Panasonic INDUSTRY

System Design Guide for Master

RTEX Technical Reference

21 May 2024

Motion Control Business Unit, Industrial Device Business Division
Panasonic Industry Co., Ltd.

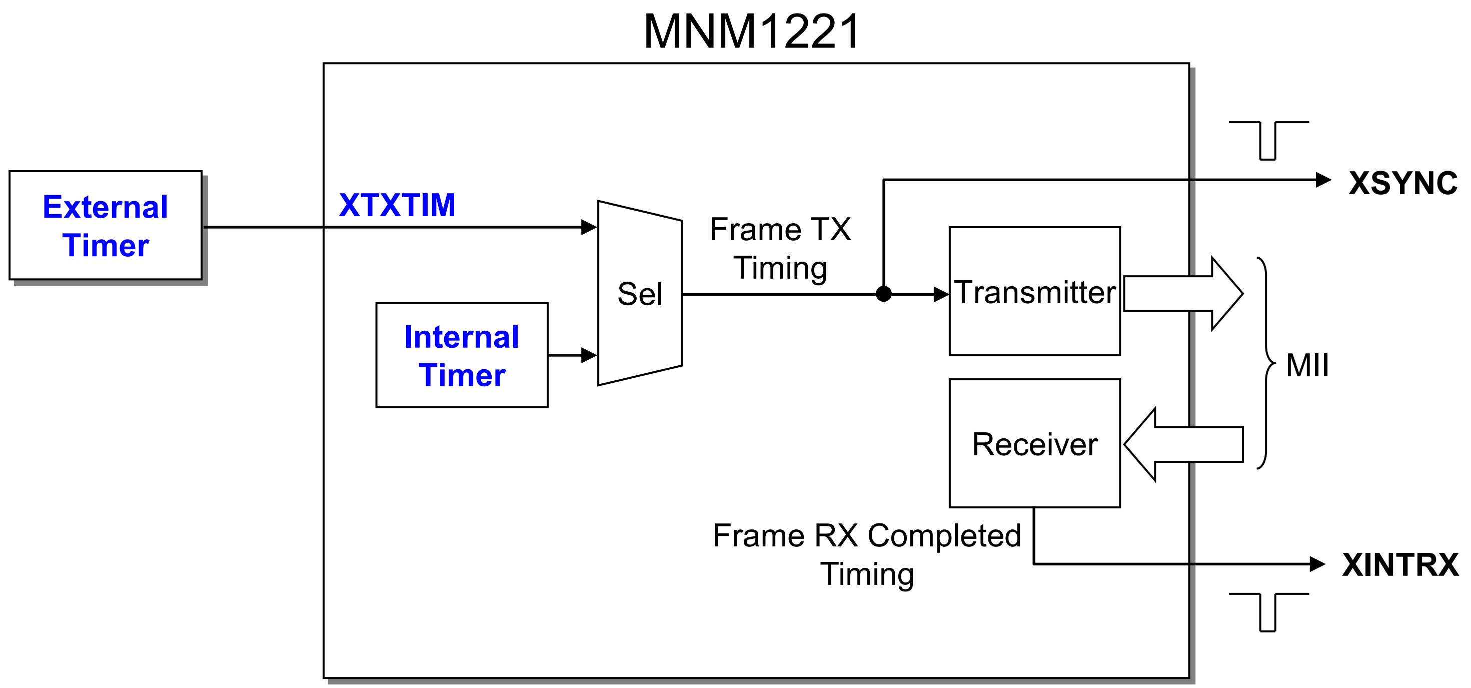
RTEX Realtime Express

Revision History

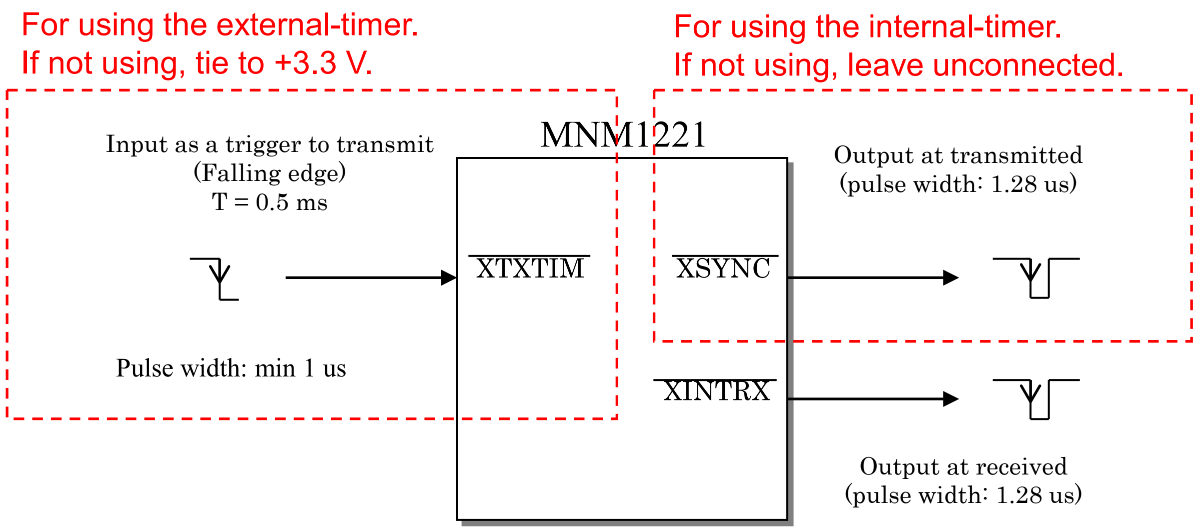
Revision	Date	Change Description
1	2005/7/14	Initial Release
2	2012/1/31	P1 Changed title from “A Guide for Firmware Development”. P3 Added introduction. P7 Added SH7216 example. P19 Added SH7145 example. P25 Added TMS320F28335 example. Deleted SH7065 example. Deleted TMS320VC33-120 example. P58 Added overview of profile position I/F. P72 Added internal-timer using system. Minor edits.
3	2024/5/21	P12-16 Added STM32H743 example. P17-38 Added TMS320F28388 example. Deleted SH7206, SH7145, and TMS320F28335 example. P48 Added the notes for “Write Buffer” and “Cache”. P51 Added the reset signal. P71 Added the description in the checking timeout. P74, 88 Replaced to A6N period setting.

Introduction

To control transmit-timing, MNM1221 has two timer sources that are an external-timer and an internal-timer.
This document describes examples of the external-timer using system in chapter 1, and the internal-timer using system in chapter 2.



Timing Signal Pins of MNM1221



Note:

If not in RUNNING state, XTXTIM input is ignored.

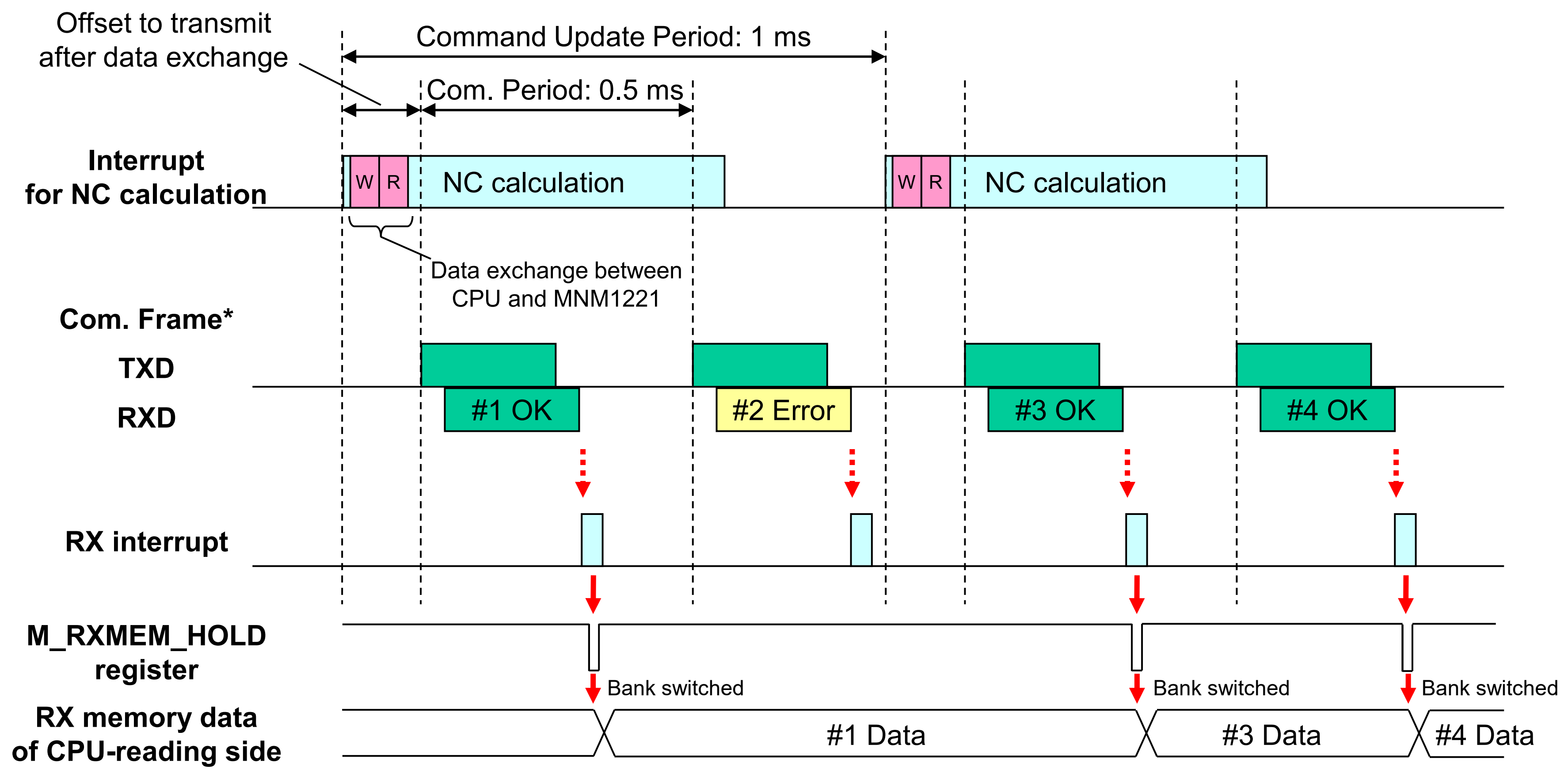
Init-A and Init-B frame in RING-CONFIG state are automatically transmitted with internal timer of MNM1221.

Chapter 1

External-Timer System

Goal of Timing Control

The goal is to make the following timing:

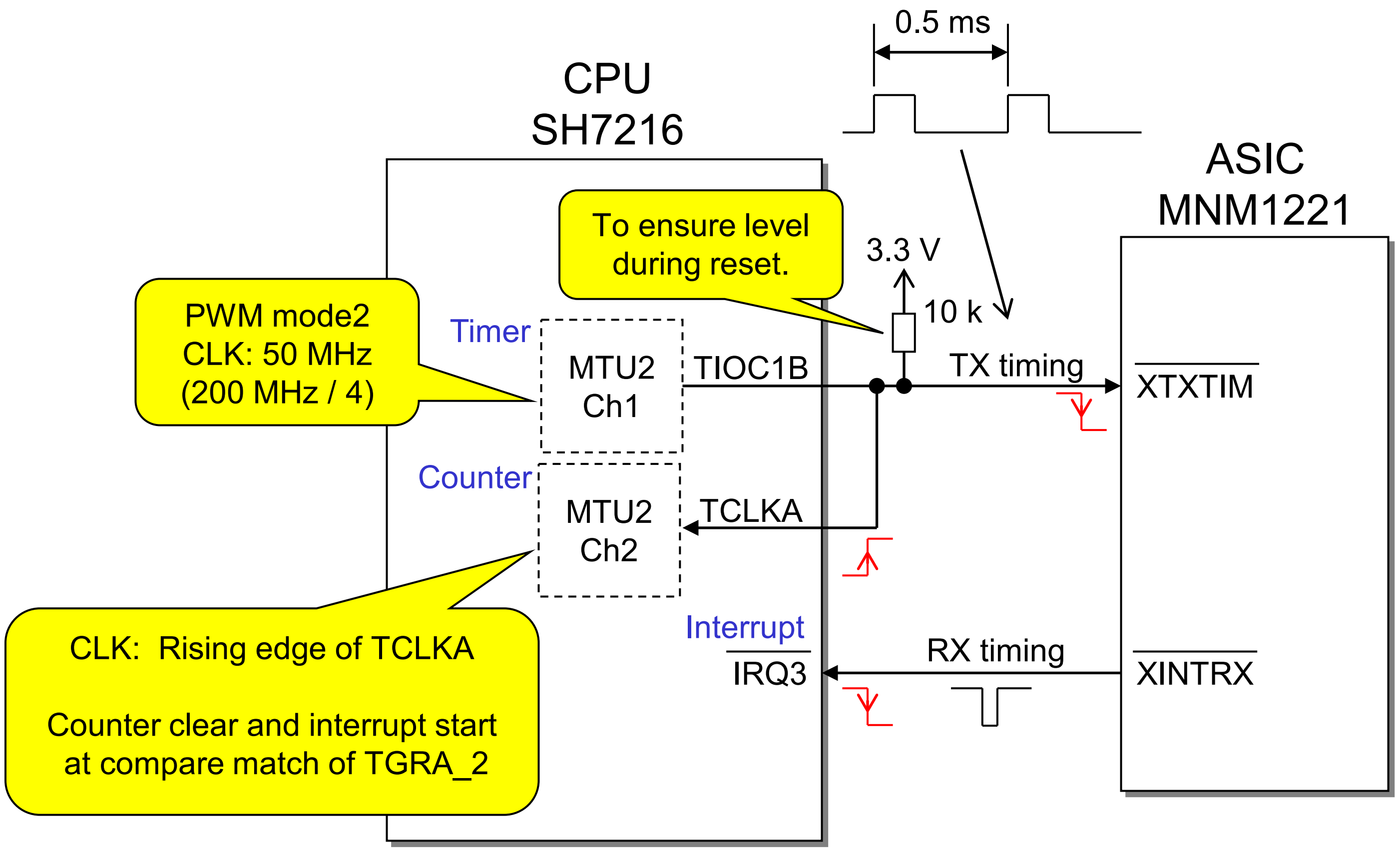


If a CRC error is detected in RX data, the RX bank is not switched in RX interrupt routine.
In this case, previous RX data is read in the data exchange at the beginning of NC calculation.

* One frame contains data of all slave nodes, and its length depends on the number of connected nodes.

