-
		EMODE	Endian Mode	✓	✓	-	-
		MPXEN	Address/Data Multiplexed I/O Interface Select	✓	✓	-	-
	CSnMOD	WRMOD	Write Access Mode Select (Byte strobe/Single write strobe)	✓	✓	-	-
		EWENB	External Wait Enable	✓	✓	-	-
		PRENB	Page Read Access Enable	-	-	✓(R)	-
		PWENB	Page Write Access Enable	-	-	✓(W)	-
		PRMOD	Page Read Access Mode Select (Normal access compatible/External data read continuous assertion)	-	-	✓(R)	-
Settings related to Recovery cycle	CSnREC	RRCV[3:0]	Read Recovery count setting	-	-	-	✓(R)
		WRCV[3:0]	Write Recovery count setting	-	-	-	✓(W)
	CSRECEN	RCVEN0~7	Separate Bus Recovery Cycle Insertion Enable	-	-	-	✓(R)
		RCVENM0~7	Multiplexed Bus Recovery Cycle Insertion Enable	-	-	-	✓(W)
Wait Control	CSnWCR1	CSPWWAIT[4:0]	Page Write Cycle Wait Select	-	-	✓(W)	-
		CSPRWAIT[4:0]	Page Read Cycle Wait Select	-	-	✓(R)	-
		CSWWAIT[4:0]	Normal Write Cycle Wait Select	✓(W)	✓(W)	-	-
		CSRWAIT[4:0]	Normal Read Cycle Wait Select	✓(R)	✓(R)	-	-
	CSnWCR2	CSROFF[2:0]	Read-Access CS Extension Cycle Select	✓(R)	✓(R)	-	-
		CSWOFF[2:0]	Write-Access CS Extension Cycle Select	✓(W)	✓(W)	-	-
		WDOFF[2:0]	Write Data Output Extension Cycle Select	✓(W)	✓(W)	-	-
		AWAIT[1:0]	Address Cycle Wait Select		✓(W)	-	-
		RDON[2:0]	RD Assert Wait Select	✓(R)	✓(R)	-	-
		WRON[2:0]	WR Assert Wait Select	✓(W)	✓(W)	-	-
		WDON[2:0]	Write Data Output Wait Select	✓(W)	✓(W)	-	-
		CSON[2:0]	CS Assert Wait Select	✓(R/W)	✓(R/W)	-	-

PIN SETTINGS FOR EXTERNAL BUS INTERFACE

- Pin settings for External bus interface is described in Multi-Function Pin Controller (MPC) section.
  After setting the MPC register as shown in the table below, it will be enabled as an external bus pin by setting it to SYSCR0.EXBE=1.

Please set it according to the registers listed here.
If you want to use it as other pin function, you need to set it differently from this one.

If there are multiple registers, you need to set all of them.
If one is not set, it will not be enabled as a pin for external bus, but can be used as other function pin.

<Smart Configurator settings>
ALE, D1 : Prohibited
CS1#, SDCS# : Prohibited
SDCS#, D0 : Prohibited
A0, BC0, DQM2 : Permitted

Table 23.46 How to Set the External Bus Interface (1/4)

Port	Output Signal	Settings of MPC Registers	
		224-Pin, 176-Pin	144-Pin
P10	ALE	PFBCR1.ALEOE = 1, PFBCR1.ALES = 1	
P12	WR3#/BC3#	PFBCR0.WR32BC32E = 1	(not supported)
P13	WR2#/BC2#	PFBCR0.WR32BC32E = 1	(not supported)
P24	CS4#	PFCSE.CS4E = 1, PFCSS1.CS4S[1:0] = 10/11	
P25	CS5#	PFCSE.CS5E = 1, PFCSS1.CS5S[1:0] = 10/11	
P26	CS6#	PFCSE.CS6E = 1, PFCSS1.CS6S[1:0] = 10/11	
P27	CS7#	PFCSE.CS7E = 1, PFCSS1.CS7S[1:0] = 10/11	
P50	WR0#/WR#	—	
P51	WR1#/BC1#	PFBCR0.WR1BC1E = 1	
	WAIT#	PFBCR1.WAITS[1:0] = 11, PFBCR3.WAITS2 = 0	
P52	RD#	—	
P53	BCLK	—	
P54	ALE	PFBCR1.ALEOE = 1, PFBCR1.ALES = 0	
	D1[A1/D1]	PFBCR2.D1S[1:0] = 10	
P55	WAIT#	PFBCR1.WAITS[1:0] = 01, PFBCR3.WAITS2 = 0	
	D0[A0/D0]	PFBCR2.D0S[1:0] = 10	
P60	CS0#	PFCSE.CS0E = 1, PFCSS0.CS0S = 0	
P61	CS1#	PFCSE.CS1E = 1, PFCSS0.CS1S[1:0] = 00	
	SDCS#	PFBCR1.MDSDE = 1	(not provided)
	D0[A0/D0]	PFBCR2.D0S[1:0] = 11	
PA0	A0	PFBCR0.ADRLE = 1, CSnMOD.WRMOD = 0	
	BC0#	PFBCR0.ADRLE = 1, CSnMOD.WRMOD = 1	
	DQM2	PFBCR0.ADRLE = 1, SDCCR.EXENB = 1, SDCCR.BSIZE[1:0] = 01	(not supported)

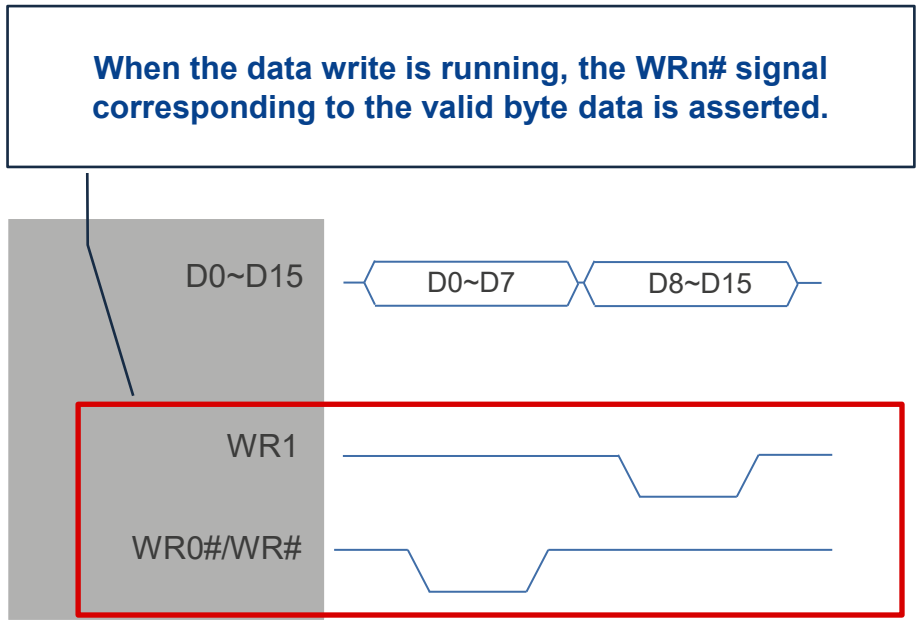
Those with a "—" are defined as "SYSCR0. EXBE" bit to 1 to enable it as a bus terminal.

If multiple pins are selected and do not work correctly, select only one of them. In the case of P61, if CS1# and SDCS# are turned on at the same time, CS1 space will also be active when accessing the SDRAM space, and it will malfunction, so simultaneous settings are prohibited.

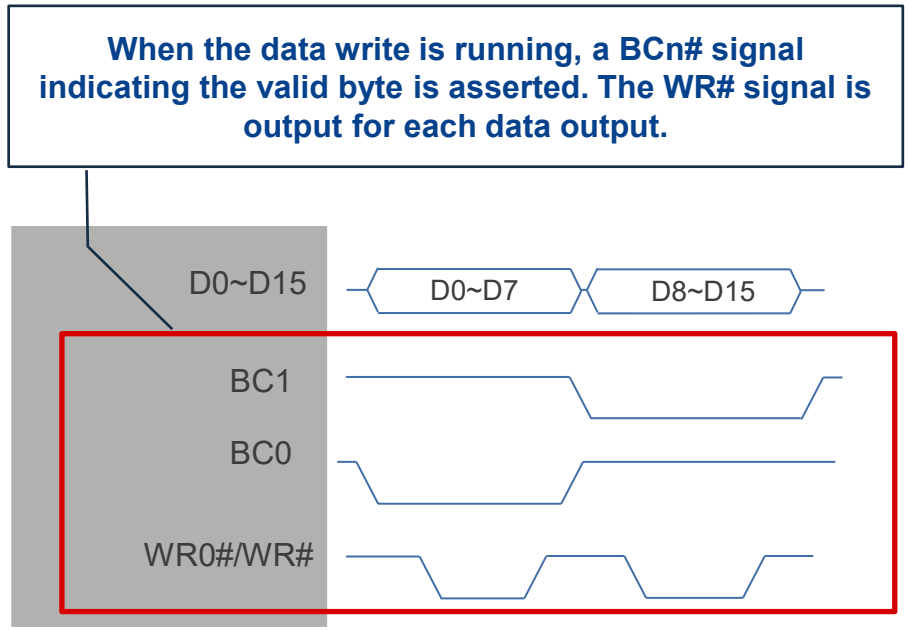
If you connect several different devices but control them exclusively, you can select multiple output signals. In the case of PA0, when DQM2 is activated with SDRAM access, the BC0# terminal is also activated, but it does not malfunction unless the corresponding CS# is activated, so it can be set at the same time.

EXTERNAL BUS
CS AREA (WRITE ACCESS MODE)

- Write access mode support Byte strobe mode and Single write strobe mode.
  An example of each signal output is shown below.



Signal output example for Byte Strobe Write Access (16bit access to 4n+1 address in 16bit bus area)



Signal output example for Single write Strobe Write Access (16bit access to 4n+1 address in 16bit bus area)

ENDIAN

- External bus can be set to endian for each CS area. In addition, multiple accesses occur in a single transfer request depending on the set endian and access address. If you access a data size of more than the bus width, page access may occur after the second time the number of accesses. By specifying the access address according to the bus width of the space to which the access is being accessed, the increase in the number of accesses can be reduced.

The order of access to data changes depending on the endian you set.

Page access may occur when accessing more than the data bus width. The applicable access is marked with “(p)”.

Data Size	Access Address	Number of Access	Bus Cycle	Unit of Data	Address		WR1#/BC1# D15–D8	WR0#/BC0# D7–D0
							RD# Data Bus D15–D8	RD# Data Bus D7–D0
8 bits	4n	One	First	8 bits	4n			7–0
	4n+1	One	First	8 bits	4n		7–0	
	4n+2	One	First	8 bits	4n+2			7–0
	4n+3	One	First	8 bits	4n+2		7–0	
16 bits	4n	One	First	16 bits	4n		15–8	7–0
	4n+1	Two	First	8 bits	4n		7–0	
			Second	8 bits	4n+2			15–8
	4n+2	One	First	16 bits	4n+2		15–8	7–0
	4n+3	Two	First	8 bits	4n+2		7–0	
			Second	8 bits	4n+4			15–8
32 bits	4n	Two	First	16 bits	4n		15–8	7–0
			Second	16 bits	4n+2	(p)	31–24	23–16
	4n+1	Three	First	8 bits	4n		7–0	
			Second	16 bits	4n+2		23–16	15–8
			Third	8 bits	4n+4			31–24
	4n+2	Two	First	16 bits	4n+2		15–8	7–0
			Second	16 bits	4n+4		31–24	23–16
	4n+3	Three	First	8 bits	4n+2		7–0	
			Second	16 bits	4n+4		23–16	15–8
			Third	8 bits	4n+6			31–24
64 bits	4n	Four	First	16 bits	4n		15–8	7–0
			Second	16 bits	4n+2	(p)	31–24	23–16
			Third	16 bits	4n+4		47–40	39–32
			Fourth	16 bits	4n+6	(p)	63–56	55–48

(p): Page access (only when page access is enabled with the PRENB and PWENB bits in CSnMOD)

Figure 16.8 Data Alignment (Little Endian) in 16-Bit Bus Space

Data Size	Access Address	Number of Access	Bus Cycle	Unit of Data	Address		WR1#/BC1# D15–D8	WR0#/BC0# D7–D0
							RD# Data Bus D15–D8	RD# Data Bus D7–D0
8 bits	4n	One	First	8 bits	4n		7–0	
	4n+1	One	First	8 bits	4n			7–0
	4n+2	One	First	8 bits	4n+2		7–0	
	4n+3	One	First	8 bits	4n+2			7–0
16 bits	4n	One	First	16 bits	4n		15–8	7–0
	4n+1	Two	First	8 bits	4n			15–8
			Second	8 bits	4n+2		7–0	
	4n+2	One	First	16 bits	4n+2		15–8	7–0
	4n+3	Two	First	8 bits	4n+2			15–8
			Second	8 bits	4n+4		7–0	
32 bits	4n	Two	First	16 bits	4n		31–24	23–16
			Second	16 bits	4n+2	(p)	15–8	7–0
	4n+1	Three	First	8 bits	4n			31–24
			Second	16 bits	4n+2		23–16	15–8
			Third	8 bits	4n+4		7–0	
	4n+2	Two	First	16 bits	4n+2		31–24	23–16
			Second	16 bits	4n+4		15–8	7–0
	4n+3	Three	First	8 bits	4n+2			31–24
			Second	16 bits	4n+4		23–16	15–8
			Third	8 bits	4n+6		7–0	
64 bits	4n	Four	First	16 bits	4n		63–56	55–48
			Second	16 bits	4n+2	(p)	47–40	39–32
			Third	16 bits	4n+4		31–24	23–16
			Fourth	16 bits	4n+6	(p)	15–8	7–0

(p): Page access (only when page access is enabled with the PRENB and PWENB bits in CSnMOD)

Figure 16.9 Data Alignment (Big Endian) in 16-Bit Bus Space

NORMAL READ CYCLE (EACH WAIT SETTINGS)

- Example of the timing when it is “Normal read". Please insert the wait to match the connected device and control the output waveform.

This is wait cycle for RD# assertion.
RD# is asserted until the next BCLK cycle of this wait cycle

This is the wait cycle until the CS# is asserted.

This is the wait cycle until the RD# is asserted.

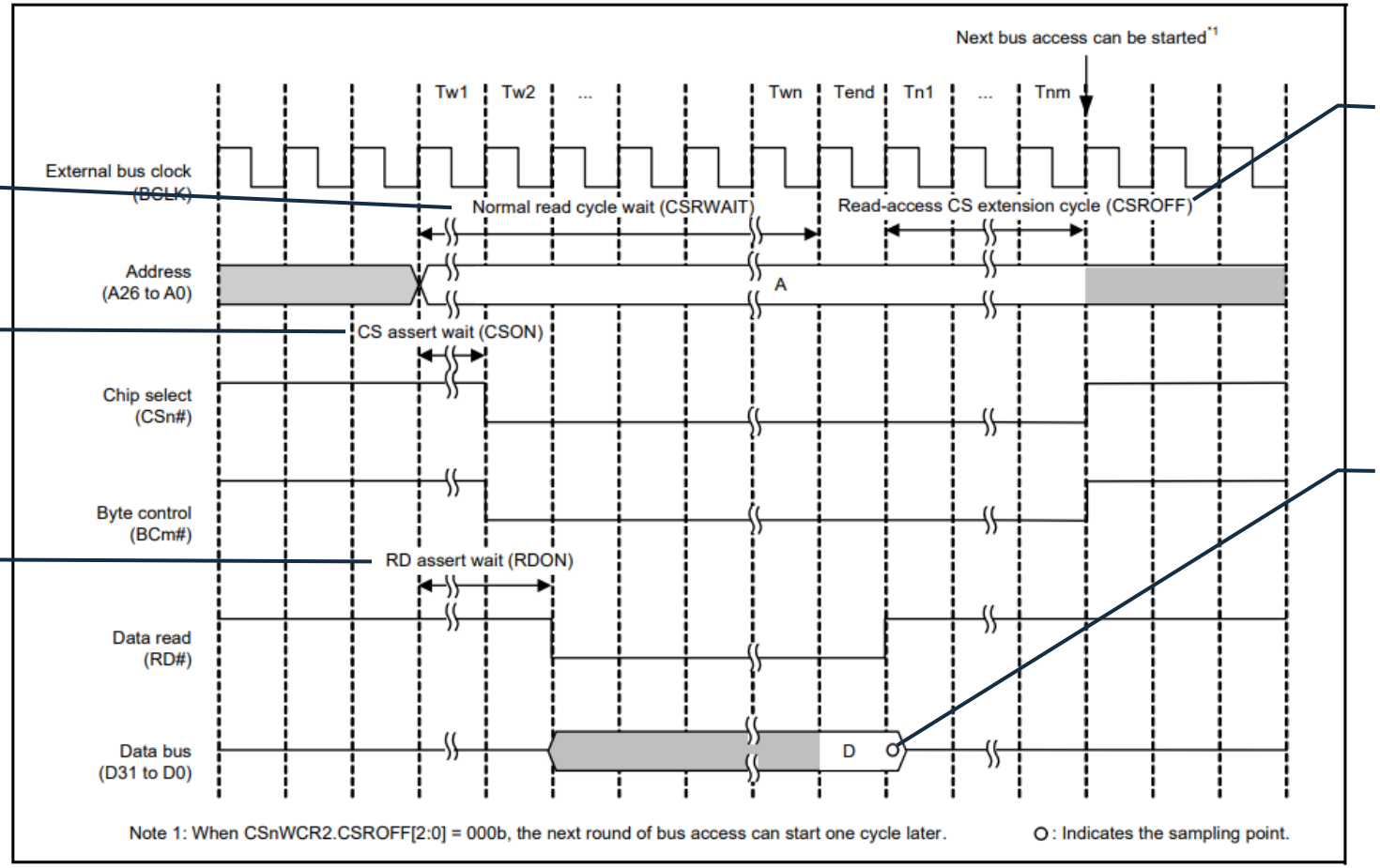


Figure 16.19 Bus Timing (Normal-Read Operation) (n = 0 to 7, m = 0 to 3)

This is the wait cycle until the CS# is negated.
After the CS# is negated, the next bus access can be started, but if this value is 0, the next access will start after one cycle.

This is the sampling point of the read.

The normal read cycle has the following constraints :
CSON[2:0]≦CSRWAIT
RDON [2:0] ≦CSRWAIT
CSON [2:0] ≦RDON

Register Name	Bit Name
CSnWCR1	CSRWAIT[4:0]
CSnWCR2	CSROFF[2:0]
	RDON[2:0]
	CSON[2:0]

NORMAL WRITE CYCLE (EACH WAIT SETTINGS)

- Example of the timing when it is “Normal write". Please insert the wait to match the connected device and control the output waveform.

This is wait cycle for WR# assertion.
WR# is asserted until the next BCLK cycle of this wait cycle

This is the wait cycle until CS# is asserted.

This is the wait cycle until WR# is asserted.

This is the wait cycle until the write data starts to be output.

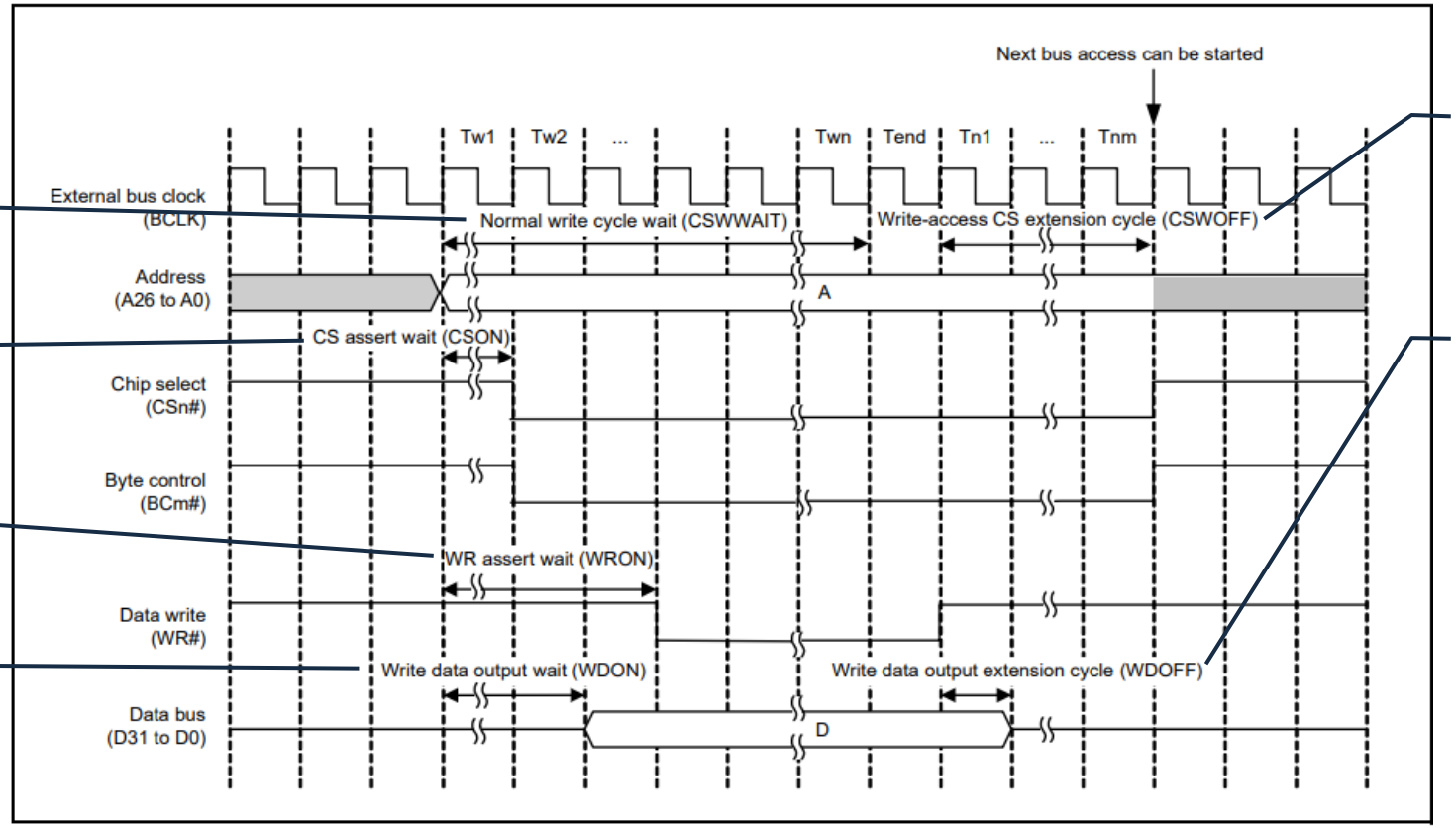


Figure 16.20 Bus Timing (Normal-Write Operation, Single Write Strobe Mode) (n = 0 to 7, m = 0 to 3)

Register Name	Bit Name
CSnWCR1	CSWWAIT[4:0]
CSnWCR2	CSWOFF[2:0]
	WDOFF[2:0]
	WRON[2:0]
	WDON[2:0]
	CSON[2:0]

It is the wait cycle from the time “WR# is negated” to the time “CS# is negated”.
After the CS# negate, the following bus access can be initiated:

This is the wait cycle from the time “WR# signal is negated” to the time “Write data is negated”.

The normal write cycle has the following constraints :
1≦WDON[2:0]
CSON[2:0]≦CSWWAIT
WRON [2:0] ≦CSWWAIT
WDON[2:0]≦CSWWAIT
WDOFF[2:0]≦CSWOFF
WDON[2:0]≦WRON
CSON [2:0] ≦WRON

EXTERNAL BUS CLOCK AND BCLK PIN

- Depending on the product, BCLK pin output must be set to 1/2 of the internal BCLK.
  Here are some examples of how this works and precaution.

Each wait count is based on internal BCLK. Please be noted that it is not based on BCLK pin output clock.

Depending on the number of wait you set, the negate timing of the data read/write may change with the fall edge of the BCLK pin output.

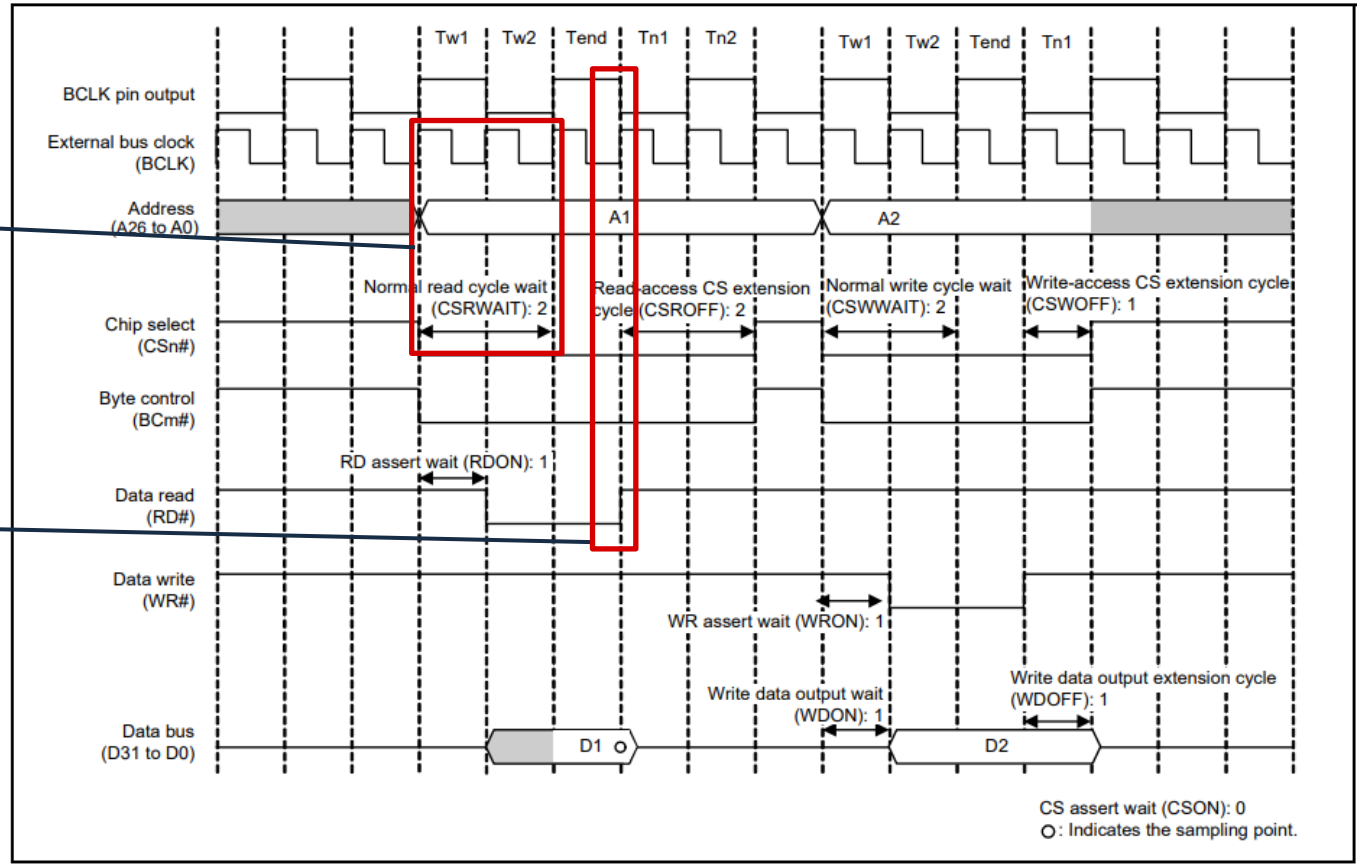


Figure 16.26 Example of Normal Access (when 1/2 BCLK is Selected with the BCLK Pin Output Select Bit) (n = 0 to 7, m = 0 to 3)

PAGE ACCESS (EACH WAIT SETTINGS)

- Page access occurs when multiple accesses occur in a single transfer request.
  Page access can have a different number of wait between data accesses than normal access.

During the second and subsequent accesses, the page read (write) cycle wait may be inserted instead of the normal read(write) cycle weight of normal access.

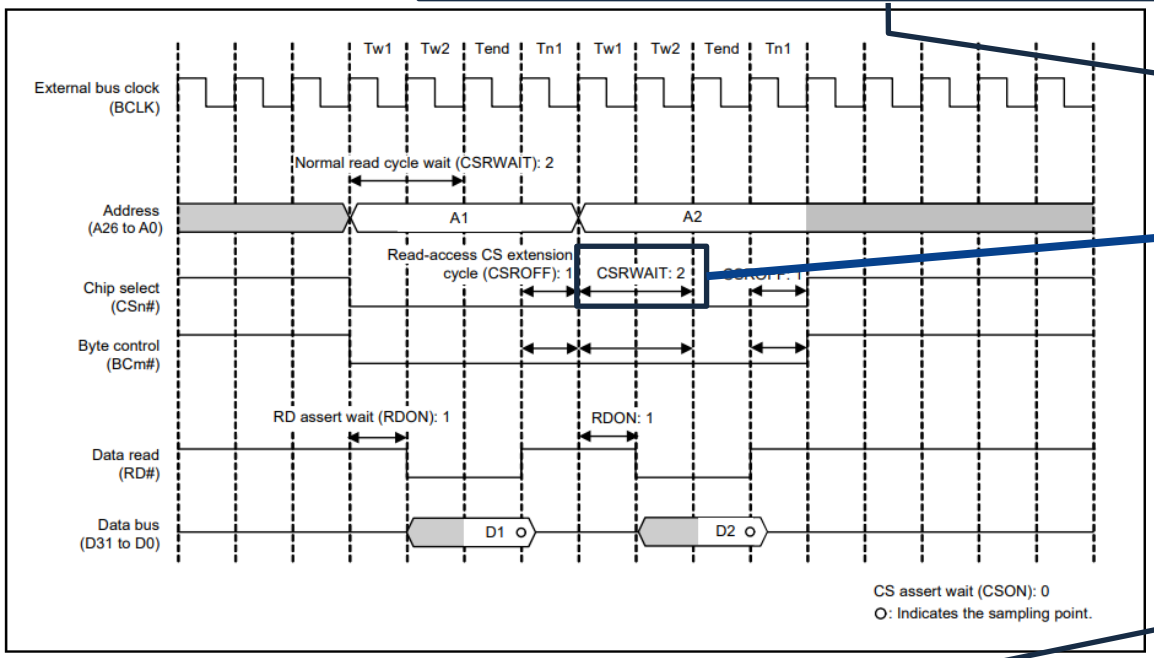


Figure 16.22 Example of Normal-Read Operation (when Two Rounds of Bus Access are Generated in Response to a Single Request for Transfer)

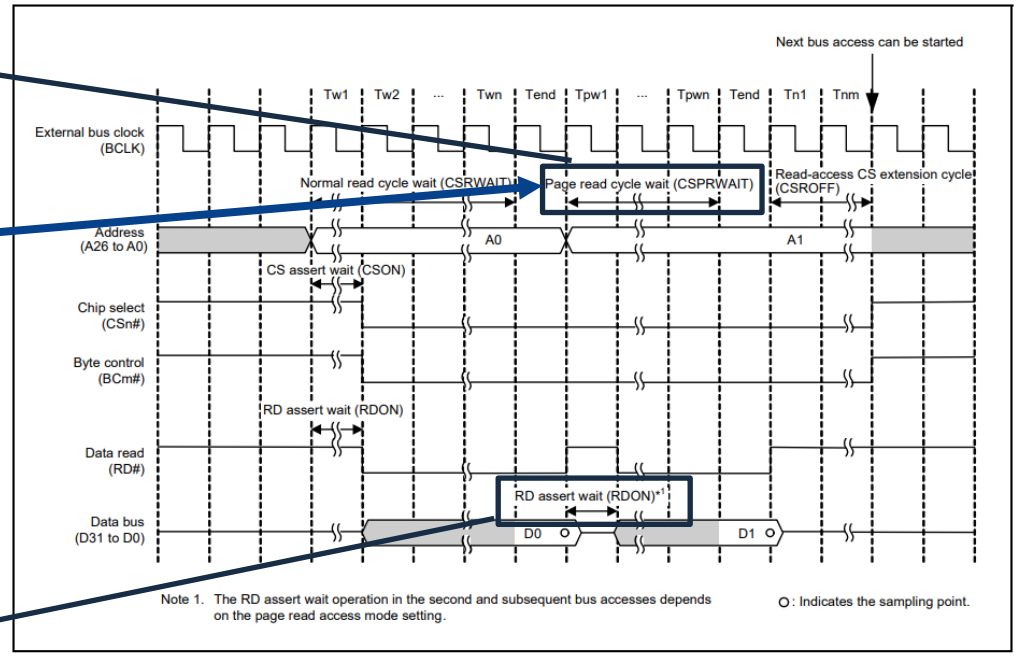


Figure 16.31 Page-Read Access Timing (n = 0 to 7, m = 0 to 3)

Register Name	Bit Name
CSnWCR1	CSPWWAIT[4:0](Write)
	CSPRWAIT[4:0](Read)

The WR assert wait is as valid as the first time. On the other hand, RD assert wait is handled as follows :
SDnMOD.PRMOD bit is 0 : RD assert is valid as in the first time, and the RD signal between them is negated.
SDnMOD.PRMOD bit is 1 : RD assert is valid as in the first time, and the RD signal continues to assert during that time.

ADDRESS/DATA MULTIPLEXED BUS

- The following is access example of Address/Data Multiplexed Bus.
  Please insert the wait to match the connected device and control the output waveform.

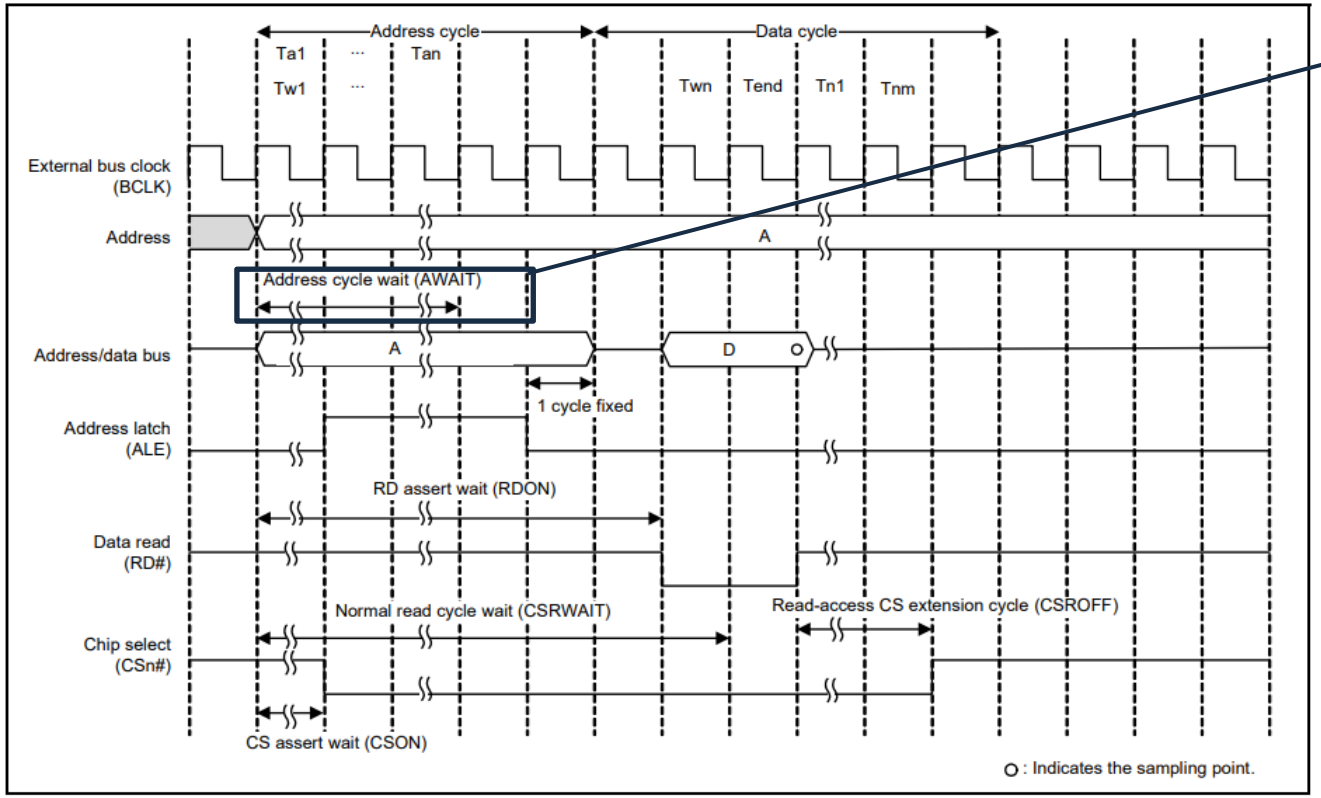


Figure 16.37 Example of Read Access Operation with Address/Data Multiplexed I/O Interface (n = 0 to 7)

In the first address cycle, set the negate timing for the ALE.

Address/Data Multiplexed Bus has the following constraints :

< Read >	< Write >
CSON[2:0]≦CSRWAIT	CSON≦CSWWAIT
RDON[2:0]≦CSRWAIT	WRON [2:0] ≦CSWWAIT
CSON [2:0] ≦RDON	WDON[2:0]≦CSWWAIT
AWAIT[1:0]+2≦RDON	WDOFF[2:0]≦CSWOFF
CSON[2:0]≦AWAIT	WDON[2:0]≦WRON
	CSON [2:0] ≦WRON
	AWAIT[1:0] +2≦WRON
	AWAIT[1:0] +2≦WDON
	CSON[2:0] ≦AWAIT

< Read >

Register Name	Bit Name
CSnWCR1	CSRWAIT[4:0]
CSnWCR2	CSROFF[2:0]
	RDON[2:0]
	CSON[2:0]
	AWAIT[4:0]

< Write >

Register Name	Bit Name
CSnWCR1	CSWWAIT[4:0]
CSnWCR2	CSWOFF[2:0]
	WDOFF[2:0]
	WRON[2:0]
	WDON[2:0]
	CSON[2:0]
	AWAIT[4:0]

RECOVERY CYCLES

- RX can set recovery cycles (extended CS# pin negate period) between external access and external access. Need to set the CS Negate period according to the connected device. In addition, if two or more external bus accesses are required for a single transfer request from the bus master, and the following conditions for inserting recovery cycles are met, the insertion of recovery cycles into the intermediate access is as follows.
  - For normal access case :
    When page access is disabled : Recovery cycles are inserted into intermediate access
    When page access is enabled : Recovery cycles are not inserted into intermediate access
  - For Address/Data Multiplexed case :
    Recovery cycles are inserted into intermediate access regardless of whether page access is enabled/disabled

Access Type	External Address Space	Insertion of Recovery Cycles	Enable/Disable recovery cycles insertion settings	
			Separate bus	Multiplexed bus
Read access after Read access	Same area	CSnREC.RRCV[3:0]*	CSRECEN.RCVEN0	CSRECEN.RCVENM0
	Different area		CSRECEN.RCVEN1	CSRECEN.RCVENM1
Write access after Read access	Same area		CSRECEN.RCVEN2	CSRECEN.RCVENM2
	Different area		CSRECEN.RCVEN3	CSRECEN.RCVENM3
Read access after Write access	Same area	CSnREC.WRCV[3:0]	CSRECEN.RCVEN4	CSRECEN.RCVENM4
	Different area		CSRECEN.RCVEN5	CSRECEN.RCVENM5
Write access after Write access	Same area		CSRECEN.RCVEN6	CSRECEN.RCVENM6
	Different area		CSRECEN.RCVEN7	CSRECEN.RCVENM7

* When CSnWCR2.CSROFF[2:0] is 000b (Read CS extend cycle is 0), The next bus start cycle will be one cycle later.
Even if you do not insert a cycle in this setting, the one cycle negate period will be inserted.

EXAMPLE : RECOVERY CYCLES INSERTION

- The following is an example of an insertion of recovery cycles.

For continuous external space accesses, recovery cycle is inserted between each access (CS terminal negate period).

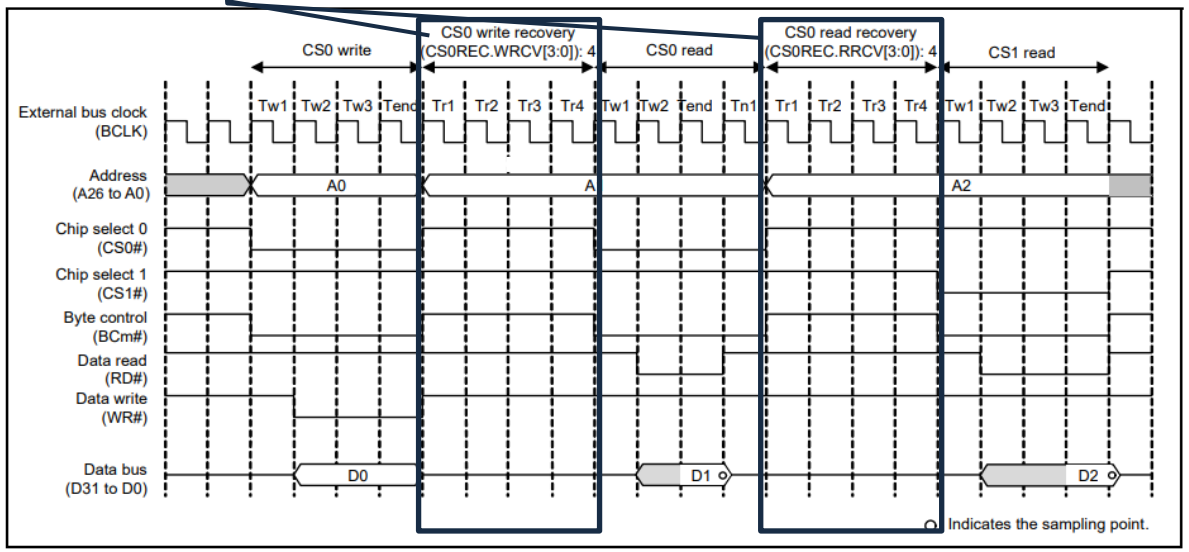


Figure 16.43 Example of Recovery Cycle Insertion with Separate Bus Interface (m = 0 to 3)

If two or more accesses occur in a single access request from the bus master, recovery cycles may be inserted depending on the conditions.

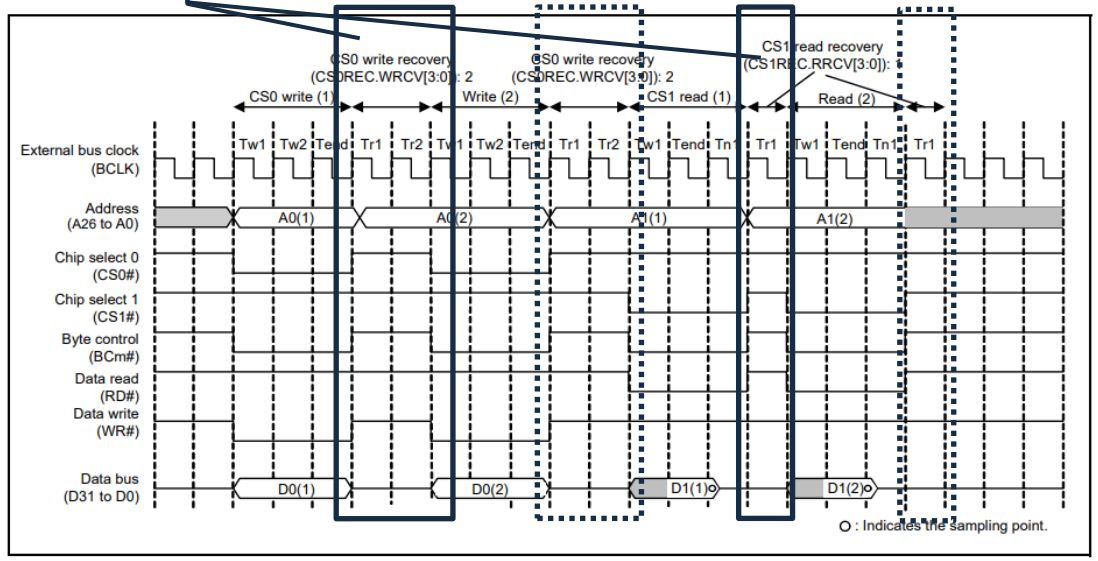


Figure 16.44 Example of Recovery Cycle Insertion When a Bus Access is Split (with Separate Bus Interface, Normal Access) (m = 0 to 3)

DETAIL OF SDRAM SPACE RELATED REGISTERS

- The following is each register's detail for SDRAM space and setting conditions.

Register Name and Description				Register setting conditions
Basic settings	SDCCR	EXENB	Operation Enable for SDRAM	
		BSIZE[1:0]	SDRAM Bus Width Select	
	SDCMOD	EMODE	Endian Mode setting for SDRAM	
	SDAMOD	BE	Continuous Access Enable for SDRAM	SDRAM Operation is disabled (SDCCR.EXENB=0) Auto-refresh operation is disabled (SDRFEN.RFEN=0) Self-refresh is disabled (SDSELF.SFEN=0)
Initialization Sequence	SDICR	INIRQ	Initialization Sequence Start	
Read/Write settings	SDADR	MXC[1:0]	Address Multiplex Select	
	SDTR	CL[2:0]	SDRAMC Column Latency	Self-refresh is enabled(SDSELF.SFEN=1) Or SDRAM Operation is disabled (SDCCR.EXENB=0) Auto-refresh operation is disabled (SDRFEN.RFEN=0) Self-refresh is disabled (SDSELF.SFEN=0)
		WR	Write Recovery Interval	
		RP[2:0]	Row Precharge Interval	
		RCD[1:0]	Row Column Latency	
		RAS[2:0]	Row Active Interval	
Refresh settings	SDSELF	SFEN	SDRAM Self-Refresh Enable	SDRAM Operation is disabled (SDCCR.EXENB=0) Auto-refresh operation is enabled (SDRFEN.RFEN=1)
	SDRFCR	RFE[11:0]	Auto-Refresh Request Interval Setting	Self-refresh is disabled (SDSELF.SFEN=0)
		REFW[3:0]	Auto-Refresh Cycle/ Self-Refresh Clearing Cycle Count Setting	
	SDRFEN	RFEN	Auto-Refresh Operation Enable	
	SDIR	ARF[3:0]	Initialization Auto-Refresh Interval	Before setting the SDICR and the same conditions as rewriting the SDICR
		ARFC[3:0]	Initialization Auto-Refresh Count	
		PRC[3:0]	Initialization Precharge Cycle Count	
Mode register settings	SDMOD	MR[4:0]	Mode Register Setting	SDRAM Operation is disabled (SDCCR.EXENB=0) Self-refresh is disabled (SDSELF.SFEN=0)
Status information	SDSR	MRSST,INIST,SRFST	SDRAM access status	

SDRAM READ/WRITE ACCESS

- SDRAM read/write access have two types : Single access and Consecutive access. There are limitations to using Consecutive access.
- Conditions for Consecutive access (If all of the following are met) :
  - ① When Continuous Access Enable in SDRAM Access Mode Register is “Enable”(SDAMOD.BE=1)*1
  - ② When there are consecutive accesses to the same raw address
  - ③ When using EXDMAC cluster transfer or block transfer in single address mode
  - ④ When single transfer data size from EXDMAC is less than/equal to external bus width and single transfer request results in a single bus access

*1 : When accessing the SDRAM area from a bus master other than EXDMAC, it is prohibited to set it to enable consecutive access. It is not guaranteed that it will work in such cases.

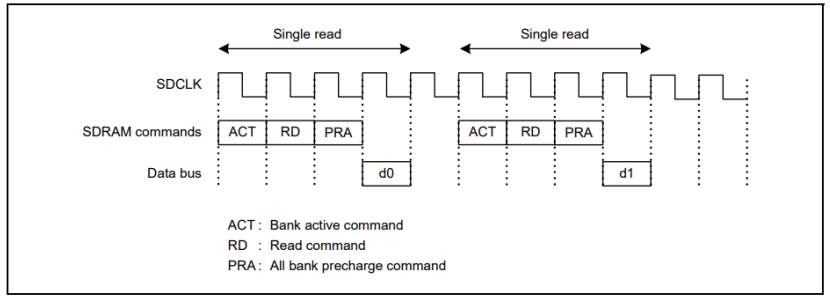


Figure 16.58 Timing Example of Single Read (SDTR.CL[2:0] = 010b: 2 Cycles)

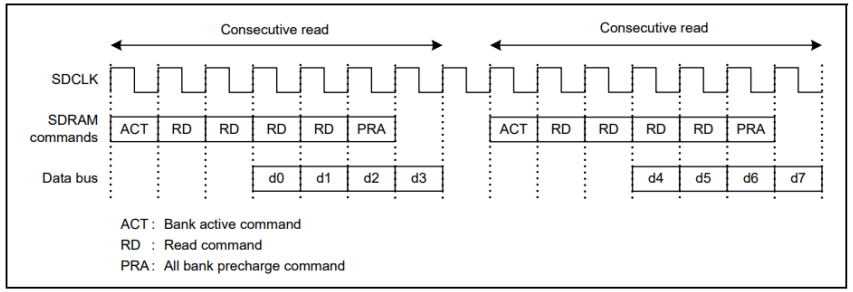


Figure 16.61 Timing Example of Consecutive Read (SDAMOD.BE = 1 and SDTR.CL[2:0] = 010b: 2 Cycles)

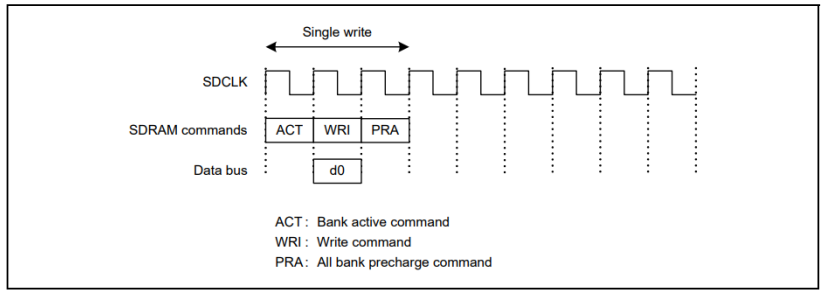


Figure 16.60 Timing Example of Single Write (when the Shortest Timing is Set)

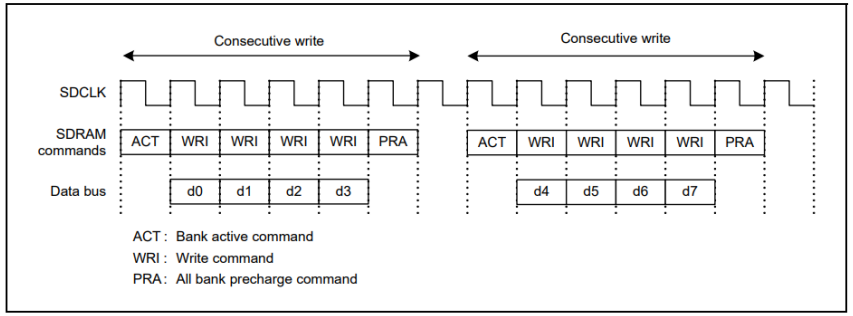


Figure 16.62 Timing Example of Consecutive Write (SDAMOD.BE = 1, when the Earliest Timing is Set)

AUTO-REFRESH CONTROL

- The specifications for Auto-Refresh control are summarized below.
  - The auto-refresh cycle can be started by setting the auto-refresh operation enable bit (RFEN) in the SDRAM auto-refresh control register (SDRFEN) to 1. Once the cycle is started, **refresh requests are generated at fixed intervals determined by the refresh counter** to start the auto-refresh cycle.
  - Since **refresh requests are not accepted during read/write access**, the auto-refresh cycle may be suspended.
  - **If an auto-refresh request is issued during consecutive access to the SDRAM**, the auto-refresh cycle starts **after bus access in response to a single transfer request from the bus master is completed**.
  - **If an SDRAM access and a refresh request are generated at the same time, the refresh request takes precedence.**
  - **A CS area access and a refresh request can be made at the same** if the SDCS#, RAS#, CAS#, WE#, and CKE signals, which are necessary for issuing the refresh command, are exclusively provided for SDRAM access.
  - **When the RFEN bit in SDRFEN is set to 1 again after the auto-refresh cycle is started, a refresh request is generated.** However, if a request is made during read/write access, a request is actually generated when access is completed.
  - **The refresh counter is halted during self-refresh operation. After recovery from self-refresh mode, the auto-refresh cycle is started and the counter value is reset thus resuming the counter operation.**
  - If an auto-refresh request occurs during DMAC transfer, **the DMAC transfer is interrupted and the auto-refresh cycle begins.**

BUS ERROR

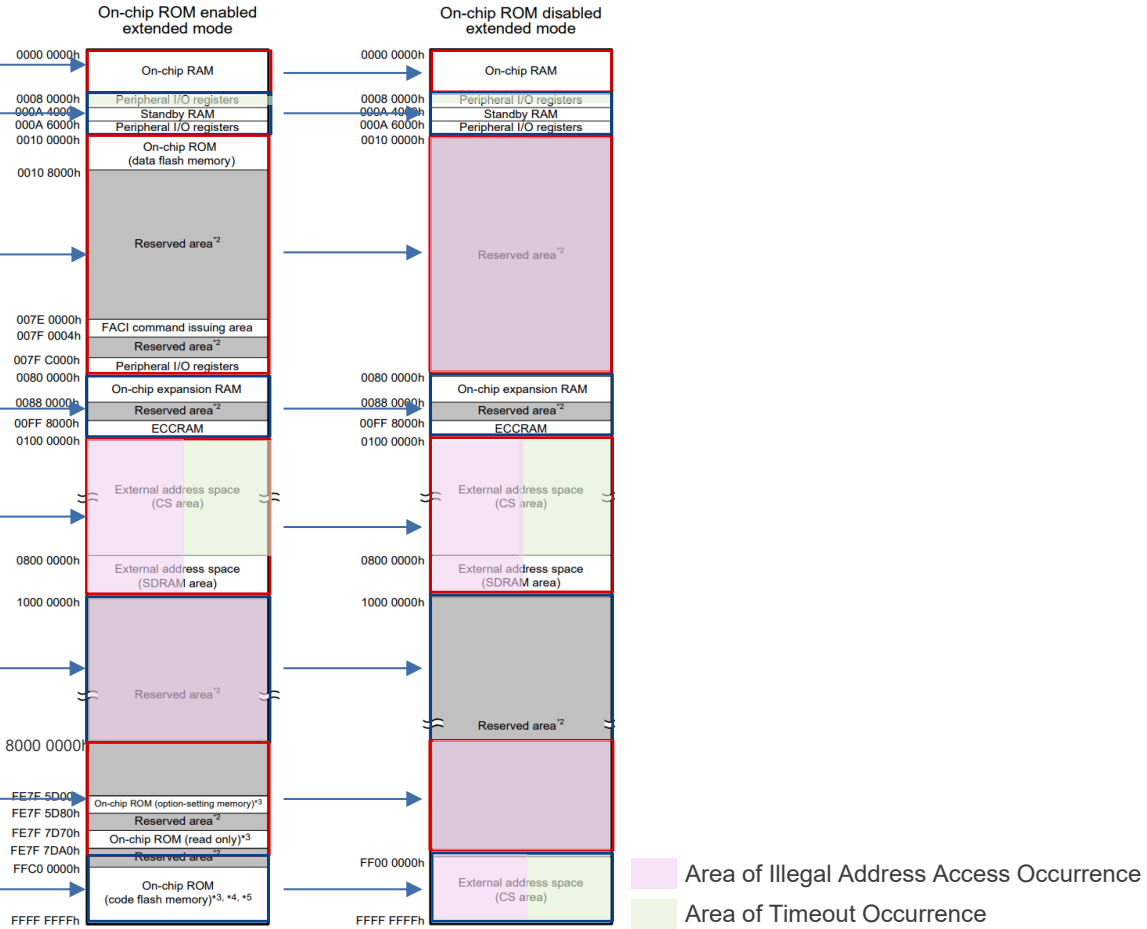
- RX can detect two types of bus errors : Illegal Address Access and Timeout.
  However, the address space that can detect bus errors is limited.
  Below is an example from the RX72M.

Table 16.21 Types of Bus Errors

Address	Type of Area		Type of Error			
			Illegal Address Access		Timeout	
	On-Chip ROM Enabled	On-Chip ROM Disabled	On-Chip ROM Enabled	On-Chip ROM Disabled	On-Chip ROM Enabled	On-Chip ROM Disabled
0000 0000h to 0007 FFFFh	Memory bus 1		—		—	
0008 0000h to 0008 7FFFh	Internal peripheral bus 1		—		—	
0008 8000h to 0009 FFFFh	Internal peripheral bus 2		Δ		—	
000A 0000h to 000B FFFFh	Internal peripheral bus 3		Δ		—	
000C 0000h to 000D FFFFh	Internal peripheral bus 4		Δ		✓	
000E 0000h to 000F FFFFh	Internal peripheral bus 5		Δ		—	
0010 0000h to 0011 FFFFh	Internal peripheral bus 6	Reserved area	—	✓	—	—
0012 0000h to 007F FFFFh			Δ	✓	—	—
0080 0000h to 00FF FFFFh	Memory bus 3		—	—	—	—
0100 0000h to 07FF FFFFh	External bus (CS1 to CS7)		[IA]		[TO]	
0800 0000h to 0FFF FFFFh	External bus (SDRAM area)		[IA]		—	
1000 0000h to 7FFF FFFFh	Reserved area		✓		—	—
8000 0000h to FEFF FFFFh	Memory bus 2	Reserved area	—	✓	—	—
FF00 0000h to FF7F FFFFh		External bus (CS0)	—	[IA]	—	[TO]
FF80 0000h to FFFF FFFFh			—		—	

—: A bus error does not result.
Δ: A bus error may or may not result.
✓: A bus error results.
[IA]: Access to this area leads to detection of a bus error if operation for this area is disabled (CSnCR.EXENB = 0; n = 0 to 7, SDCCR.EXENB = 0).
[TO]: Bus access not being completed within 768 cycles leads to detection of a bus error.
Note: The capacity of the code flash memory differ depending on the product. For details, refer to section 64, Flash Memory (FLASH).

SDRAM has no external WAIT and no access beyond 768 cycles, so there is no factor that causes timeout.



APPLICATION NOTE FOR EXTERNAL BUS USAGE

- Application note for External bus usage is as follows. For more detailed usage, please refer to the following :
  - RX Family Example of Using the External Bus R01AN6594
  - RX Family Read/Write Operations in SDRAM Using the SDRAMC R01AN5441

I/O PORT

This chapter is created with reference to RX66T. However, it can be used as a reference for all RX family products.