Example for SH7216 (Renesas)

Timing Circuit

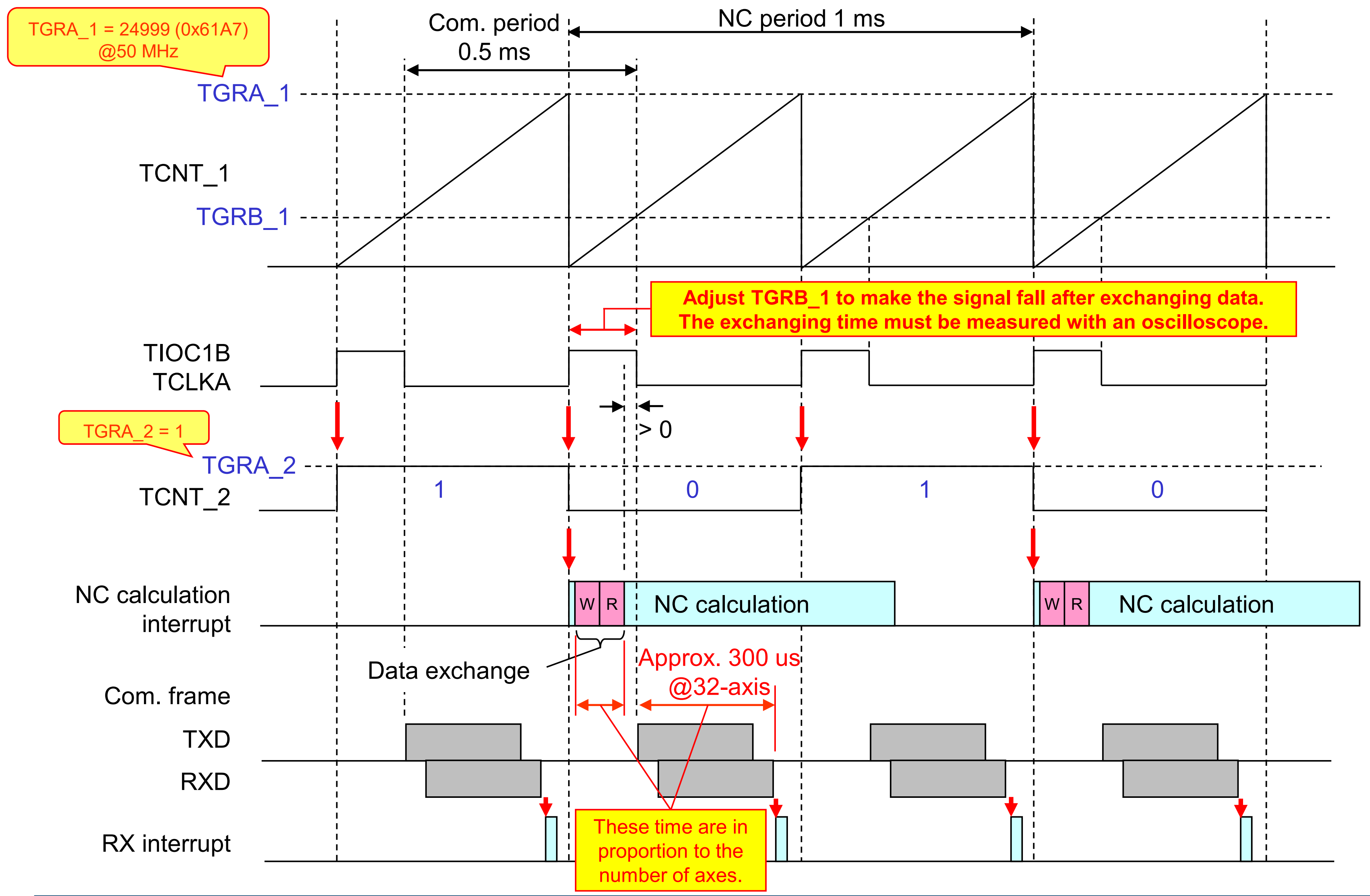


- MTU2-Ch1 generates TX timing signal. For 0.5 ms, **TGRA_1 = 24999(0x61A7)@50MHz**
- MTU2-Ch2 divides this signal, and generates the start signal for NC calculating interrupt. For 1 ms, **TGRA_2 = 1**
- IRQ3 by XINTRX of MNM1221 causes RX interrupt.

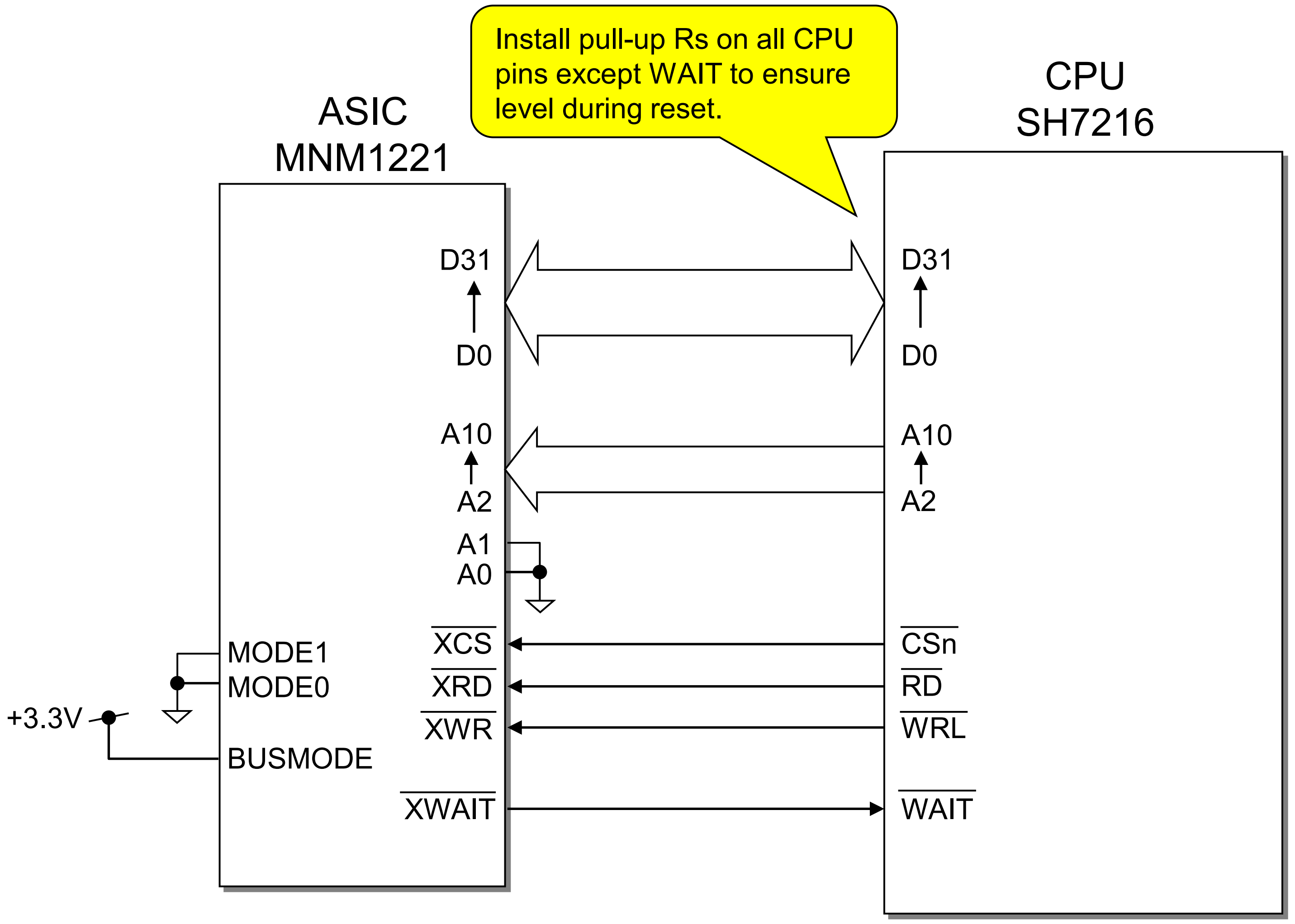
Multiplex Interrupt Setting

Source	Trigger	Priority	Period	Operation
TGIA_2	Compare match of MTU2 Ch2	-	1 ms	- Communication data exchange - NC calculation
/IRQ3	RX complete	Higher than TGIA_2	0.5 ms	- Communication status check - RX memory bank switch

Timing Chart

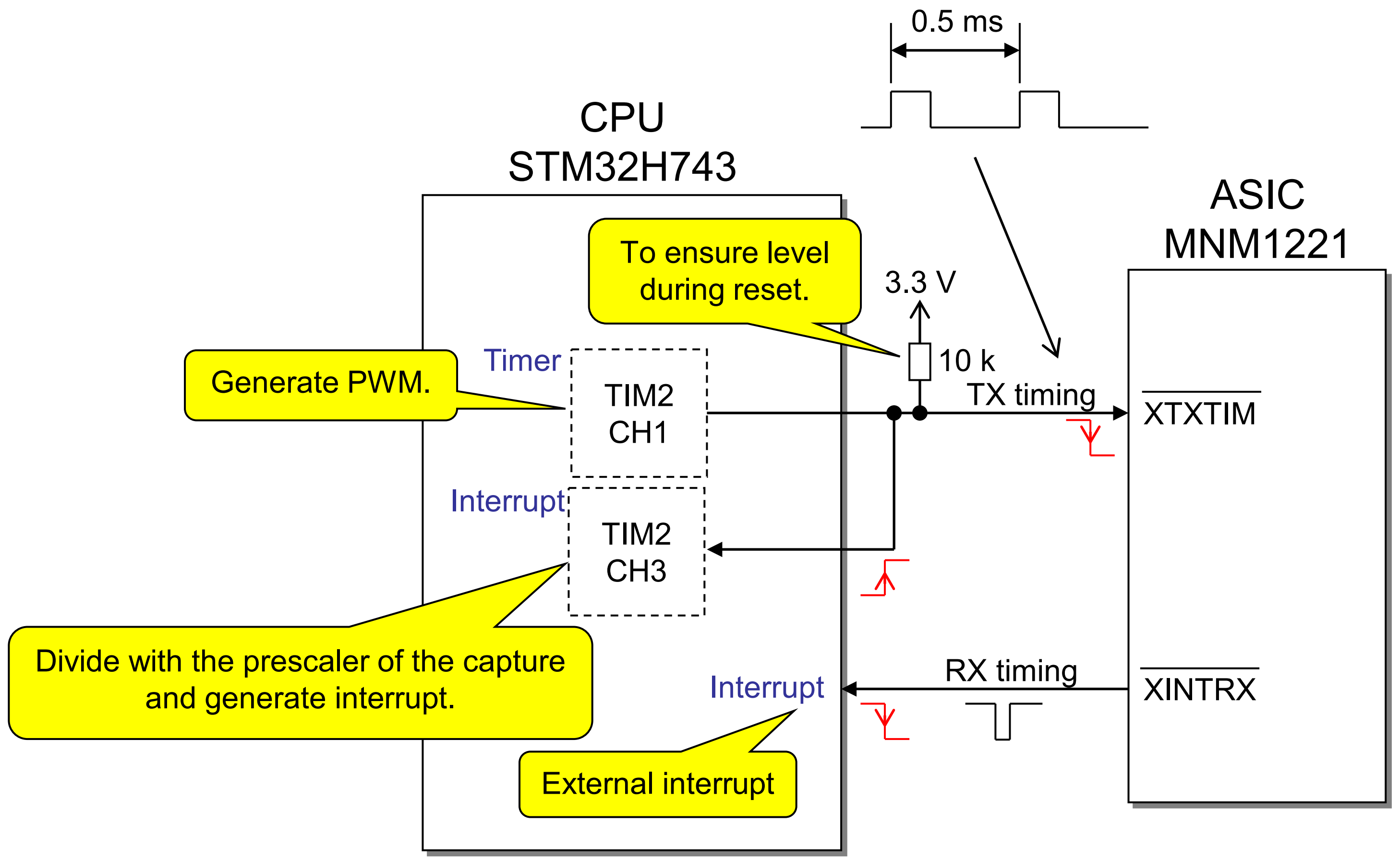


Bus Connection



Example for STM32H743 (STMicro)