LIST OF PORT FUNCTIONS

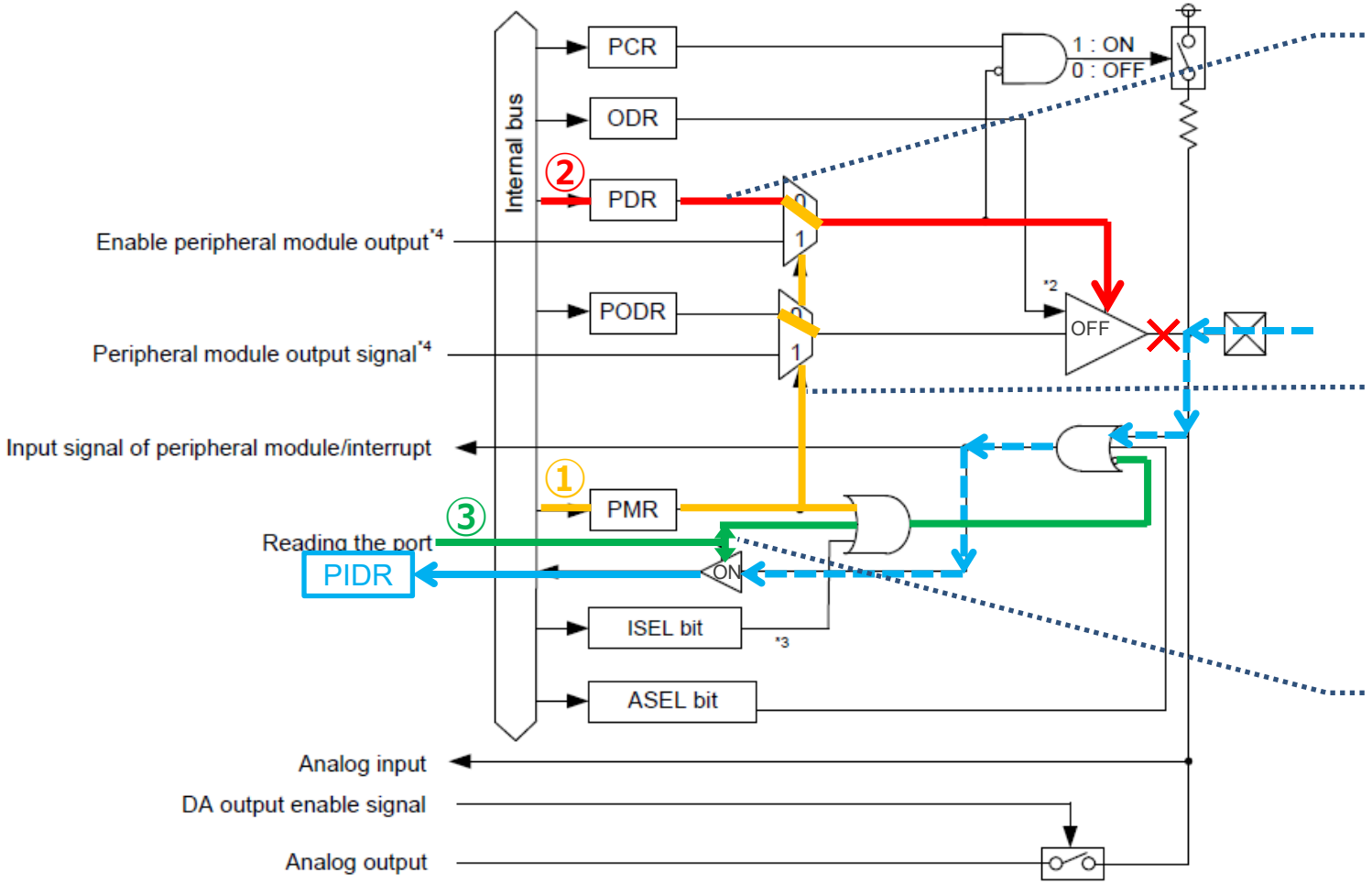
Port	Pin	Input Pull-up	Open-Drain Output	Driving Ability Switching	5-V Tolerant
PORT0	P00, P01	✓	✓	Normal drive/high drive	—
PORT1	P10, P11	✓	✓	Normal drive/high drive	—
PORT2	P20 to P24, P27	✓	✓	Normal drive/high drive	—
PORT3	P30 to P33	✓	✓	Normal drive/high drive	—
	P36, P37	✓	✓	Fixed to normal output	—
PORT4	P40 to P47	✓	✓	Fixed to normal output	—
PORT5	P50 to P55	✓	✓	Fixed to normal output	—
PORT6	P60 to P65	✓	✓	Fixed to normal output	—
PORT7	P70	✓	✓	Normal drive/high drive	—
	P71 to P76	✓	✓	Normal drive/high drive/ large current output	—
PORT8	P80, P82	✓	✓	Normal drive/high drive	—
	P81	✓	✓	Normal drive/high drive/ large current output	—
PORT9	P90 to P95	✓	✓	Normal drive/high drive/ large current output	—
	P96	✓	✓	Normal drive/high drive	—
PORTA	PA0 to PA5	✓	✓	Normal drive/high drive	—
PORTB	PB0, PB3, PB4, PB6, PB7	✓	✓	Normal drive/high drive	—
	PB1, PB2	✓	✓	Fixed to normal output	✓
	PB5	✓	✓	Normal drive/high drive/ large current output	—
PORTD	PD0 to PD2, PD4 to PD7	✓	✓	Normal drive/high drive	—
	PD3	✓	✓	Normal drive/high drive/ large current output	—
PORTE	PE0, PE1, PE3 to PE5	✓	✓	Normal drive/high drive	—
	PE2	—	—	—	—
PORTN	PN6*1	✓	✓	Normal drive/high drive	—
	PN7*2	✓	✓	Normal drive/high drive	—

Note 1. This pin is initially set to input and is pulled up.
Note 2. This pin is initially set to input and is pulled down.

The following port functions are also valid for other signals (such as serial and other peripheral functions) that share pins with general-purpose I/O ports.

- Input pull-up function
- Open-drain output function
- Drive ability switching function
- 5V tolerant setting

I/O PORT OPERATION - GENERAL I/O INPUT PORT -

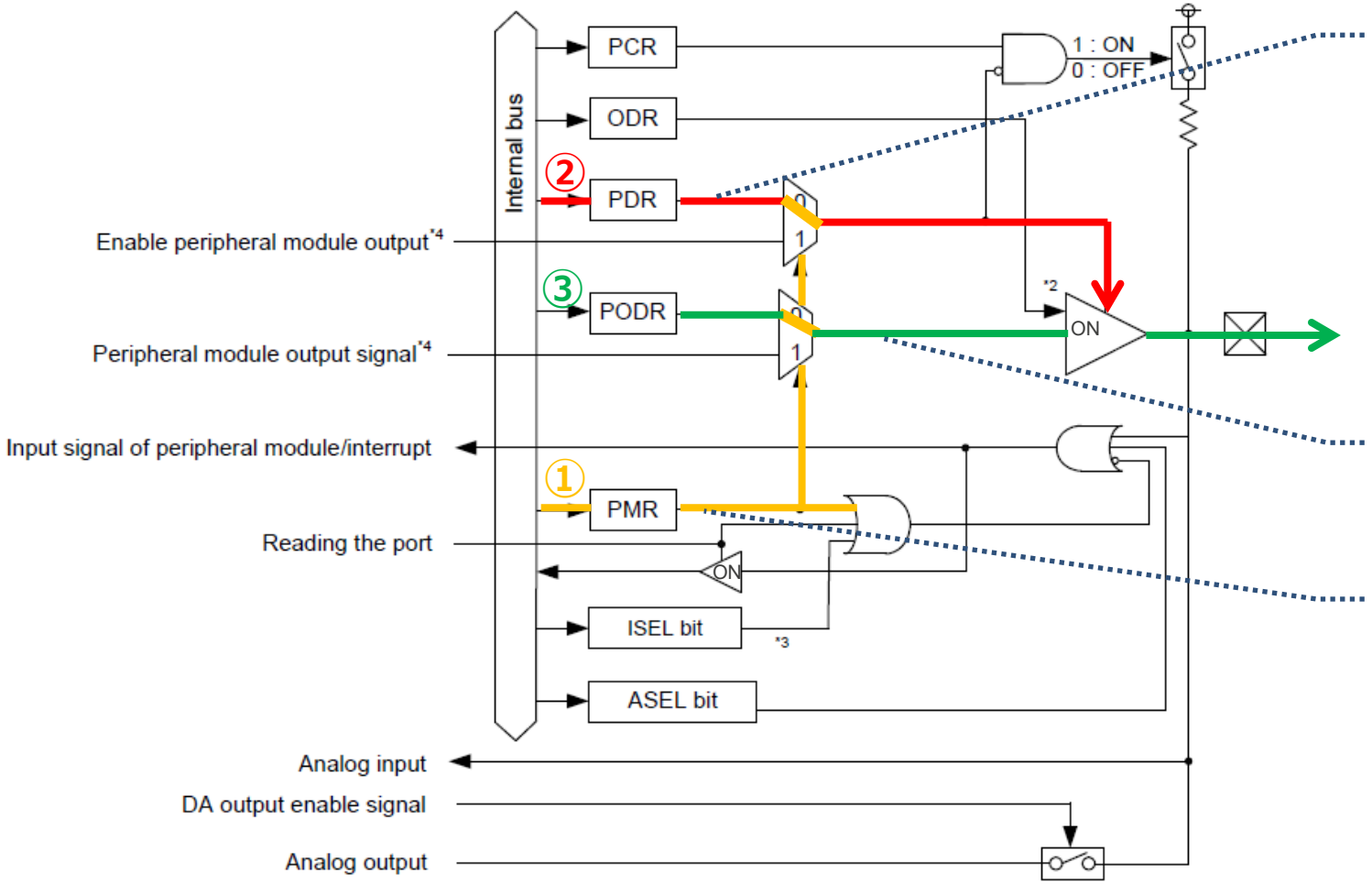


② The output buffer is turned off by setting the PDR register to the “Input” port (PORTn.PDR.Bm = “0”).

① The port is switched to a general I/O pin by setting the PMR register (PORTn.PMR.Bm = “0”).

③ When the port(PORTn.PIDR) is read, the input buffer is turned on and the High/Low of the target port is set to PIDR.

I/O PORT OPERATION - GENERAL I/O OUTPUT PORT -



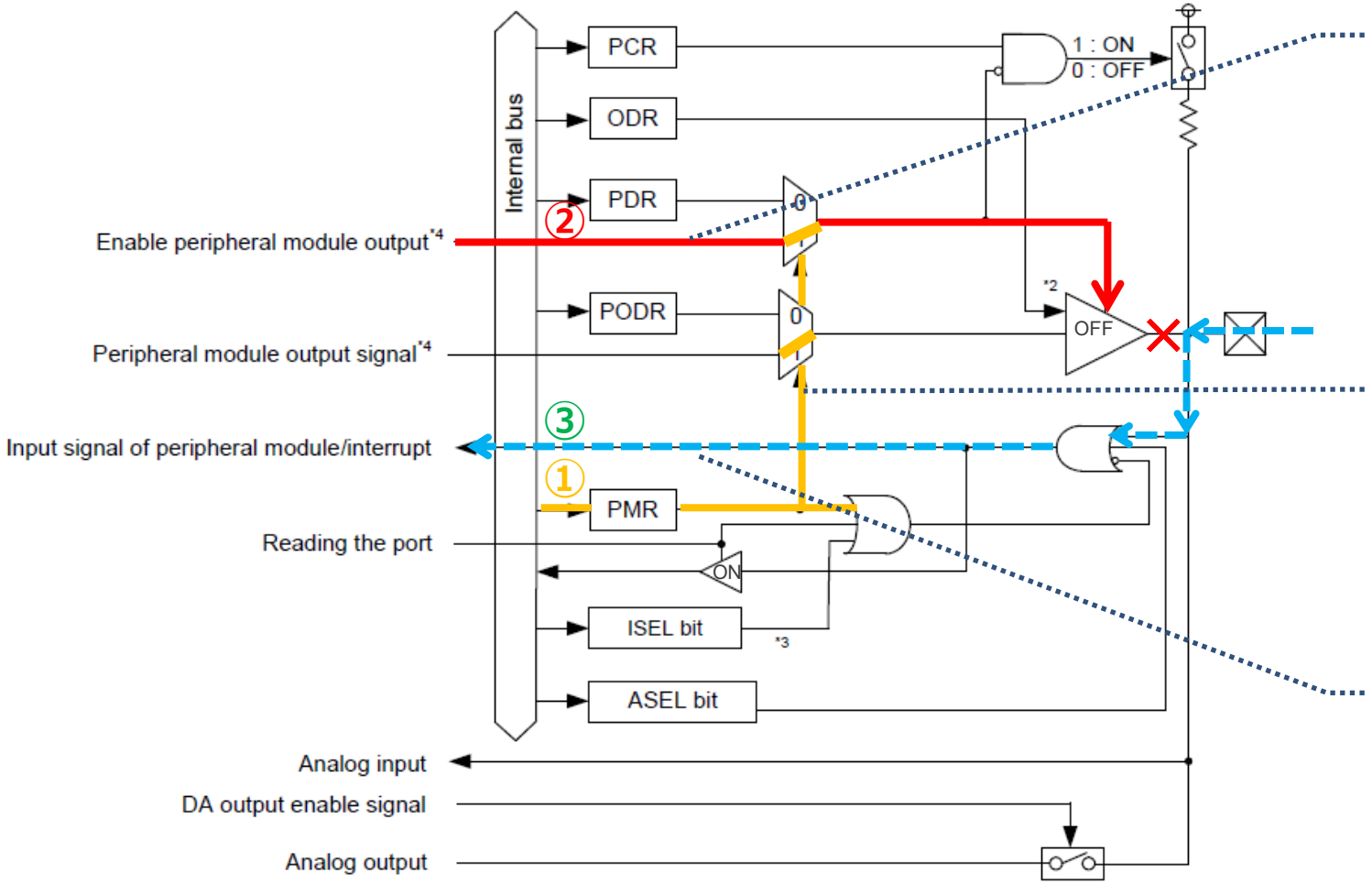
② The output buffer is turned on by setting the PDR register to the “Output” port (PORTn.PDR.Bm = “0”).

③ The value (“1” : High, “0” : Low) of the port output data register (PORTn.PODR) is output from the target port.

① The pin is switched to a general I/O pin by setting the PMR register (PORTn.PMR.Bm = “0”).

I/O PORT OPERATION - PERIPHERAL FUNCTION INPUT -

- When the pin function of the port is set to "Use the pin as peripheral function" and the peripheral function to be used is the **input** operation.



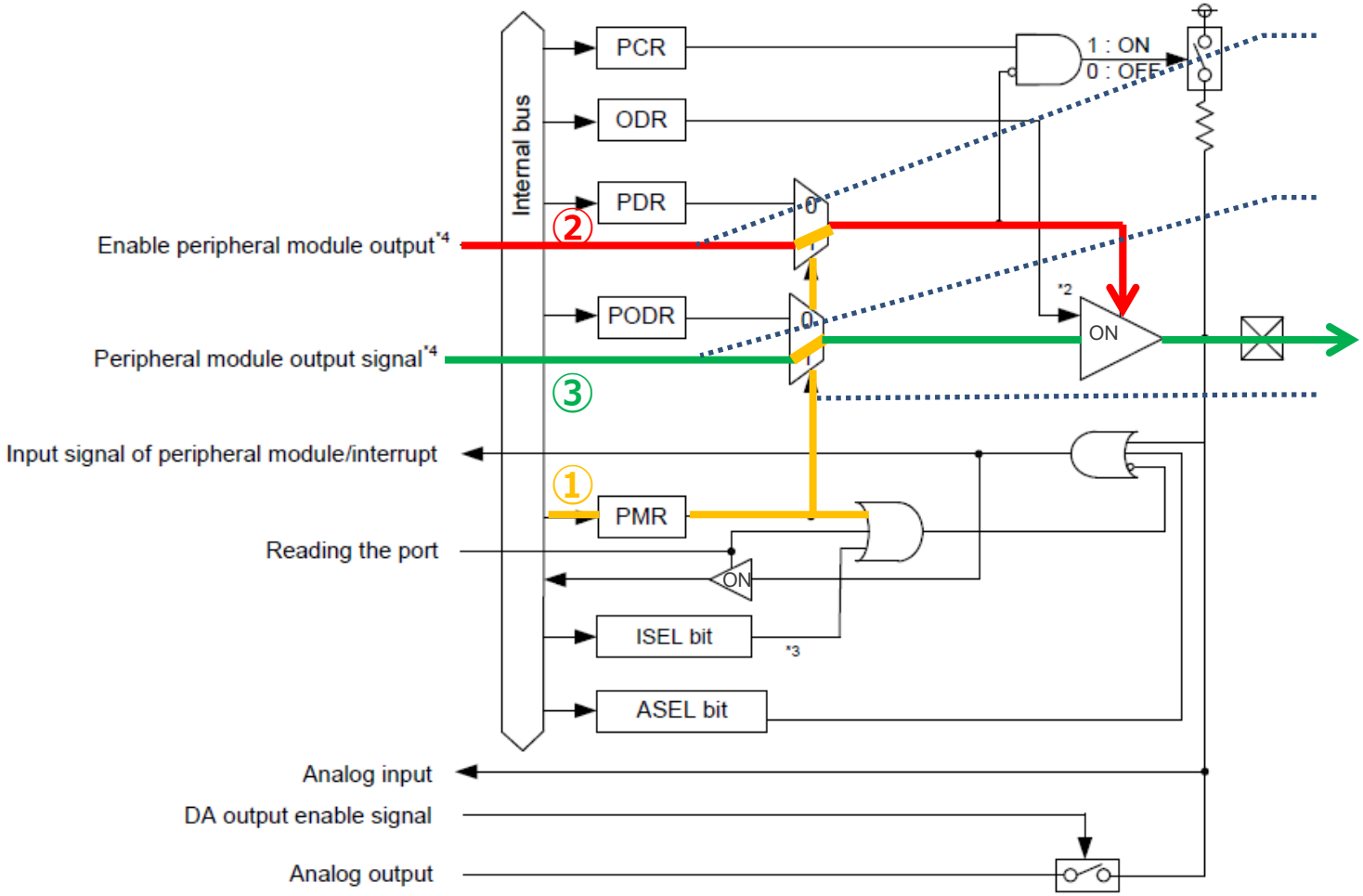
② The buffer is turned off by the output disable operation of peripheral functions.

① The pin is switched to a peripheral functions by setting the PMR register (PORTn.PMR.Bm = “1”).

③ Setting the pin function control register (PmnPFS.PSEL) to a peripheral function, High/Low of the target port is input to the peripheral function.

I/O PORT OPERATION - PERIPHERAL FUNCTION OUTPUT -

- When the pin function of the port is set to "Use the pin as peripheral function" and the peripheral function to be used is the **output** operation.

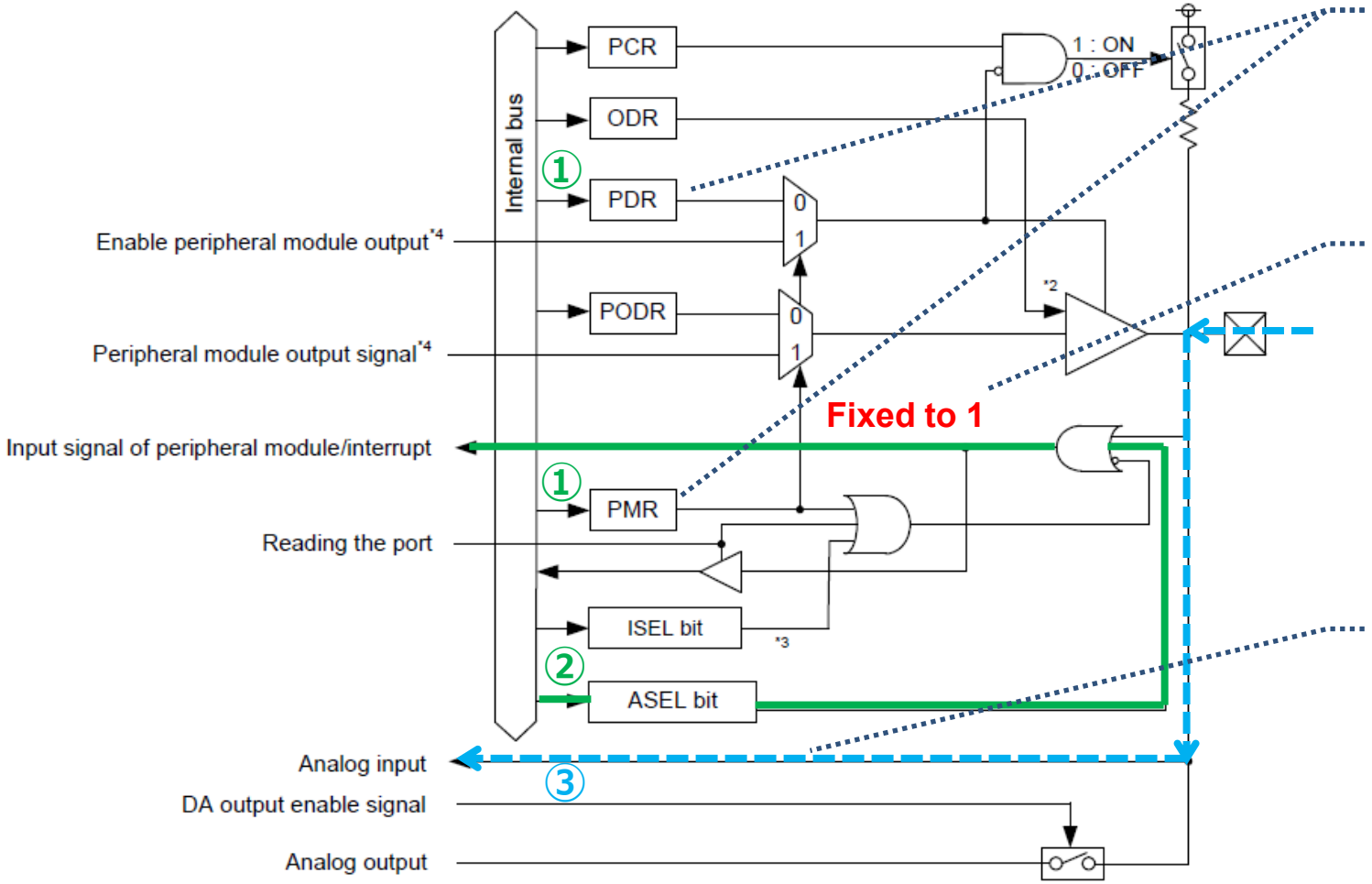


② The buffer is turned on by the output enable operation of the peripheral function.

③ Setting the pin function control register (PmnPFS.PSEL) to a peripheral function enables the port output operation from the peripheral function to output High/Low to the target port.

① By setting the PMR register to the peripheral function port (PORTn.PMR.Bm = “1”), the pin control is set to the peripheral function.

I/O PORT OPERATION - ANALOG FUNCTION INPUT -

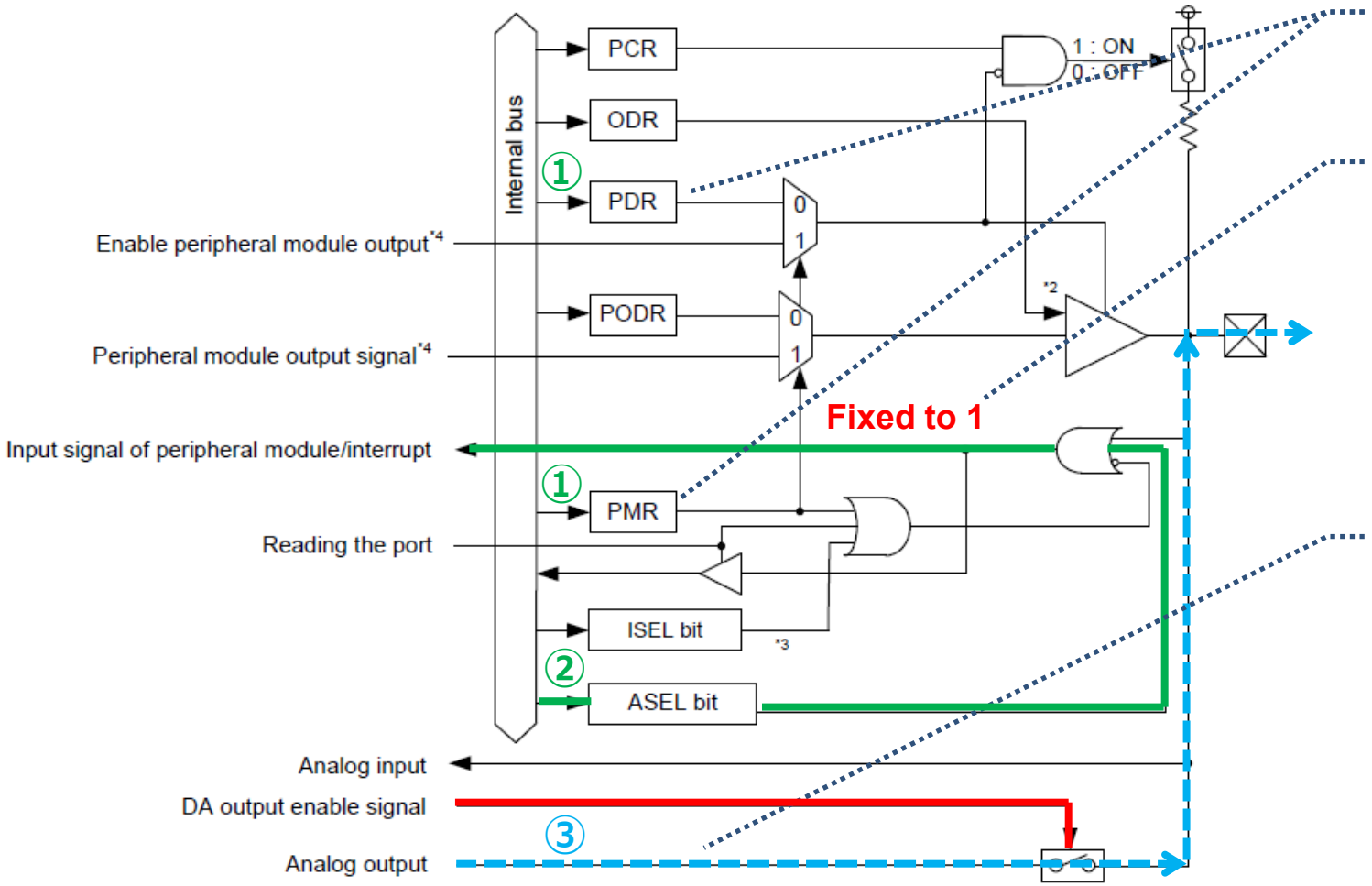


①Set the port mode register (pmr) to “0” and the port direction register (pdr) to “0” to change to a general-purpose input port.

② Setting this pin to analog port in the pin function control register (PmnPFS.ASEL) fixes the internal digital input signal to “1”.

③ Analog input is enabled by starting an analog input operation (e.g., A/D sampling operation).

I/O PORT OPERATION - ANALOG FUNCTION OUTPUT -

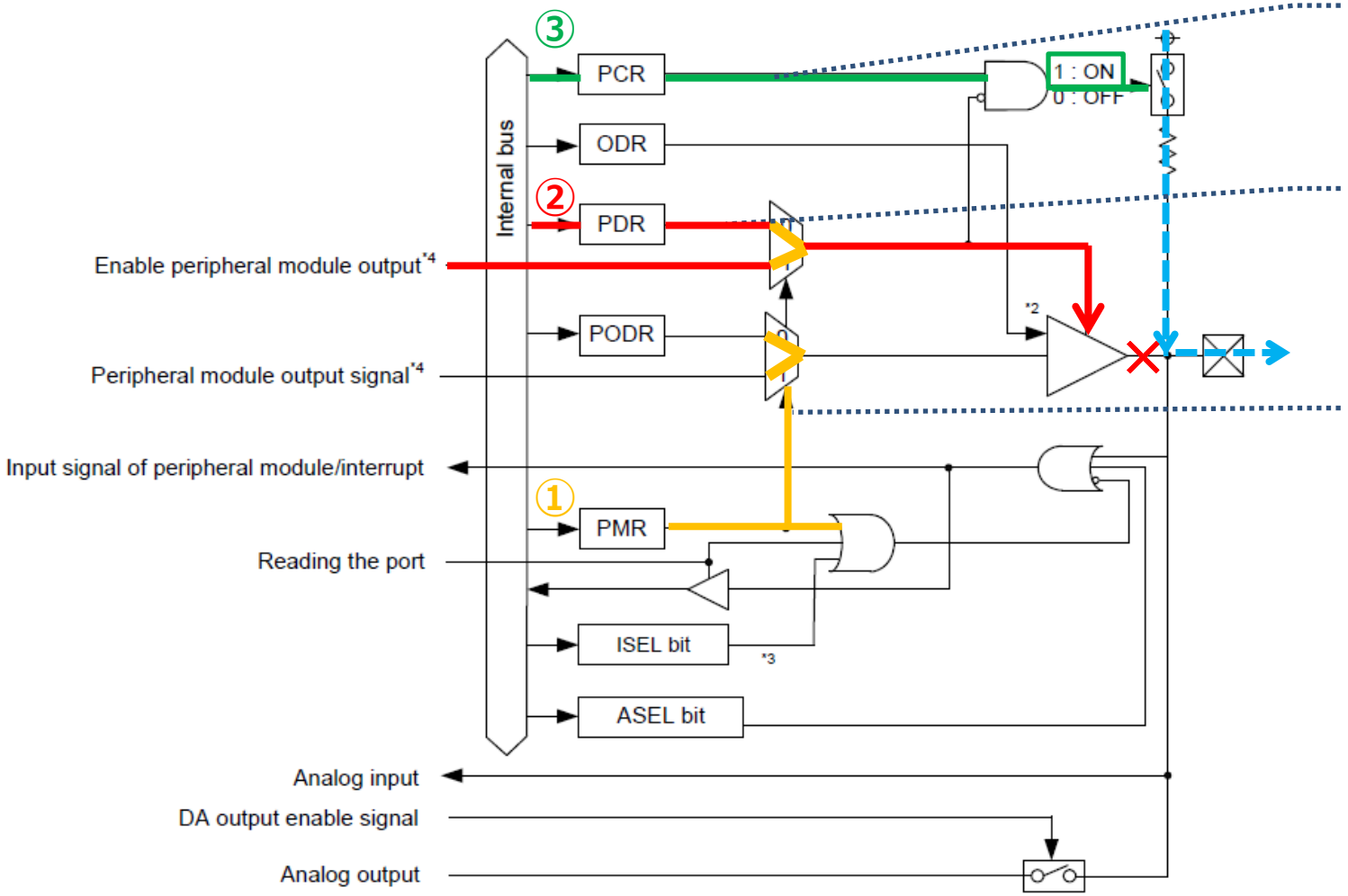


①Set the port mode register (pmr) to “0” and the port direction register (pdr) to “0” to change to a general-purpose input port.

② Setting this pin to analog port in the pin function control register (PmnPFS.ASEL) fixes the internal digital input signal to “1”.

③ Setting the D/A output destination select register (DADSELR.OUTDAn) to “1” enables the D/A signal to be output from the target port.

I/O PORT OPERATION - PULL-UP -

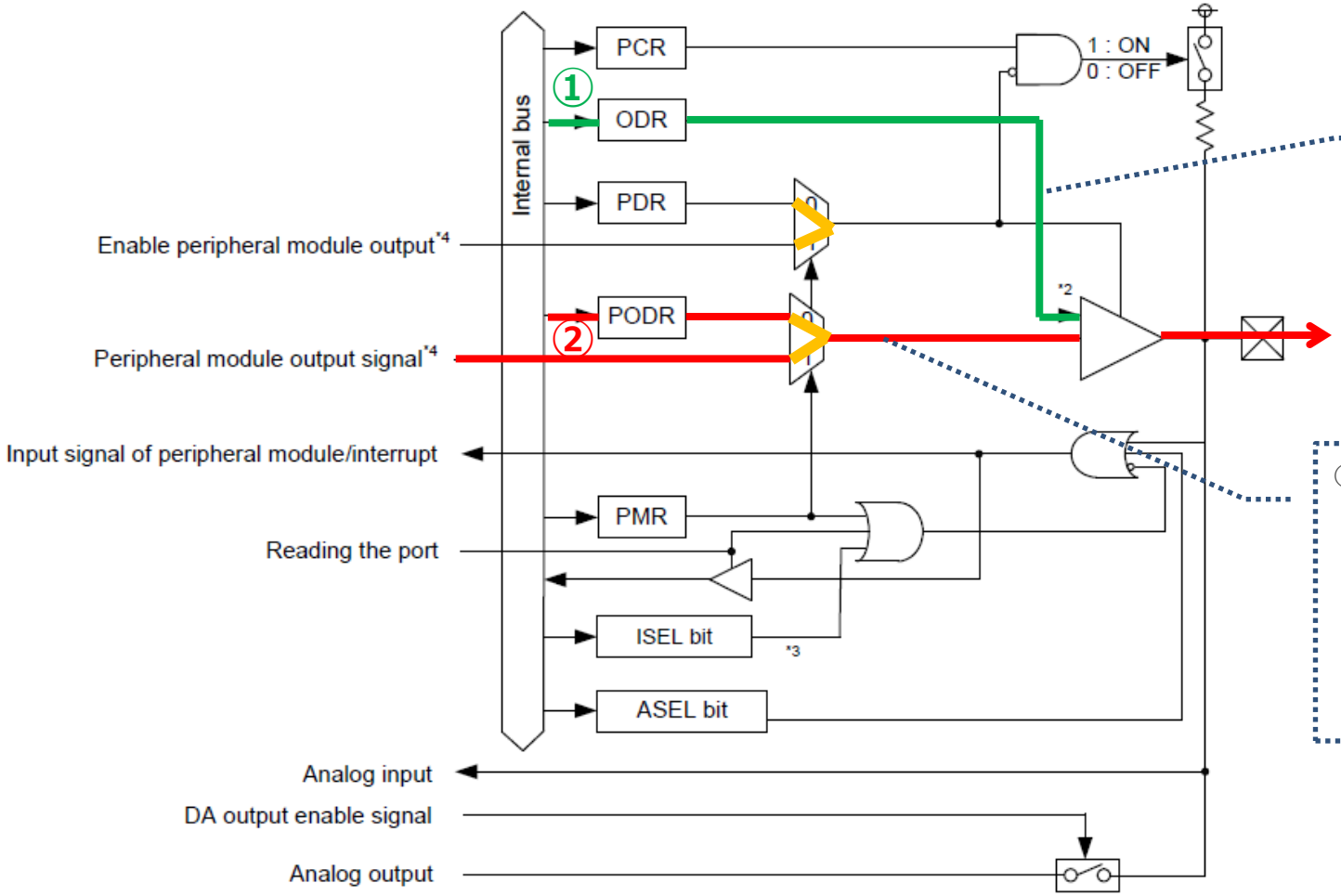


③ After setting ① and ②, pull-up is turned ON by setting the pull-up control register (PCR) to “1” (pull-up control should be turned ON after set to the input port).

② The output buffer is turned off when set to the input port by the PDR register (PORTn.PDR.Bm = “0”) or in the input operation of a peripheral function.

① Setting the PMR register to a general I/O pin or peripheral function (PORTn.PMR.Bm = “0” or “1”).

I/O PORT OPERATION - N-CHANNEL OPEN-DRAIN -



① By setting the Open-Drain Control Register (PORTn.ODRm), the pin function is set to the NMOS open-drain output.

② In N-channel open drain operation, the port output are as follows.
•High output :
N-MOS turns OFF and the pin output becomes Hi-Z.
•Low output :
N-MOS turns ON and the pin output becomes Low.
It can be used for both general-purpose output and peripheral function operation.

NOTICE FOR I/O PORT SETTINGS

- Common
  - When PORTm.PIDR is read, the pin status is read regardless of PORTm.PDR and PORTm.PMR registers.
  - Pull-up enables the input pull-up resistor of the pin corresponding to the bit with the PORTm.PCR register set to “1” when the pin is in the input state. The pull-up resistor is disabled during a reset.
  - The input-pull-up function, open-drain output function, drive capability switching function, and 5V tolerant setting are also valid for other signals that share a port with the general-purpose I/O port.

- Interrupt pins
  - The ISEL bit is set when used as an IRQ input pin (external pin interrupt). It can also be used in combination with peripheral functions. However, it is prohibited to enable IRQn of the same number by two or more pins.

- Analog port
  - When setting the pins as analog pins with ASEL bit, set the relevant bit of the port mode register (PORTm.PMR) and the relevant bit of the port direction register (PORTm.PDR) to “0” to set the relevant pins as general-purpose inputs, set PmnPFS.ASEL bit to “1”. The pin status cannot be read at this time.

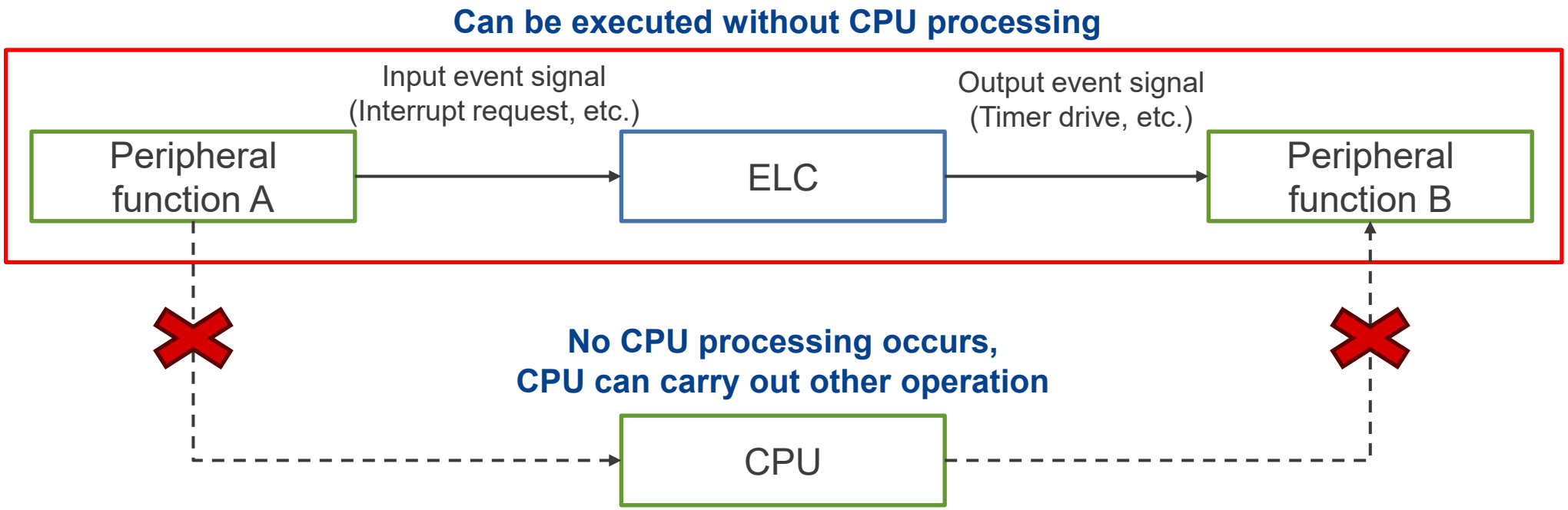
- Peripheral function port
  - The PmnPFS.PSEL[5:0] bit should be changed with the PMR.Bn bit set to “0”.

- Other
  - For ports to which RIIC and RI3C are assigned, set the PCR.Bn bit to “0”.
    (Pull-up is automatically turned off for peripheral function outputs other than RIIC and RI3C.)

EVENT LINK CONTROLLER(ELC)

ELC OVERVIEW

Event Link Controller (ELC) is a function that triggers an event signal from a peripheral function to activate another peripheral function without involving the CPU. It is possible to trigger various other functions in conjunction with an event from one module.



COMPARING ELC AND CPU INTERRUPTS

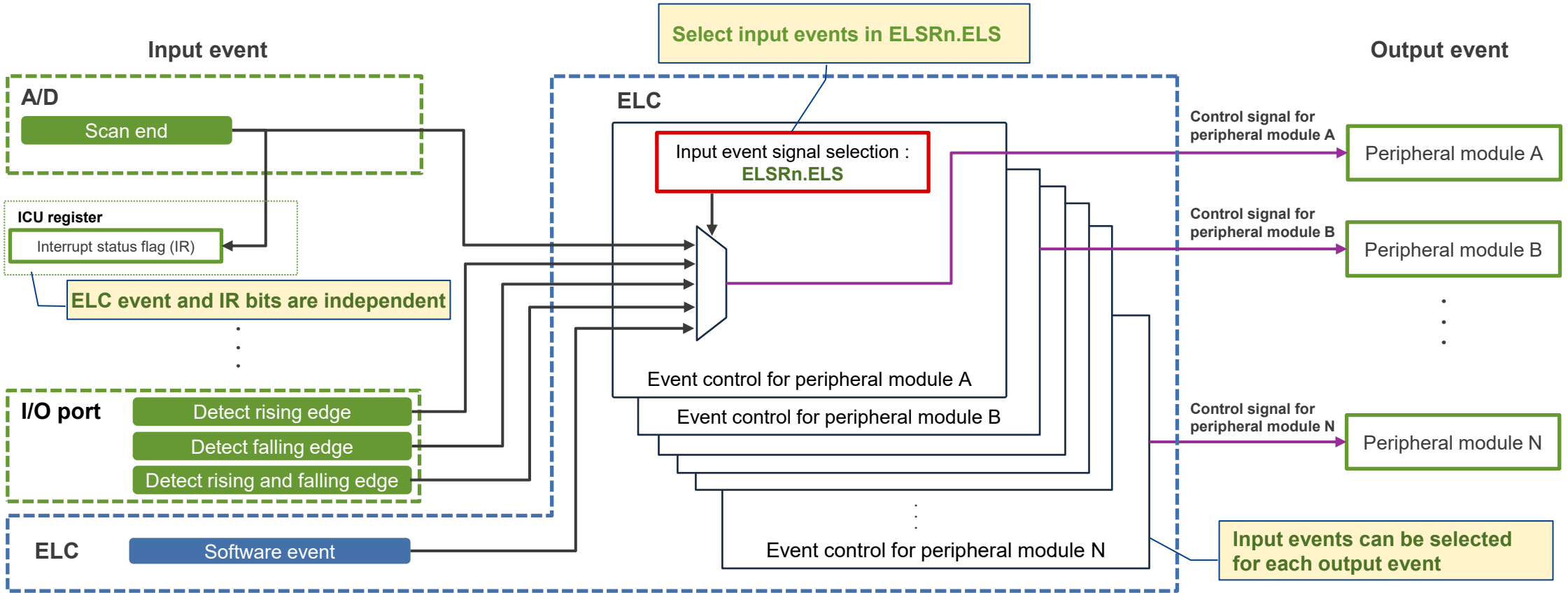
The following compares the case where the event link controller is used with the case where CPU interrupt is used.

Blue : Benefit

Item	ELC	CPU interrupt
CPU usage	**No**	Yes
Number of simultaneous output events for one input event	**Multiple**	Multiple (However, executed one by one depending on the instruction)
Number of output events executed when multiple input events occur	**Multiple output events can be executed in parallel**	Sequentially executed, one by one, in order of priority

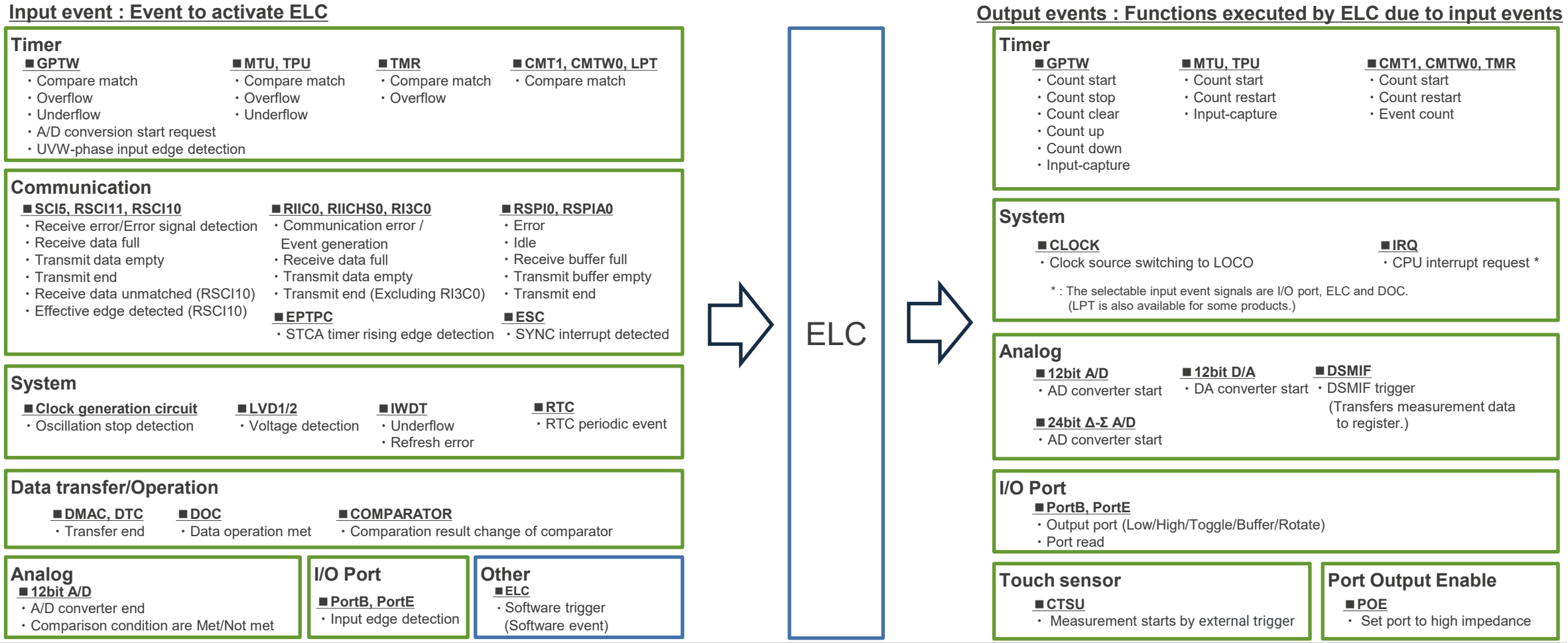
HOW TO CONNECT ELC INPUT AND OUTPUT EVENT

The figure below shows the structure of the Event Link Controller.
ELC is controlled by one control register (ELSRn) for each output event. Multiple input events can be set for one output event.
Input events and interrupt status flags (IR bits) are independent and have no effect on CPU interrupts.



SUMMARY OF ELC INPUT AND OUTPUT EVENTS

Summary of input and output events (The following is an example. Please refer to the “ELC” chapter of each RX product for details).



RELATED REGISTER WHEN USING ELC

ELC is controlled by the ELCR and ELSRn registers. When using the events in the blue frame, the other registers of ELC also need to be set. When using the functions in the red frame, the register settings within each function are also required (Please refer to the next page.).

Input event : Event to activate ELC

Timer

■GPTW
- Compare match
- Overflow
- Underflow
- A/D conversion start request
- UVW-phase input edge detected

■MTU, TPU
- Compare match
- Overflow
- Underflow

■TMR
- Compare match
- Overflow

■CMT1, CMTW0, LPT
- Compare match

Communication

■SCI5, RSCI11, RSCI10
- Receive error/Error signal detection
- Receive data full
- Transmit data empty
- Transmit end
- Receive data unmatched (RSCI10)
- Effective edge detected (RSCI10)

■RIIC0, RIICHS0, RI3C0
- Communication error / Event generation
- Receive data full
- Transmit data empty
- Transmit end (Excluding RI3C0)

■RSPI0, RSPIA0
- Error
- Idle
- Receive buffer full
- Transmit buffer empty
- Transmit end

■EPTPC
- STCA timer rising edge detection

■ESC
- SYNC interrupt detected

System

■Clock generation circuit
- Oscillation stop detection

■LVD1/2
- Voltage detection

■IWDT
- Underflow
- Refresh error

■RTC
- RTC periodic event

Data transfer/Operation

■DMAC, DTC
- Transfer end

■DOC
- Data operation met

■COMPARATOR
- Comparation result change of comparator

Analog

■12bit A/D
- A/D converter end
- Comparison condition are Met/Not met

I/O Port

■PortB, PortE
- Input edge detection

Other

■ELC
- Software trigger (Software event)

ELC Register

- **ELCR** : ELC Enable/Disable setting
- **ELSRn** : Input event settings
- **ELOPn** : Timer output event setting
- **PGR/PDFBn/PGCn/PELm** : I/O port-related I/O event settings
- **ELSEGR** : Software Event Settings

Output events : Functions executed by ELC due to input events

Timer

■GPTW
- Count start
- Count stop
- Count clear
- Count up
- Count down
- Input-capture

■MTU, TPU
- Count start
- Count restart
- Input-capture

■CMT1, CMTW0, TMR
- Count start
- Count restart
- Event count

System

■CLOCK
- Clock source switching to LOCO

■IRQ
- CPU interrupt request *

* : The selectable input event signals are I/O port, ELC and DOC. (LPT is also available for some products.)

Analog

■12bit A/D
- AD converter start

■24bit Δ-Σ A/D
- AD converter start

■12bit D/A
- DA converter start

■DSMIF
- DSMIF trigger (Transfers measurement data to register.)

I/O Port

■PortB, PortE
- Output port (Low/High/Toggle/Buffer/Rotate)
- Port read

Touch sensor

■CTSU
- Measurement starts by external trigger

Port Output Enable

■POE
- Set port to high impedance

RELATED REGISTER WHEN USING ELC

The following figure shows the registers that need to be set on the function side other than ELC.

Input event

Timer

■ **GPTW**

・A/D converter start request
GTINTAD.ADTRnUEN :
Set the conditions of event generation (Up count compare match)
GTINTAD.ADTRnDEN :
Set the conditions of event generation (Down count compare match)

Communication

■ **ESC**

・SYNCn signal
ESCICR.SYNCnC : Set the edge condition (Up count/Down count)

Output event

Timer

■ **GPTW**

・Count start
GTSSR : Count start factor setting
・Count stop
GTPSR : Count stop factor setting
・Count clear
GTCSR : Count clear factor setting
・Count up
GTUPSR : Count up factor setting
・Count down
GTDNSR : Count down factor setting
・Capture operation
GTICnSR : Input capture factor setting

System

■ **Clock**

・Clock source switching LOCO (Low-speed on-chip oscillator)
RSTCKCR.RSTCKEN : Set bit to 0 when using this event

Analog

■ **12bit A/D**

・A/D converter start
ADSTRGR.TRSAn/**ADGCTRGR.TRSC** :
A/D conversion start trigger selection

■ **24bit Δ-Σ A/D**

・A/D converter start
MR.TRGMD :
A/D conversion start trigger selection

Touch sensor

■ **CTSU**

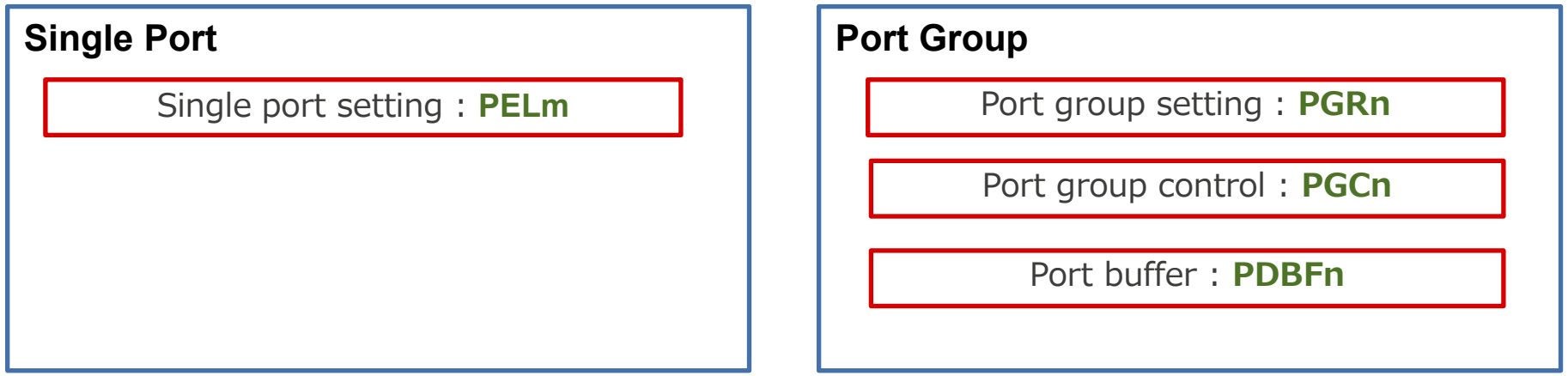
・Measurement starts by external trigger
CTSUCR0.CTSUCAP : Measurement start trigger selection

HOW TO SET ELC
I/O PORT : SINGLE PORT AND PORT GROUP

There are two types of input/output events using I/O port : “Single Port" and “Port Group".

- Single Port : This function connects I/O port pins and events on a one-to-one basis.
- Port Group: This is a function to connect multiple ports of an I/O port as a single group to an event.

Note that the registers set by "Single Port" and "Port Group" differ as follows.



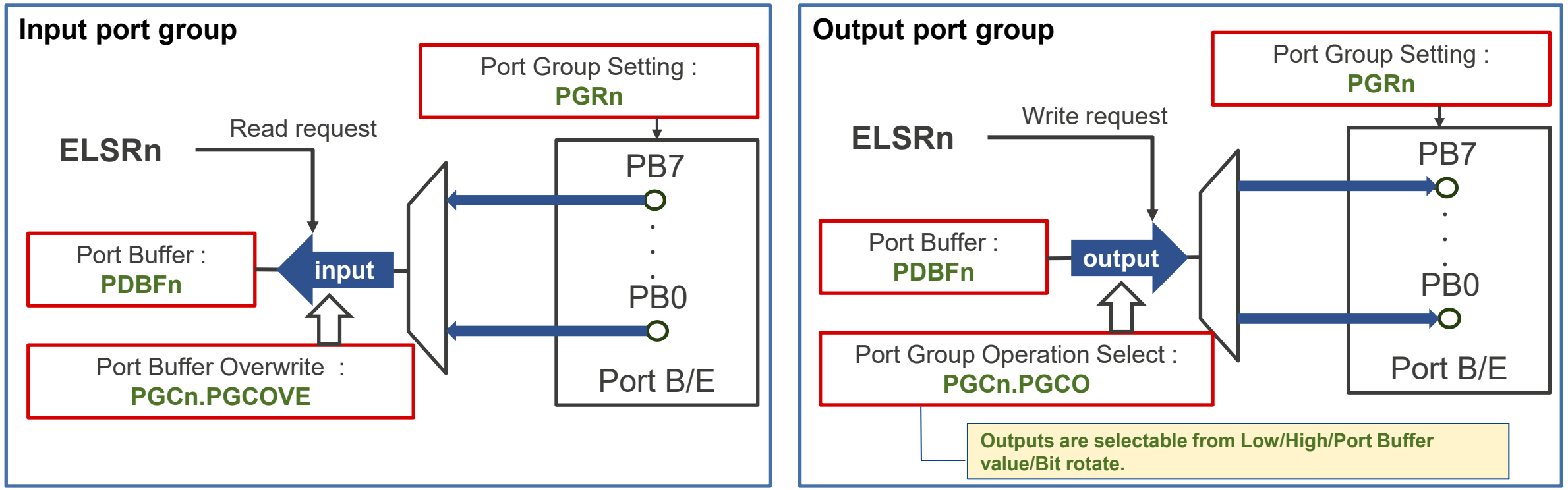
HOW TO SET ELC

I/O PORT : PORT GROUP OUTPUT EVENTS

There are two types of output events for a port group :

- Input port group : When an event is input, the port status(High/Low) of the port selected by PGRn is transferred to the port buffer (PDBFn).
  (Select whether to store only once by PGCn.PGCOVE or overwrite it and store it many times.)
- Output port group : Output from the pin selected by PGRn as specified by PGCn.PGCO when an event is input.
  (Use PGCn.PGCO to select the output mode.)

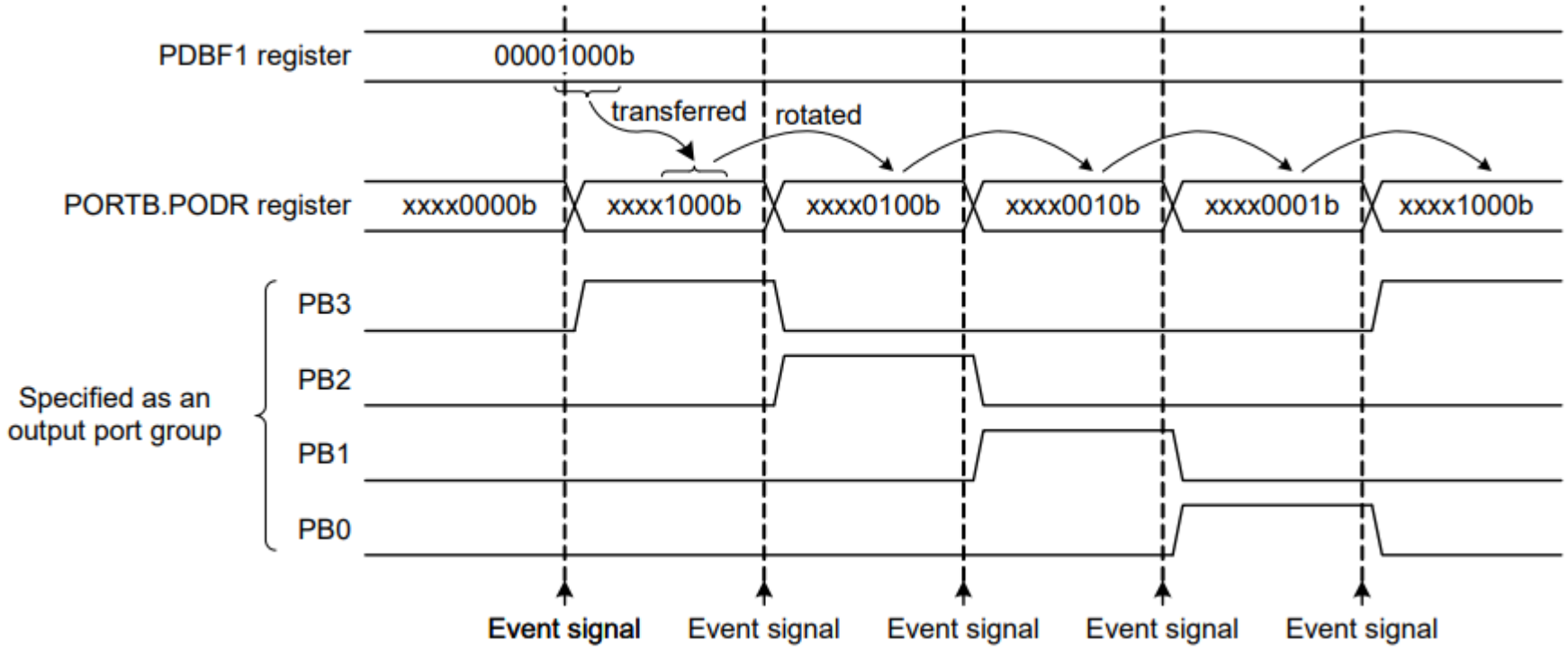
: Registers that need to be set



HOW TO SET ELC

SUPPLEMENTAL INFORMATION) WHAT IS BIT ROTATE OUTPUT?

- When an output event is used in a port group, the value set in PDBFn register is rotated to MSB→LSB for each input event, and the value is output from each port. For example, if PDBFn register is set to 00001000b as shown below, High output-port can be switched for each event-input.



SUPPLEMENTARY INFORMATION & NOTES FOR ELC SETTINGS

■ Supplementary information

- ELC can operate even when CPU is stopped, such as in sleep mode.
  However, please note that it will not operate in a mode where the clock supply to the ELC and the target peripheral module is stopped, as described in the "Usage Notes" section of the ELC chapter.
- Input events used for ELC can also be used as transferring source for DMAC and DTC.

■ Precautions

- Be sure to read the precautions for using ELC in the section "Usage Notes" in ELC chapter.
- Be sure to read the "Usage Notes" for each peripheral modules to be used.
- When ICU interrupts are used as output events, there are limitations on the input events that can be used.
  For details, refer to the hardware manual.

USE CASE

ELC can be used for a variety of purposes, the following is the example :

① : Execute peripheral function without CPU operation

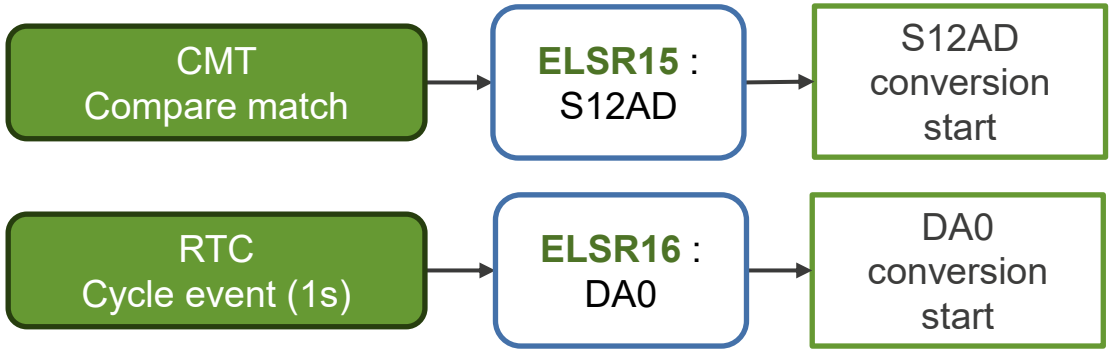
② : Execute several different peripherals with one event signal

③ : Execute sequentially of peripherals by connecting them together

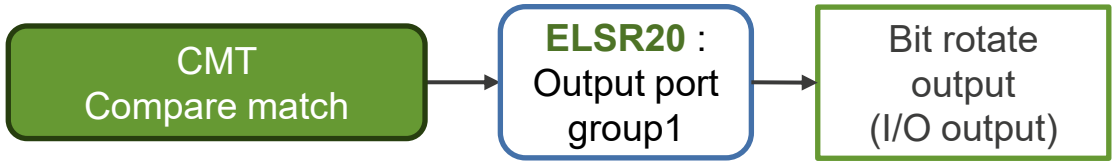
This chapter is written with reference to RX72M, however, can be used as a reference for all products.