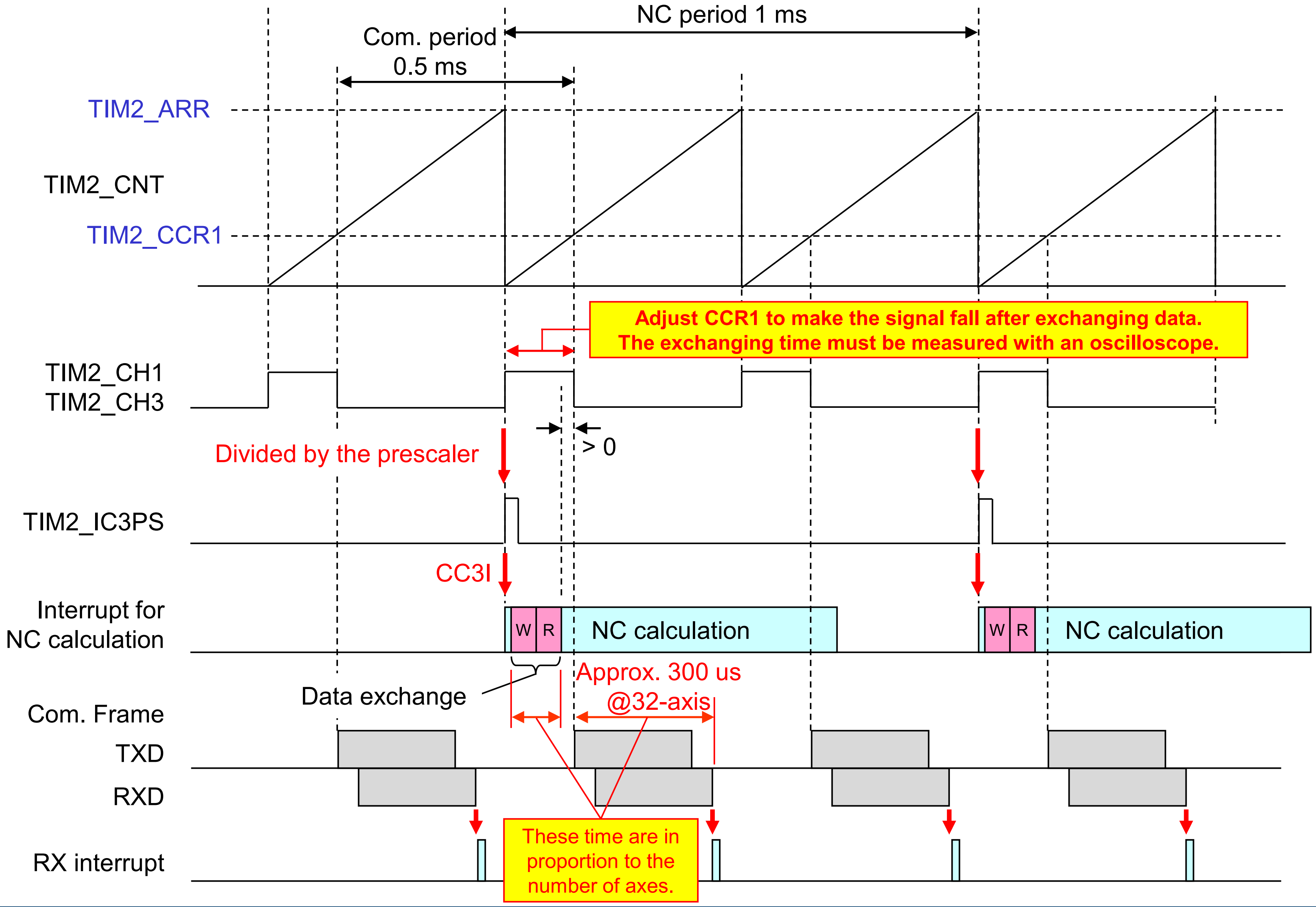
Timing Circuit



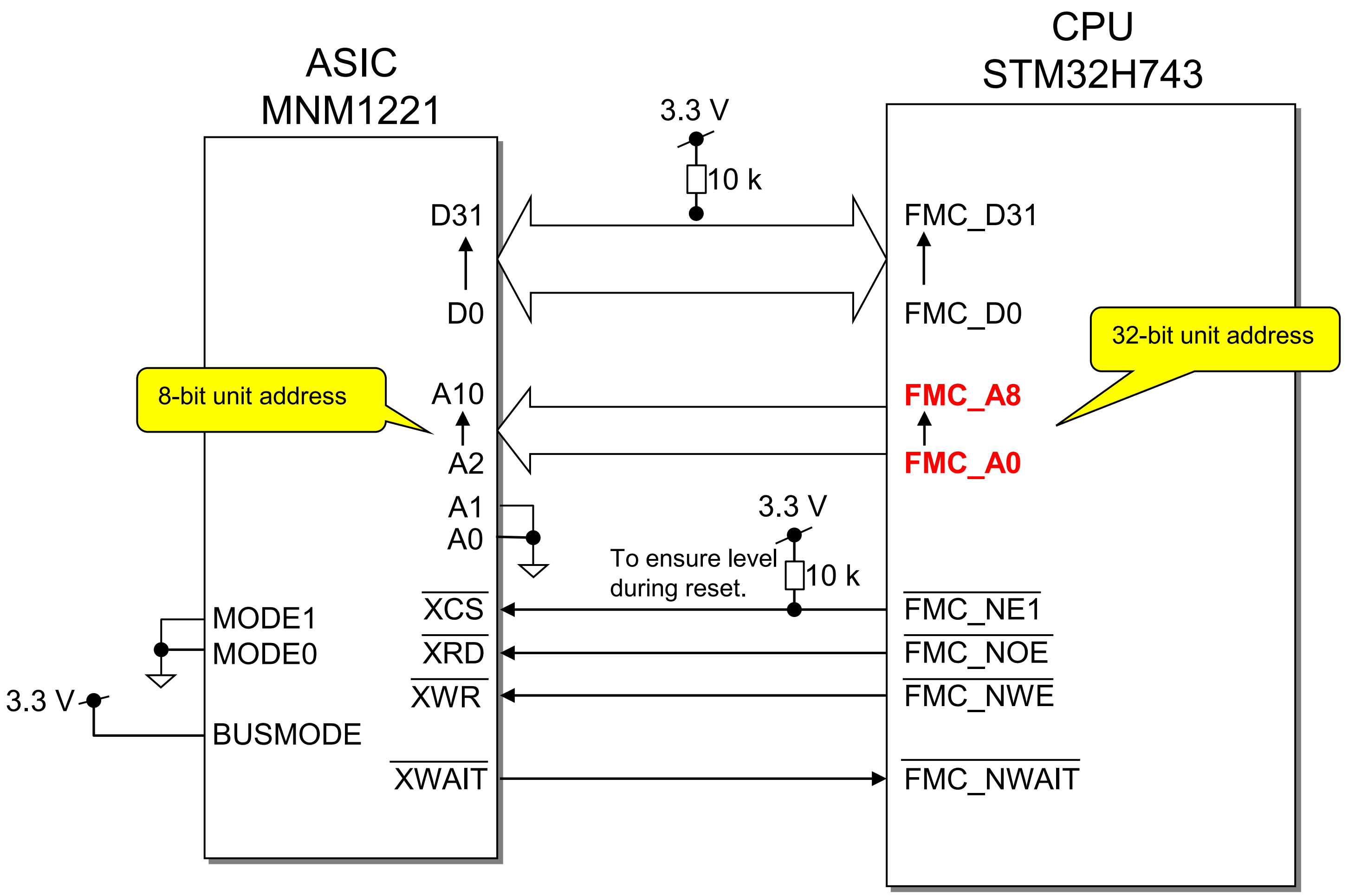
Multiplex Interrupt Setting

Source	Trigger	Priority	Period	Operation
TIM2 CC3I	Divided PWM Signal	-	1 ms	- Communication data exchange - NC calculation
External Interrupt	RX complete (XINTRX)	Higher than CC3I	0.5 ms	- Communication status check - RX memory bank switch

Timing Chart

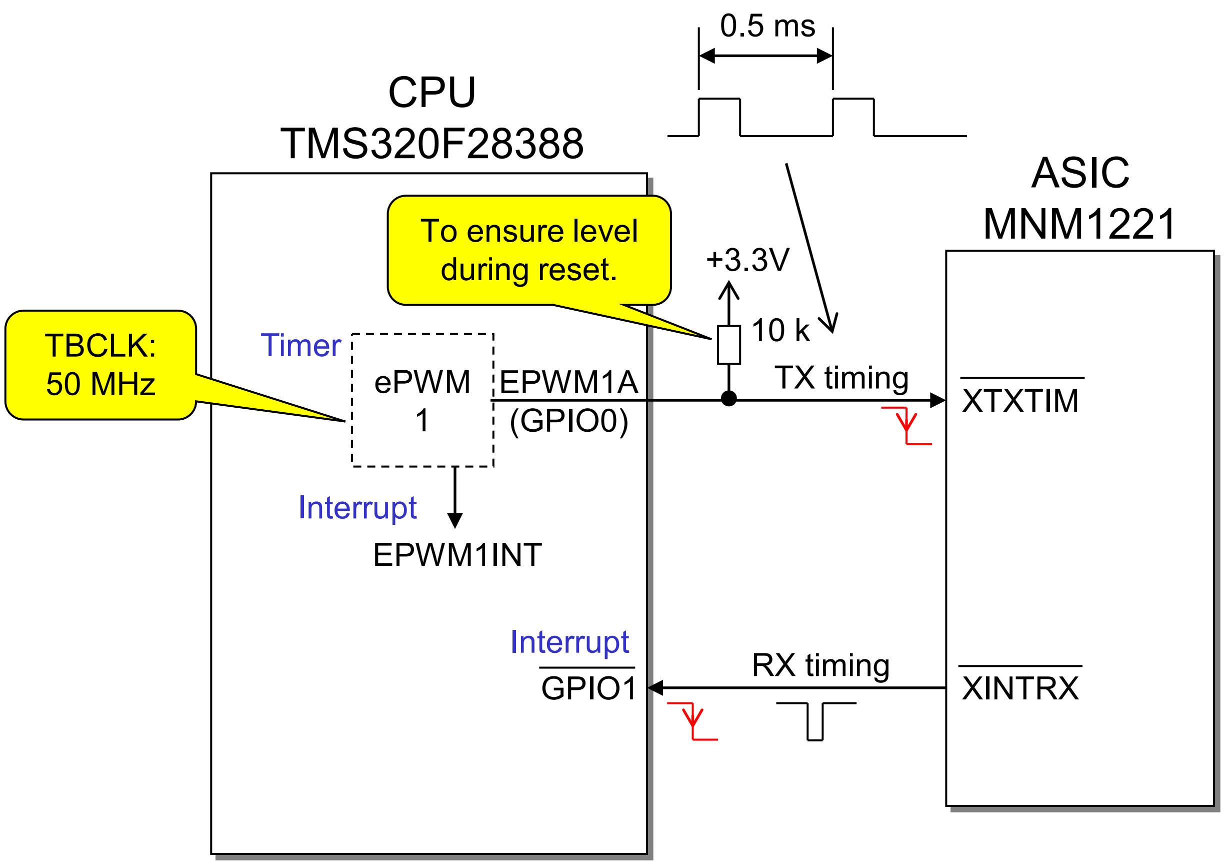


Bus Connection



Example for TMS320F28388 (TI)

Timing Circuit

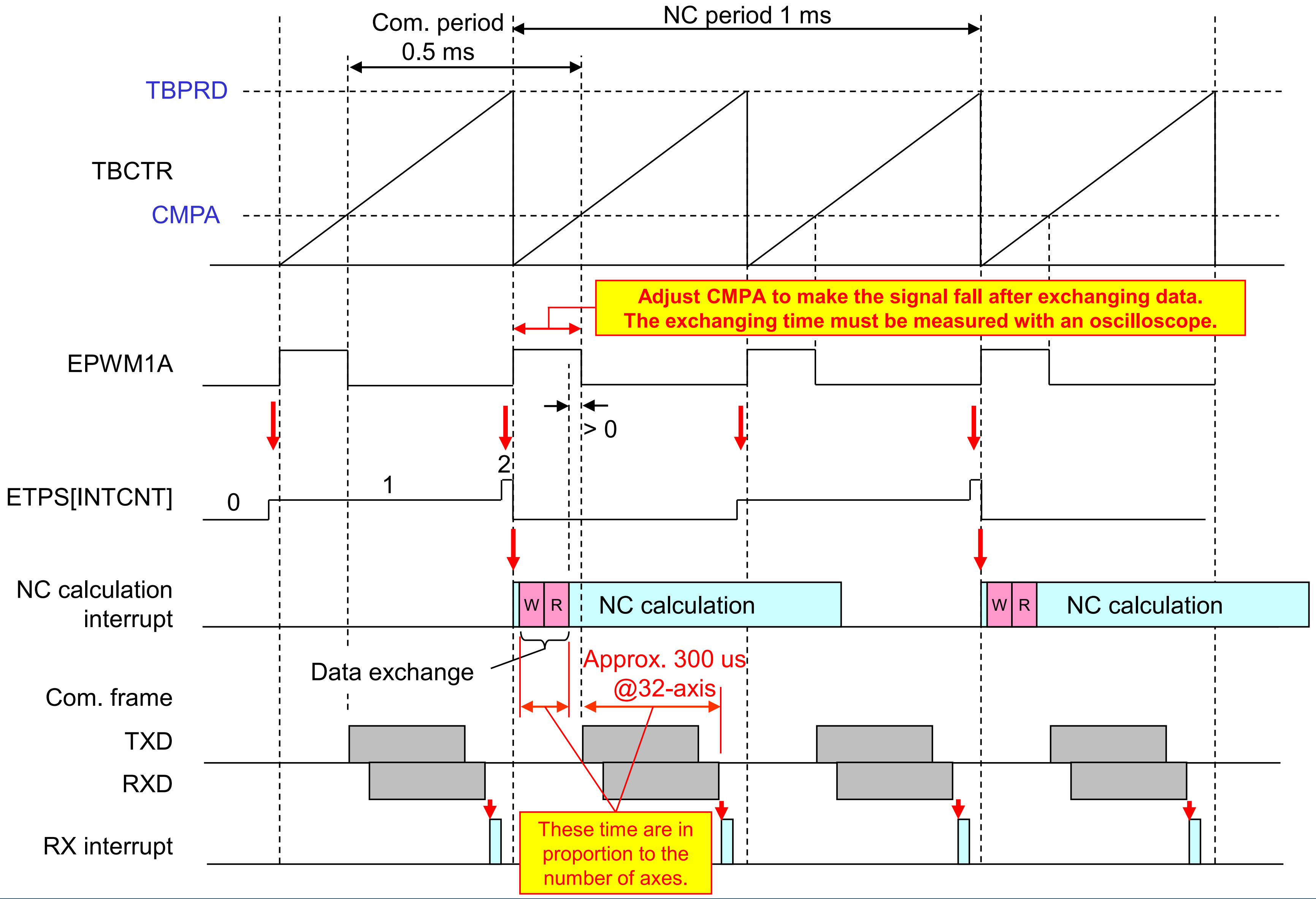


- ePWM1 generates TX timing signal. For 0.5 ms, **TBPRD = 24999 (0x61A7) @TBCLK 50 MHz**
- The interrupt generator of Event-Trigger divides this signal, and NC calculation interrupt starts.
- GPIO1 by XINTRX of MNM1221 causes RX interrupt.

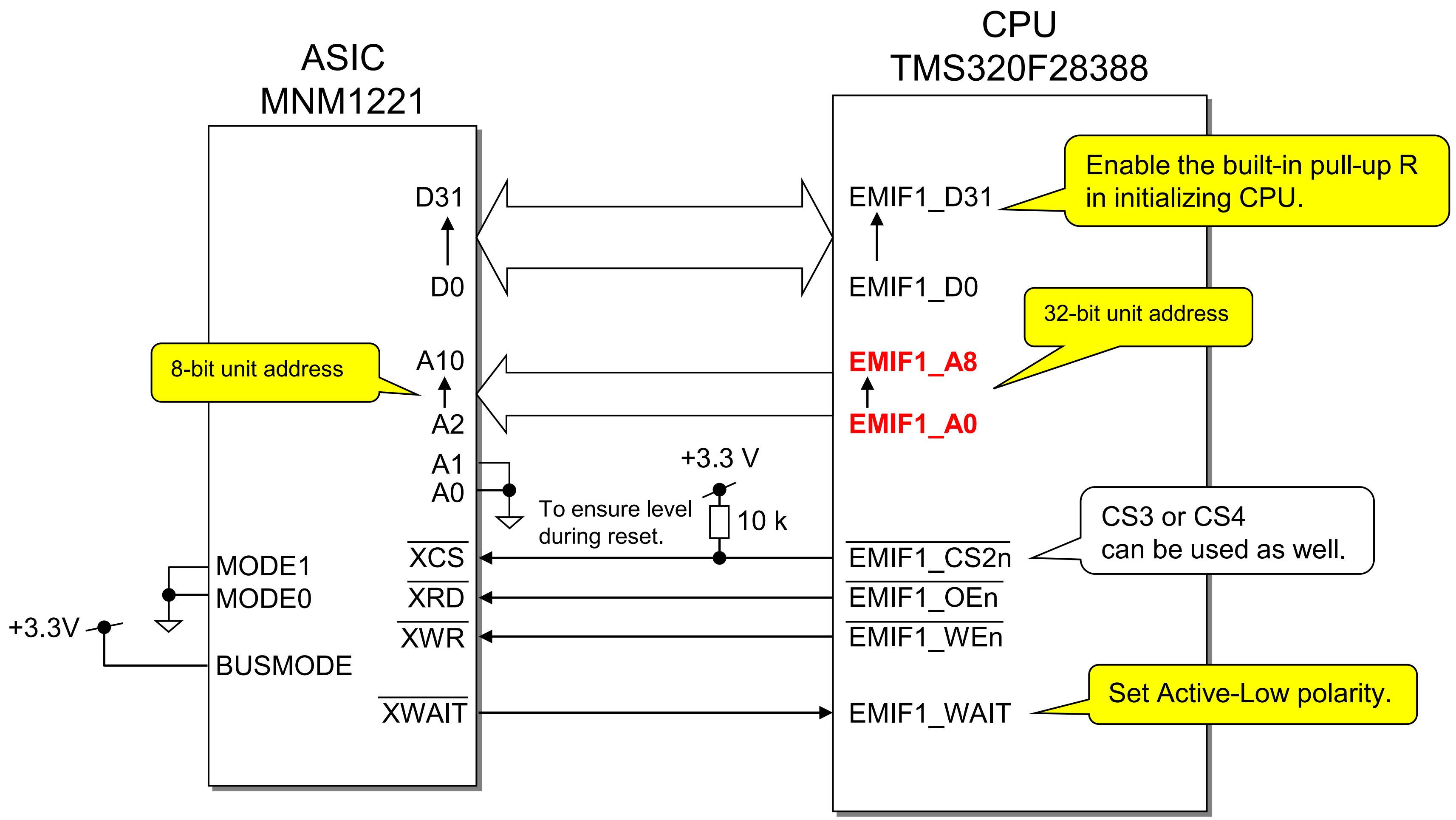
Multiplex Interrupt Setting

Source	Trigger	Priority	Period	Operation
EPWM1INT	Divide the event of "TBCTR = TBPRD"	-	1 ms	- Communication data exchange - NC calculation
XINT via GPIO1	RX complete	Higher than EPWM1INT	0.5 ms	- Communication status check - RX memory bank switch

Timing Chart



Bus Connection



Note:TMS320F28388 has 32-bit unit address bus.

Unit of EMIF1_A[x]

12.2.6.1 Interfacing to Asynchronous Memory

Figure 12-8 shows the EMIF's external pins used in interfacing with an asynchronous device. In $\overline{\text{EM1CS}}$[n], n = 2, 3, or 4.

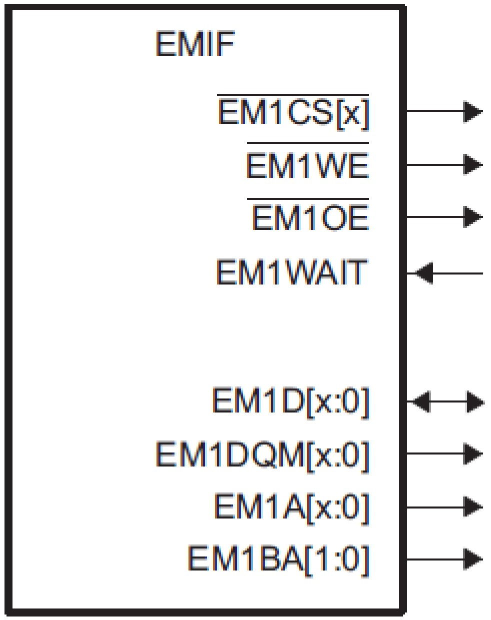
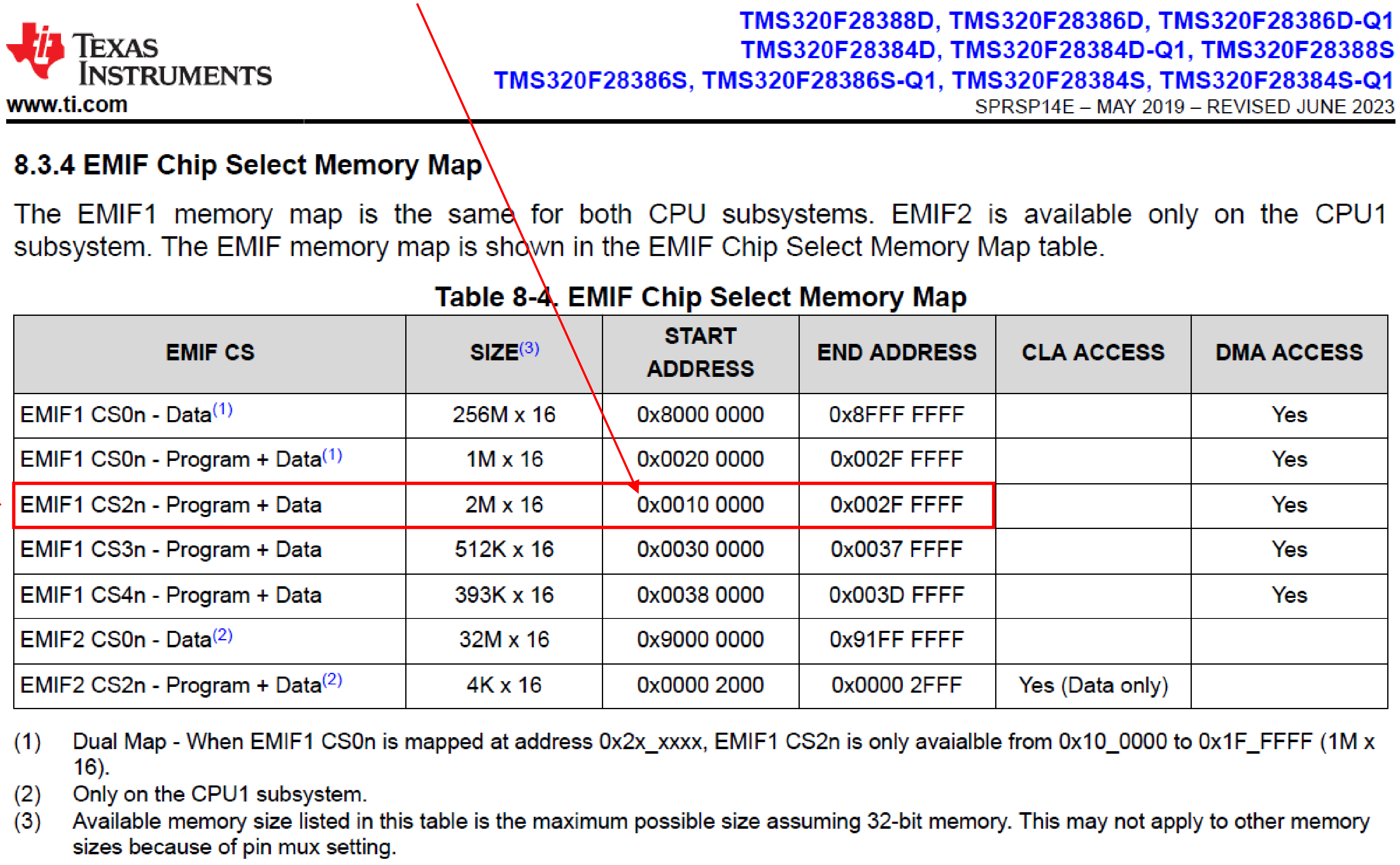

EMIF
$\overline{\text{EM1CS[x]}}$
$\overline{\text{EM1WE}}$
$\overline{\text{EM1OE}}$
EM1WAIT
EM1D[x:0]
EM1DQM[x:0]
EM1A[x:0]
EM1BA[1:0]

Figure 12-8. EMIF Asynchronous Interface

Of special note is the connection between the EMIF and the external device's address bus. The EMIF address pin EM1A[0] always provides the least-significant bit of a 32-bit word address. Therefore, when interfacing to a 16-bit or 8-bit asynchronous device, the EM1BA[1] and EM1BA[0] pins provide the least-significant bits of the halfword or byte address, respectively. Figure 12-9 and Figure 12-10 show the mapping between the EMIF and the connected device's data and address pins for various programmed data bus widths. The data bus width can be configured in the asynchronous *n* configuration register (ASYNC_CS*n*_CR).

Memory Map

16-bit unit address



TMS320F28388D, TMS320F28386D, TMS320F28386D-Q1
TMS320F28384D, TMS320F28384D-Q1, TMS320F28388S
TMS320F28386S, TMS320F28386S-Q1, TMS320F28384S, TMS320F28384S-Q1

www.ti.com

SPRSP14E – MAY 2019 – REVISED JUNE 2023

8.3.4 EMIF Chip Select Memory Map

The EMIF1 memory map is the same for both CPU subsystems. EMIF2 is available only on the CPU1 subsystem. The EMIF memory map is shown in the EMIF Chip Select Memory Map table.

Table 8-4. EMIF Chip Select Memory Map

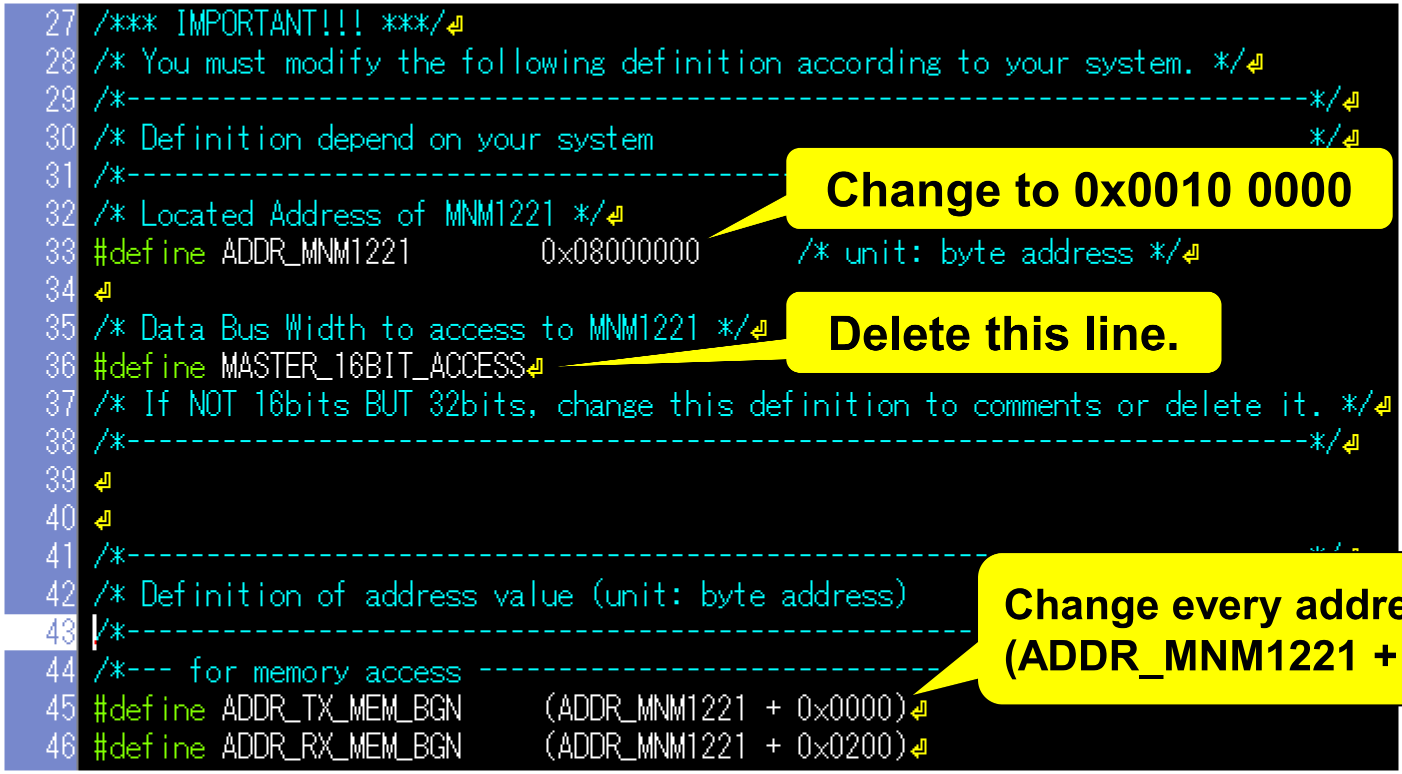
EMIF CS	SIZE[3]	START ADDRESS	END ADDRESS	CLA ACCESS	DMA ACCESS
EMIF1 CS0n - Data[1]	256M x 16	0x8000 0000	0x8FFF FFFF		Yes
EMIF1 CS0n - Program + Data[1]	1M x 16	0x0020 0000	0x002F FFFF		Yes
EMIF1 CS2n - Program + Data	2M x 16	0x0010 0000	0x002F FFFF		Yes
EMIF1 CS3n - Program + Data	512K x 16	0x0030 0000	0x0037 FFFF		Yes
EMIF1 CS4n - Program + Data	393K x 16	0x0038 0000	0x003D FFFF		Yes
EMIF2 CS0n - Data[2]	32M x 16	0x9000 0000	0x91FF FFFF		
EMIF2 CS2n - Program + Data[2]	4K x 16	0x0000 2000	0x0000 2FFF	Yes (Data only)	

(1) Dual Map - When EMIF1 CS0n is mapped at address 0x2x_xxxx, EMIF1 CS2n is only avaialble from 0x10_0000 to 0x1F_FFFF (1M x 16).
(2) Only on the CPU1 subsystem.
(3) Available memory size listed in this table is the maximum possible size assuming 32-bit memory. This may not apply to other memory sizes because of pin mux setting.