USE CASE ① :
EXECUTE PERIPHERAL FUNCTION WITHOUT CPU OPERATION

- The following shows an example in which a peripheral module is executed at a fixed period without CPU operation by utilizing ELC.



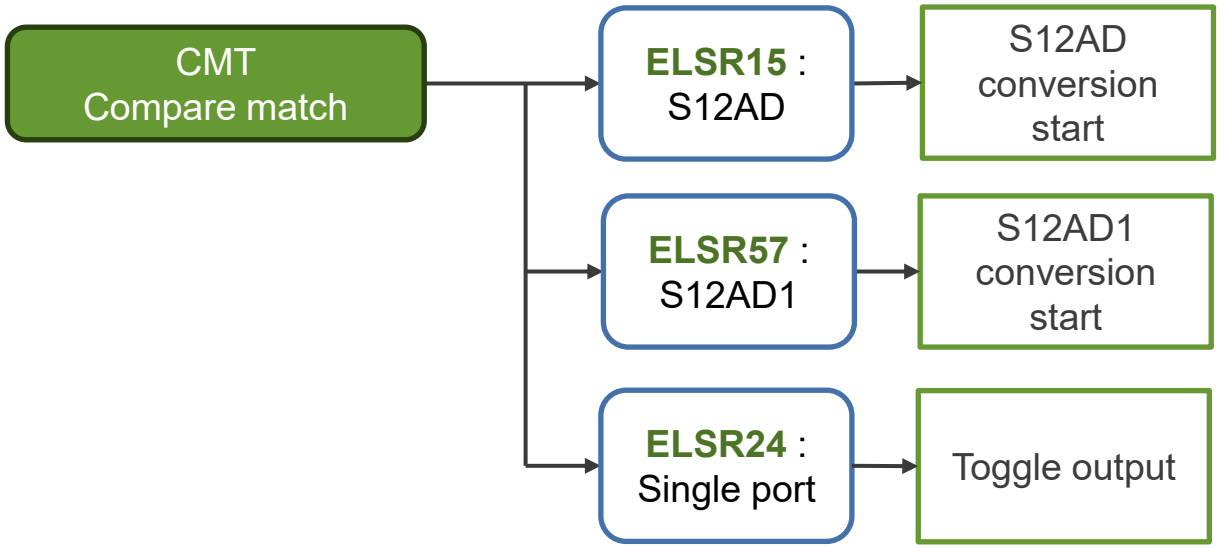
Example 1 : Starting 12 bits A/D or D/A converter by compare match timer or RTC cycle event



Example 2 : Bit Rotate Output at Period
(The value set in PDBF register is rotation from MSB to LSB.)

USE CASE ② :
EXECUTE SEVERAL DIFFERENT PERIPHERALS WITH ONE EVENT SIGNAL

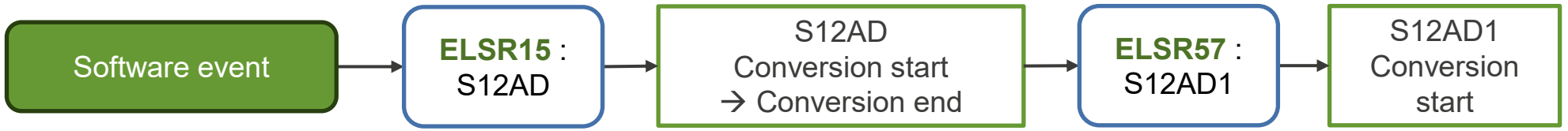
- The following shows how to use ELC to activate several peripheral modules with a single input-event signal.



e.g. : Simultaneously operating S12AD, S12AD1 and I/O outputting (toggling) at CMT compare match

USE CASE ③ :
EXECUTE SEQUENTIALLY OF PERIPHERALS BY CONNECTING THEM TOGETHER

- The following shows an example of starting several peripheral modules in succession with a single input-event signal by utilizing ELC.



e.g. : Conversion of S12AD is started by software event, conversion of S12AD1 is started by S12AD conversion end as input event.

ADVANCED EXAMPLE

Detailed examples are introduced on the following.

① : Operate SCI data transfer and re-set after timeout without using the CPU

② : Key-scanning without CPU operation

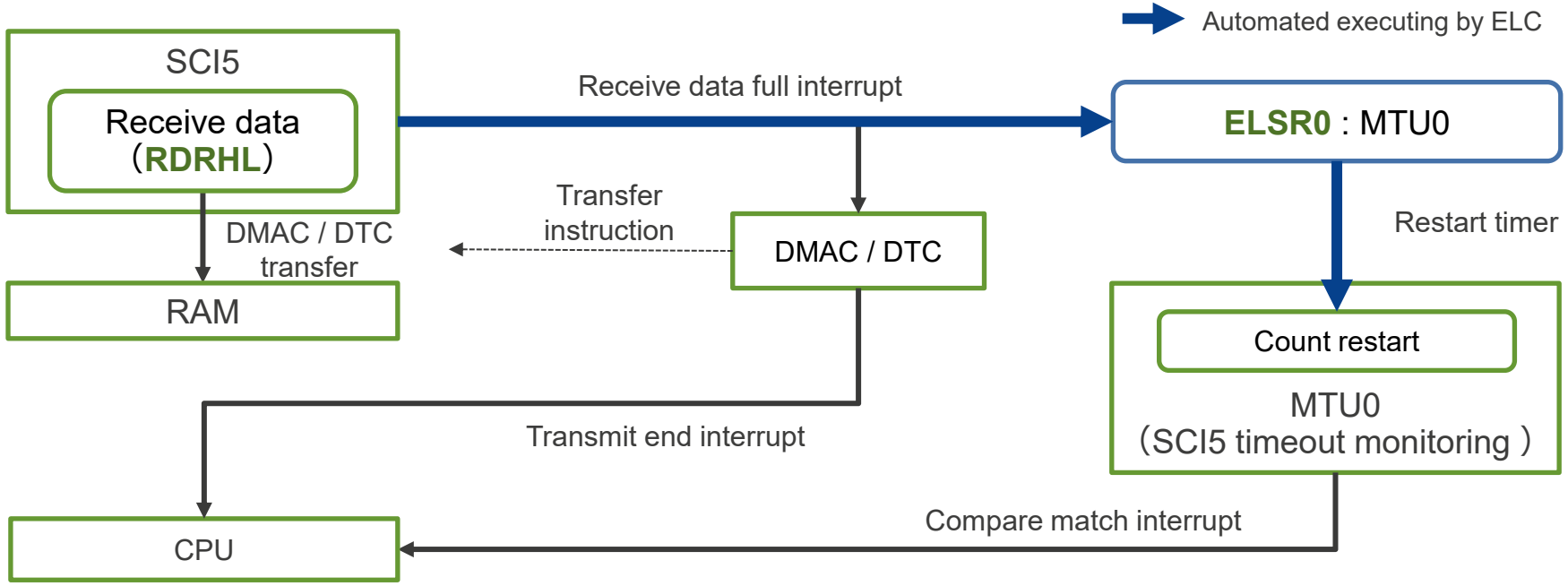
③ : Periodically AD converting & save the conversion result in RAM without CPU operation

This chapter is written with reference to RX72M, however, can be used as a reference for all products.

ADVANCED EXAMPLE ① :

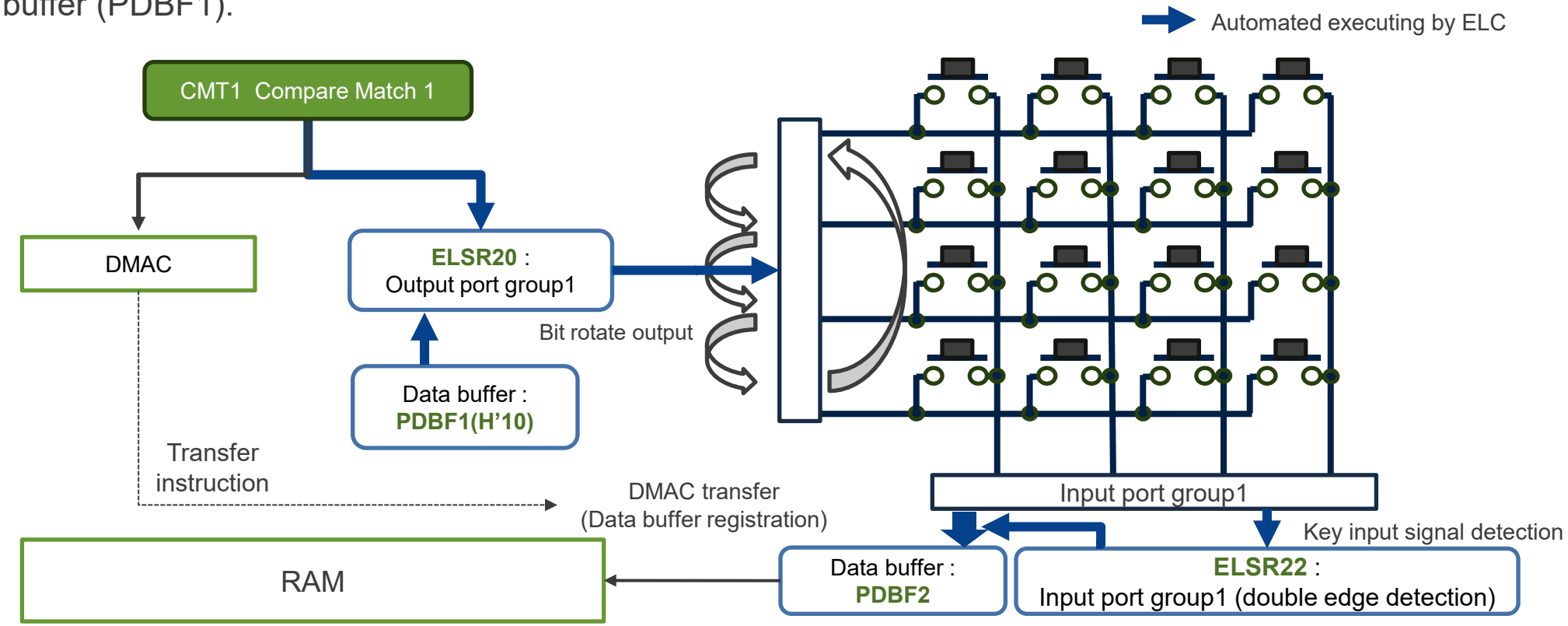
PERFORM SCI DATA TRANSFER AND RESET AFTER TIMEOUT WITHOUT USING THE CPU

When SCI reception is required at regular intervals, timeout detection is automatically performed using MTU after SCI reception until the next reception is completed. There is no operation in CPU in this example.
SCI receive data full interrupt is used as ELC input event, and ELC restarts MTU count. MTU is set to perform a compare match according to the reception interval. If “Restart” is not in time, CPU is notified that a compare match interrupt occurred and timed out. In addition, the received data is transferred to RAM by DMAC/DTC at the same time with the received data full interrupt as the transfer factor.



ADVANCED EXAMPLE ② :
KEY-SCANNING WITHOUT CPU OPERATION

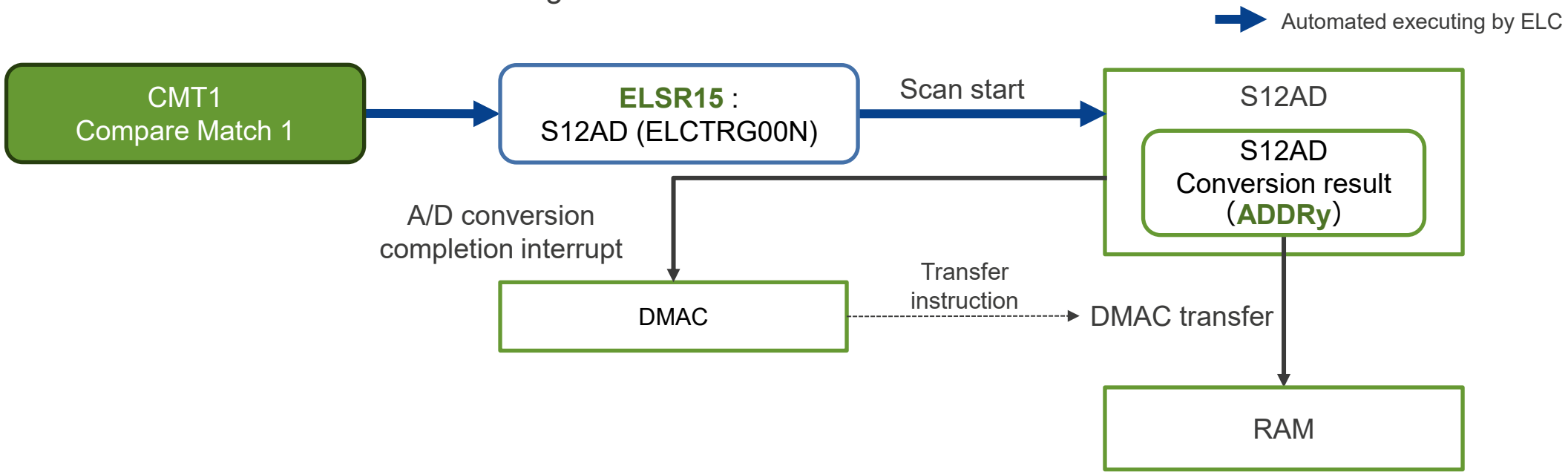
This is an example of implementing the key scan function. When an input-event CMT compare match occurs, ELC is activated, the output port group is rotated, and the rows are enabled in order. In addition, DMAC transfers data from the data buffer (PDBF2) to RAM using this input event as the transfer factor. In the operation of the column side, the input port group pins change by pressing the button, and an input event occurs. This activates ELC and stores the values of the input port group pins in the data buffer (PDBF1).



ADVANCED EXAMPLE ③ :

PERIODICALLY AD CONVERTING & SAVE THE CONVERSION RESULT IN RAM WITHOUT CPU OPERATION

By using a combination of ELC and DMAC, it is possible to periodically perform from A/D converting process to change data storing without going through CPU. When a compare match occurs on CMT, ELC starts A/D conversion. Also, A/D conversion result is saved to RAM by DMAC using A/D conversion end interrupt as the transfer factor. By setting DMAC to the free-run mode, you do not need to set the transfer count again.



LIST OF ELC APPLICATION NOTES

Please refer to the following for more detailed ELC usage.

- RX200 Series Using the ELC on the RX210 R01AN1099
- RX Family Tamper Detection Method Utilizing Existing Peripheral Functions R01AN7654

MTU

TIMER FUNCTION COMPARISON

- The following is a comparison of each timer function in RX family.

* Varies depending on the product

Operation mode	GPT/GPTW	MTU3	TPU	TMR	CMT	CMTW
Bits size	16/32	16	16	8	16	32
Maximum number of channels*	10	9	6	4	4	2
Maximum number of PWM outputs*	20	14	15	4	-	2
Maximum operating clock	Equivalent to ICLK	Equivalent to ICLK	PCLK	PCLK	PCLK	PCLK
PWM output	✓	✓	✓	✓	-	✓
Complementary PWM output	✓	✓	-	-	-	-
Positive-phase/negative-phase independently compare operation	✓	-	-	-	-	-
Input capture	✓	✓	✓	-	-	✓
Phase counting mode	✓	✓	✓	-	-	-
Output compare	✓	✓	✓	✓	-	✓
External clock count	✓	✓	✓	✓	-	-
Free-running operation	✓	✓	✓	✓	✓	✓
Compare match operation	✓	✓	✓	✓	✓	✓
High resolution output control	✓	-	-	-	-	-

MTU FUNCTION LIST

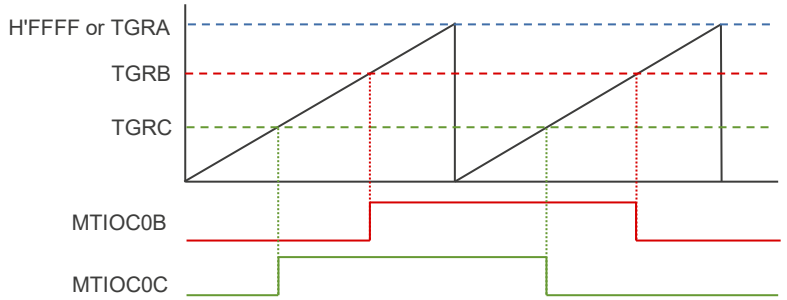
- The following table lists the operation modes held by MTU, registers supporting buffer operation, and optional operations.

Operation mode	Registers for buffer operation (MTU0,3,4,6,7,9)	Optional operation	Channel
Normal mode Output compare Input capture	TGR (Output compare/ Input capture)	Synchronous operation Noise Filter A/D conversion start request / A/D conversion Delaying	MTU0, 1, 2, 3, 4, 6, 7, 9 * A/D conversion start request is only available for MTU4, 7
PWM 1/2 mode Output compare Input capture	TGR (Output compare/ Input capture)	Synchronous operation Noise Filter A/D conversion start request / A/D conversion Delaying	MTU0, 1, 2, 3, 4, 6, 7, 9 * A/D conversion start request is only available for MTU4, 7
Reset-Synchronized PWM Mode	TGR (Output compare) TOCR2 (Output control)	Synchronous operation Noise Filter A/D conversion start request / A/D conversion Delaying AC synchronous motor drive mode	MTU3, 4, 6, 7 * A/D conversion start request is only available for MTU4, 7 * AC synchronous motor drive mode is only available for MTU3, 4
Complementary PWM Mode	TGR (Output compare/ Input capture) *Support for double buffer TOCR2 (Output control) TCDR (Cycle register)	Synchronous operation Noise Filter A/D conversion start request / A/D conversion Delaying AC synchronous motor drive mode Interrupt skipping	MTU3, 4, 6, 7 * A/D conversion start request is only available for MTU4, 7 * AC synchronous motor drive mode is only available for MTU3, 4
Phase Counting Mode	TGR (Output compare/ Input capture)	Noise Filter	MTU1, 2
Phase Counting Mode (Cascade Connection)	TGR (Output compare/ Input capture)	Noise Filter	MTU1, 2

OUTPUT WAVEFORM
NORMAL MODE, PWM MODE 1/2

Normal mode

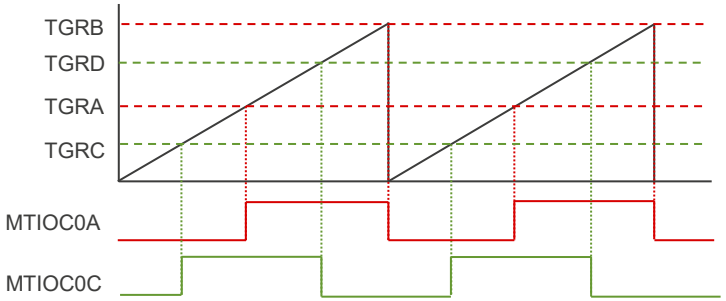
- TCNT clear condition : H'FFFF or TGR register
- PWM output waveform : Output waveforms corresponding to TGR
- Duty : 50% duty waveform only



Example : MTU0 normal mode operating (TGRB,TBRC are toggle operating)

PWM mode 1

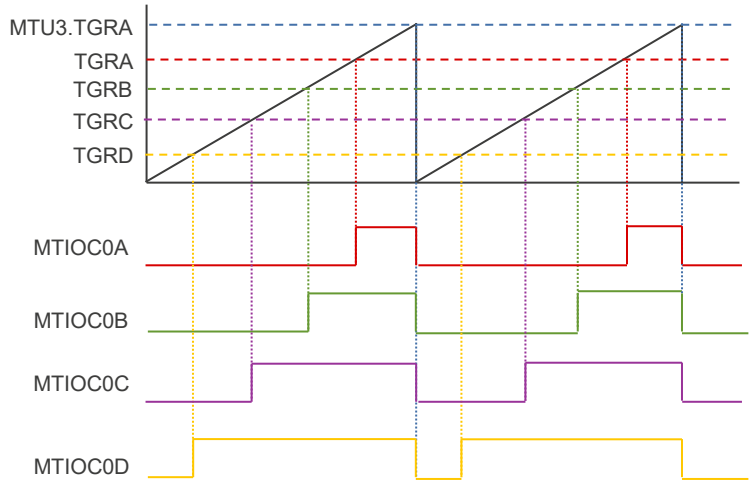
- TCNT clear condition : Any TGR register
- PWM output waveform : Outputs waveform by combination of TGRA&TGRB,TGRC&TGRD
- Duty : It is available PWM waveform output from 0%~100%



Example : MTU0 PWM mode 1 operating (Counter clear by TGRB)

PWM mode 2

- TCNT clear condition : Any TGR register or TGR register of other mode
- PWM output waveform : Only condition for clearing is TCNT counter clearing, the cycle is set by each TGR register.
- Duty : Available PWM waveform output from 0%~100%



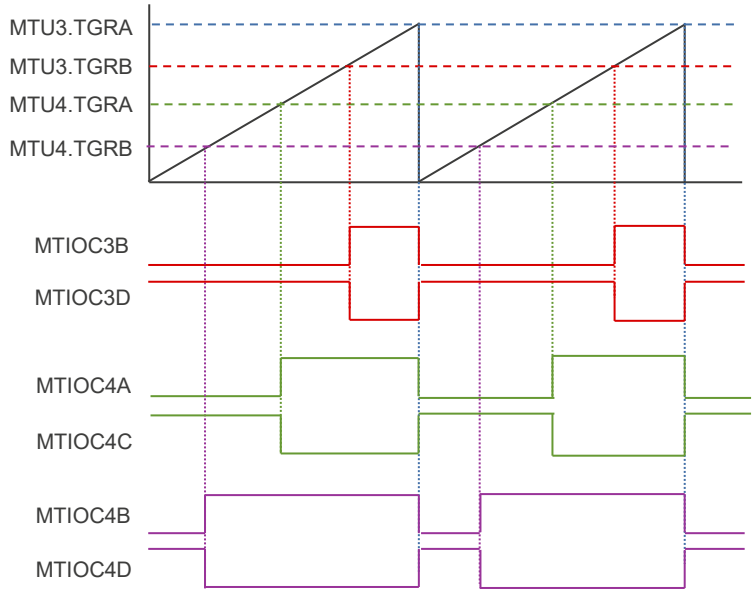
Example : MTU0 PWM mode 2 operating (TCNT count clear condition is MTU3.TGRA at operating mode 1)

OUTPUT WAVEFORM

RESET-SYNCHRONIZED PWM MODE, COMPLEMENTARY PWM MODE

Reset-synchronized PWM mode

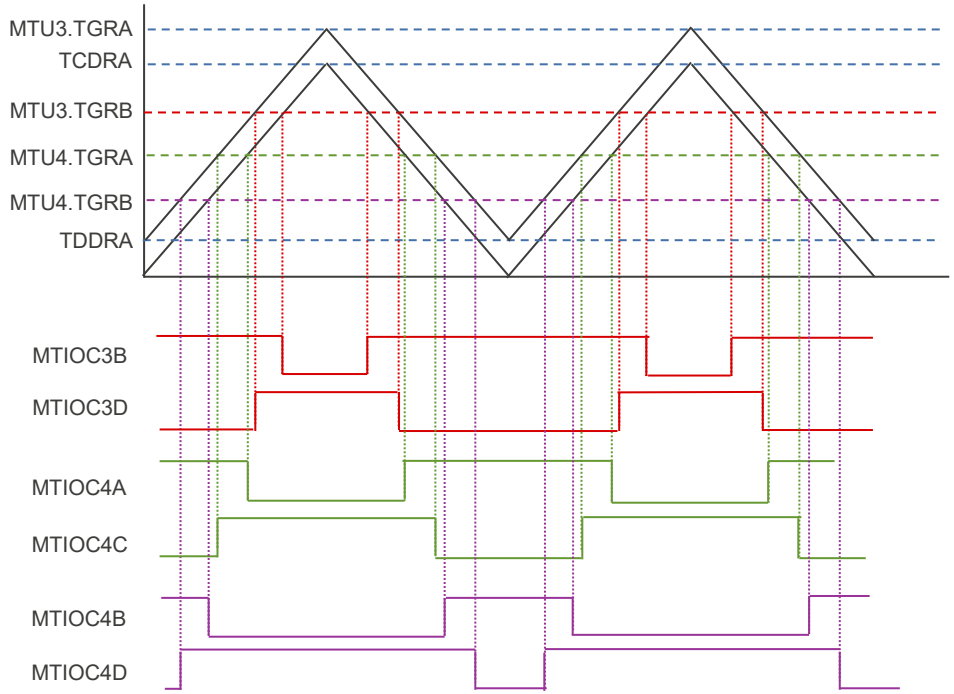
- TCNT clear condition : MTU3(6).TGRA register
- PWM output waveform : Toggle output (Positive and Negative) by compare-match of MTU3(6).TGRB/MTU4(7).TGRA/MTU4(7).TGRB and counter clear
- Duty : Available PWM waveform output between 0%~100%
- Dead time : Without



Example : MTU3, MTU4 Reset-synchronized PWM mode operating

Complementary PWM mode

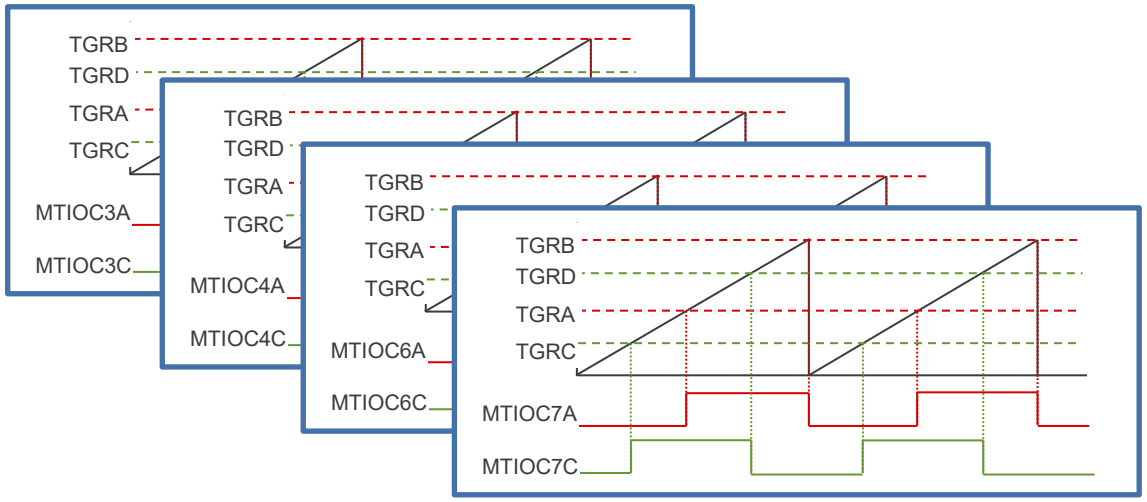
- TCNT clear condition : MTU3(6).TGRA register
- PWM output waveform : Toggle output (Positive and Negative) by compare-match of MTU3(6).TGRB/MTU4(7).TGRA/MTU4(7).TGRB.
- Duty : Available PWM waveform output between 0%~100%
- Dead-time : Retains the dead-time of value set in TDDRA



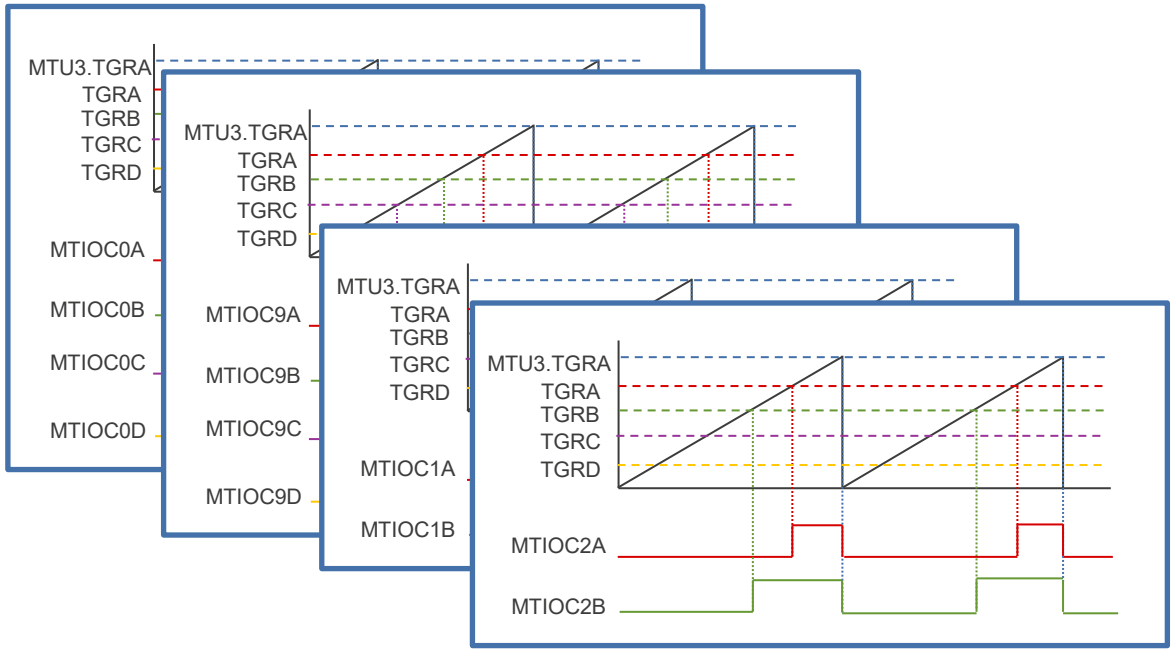
Example : MTU3, MTU4 complementary PWM mode operating