Modify “mnm1221_m.h”

Although the example code defines addresses with 8-bit unit, TMS320F28388 has 16-bit unit. Therefore, modify every address definition as follows:

mnm1221_m.h



```
/*** IMPORTANT!!! ***/
/* You must modify the following definition according to your system. */
/*--------------------------------------------------------------------------*/
/* Definition depend on your system                                         */
/*----------------------------------------
/* Located Address of MNM1221 */
#define ADDR_MNM1221        0x08000000       /* unit: byte address */

/* Data Bus Width to access to MNM1221 */
#define MASTER_16BIT_ACCESS
/* If NOT 16bits BUT 32bits, change this definition to comments or delete it. */
/*--------------------------------------------------------------------------*/


/*------------------------------------------------
/* Definition of address value (unit: byte address)
/*------------------------------------------------
/*--- for memory access ---------------------------
#define ADDR_TX_MEM_BGN     (ADDR_MNM1221 + 0x0000)
#define ADDR_RX_MEM_BGN     (ADDR_MNM1221 + 0x0200)
```

Change to 0x0010 0000

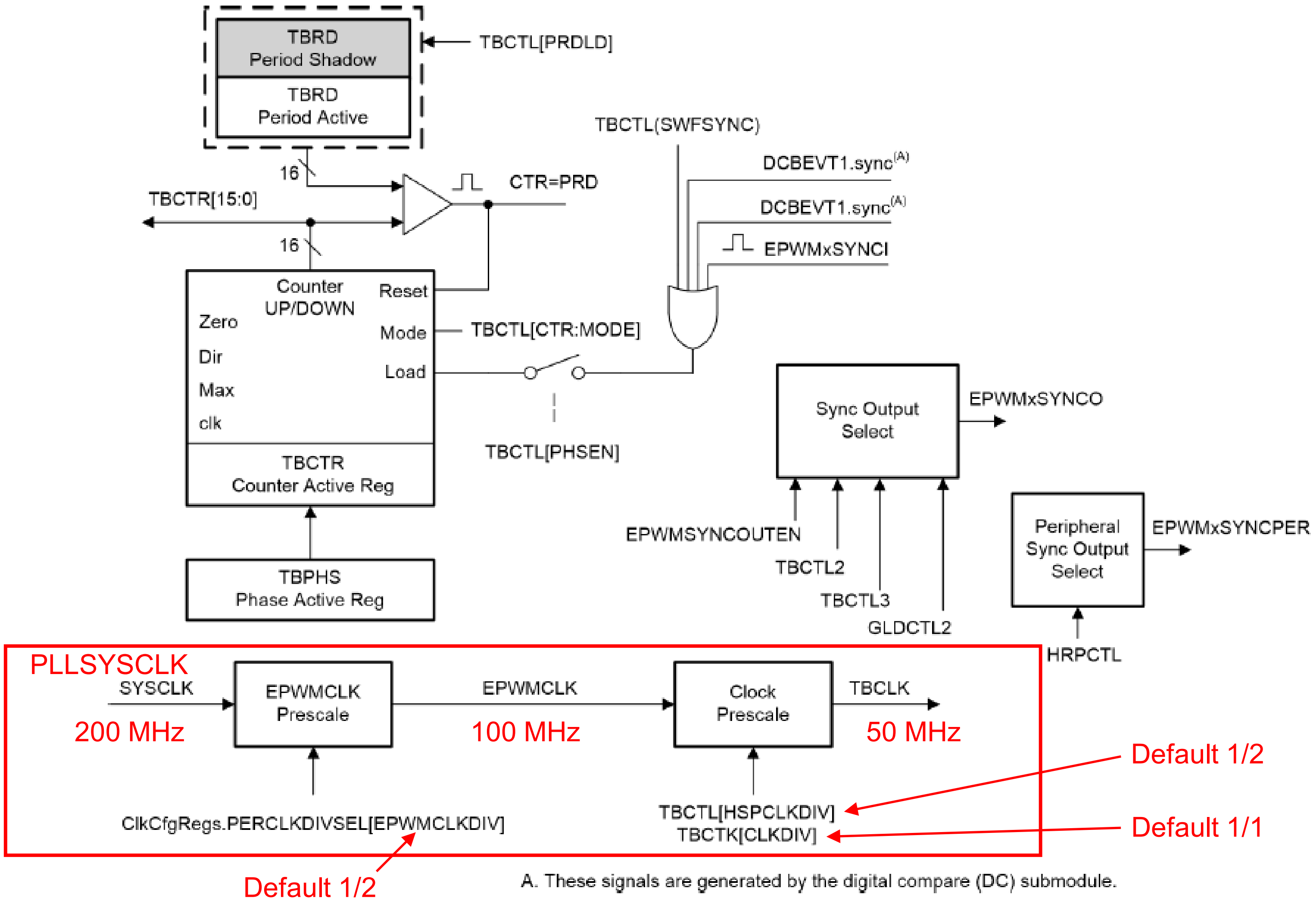
Delete this line.

Change every address like this:
(ADDR_MNM1221 + (0x0000 >> 1))

Details of TMS320F28388 Configuration

Source:
TMS320F2838x Technical Reference Manual SPRUII0E

TBCLK Setting



A. These signals are generated by the digital compare (DC) submodule.

Figure 26-5. Time-Base Submodule Signals and Registers

TBCTL register

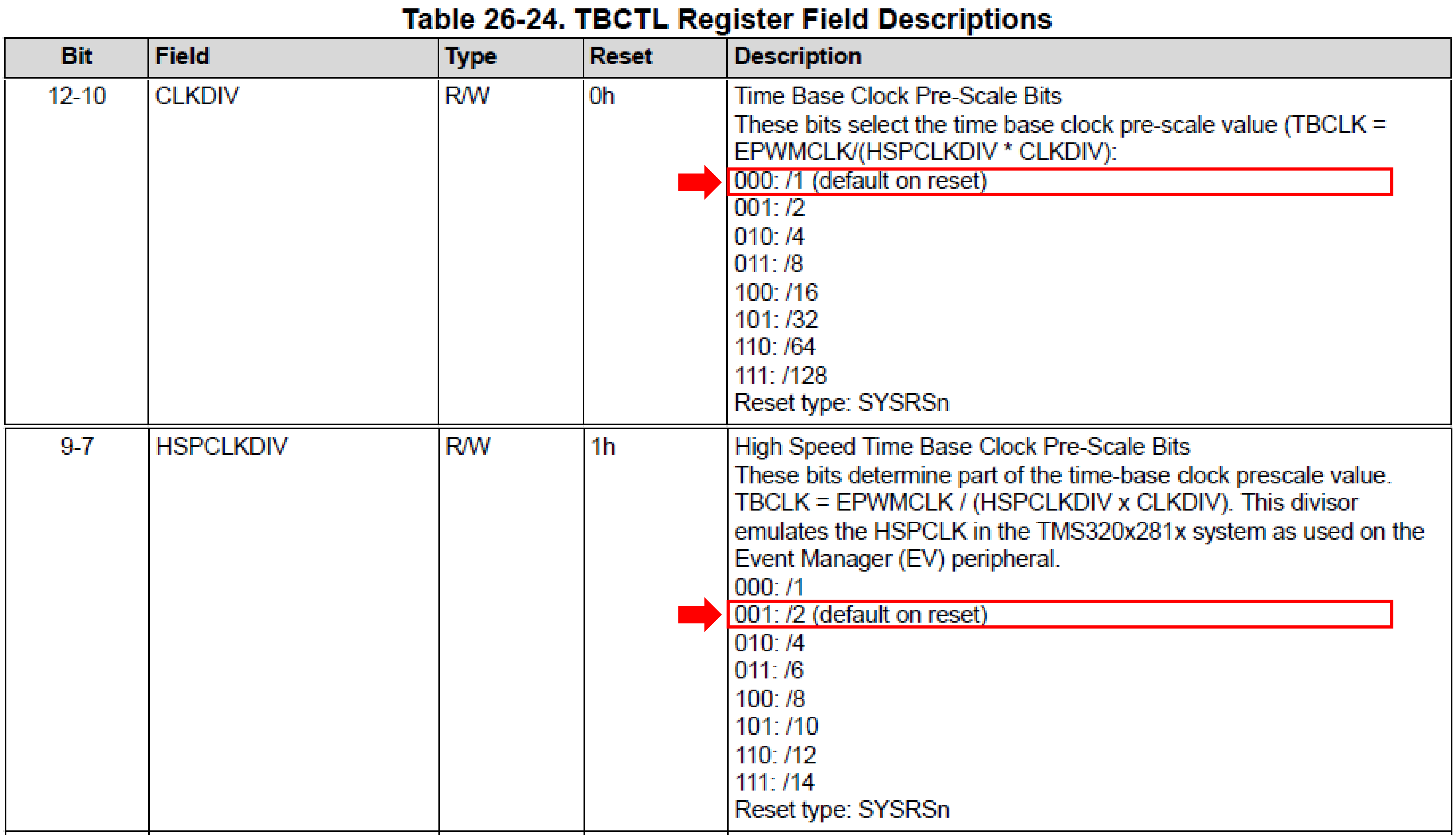


Table 26-24. TBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
12-10	CLKDIV	R/W	0h	Time Base Clock Pre-Scale Bits These bits select the time base clock pre-scale value (TBCLK = EPWMCLK/(HSPCLKDIV * CLKDIV): 000: /1 (default on reset) 001: /2 010: /4 011: /8 100: /16 101: /32 110: /64 111: /128 Reset type: SYSRSn
9-7	HSPCLKDIV	R/W	1h	High Speed Time Base Clock Pre-Scale Bits These bits determine part of the time-base clock prescale value. TBCLK = EPWMCLK / (HSPCLKDIV x CLKDIV). This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral. 000: /1 001: /2 (default on reset) 010: /4 011: /6 100: /8 101: /10 110: /12 111: /14 Reset type: SYSRSn

PWM Period Setting

Up-Count Mode: In up-count mode, the time-base counter starts from zero and increments until the counter reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.

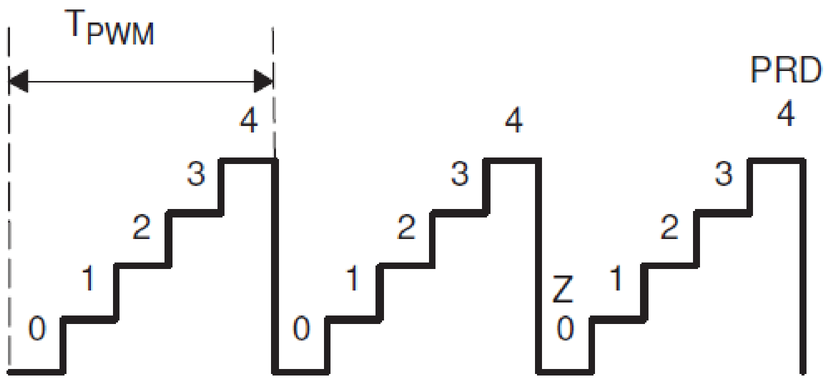


Figure 26-6. Time-Base Frequency and Period

For Up Count and Down Count

$T_{PWM} = (TBPRD + 1) \times T_{TBCLK}$

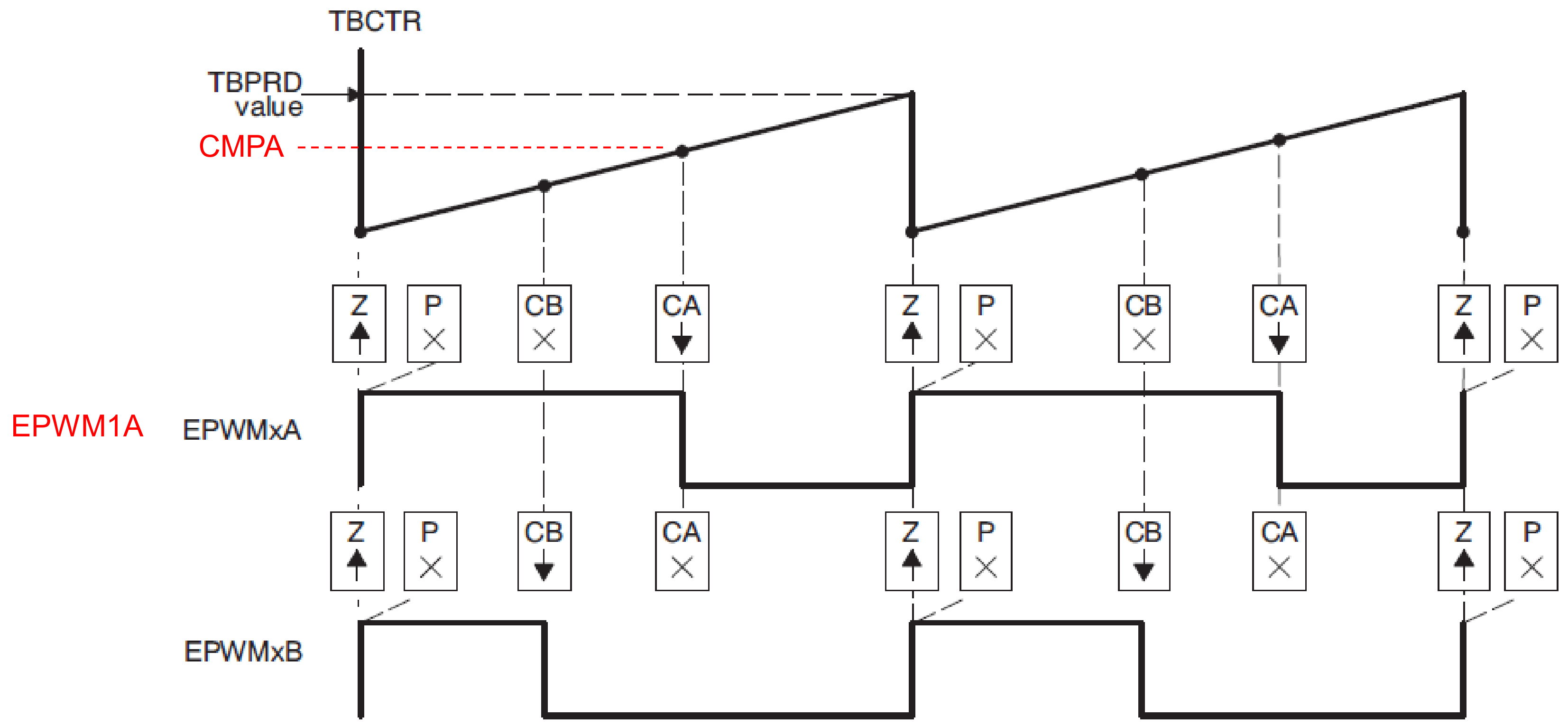
$F_{PWM} = 1/(T_{PWM})$

Setting for 0.5 ms:

TBCLK	TBPRD
50 MHz	24999 (0x61A7)

TBCLK = EPWMCLK / (HSPCLKDIV x CLKDIV)

Generating PWM Signal



A. PWM period = (TBPRD + 1) × T_{TBCLK}

B. Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).

C. Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).

D. The "Do Nothing" actions (X) are shown for completeness, but are not shown on subsequent diagrams.

E. Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

Figure 26-27. Up, Single Edge Asymmetric Waveform, with Independent Modulation on EPWMxA and EPWMxB—Active High

Generating Interrupt Start Signal

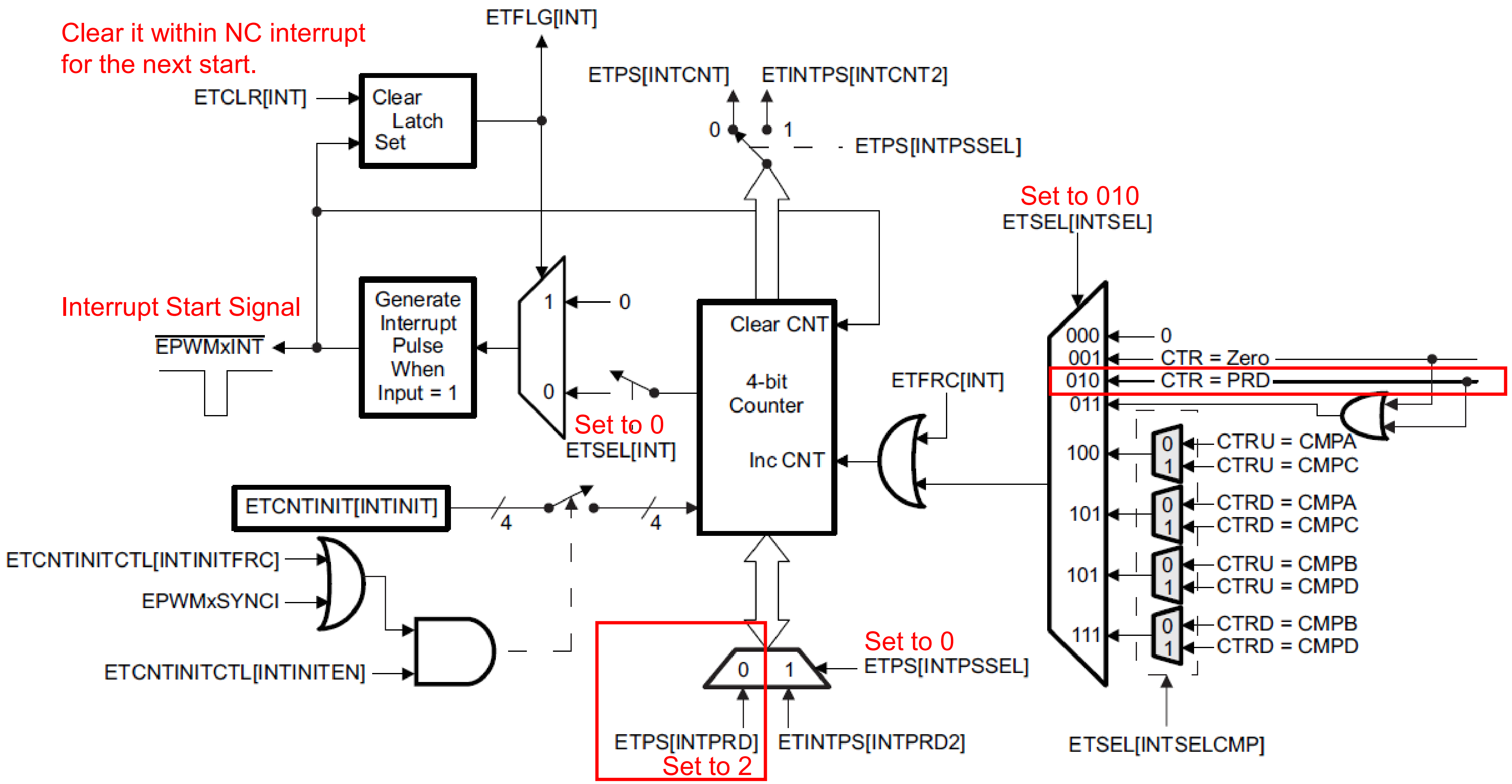


Figure 26-47. Event-Trigger Interrupt Generator

ETSEL register

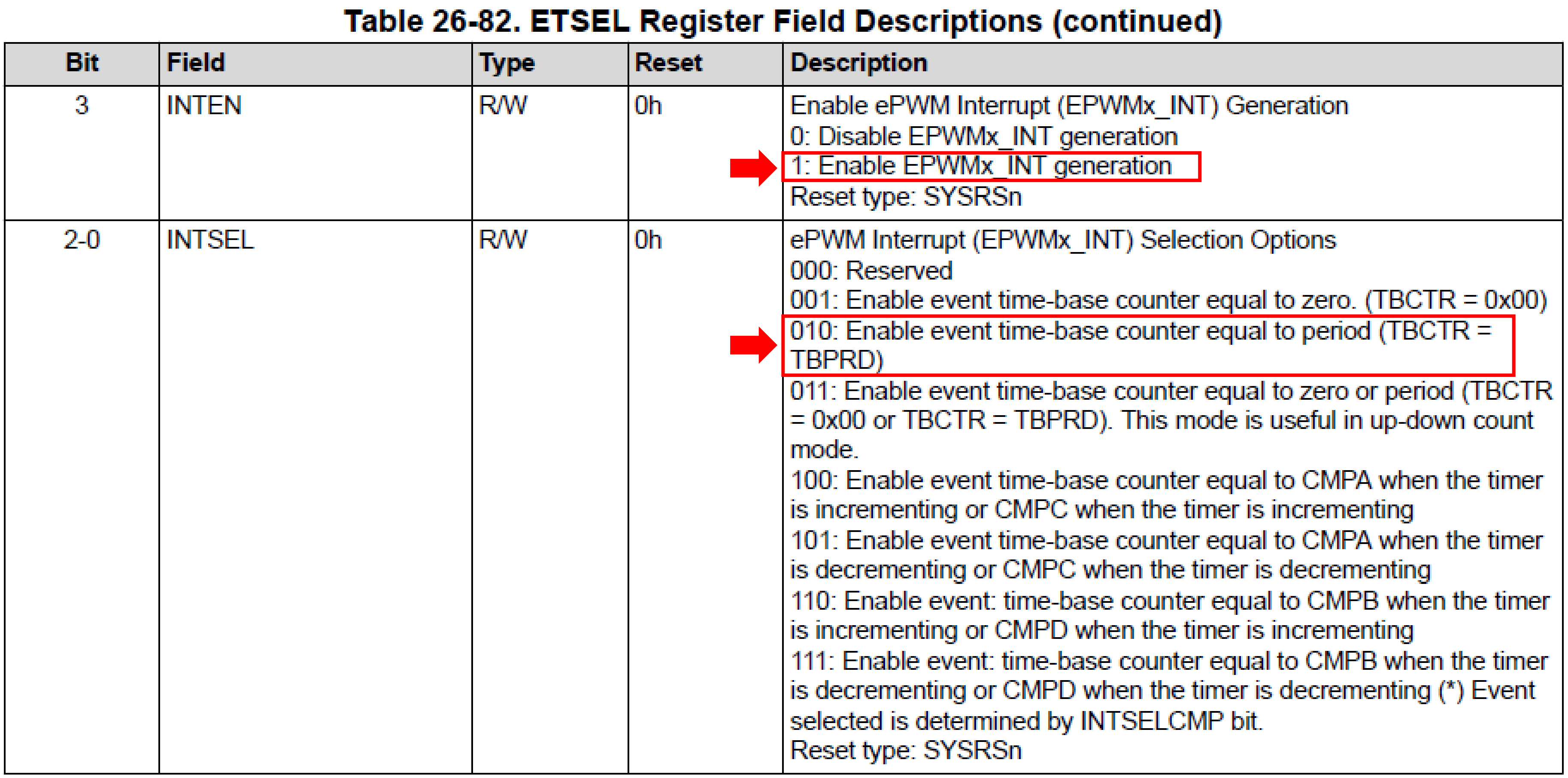


Table 26-82. ETSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	INTEN	R/W	0h	Enable ePWM Interrupt (EPWMx_INT) Generation 0: Disable EPWMx_INT generation 1: Enable EPWMx_INT generation Reset type: SYSRSn
2-0	INTSEL	R/W	0h	ePWM Interrupt (EPWMx_INT) Selection Options 000: Reserved 001: Enable event time-base counter equal to zero. (TBCTR = 0x00) 010: Enable event time-base counter equal to period (TBCTR = TBPRD) 011: Enable event time-base counter equal to zero or period (TBCTR = 0x00 or TBCTR = TBPRD). This mode is useful in up-down count mode. 100: Enable event time-base counter equal to CMPA when the timer is incrementing or CMPC when the timer is incrementing 101: Enable event time-base counter equal to CMPA when the timer is decrementing or CMPC when the timer is decrementing 110: Enable event: time-base counter equal to CMPB when the timer is incrementing or CMPD when the timer is incrementing 111: Enable event: time-base counter equal to CMPB when the timer is decrementing or CMPD when the timer is decrementing (*) Event selected is determined by INTSELCMP bit. Reset type: SYSRSn

ETPS register

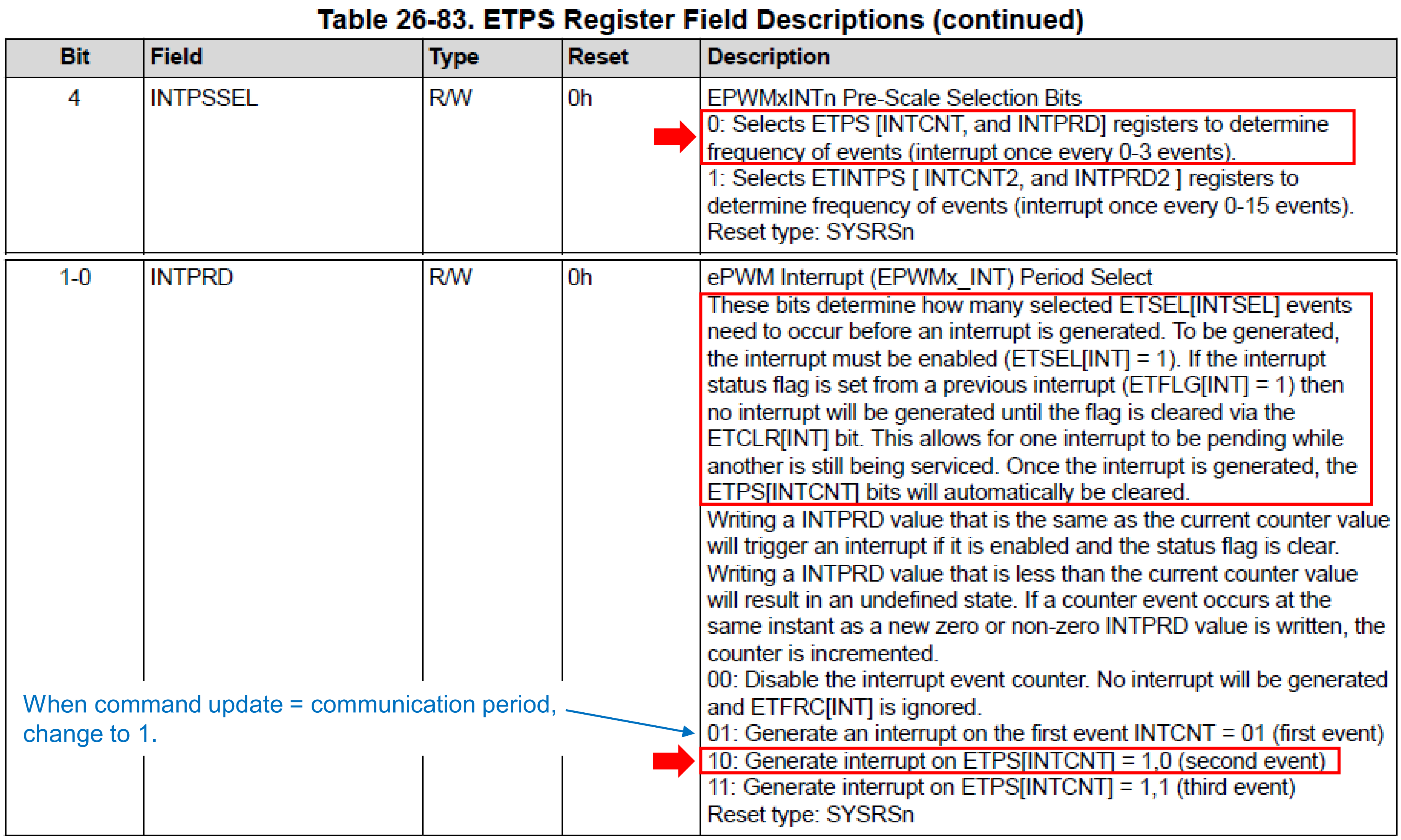


Table 26-83. ETPS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	INTPSSEL	R/W	0h	EPWMxINTn Pre-Scale Selection Bits 0: Selects ETPS [INTCNT, and INTPRD] registers to determine frequency of events (interrupt once every 0-3 events). 1: Selects ETINTPS [INTCNT2, and INTPRD2] registers to determine frequency of events (interrupt once every 0-15 events). Reset type: SYSRSn
1-0	INTPRD	R/W	0h	ePWM Interrupt (EPWMx_INT) Period Select These bits determine how many selected ETSEL[INTSEL] events need to occur before an interrupt is generated. To be generated, the interrupt must be enabled (ETSEL[INT] = 1). If the interrupt status flag is set from a previous interrupt (ETFLG[INT] = 1) then no interrupt will be generated until the flag is cleared via the ETCLR[INT] bit. This allows for one interrupt to be pending while another is still being serviced. Once the interrupt is generated, the ETPS[INTCNT] bits will automatically be cleared. Writing a INTPRD value that is the same as the current counter value will trigger an interrupt if it is enabled and the status flag is clear. Writing a INTPRD value that is less than the current counter value will result in an undefined state. If a counter event occurs at the same instant as a new zero or non-zero INTPRD value is written, the counter is incremented. 00: Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored. 01: Generate an interrupt on the first event INTCNT = 01 (first event) 10: Generate interrupt on ETPS[INTCNT] = 1,0 (second event) 11: Generate interrupt on ETPS[INTCNT] = 1,1 (third event) Reset type: SYSRSn

When command update = communication period, change to 1.

ETCLR register

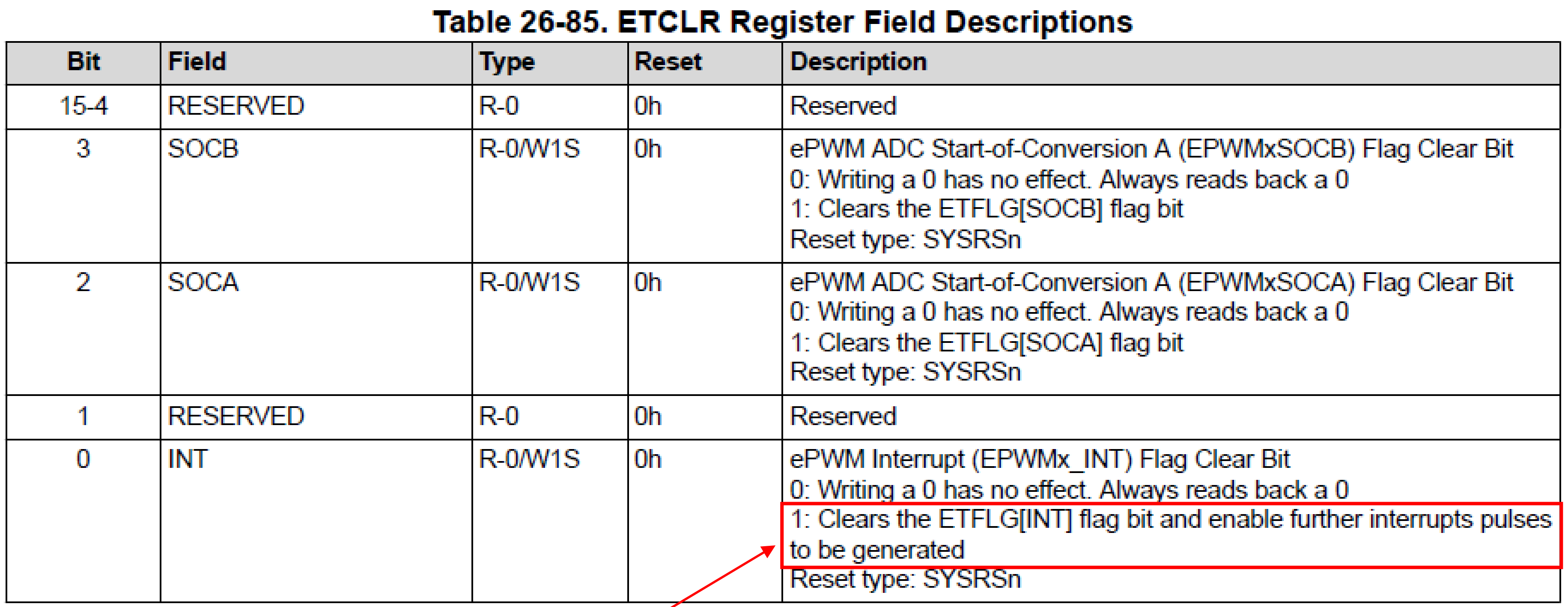


Table 26-85. ETCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R-0	0h	Reserved
3	SOCB	R-0/W1S	0h	ePWM ADC Start-of-Conversion A (EPWMxSOCB) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[SOCB] flag bit Reset type: SYSRSn
2	SOCA	R-0/W1S	0h	ePWM ADC Start-of-Conversion A (EPWMxSOCA) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[SOCA] flag bit Reset type: SYSRSn
1	RESERVED	R-0	0h	Reserved
0	INT	R-0/W1S	0h	ePWM Interrupt (EPWMx_INT) Flag Clear Bit 0: Writing a 0 has no effect. Always reads back a 0 1: Clears the ETFLG[INT] flag bit and enable further interrupts pulses to be generated Reset type: SYSRSn

Set to 1 within NC interrupt
for the next start.