MAXIMUM NUMBER OF PWM OUTPUTS

The maximum number of PWM outputs is up to 14* in PWM mode 1 and up to 12* in PWM mode 2.
When PWM Mode 1 and PWM Mode 2 are mixed, up to 20* outputs are possible.
* Varies depending on the product

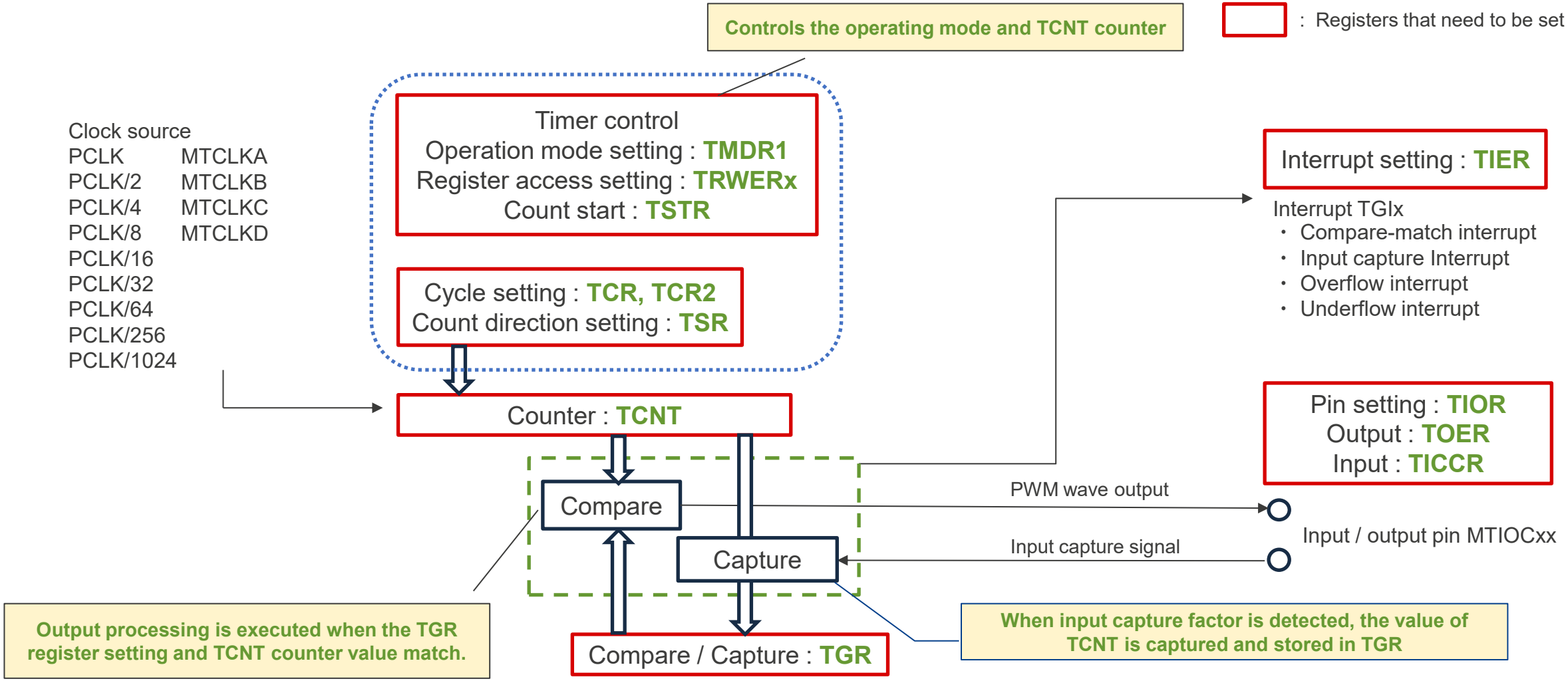


PWM mode 1 (MTU3, MTU4, MTU6, MTU7) : Total 8 outputs

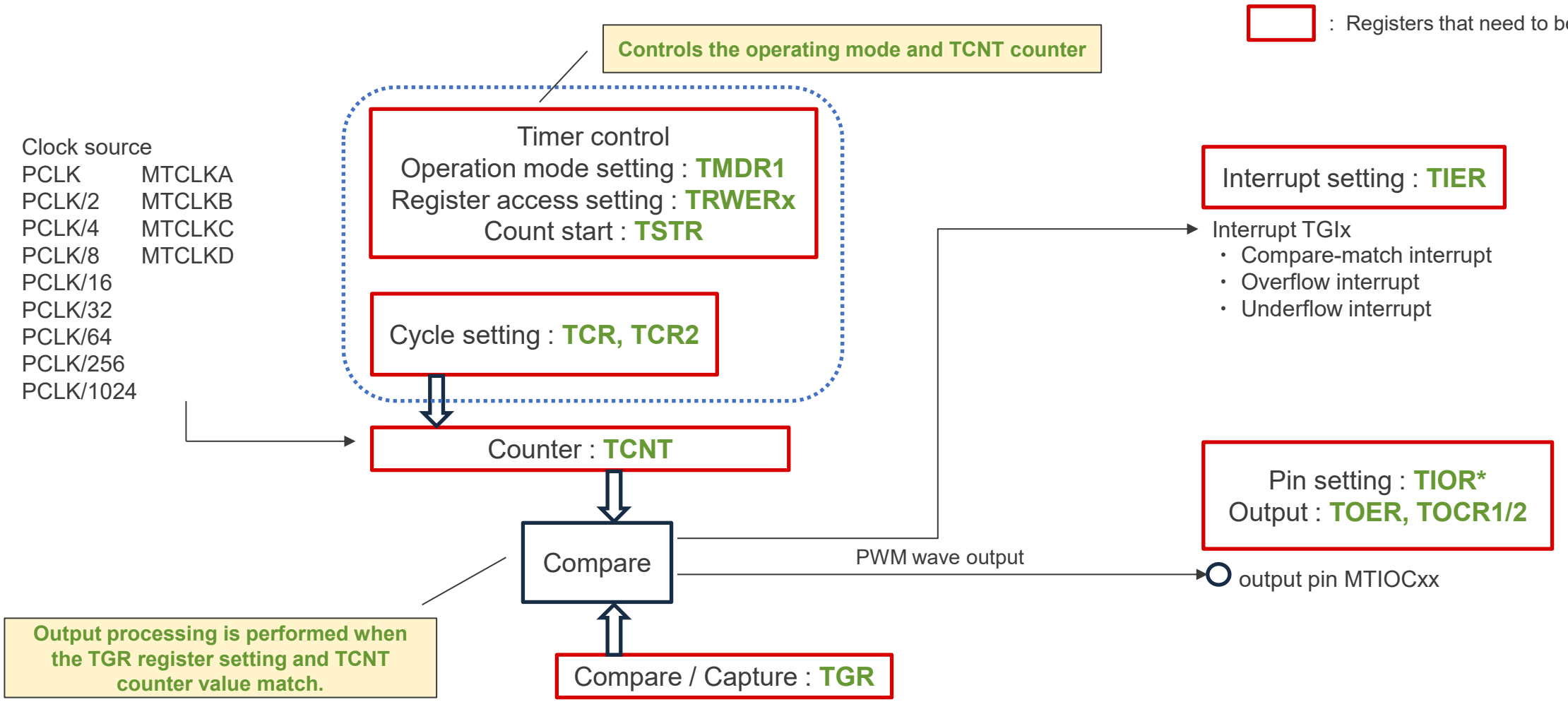


PWM mode 2 (MTU0~MTU2, MTU9) : Total 12 outputs
***TCNT count clearing is MTUn.TGRA of mode 1 operating**
n = 3, 4, 6, 7

NORMAL MODE, PWM MODE 1/2 RELATED REGISTERS

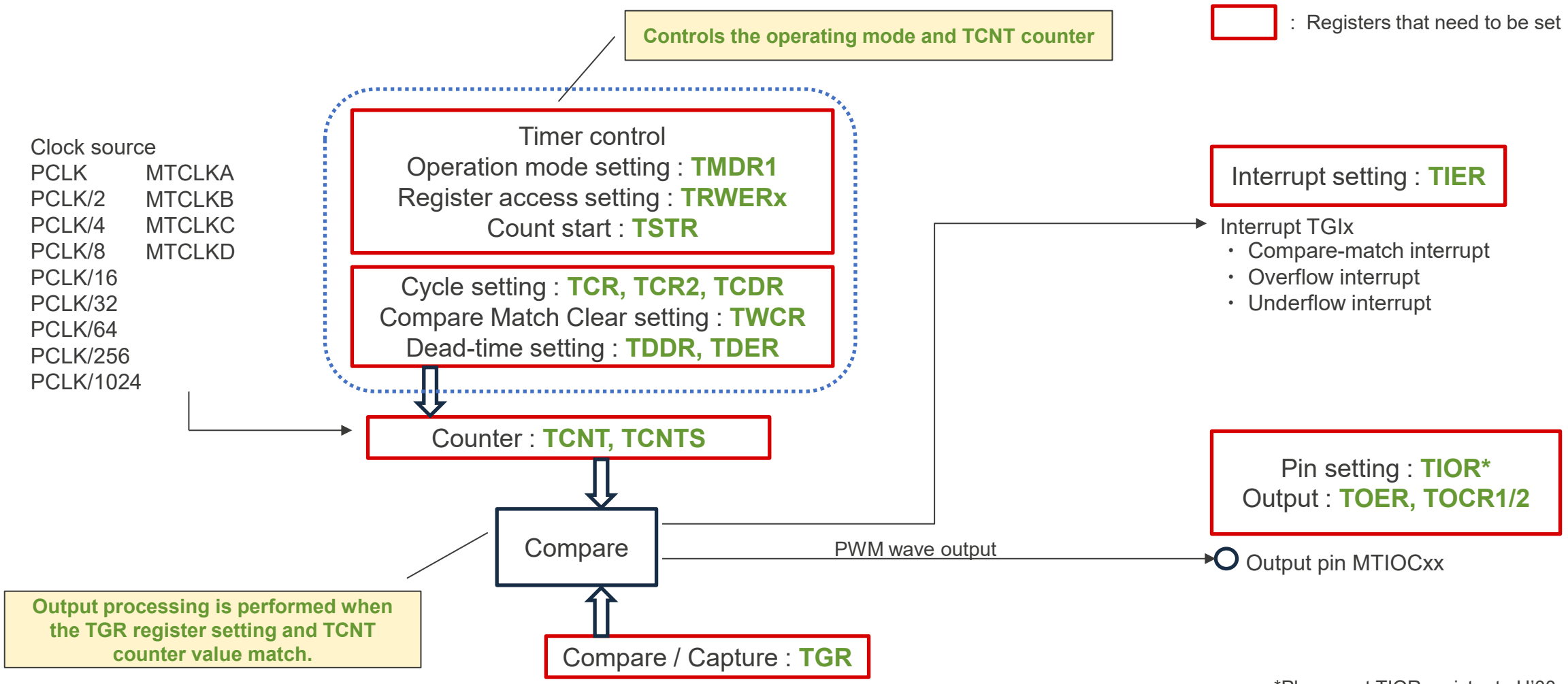


RESET-SYNCHRONIZED PWM MODE RELATED REGISTERS



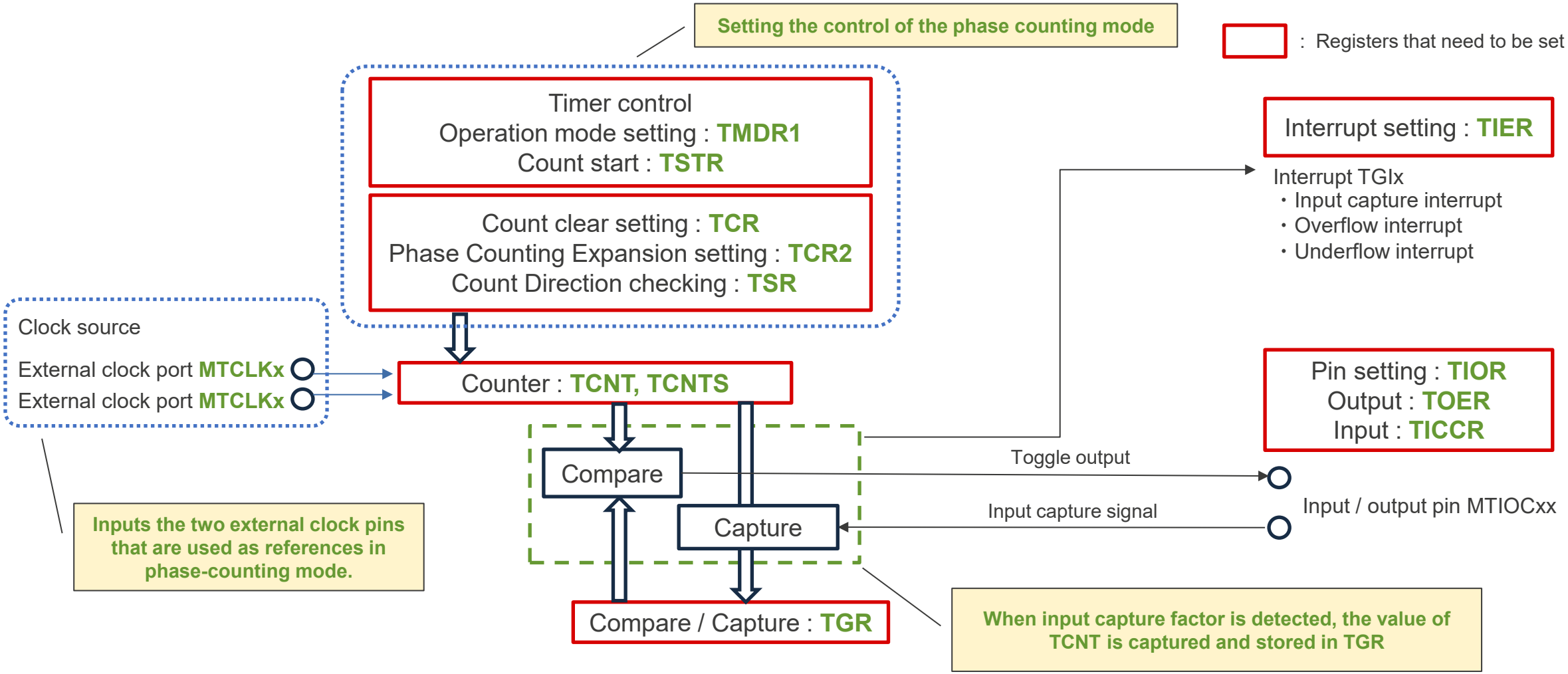
*Please set TIOR register to H'00

COMPLEMENTARY PWM MODE RELATED REGISTERS

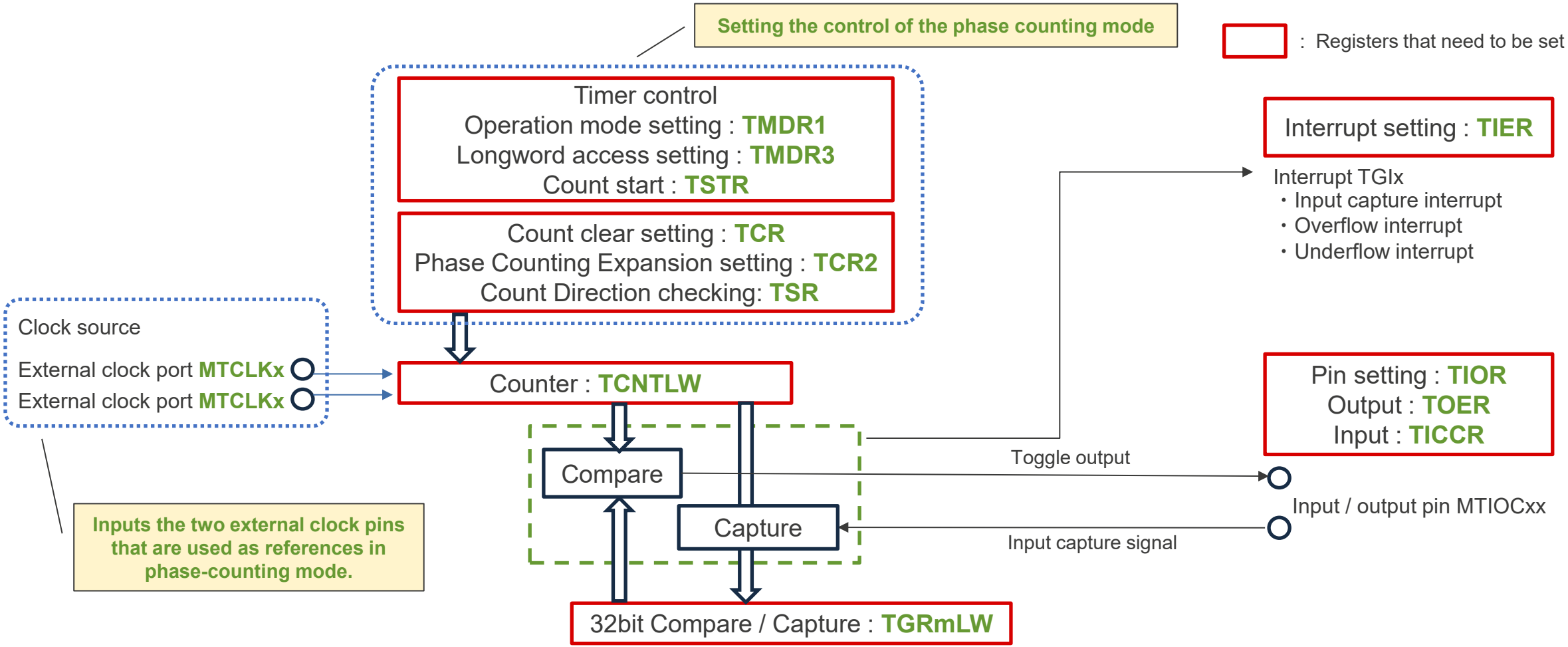


*Please set TIOR register to H'00

PHASE COUNTING MODE RELATED REGISTERS



PHASE COUNTING MODE (CASCADE OPERATION) RELATED REGISTERS



* TCR, TCR2, TMDR and TIOR registers are enabled for MTU1 only.

BUFFER OPERATION RELATED REGISTERS (1/2)

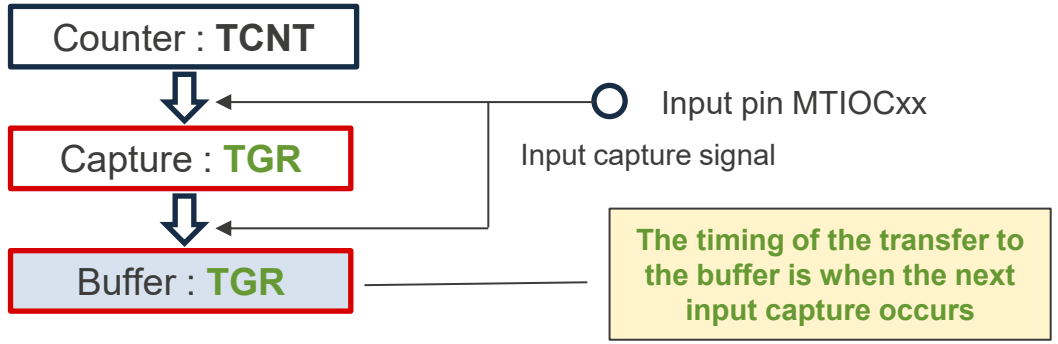
: Registers that need to be set

: Buffer registers

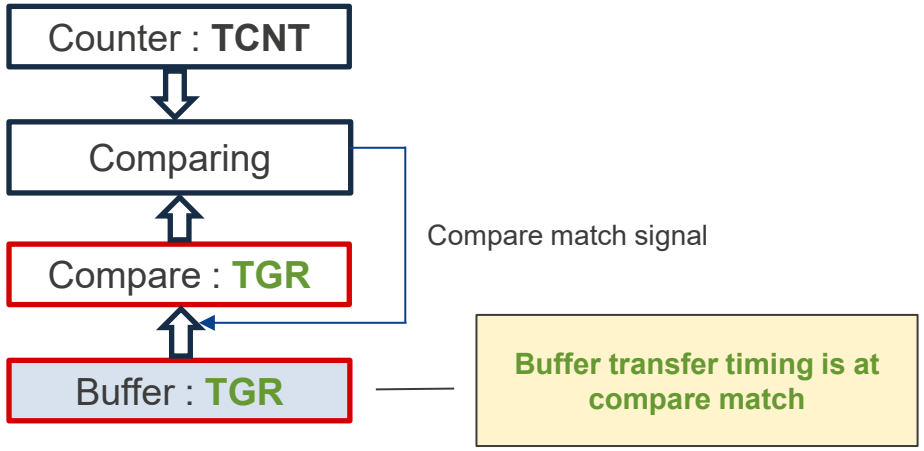
■ Common settings for normal mode, PWM mode 1/2 and phase counting mode

Timer control
Buffer setting : **TMDR1**

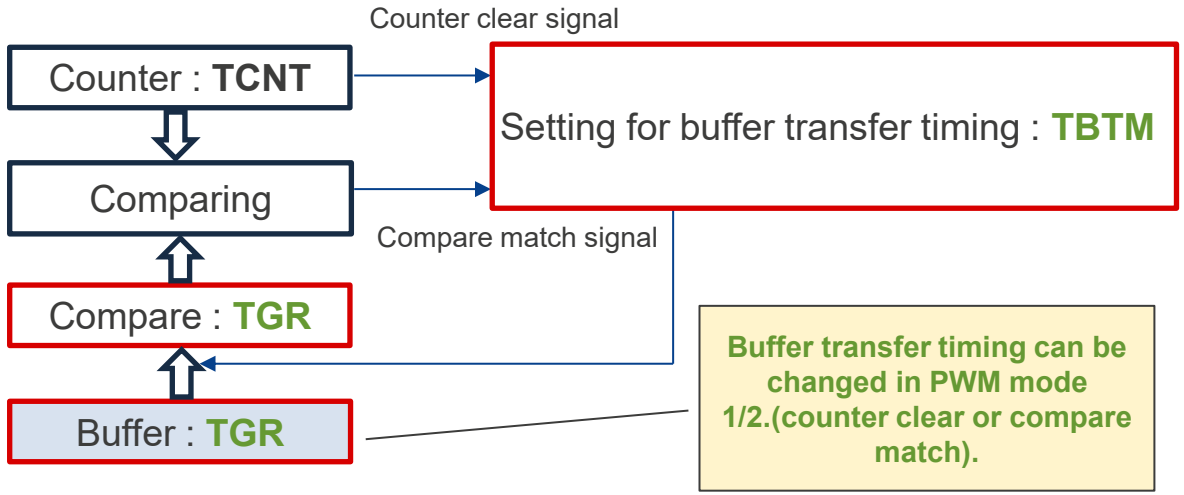
■ Input capture buffer operation (Normal mode, PWM mode 1/2, phase counting mode)



■ Output compare buffer operation (normal mode, phase counting mode)



■ Buffered operation of output compare (PWM mode 1/2)

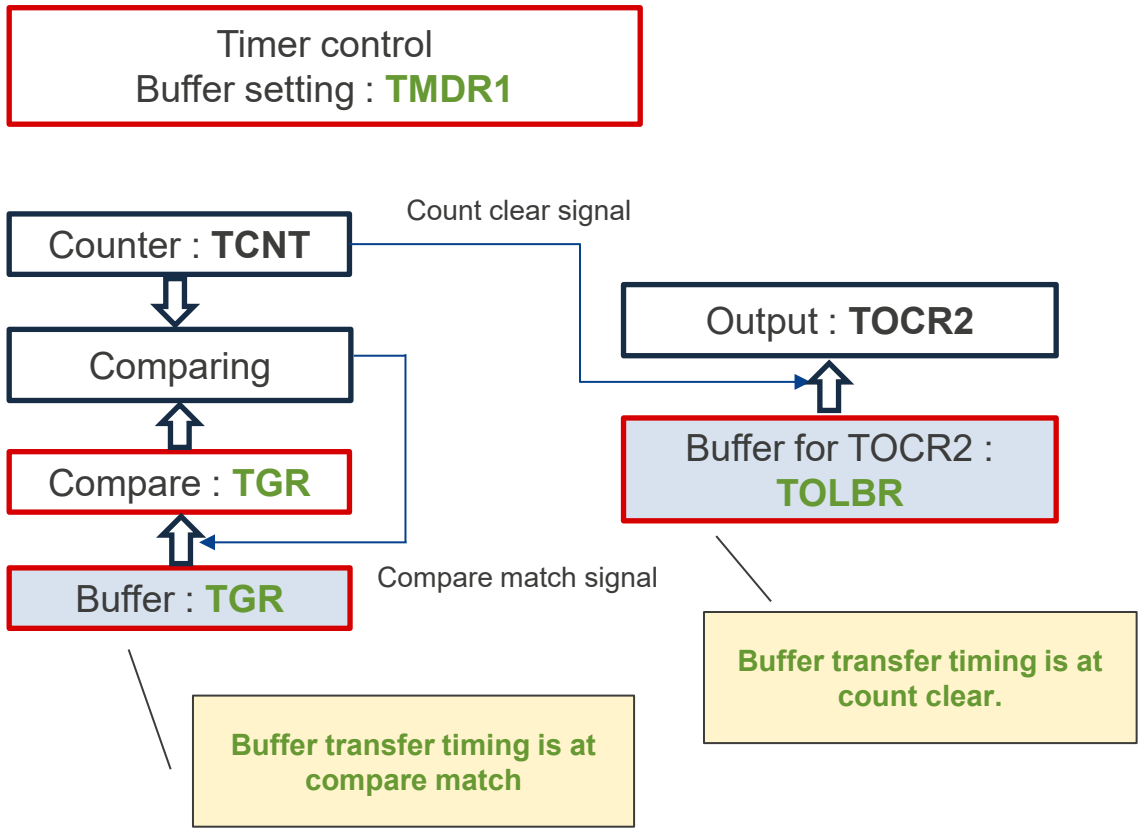


BUFFER OPERATION RELATED REGISTERS (2/2)

: Registers that need to be set

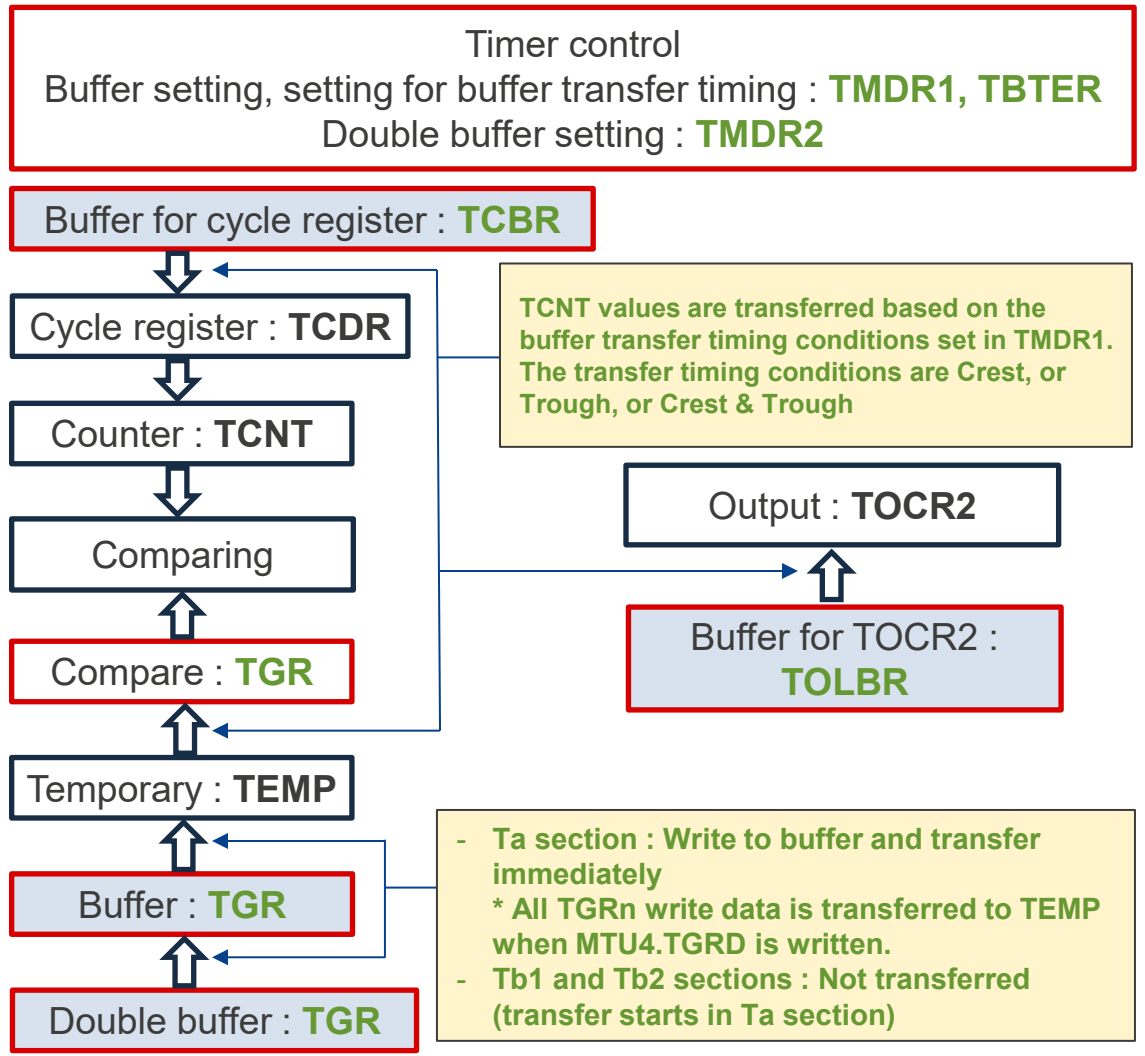
: Buffer registers

■ Buffer operation in reset synchronous PWM mode

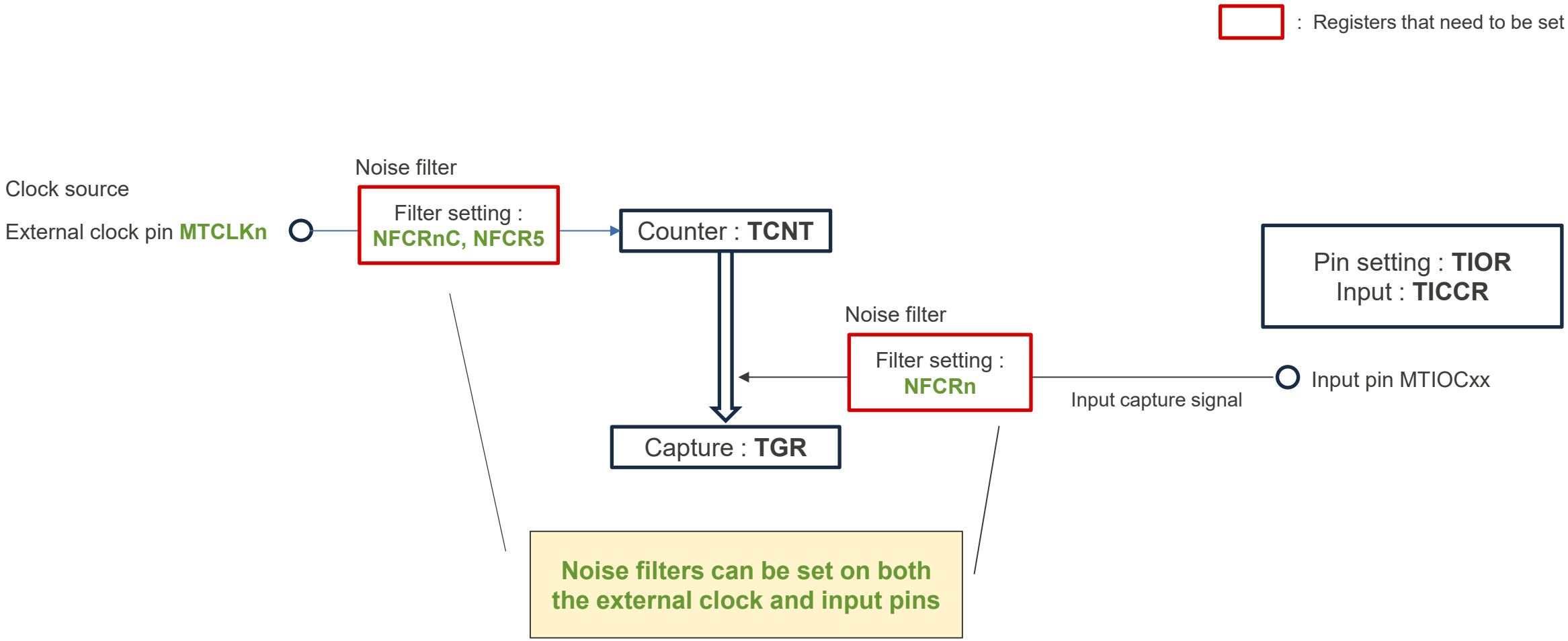


**In reset synchronous PWM mode, both period buffer and duty buffer transfers are performed at the compair timing of the period register.

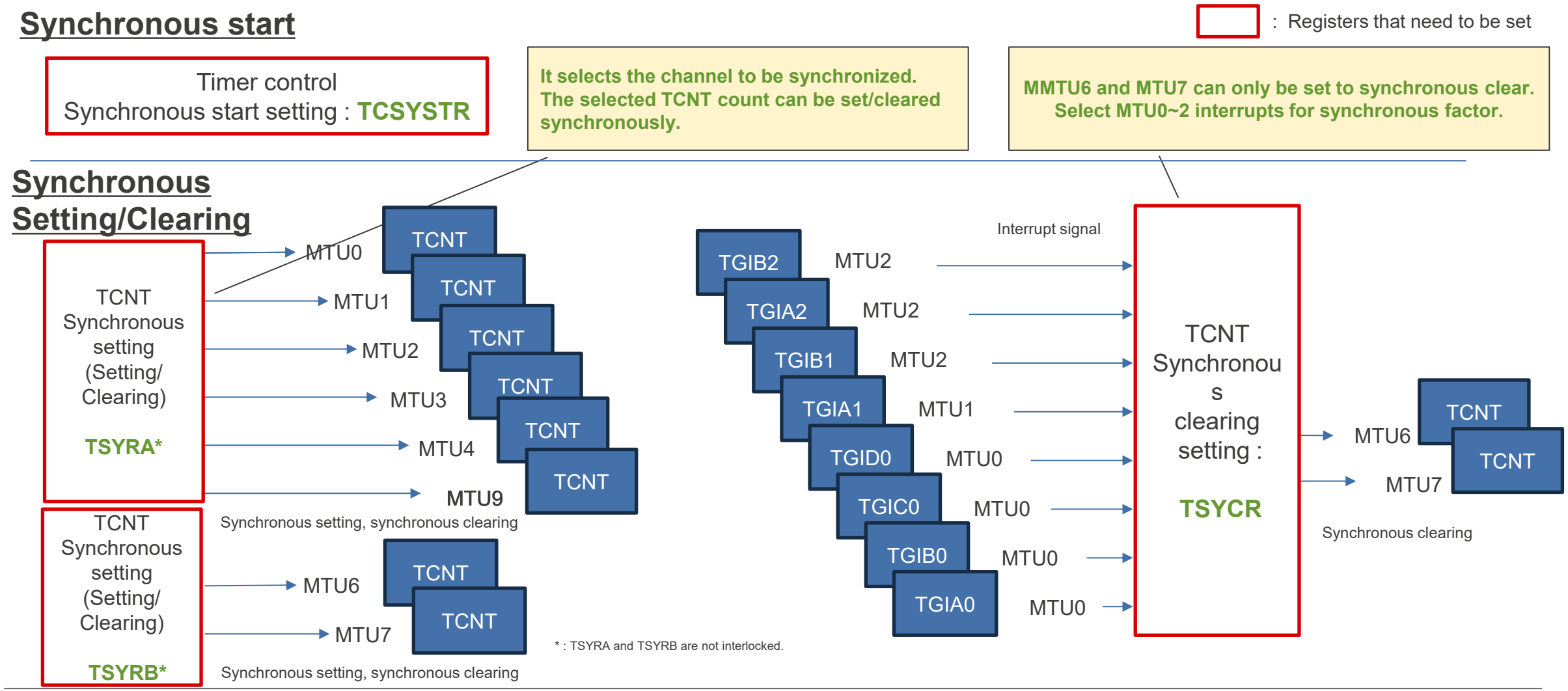
■ Buffer operation in complementary PWM mode



NOISE FILTER RELATED REGISTERS (COMMON TO ALL MODES)



SYNCHRONOUS OPERATION RELATED REGISTERS
(NORMAL MODE, PWM MODE 1/2, RESET SYNCHRONOUS PWM MODE)



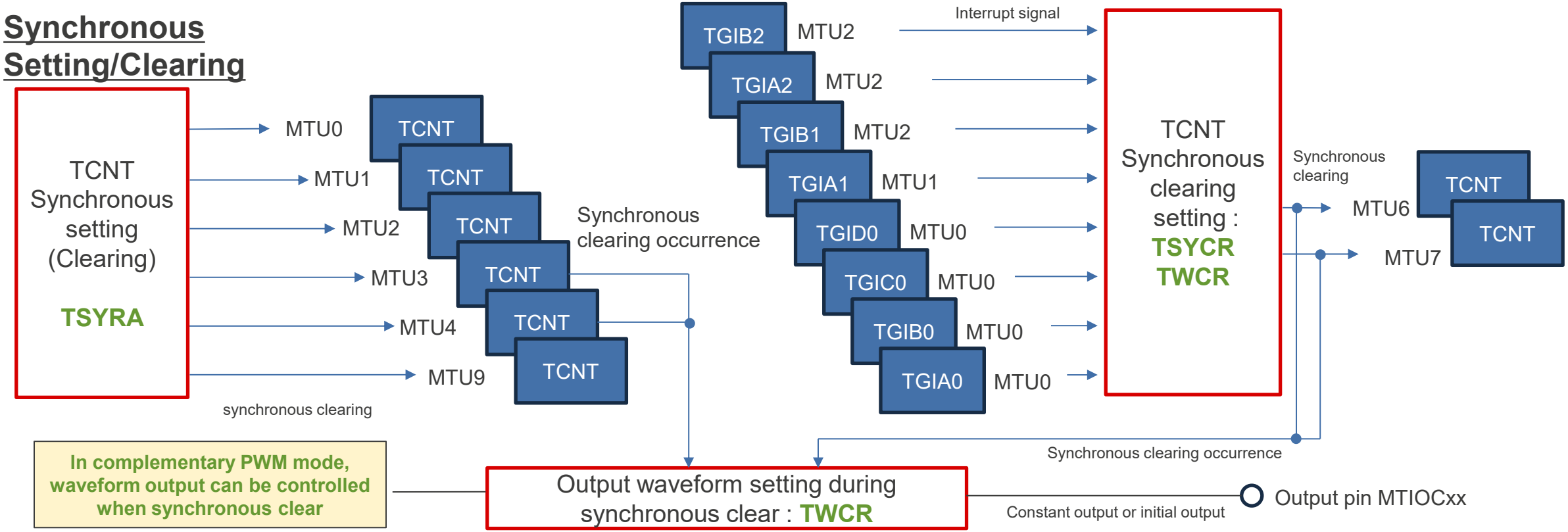
SYNCHRONOUS OPERATION RELATED REGISTERS
(COMPLEMENTARY PWM MODE)

Synchronous start

: Registers that need to be set

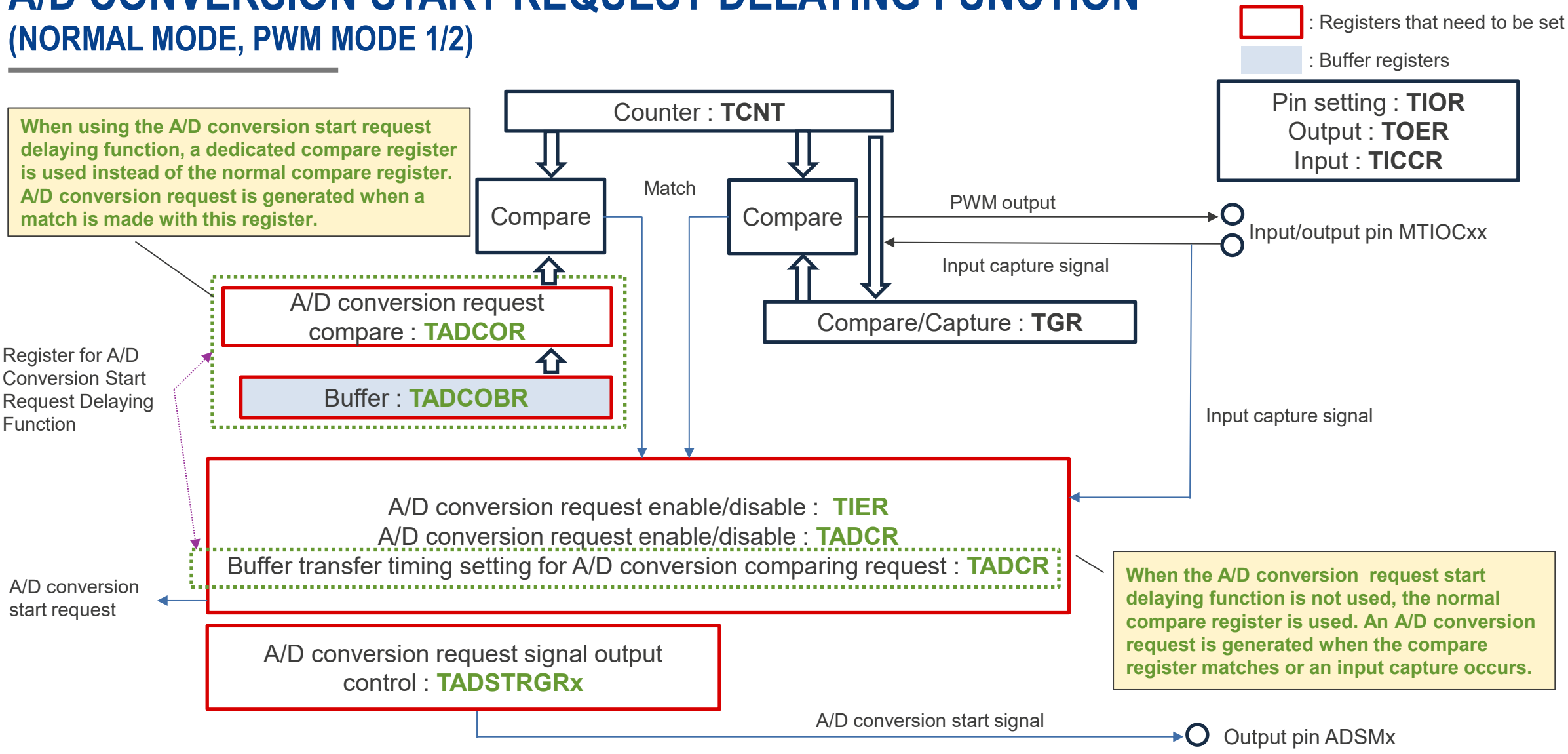
Timer control
Synchronous start setting : **TCSYSTR**

Synchronous Setting/Clearing

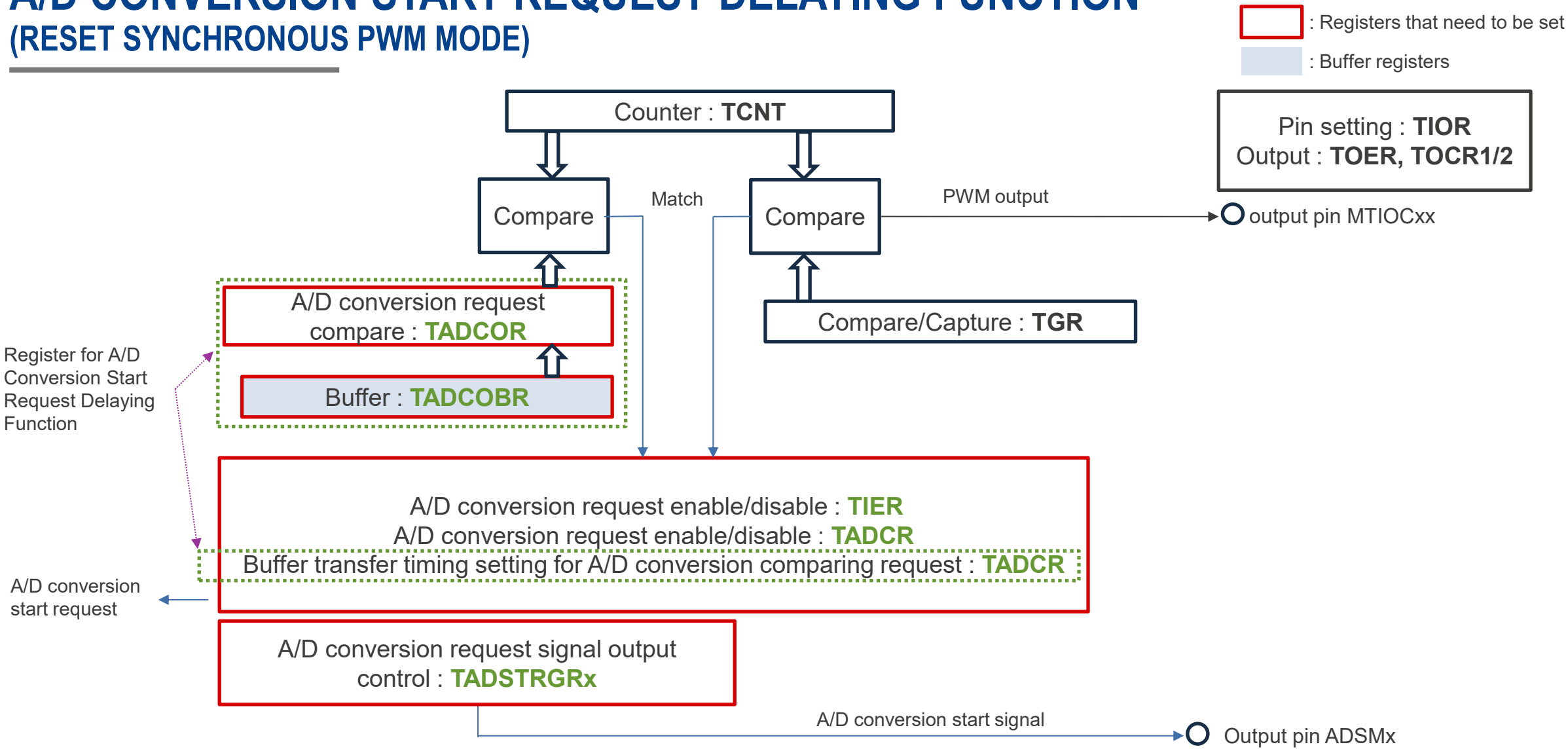


A/D CONVERSION START REQUEST, A/D CONVERSION START REQUEST DELAYING FUNCTION

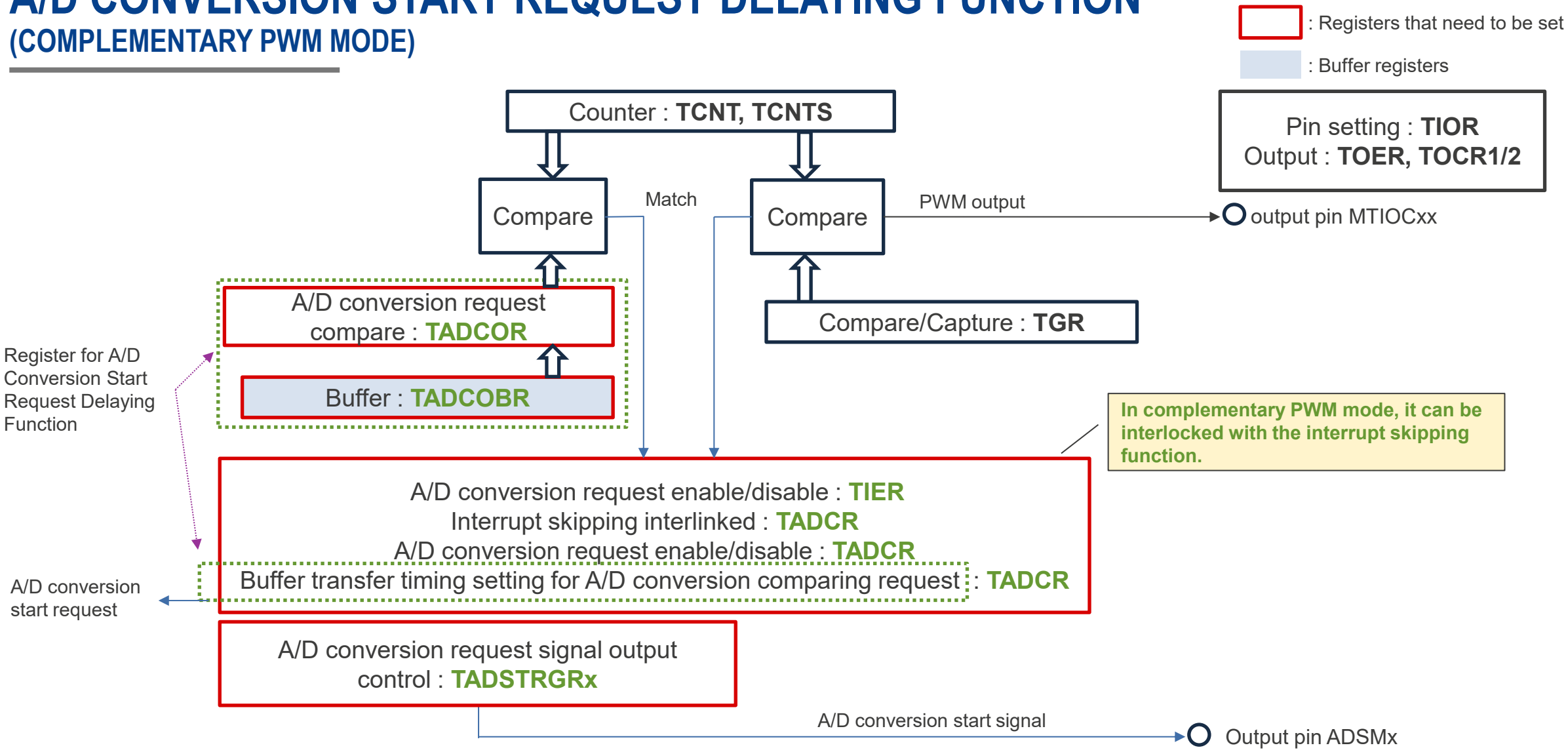
(NORMAL MODE, PWM MODE 1/2)



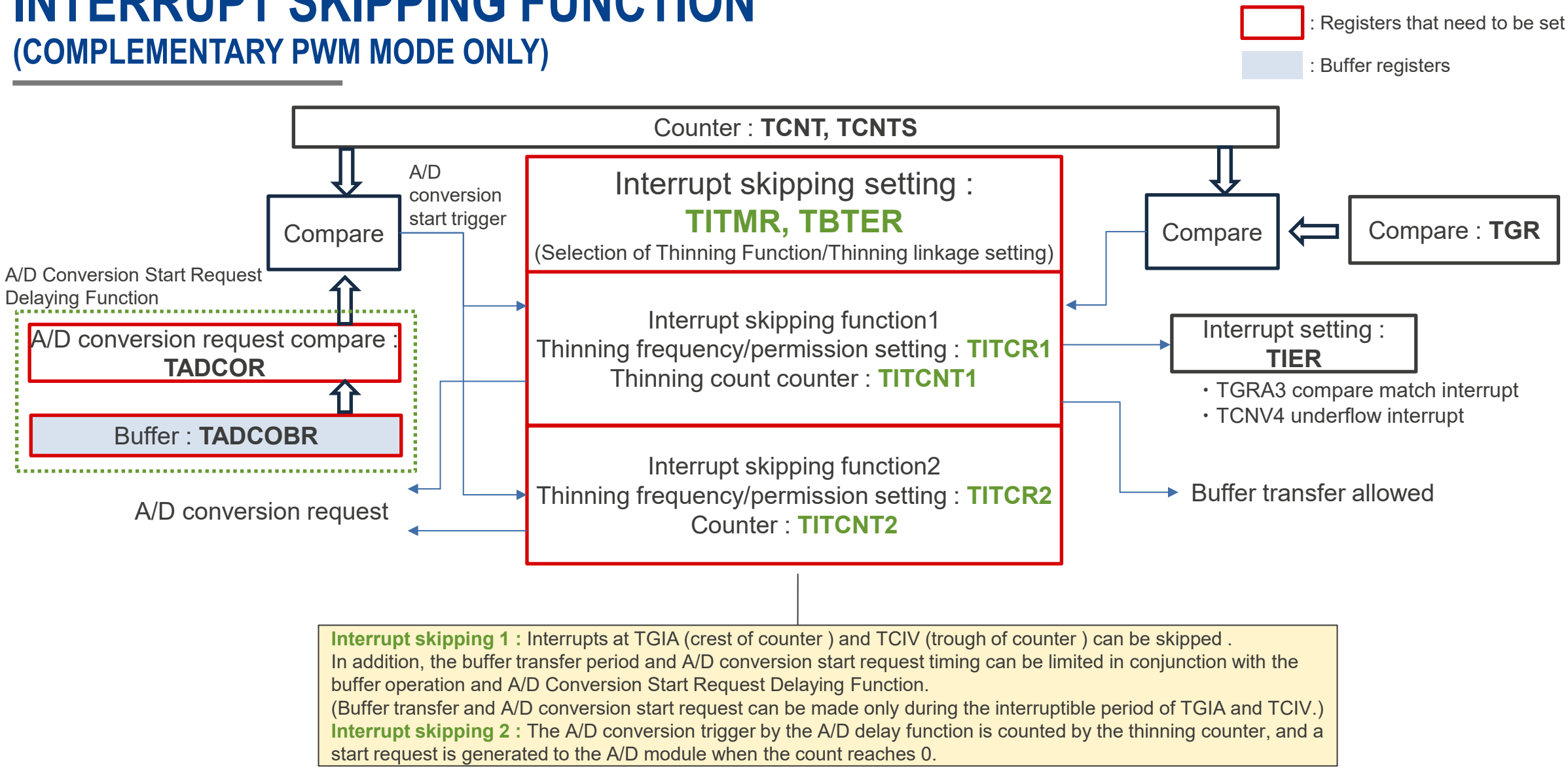
A/D CONVERSION START REQUEST, A/D CONVERSION START REQUEST DELAYING FUNCTION (RESET SYNCHRONOUS PWM MODE)



A/D CONVERSION START REQUEST, A/D CONVERSION START REQUEST DELAYING FUNCTION (COMPLEMENTARY PWM MODE)



INTERRUPT SKIPPING FUNCTION
(COMPLEMENTARY PWM MODE ONLY)



LIST OF APPLICATION NOTES USING MTU

Please refer to the following for more detailed MTU usage.

- RX Family PWM Output Methods Using MTU3/GPTW R01AN5995
- RX Family Example of Operation Near PWM 0% and 100% Duty Cycles Using MTU3 or GPTW R01AN6539
- A/D conversion start request delayed function using MTU3/GPTW R01AN6643
- RX Family Pulse Period Measurement Using MTU2/MTU3 R01AN6644
- RX Family Pulse Width Measurement Using MTU2/MTU3 R01AN6748
- RX Family Usage Examples for Phase Counting Modes Using MTU3/GPTW R01AN6387
- RX Family Synchronous Operation Using MTU3/GPTW R01AN6282

POE3

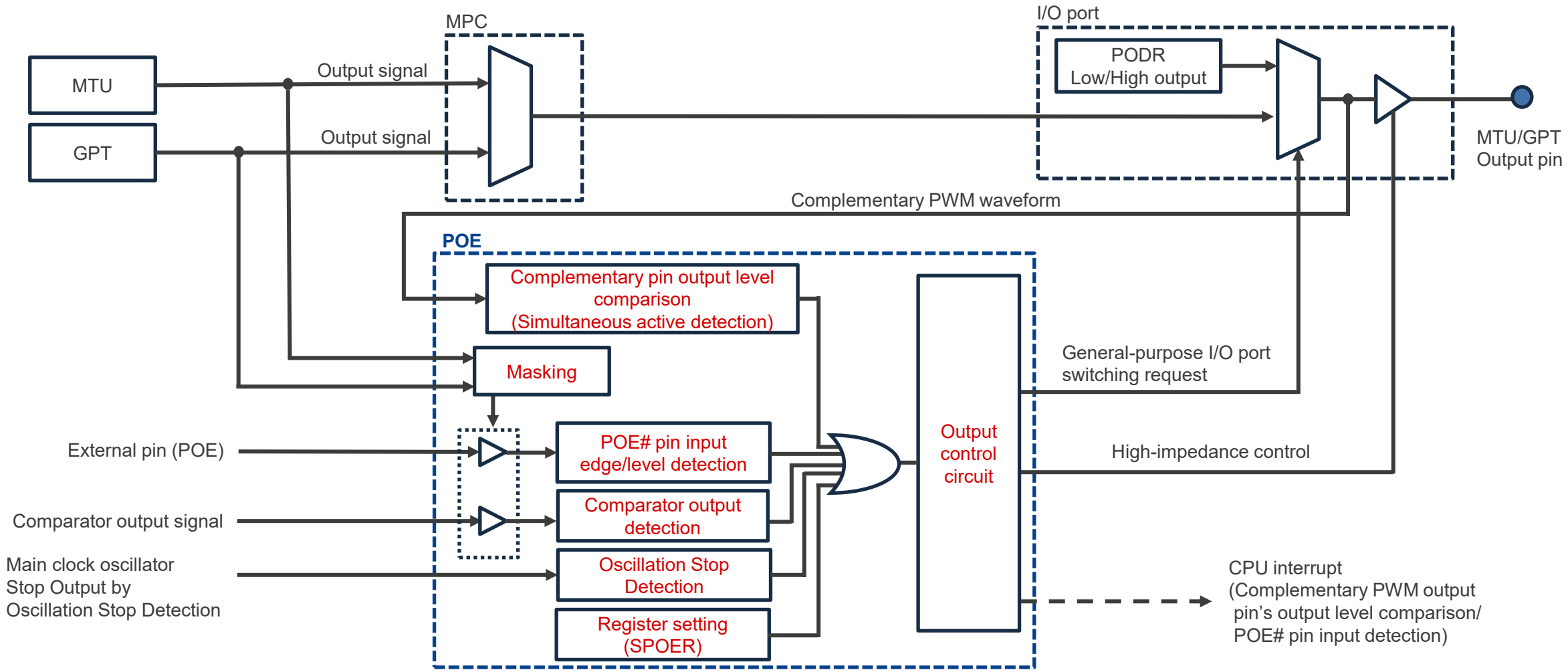
PORT OUTPUT ENABLE(POE) FUNCTION/SPECIFICATIONS

- POE function automatically switches target pins to high-impedance or GPIO ports and disables PWM output upon detection of an output disable condition.
  The following is POE function specification list. (This may vary depending on the product, please refer to the hardware manual for details.)

POE specification		
Status of target pin for Output disable when output disable condition is detected		Switch to high-impedance or general-purpose I/O port
Conditions for disabling the output	Condition 1 : Output-Level Compare of complementary PWM output pin	Positive and reverse phase output levels can be monitored, and a short circuit can be set as a stop condition (refer to P.60 for target terminals)
	Condition 2 : Change of POE# pin input	Trigger input (falling edge or low detection) on POE0, 4, 8~12# pins can be set as stop condition *Rising edge or High detection is also possible by the logic inversion function of the input signal
	Condition 3 : Comparator output detection	Output signal from comparator function can be set as stop condition
	Condition 4 : Oscillation Stop Detection (Main Clock)	Oscillation stop detection of main clock can be set as stop condition
	Condition 5 : Register setting (SPOER)	Stop condition can be generated by software
Target pins for switching to disabling of signal output		Refer to P.60
Target pins for switching to disabling of signal output for each output disable condition	When comparing output levels (output pins short-circuited) of complementary PWM output pins	When the output level comparison result is the MTU pin: All target pins of MTU When the output level comparison result is the GPT pin: All the target pins of GPT
	When POE# pin input is changed	Select from all MTU and GPT target pins
	When comparator output is detected	Select from all MTU and GPT target pins
	When oscillation (Main Clock) stop is detected	Select from all MTU and GPT target pins
	When register setting (SPOER)	Select from all MTU and GPT target pins
Masking of output conditions (POE# pin input change, comparator output detection)		The conditions of “POE# input terminal change” and “comparator output detection” can be masked depending on the MTU/GPT output terminal state. Refer to P.60 for target pins.
Interrupt factor		In case POE# pin is input In case comparing output levels of complementary pin

POE DETECT FUNCTION SYSTEM BLOCK DIAGRAM

- The figure below shows the overall block diagram of the POE detect function system.