PCLKCR1 register

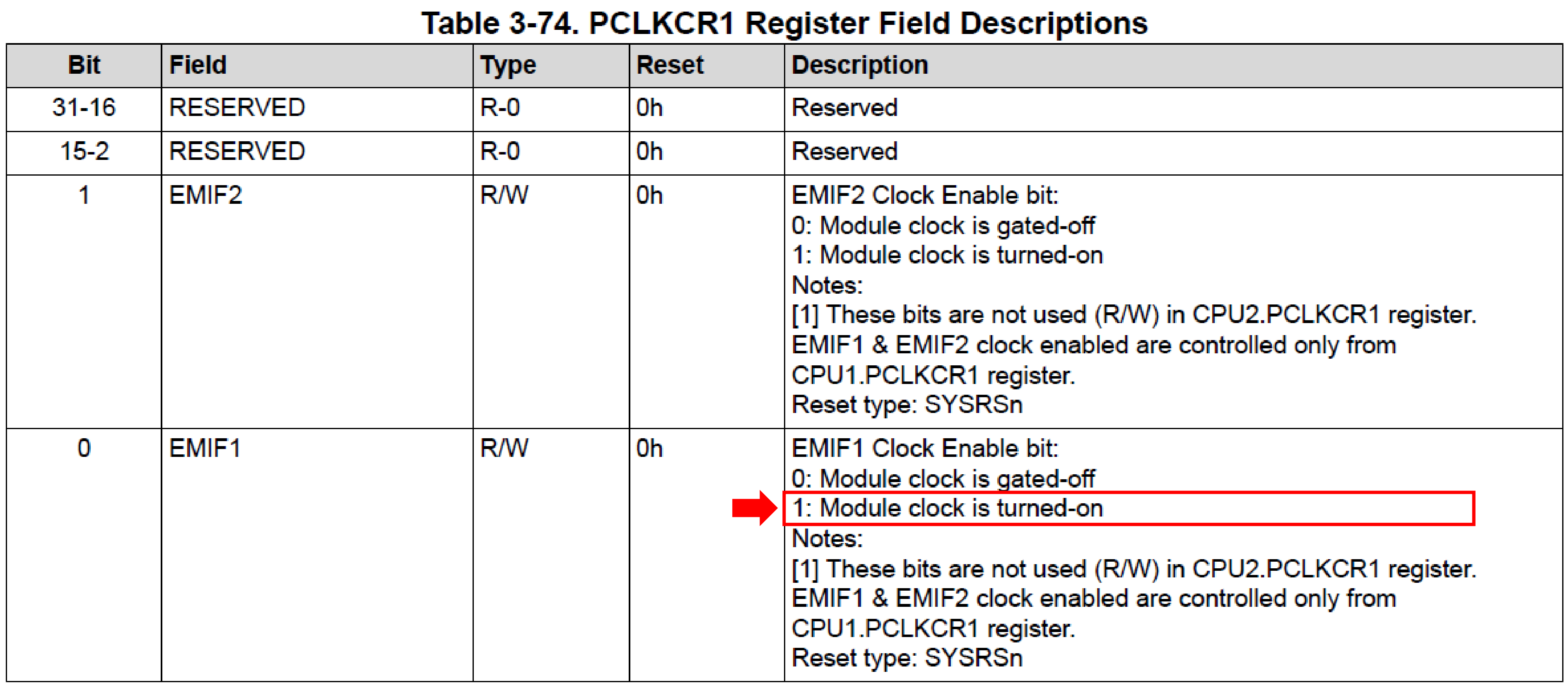
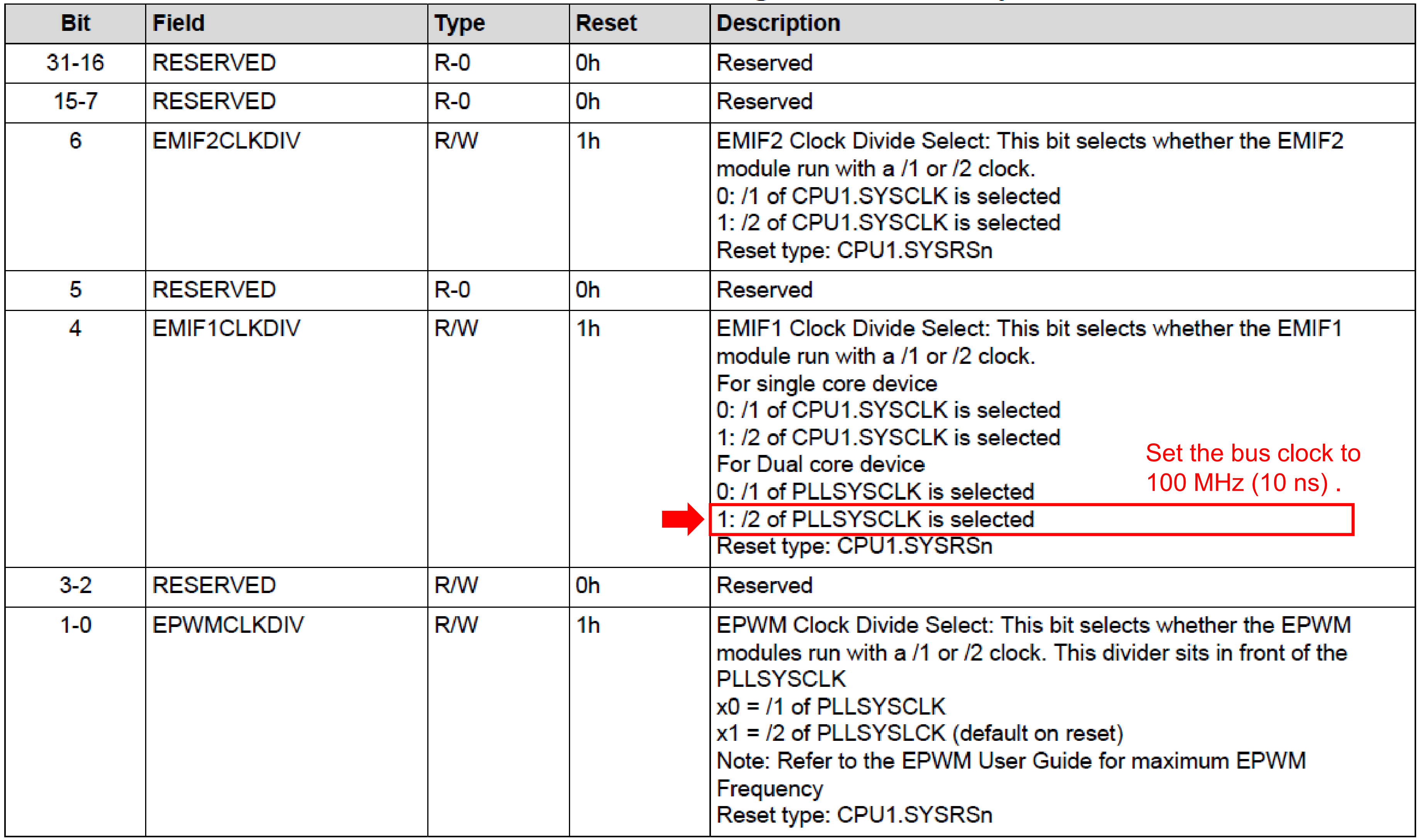


Table 3-74. PCLKCR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-2	RESERVED	R-0	0h	Reserved
1	EMIF2	R/W	0h	EMIF2 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Notes: [1] These bits are not used (R/W) in CPU2.PCLKCR1 register. EMIF1 & EMIF2 clock enabled are controlled only from CPU1.PCLKCR1 register. Reset type: SYSRSn
0	EMIF1	R/W	0h	EMIF1 Clock Enable bit: 0: Module clock is gated-off 1: Module clock is turned-on Notes: [1] These bits are not used (R/W) in CPU2.PCLKCR1 register. EMIF1 & EMIF2 clock enabled are controlled only from CPU1.PCLKCR1 register. Reset type: SYSRSn

PERCLKDIVSEL register

Table 3-52. PERCLKDIVSEL Register Field Descriptions



Bit	Field	Type	Reset	Description
31-16	RESERVED	R-0	0h	Reserved
15-7	RESERVED	R-0	0h	Reserved
6	EMIF2CLKDIV	R/W	1h	EMIF2 Clock Divide Select: This bit selects whether the EMIF2 module run with a /1 or /2 clock. 0: /1 of CPU1.SYSCLK is selected 1: /2 of CPU1.SYSCLK is selected Reset type: CPU1.SYSRSn
5	RESERVED	R-0	0h	Reserved
4	EMIF1CLKDIV	R/W	1h	EMIF1 Clock Divide Select: This bit selects whether the EMIF1 module run with a /1 or /2 clock. For single core device 0: /1 of CPU1.SYSCLK is selected 1: /2 of CPU1.SYSCLK is selected For Dual core device 0: /1 of PLLSYSCLK is selected 1: /2 of PLLSYSCLK is selected Reset type: CPU1.SYSRSn
3-2	RESERVED	R/W	0h	Reserved
1-0	EPWMCLKDIV	R/W	1h	EPWM Clock Divide Select: This bit selects whether the EPWM modules run with a /1 or /2 clock. This divider sits in front of the PLLSYSCLK x0 = /1 of PLLSYSCLK x1 = /2 of PLLSYSLCK (default on reset) Note: Refer to the EPWM User Guide for maximum EPWM Frequency Reset type: CPU1.SYSRSn

Set the bus clock to 100 MHz (10 ns) .

ASYNC_CS2_CR register

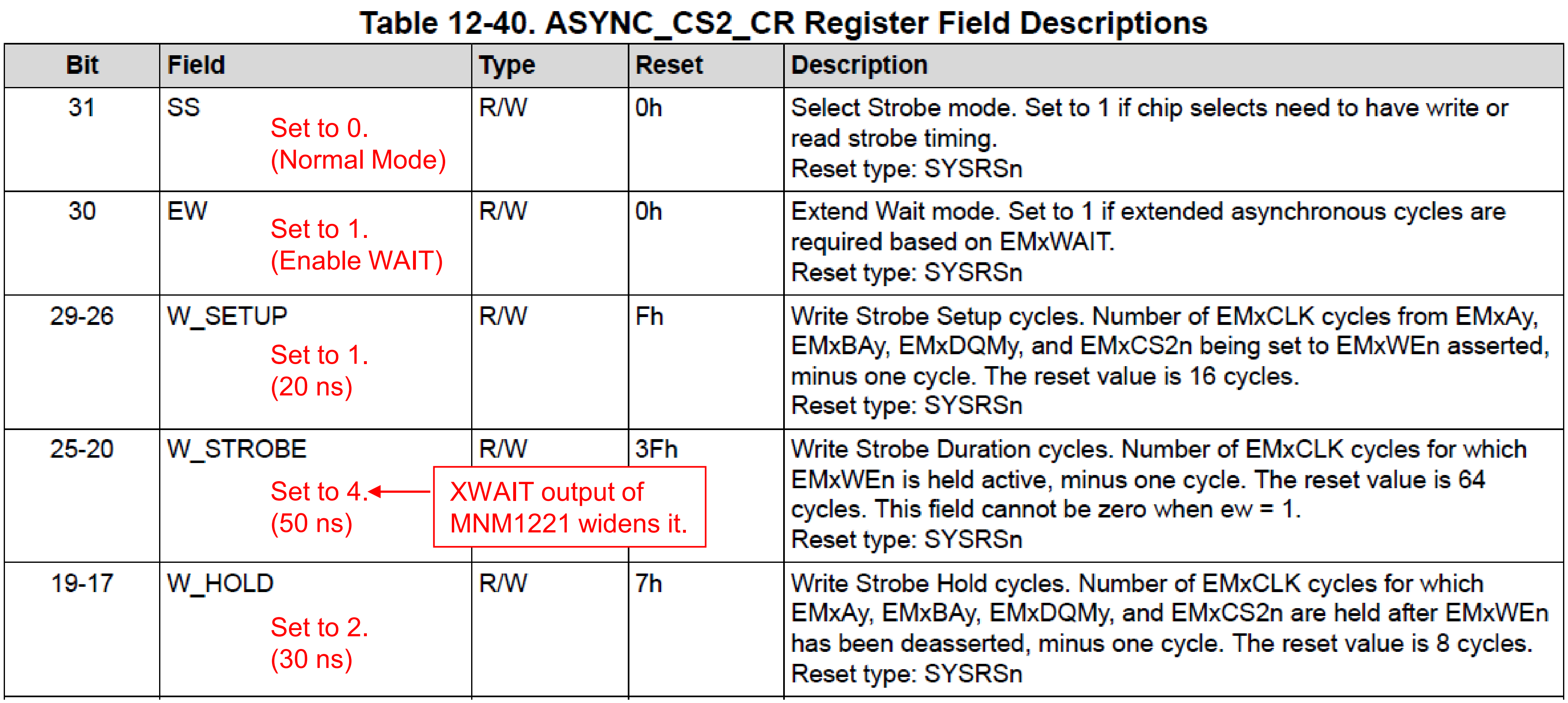


Table 12-40. ASYNC_CS2_CR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	SS Set to 0. (Normal Mode)	R/W	0h	Select Strobe mode. Set to 1 if chip selects need to have write or read strobe timing. Reset type: SYSRSn
30	EW Set to 1. (Enable WAIT)	R/W	0h	Extend Wait mode. Set to 1 if extended asynchronous cycles are required based on EMxWAIT. Reset type: SYSRSn
29-26	W_SETUP Set to 1. (20 ns)	R/W	Fh	Write Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBAy, EMxDQMy, and EMxCS2n being set to EMxWEn asserted, minus one cycle. The reset value is 16 cycles. Reset type: SYSRSn
25-20	W_STROBE Set to 4. (50 ns)	R/W	3Fh	Write Strobe Duration cycles. Number of EMxCLK cycles for which EMxWEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1. Reset type: SYSRSn
19-17	W_HOLD Set to 2. (30 ns)	R/W	7h	Write Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBAy, EMxDQMy, and EMxCS2n are held after EMxWEn has been deasserted, minus one cycle. The reset value is 8 cycles. Reset type: SYSRSn

XWAIT output of MNM1221 widens it.