OVERVIEW OF MASKING, OUTPUT LEVEL COMPARISON, AND PINS TARGETED FOR STOP CONTROL

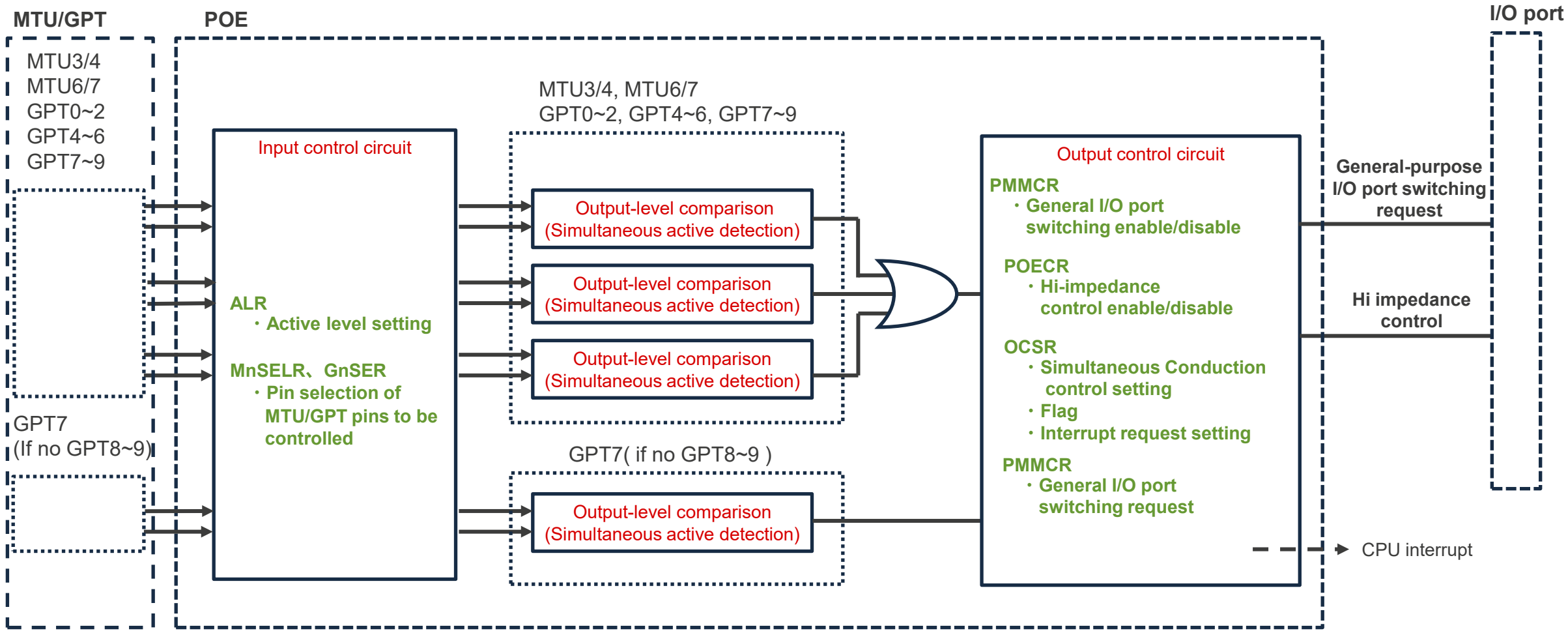
MTU function pins		Masking pins	Output level comparison pins	Stop control target pins
MTU0	MTIOC0A	✓		✓
	MTIOC0B	✓		✓
	MTIOC0C	✓		✓
	MTIOC0D	✓		✓
MTU1	MTIOC1A	✓		
	MTIOC1B	✓		
MTU2	MTIOC2A	✓		
	MTIOC2B	✓		
MTU3	MTIOC3A	✓		
	MTIOC3B	✓	✓	✓
	MTIOC3C	✓		
	MTIOC3D	✓	✓	✓
MTU4	MTIOC4A	✓	✓	✓
	MTIOC4B	✓	✓	✓
	MTIOC4C	✓	✓	✓
	MTIOC4D	✓	✓	✓
MTU6	MTIOC6A	✓		
	MTIOC6B	✓	✓	✓
	MTIOC6C	✓		
	MTIOC6D	✓	✓	✓
MTU7	MTIOC7A	✓	✓	✓
	MTIOC7B	✓	✓	✓
	MTIOC7C	✓	✓	✓
	MTIOC7D	✓	✓	✓

MTU function pins		Masking pins	Output level comparison pins	Stop control target pins
MTU9	MTIOC9A	✓		✓
	MTIOC9B	✓		✓
	MTIOC9C	✓		✓
	MTIOC9D	✓		✓

MTU function pins		Masking pins	Output level comparison pins	Stop control target pins
GPTW0	GTIOC0A	✓	✓	✓
	GTIOC0B	✓	✓	✓
GPTW1	GTIOC1A	✓	✓	✓
	GTIOC1B	✓	✓	✓
GPTW2	GTIOC2A	✓	✓	✓
	GTIOC2B	✓	✓	✓
GPTW3	GTIOC3A	✓		✓
	GTIOC3B	✓		✓
GPTW4	GTIOC4A	✓	✓	✓
	GTIOC4B	✓	✓	✓
GPTW5	GTIOC5A	✓	✓	✓
	GTIOC5B	✓	✓	✓
GPTW6	GTIOC6A	✓	✓	✓
	GTIOC6B	✓	✓	✓
GPTW7	GTIOC7A	✓	✓	✓
	GTIOC7B	✓	✓	✓
GPTW8	GTIOC8A	✓	✓	✓
	GTIOC8B	✓	✓	✓
GPTW9	GTIOC9A	✓	✓	✓
	GTIOC9B	✓	✓	✓

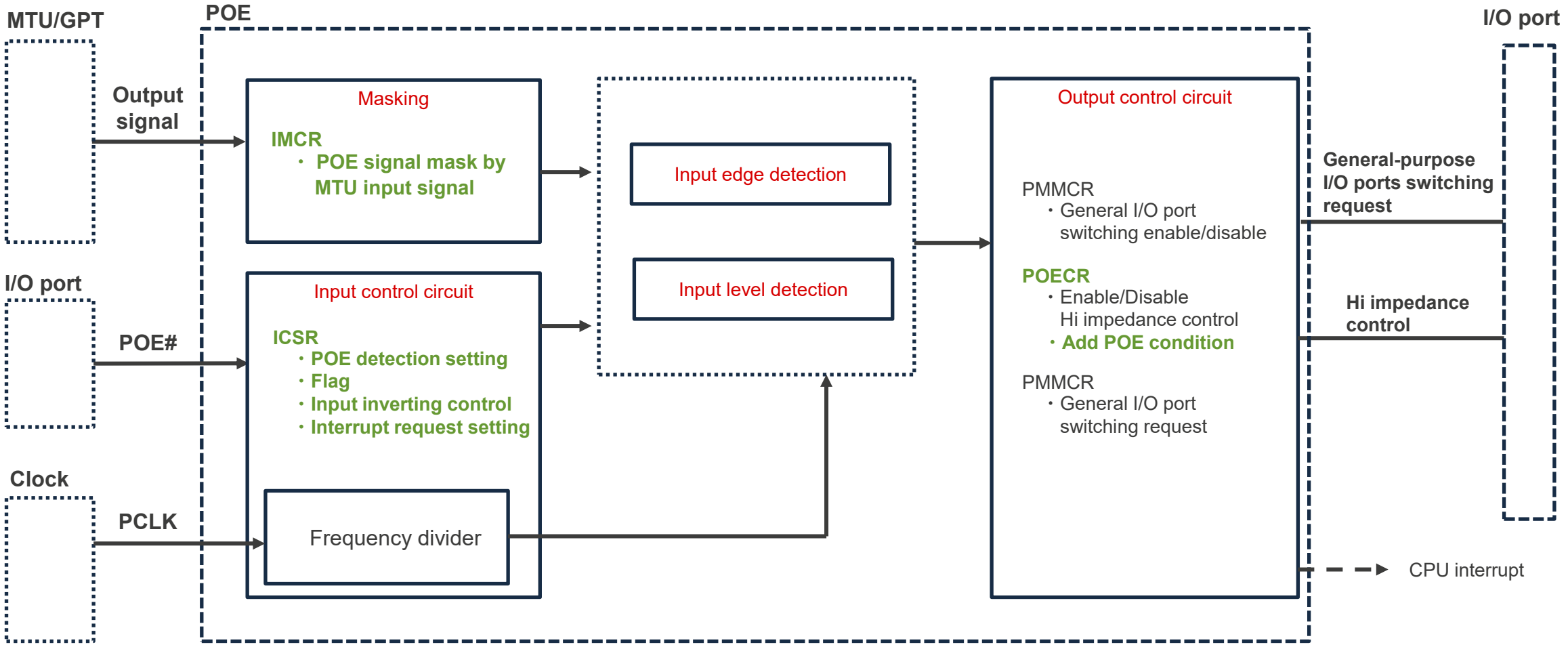
IN CASE OF USING THE OUTPUT LEVEL COMPARISON OF THE COMPLEMENTARY PWM OUTPUT PIN AS THE STOP CONDITION

- The control block and registers that need to be set when the output level comparison of the complementary PWM output pins is used as the stop condition are shown below.



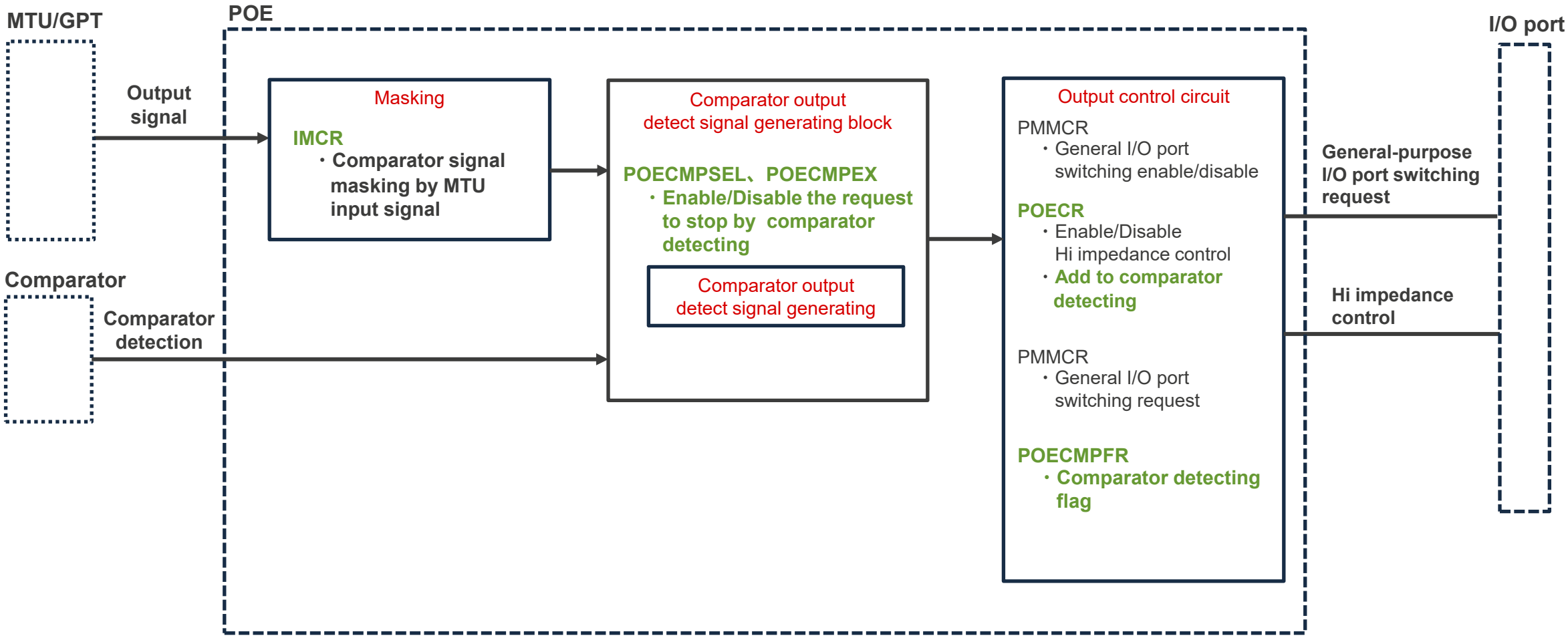
IN CASE OF USING THE POE# PIN INPUT AS THE STOP CONDITION

- The control block and registers that need to be set when the POE# pin input is used as the stop condition are shown below.



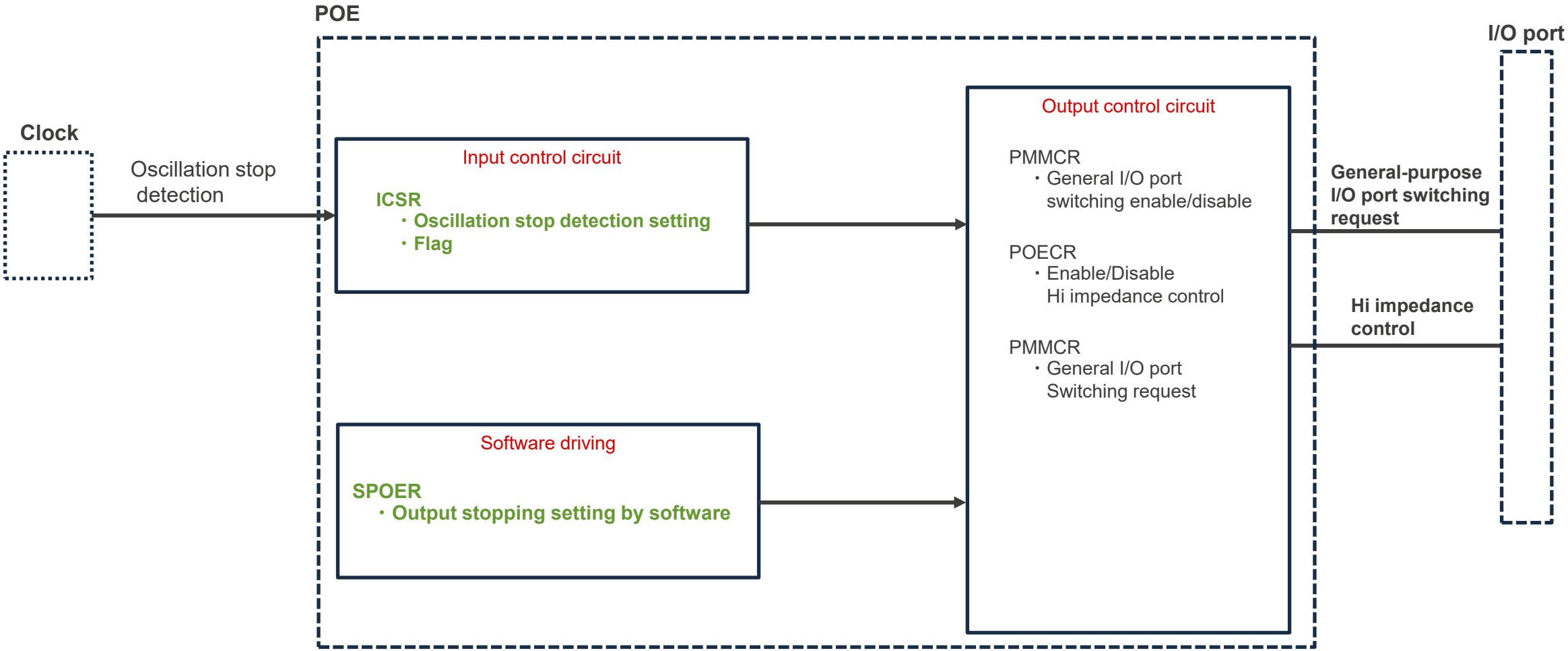
IN CASE OF USING THE COMPARATOR OUTPUT SIGNAL AS THE STOP CONDITION

- The control block and registers that need to be set when the comparator output signal is used as the stop condition are shown below.



IN CASE OF OSCILLATION STOP DETECTION SIGNAL AND CONTROL BY SOFTWARE

- The control block and registers that need to be set when the oscillation stop detection signal or software is used as the stop condition are shown below.



CONDITIONS FOR RELEASE OF POE OUTPUT

■ **The pins with output stopped by input level detection are released by one of the follows.**

- Reset to initial state
- Clear ICSRn.POExF flag

* When set to Low sampling by ICSRn.POExM[3:0] bits, only after a High is input from the POEn# pin and a High is detected, writing '0' to the flag is invalid and the flag is not set to '0'.

■ **The pins with output stopped due to output short-circuit detection are released by one of the follows.**

- Reset to initial state
- Set OCSRn.OSFn flag to "0"

* Note that just writing 0 to a flag is ignored (the flag is not set to 0); the flags can be cleared by writing 0 to it only after setting the pin to the inactive level. In the MTU, the inactive level (initial output level) can be output by stopping the count operation. In the GPTW, the inactive level can be output in accordance with the procedure described in 'Pin Initialization Due to Error during Operation'.

■ **The pins with output stopped by comparator output detection are released by one of the follows**

- Reset to initial state
- Set POECMPFR.CjFLAG flag(j = 0 ～ 5) to "0"

* When setting the POECMPFR.CjFLAG flag to "0", make sure that the analog input signal that was detected by the comparator output has returned to the appropriate value after confirming by performing an A/D conversion, etc. Note that if the flag is cleared without confirming that the analog input signal has returned to the appropriate value and the analog input signal remains higher than the reference voltage when the comparator is inverting output, or remains lower than the reference voltage when the comparator is inverting output, the POECMPFR.CjFLAG flag described above will not become "1" again.

■ **The pins with output stopped by oscillation stop detection are released by one of the follows**

- Reset to initial state
- Set SYSTEM.OSTDSR.OSTDF flag to "0" , then after that set ICSR6.OSTSTF flag to "0"

NOTES

- Depending on the product, the MTU3, 4, 6, and 7 pins are set to high impedance after resetting, or the output condition using some POE pins is enabled. Some products also enable the high-impedance state even when the MTU3, 4, 6, and 7 pins are not used (when the corresponding pins are used as other modules). If this is a problem, the corresponding POECR2 and MnSER registers must be cleared after reset. Please refer to the hardware manual for details.

LIST OF APPLICATION NOTES USING POE

Please refer to the following for more detailed POE usage.

- RX24U Group Disabling and Restoring PWM Output Using POE3A and MTU3d R01AN4140
- RX63T Group POE PWM Output Cutoff by Comparator Detection R01AN1404

WATCHDOG TIMER/ INDEPENDENT WATCHDOG TIMER (WDT/IWDT)

THE FUNCTIONAL COMPARISON OF WDT AND IWDT

WDT
IWDT

The functional comparison between WDT and IWDT is as follows.

Blue text : Functional difference

Item	WDT	IWDT
Count source	Peripheral module clock	**IWDT-dedicated clock (IWDTCLK)**
Clock divide ratio	4/64/128/512/2048/8192	**1/16/32/64/128/256**
Conditions for starting the counter	・Auto-start mode: Counting automatically starts after a reset is released ・Register start mode: Counting is started by refresh operation	・Auto-start mode: Counting automatically starts after a reset is released ・Register start mode: Counting is started by refresh operation
Conditions for stopping the counter	・Reset ・In low power consumption states ・A counter underflows or a refresh error occurs (only in register start mode)	・Reset **・In low power consumption states(Depends on register setting)** ・A counter underflows or a refresh error occurs (only in register start mode)
Window function	Yes	Yes
Reset output sources	・Down-counter underflows ・Refreshing outside the refresh-permitted period (refresh error)	・Down-counter underflows ・Refreshing outside the refresh-permitted period (refresh error)
Non-maskable interrupt/ interrupt sources	・Down-counter underflows ・Refreshing outside the refresh-permitted period (refresh error)	・Down-counter underflows ・Refreshing outside the refresh-permitted period (refresh error)
Reading the counter value	Enable	Enable
Event link function	No	**Yes**
Output signal (Internal signal)	・Reset output ・Interrupt request output	・Reset output ・Interrupt request output **・Sleep mode count stop control output**
Operations during low power consumption mode	Stop	**Run / Stop (Selectable)**

WDT SETTINGS LIST

The watchdog timer has two modes of operation: auto start mode and register start mode.
Each mode is operated by setting registers in the Option Setting Memory (OFS) or WDT.
The following figure shows the registers that need to be set in each operation mode.

: OFS setting
: WDT setting

	Auto-start mode	Register start mode
Start mode setting	Auto start mode enabled OFS0.WDTSTRT	Register start mode enabled OFS0.WDTSTRT
Timer setting	Timeout period selection OFS0.WDTTOPS	Timeout period selection WDTCR.TOPS
	Clock divide ratio selection OFS0.WDTCKS	Clock divide ratio selection WDTCR.CKS
Window mode setting	Window end position selection OFS0.WDTRPES	Window end position selection WDTCR.RPES
	Window start position selection OFS0.WDTRPSS	Window start position selection WDTCR.RPSS
Interrupt output setting when detect the out of control	Reset interrupt request selection OFS0.WDTRSTIRQS	Reset interrupt request selection WDTRCR.RSTIRQS
Count start	Automatic start after reset release	Refresh operation WDT.WDTRR Write “00h” and then “FFh

IWDT SETTINGS LIST

IWDT

IWDT has two modes of operation: auto start mode and register start mode.
Each mode is operated by setting registers in the Option Setting Memory (OFS) or IWDT.
The following figure shows the registers that need to be set in each operation mode.

: OFS setting
: WDT setting

	Auto-start mode	Register start mode
Start mode setting	Auto start mode enabled OFS0.IWDTSTRT	Register start mode enabled OFS0.IWDTSTRT
Timer setting	Timeout period selection OFS0.IWDTTOPS[1:0]	Timeout period selection IWDTCR.TOPS[1:0]
	Clock divide ratio selection OFS0.IWDTCKS[3:0]	Clock divide ratio selection IWDTCR.CKS[3:0]
Window mode setting	Window end position selection OFS0.IWDTRPES[1:0]	Window end position selection IWDTCR.RPES[1:0]
	Window start position selection OFS0.IWDTRPSS[1:0]	Window start position selection IWDTCR.RPSS[1:0]
Interrupt output setting when detect the out of control	Reset interrupt request selection OFS0.IWDTRSTIRQS	Reset interrupt request select IWDTRCR.RSTIRQS
Count stop setting in sleep mode	Sleep Mode Count Stop Control OFS0.IWDTSLCSTP	Sleep Mode Count Stop Control IWDTCSTPR.SLCSTP
Count start	Automatic start after reset release	Refresh Operation IWDT.IWDTRR Write “00h” and then “FFh

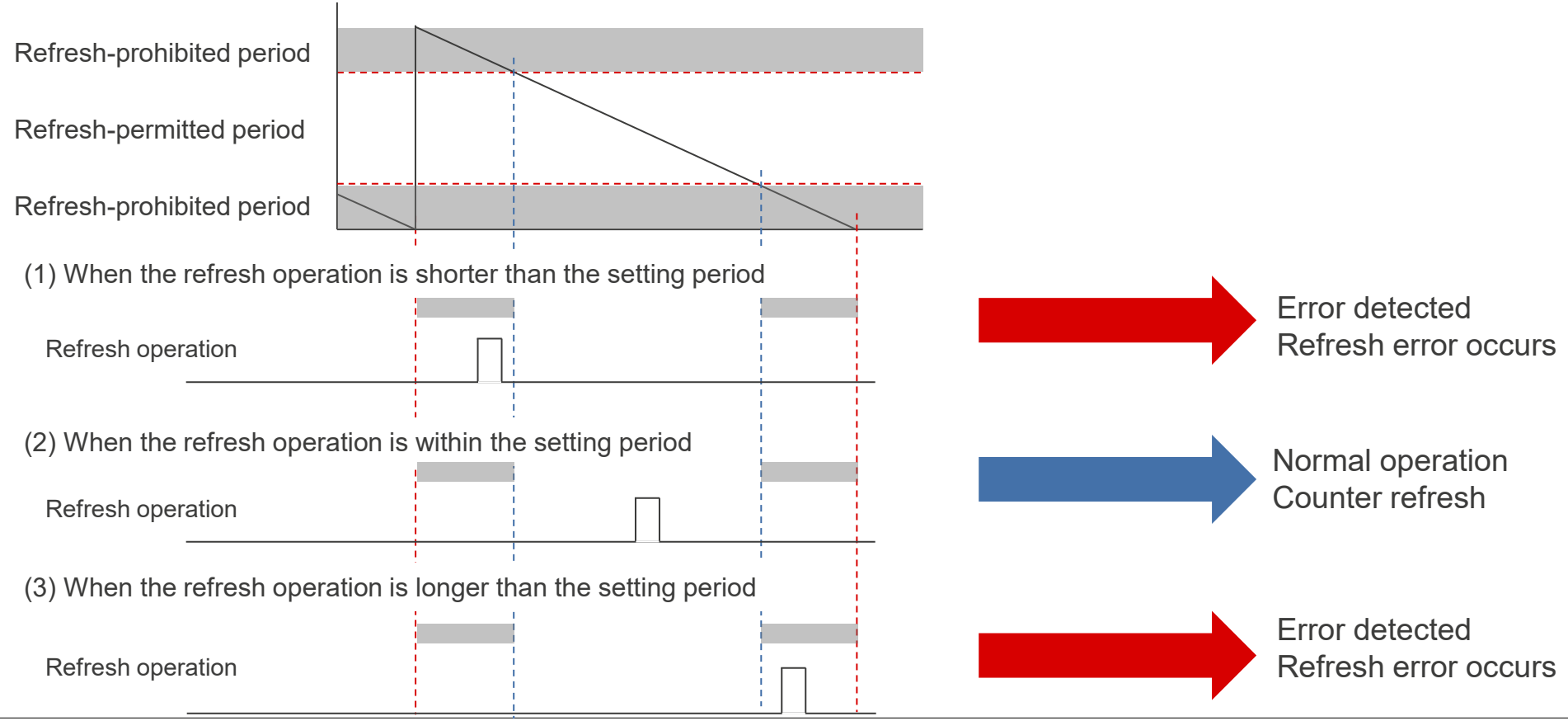
FUNCTION EXPLANATION : WINDOW MODE

WDT
IWDT

In window mode, set the permitted and prohibited periods for refresh operations.

If a refresh operation is executed during the prohibited period, a refresh error occurs and the system is judged to be out of control.

In addition to CPU running out of control, this function can also detect deviations in the processing cycle, making it suitable for applications with high periodicity and reliability requirements.



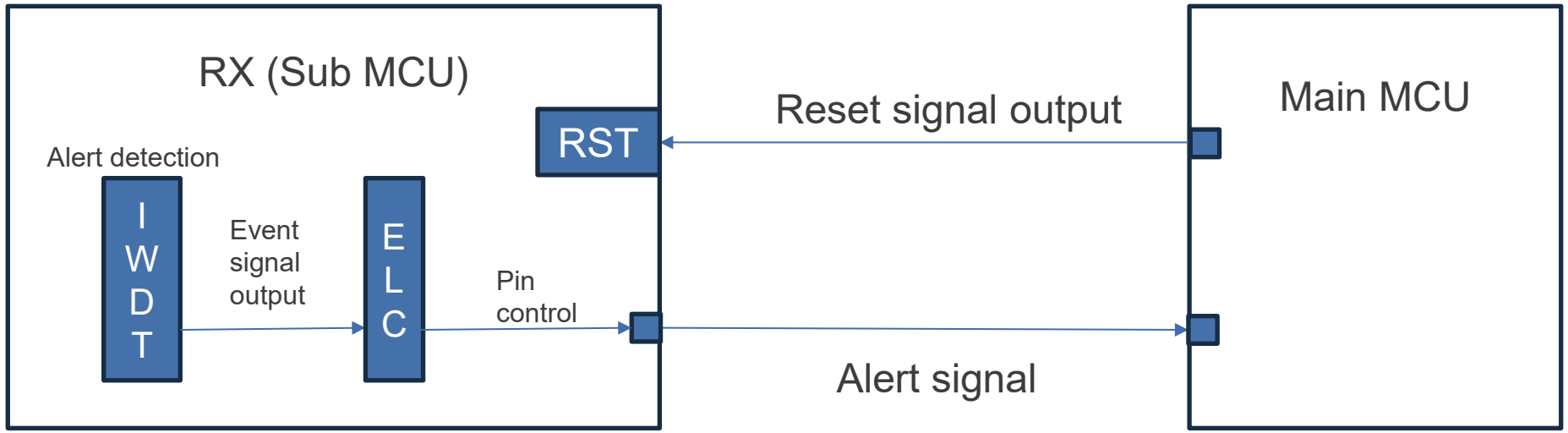
USE CASE :

OUTPUT ERROR ALERT TO EXTERNAL MAIN MCU USING EVENT LINK CONTROLLER (ELC)

IWDT

IWDT can output signals to the ELC when a refresh error or underflow occurs and can output external signals from I/O ports using the ELC without going through the CPU.

The figure below shows how to detect an alert in IWDT, send an alert notification to the external main MCU, and execute a hardware reset from the main MCU. The main MCU can check the status of the sub-MCUs and monitor the number of alerts.



*When using ELC, non-maskable interrupt request or interrupt request output must be enabled (IWDTRCR.RSTIRQS = 0).

NOTE

WDT
IWDT

- When WDT or IWDT is operated, it cannot be stopped except under the count stop condition. The stop conditions are as follows.
  - Reset
  - Low power consumption state (IWDT can be enabled/disabled by register setting)
  - In case underflow or refresh error occurs (only in register start mode)

- When debugging on the emulator, the WDT and IWDT counts stop during the break. The stopped counts are restarted during program execution.

REVISION HISTORY

Revision	Date	Contents
1.00	2025/1	First edition, issued
2.00	2026/2	Function was added (Low Power consumption, Interrupt Controller, External bus controller)