ASYNC_CS2_CR register

Table 12-40. ASYNC_CS2_CR Register Field Descriptions

Bit	Field	Type	Reset	Description
16-13	R_SETUP Set to 1. (20 ns)	R/W	Fh	Read Strobe Setup cycles. Number of EMxCLK cycles from EMxAy, EMxBAy, EMxDQMy, and EMxCS2n being set to EMxOEn asserted, minus one cycle. The reset value is 16 cycles. Reset type: SYSRSn
12-7	R_STROBE Set to 4. (50 ns)	R/W XWAIT output of MNM1221 widens it.	3Fh	Read Strobe Duration cycles. Number of EMxCLK cycles for which EMxOEn is held active, minus one cycle. The reset value is 64 cycles. This field cannot be zero when ew = 1. Reset type: SYSRSn
6-4	R_HOLD Set to 2. (30 ns)	R/W	7h	Read Strobe Hold cycles. Number of EMxCLK cycles for which EMxAy, EMxBAy, EMxDQMy, and EMxCS2n are held after EMxOEn has been deasserted, minus one cycle. The reset value is 8 cycles. Reset type: SYSRSn
3-2	TA Set to 0. (10 ns)	R/W	3h	Turn Around cycles. Number of EMxCLK cycles between the end of one asynchronous memory access and the start of another asynchronous memory access, minus one cycle. This delay is not incurred between a read followed by a read, or a write followed by a write to the same chip select. The reset value is 4 cycles. Reset type: SYSRSn
1-0	ASIZE	R/W	1h	Asynchronous Memory Size. Defines the width of the asynchronous device's data bus : 00: 8 Bit data bus. 01: 16 Bit data bus. 10: 32 Bit data bus. 11: Reserved. Reset type: SYSRSn

ASYNC_WCCR register

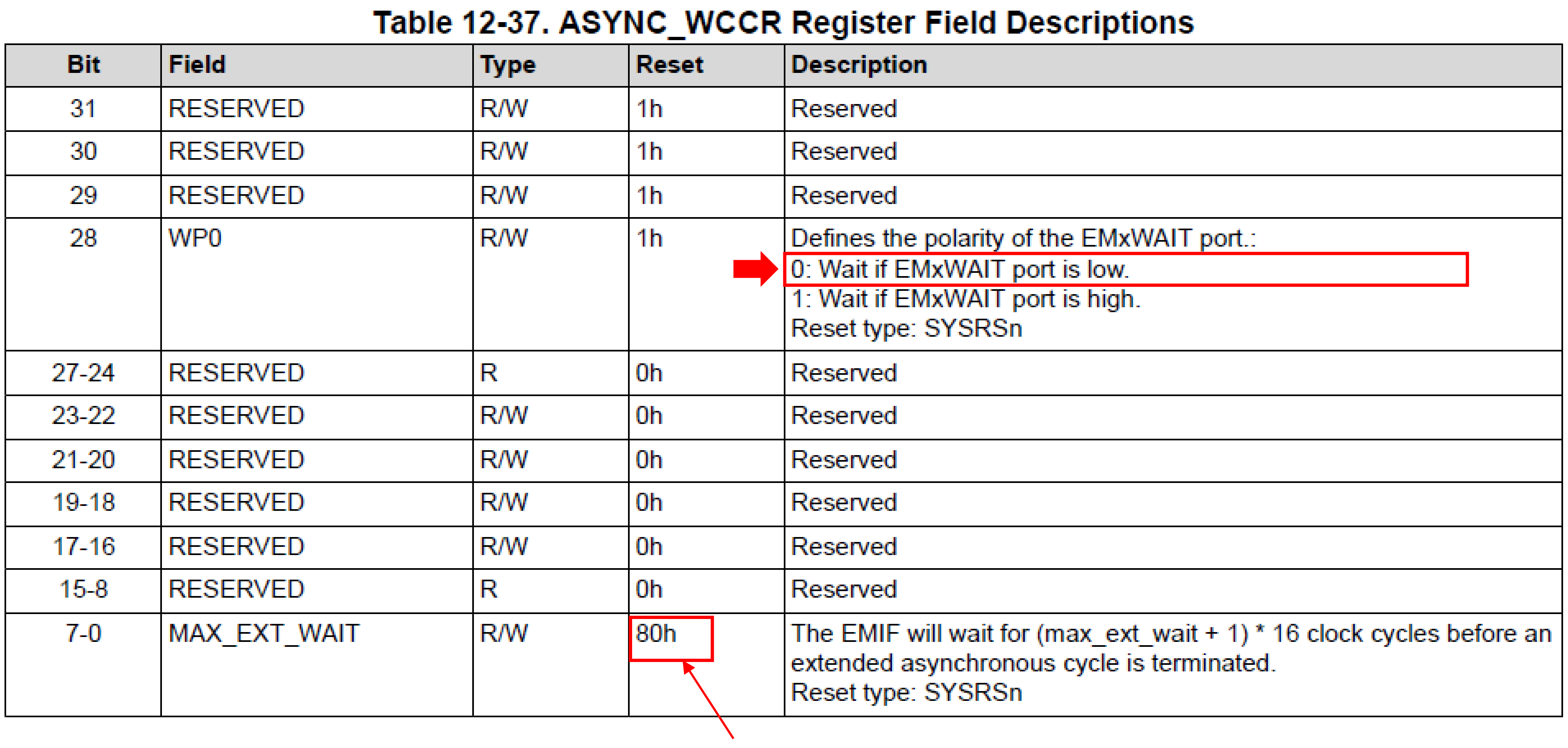


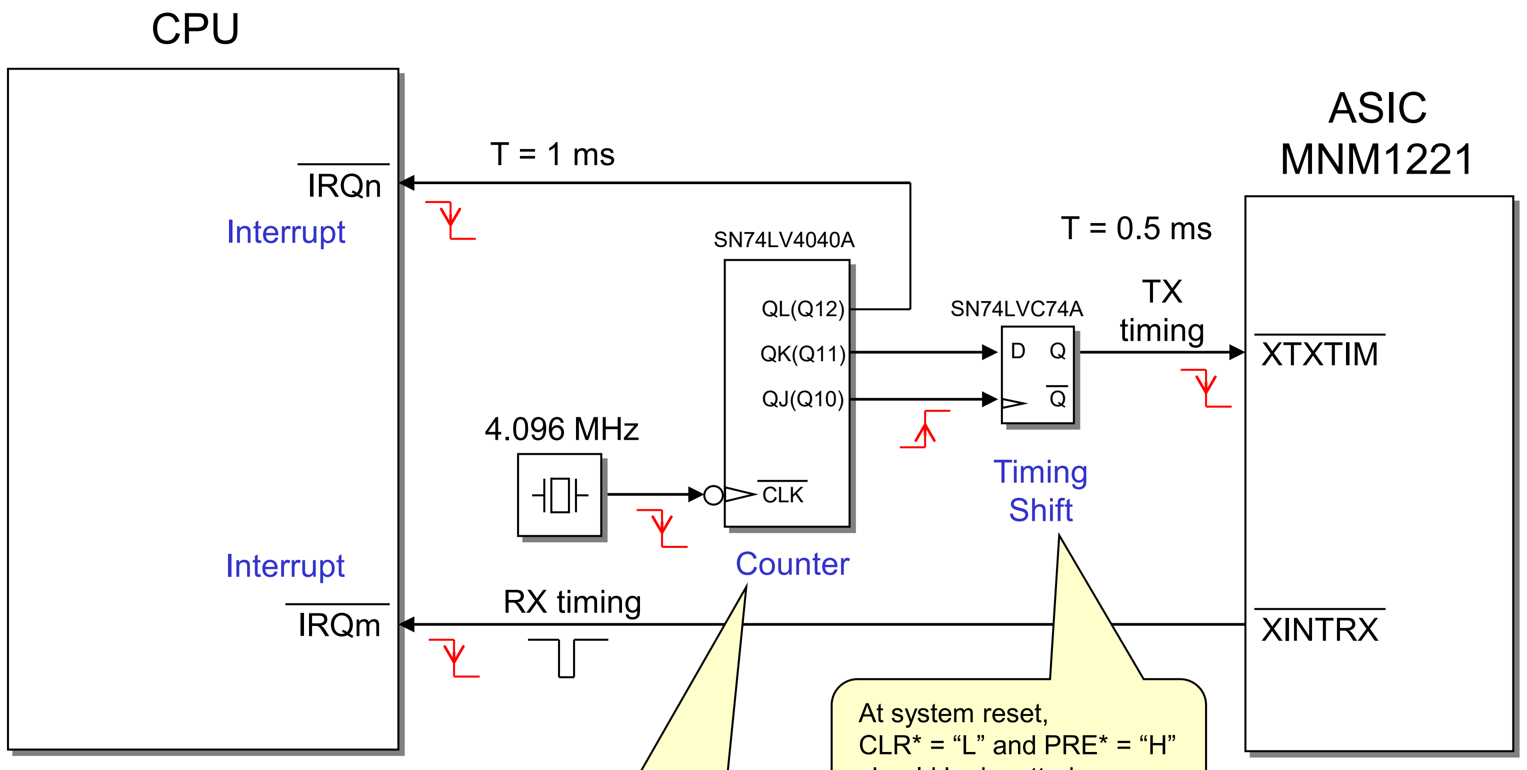
Table 12-37. ASYNC_WCCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	RESERVED	R/W	1h	Reserved
30	RESERVED	R/W	1h	Reserved
29	RESERVED	R/W	1h	Reserved
28	WP0	R/W	1h	Defines the polarity of the EMxWAIT port.: 0: Wait if EMxWAIT port is low. 1: Wait if EMxWAIT port is high. Reset type: SYSRSn
27-24	RESERVED	R	0h	Reserved
23-22	RESERVED	R/W	0h	Reserved
21-20	RESERVED	R/W	0h	Reserved
19-18	RESERVED	R/W	0h	Reserved
17-16	RESERVED	R/W	0h	Reserved
15-8	RESERVED	R	0h	Reserved
7-0	MAX_EXT_WAIT	R/W	80h	The EMIF will wait for (max_ext_wait + 1) * 16 clock cycles before an extended asynchronous cycle is terminated. Reset type: SYSRSn

Leave the default.

Example for
CPU without Internal Timer

Timing Circuit

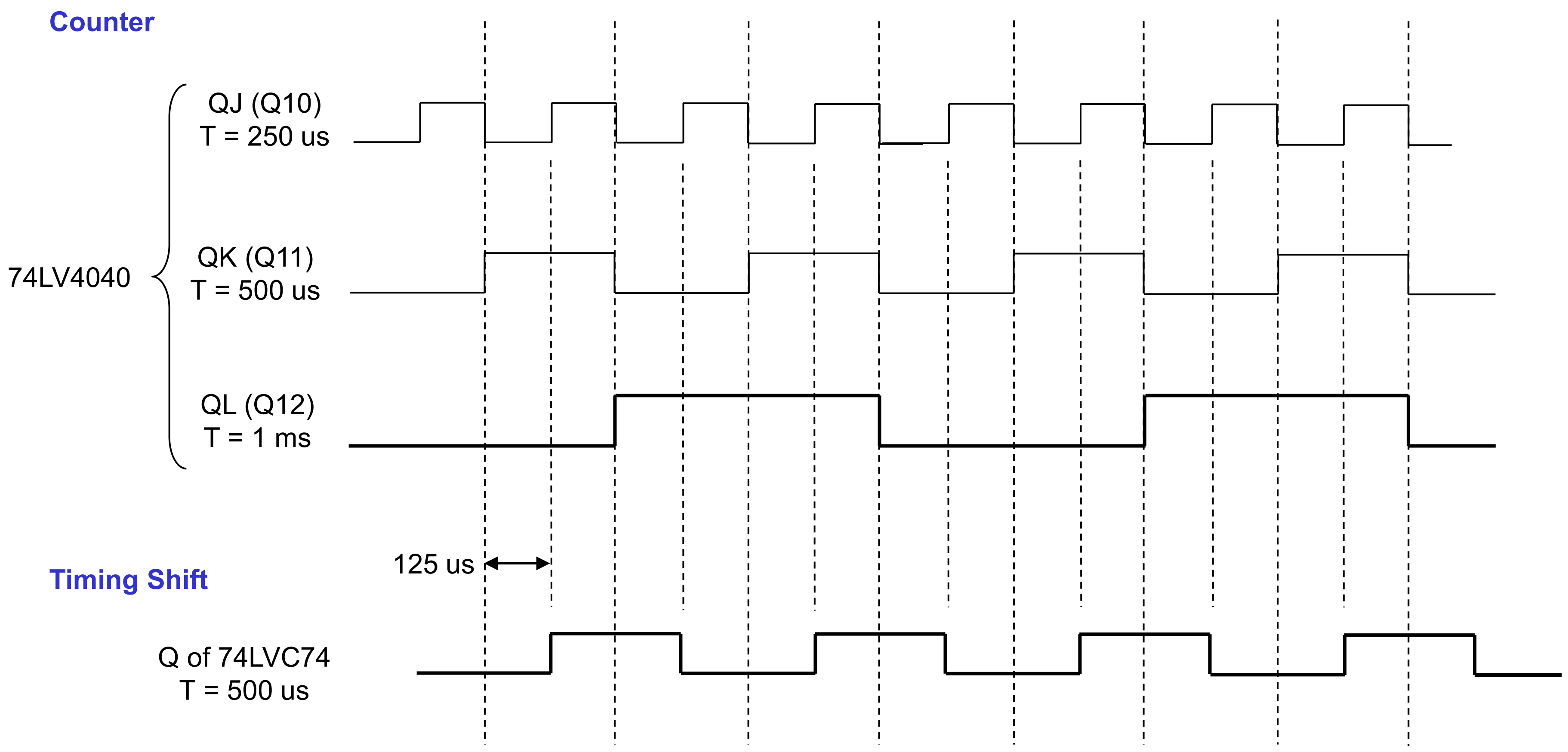


At system reset,
CLR = “H” should be inputted
to clear all outputs.

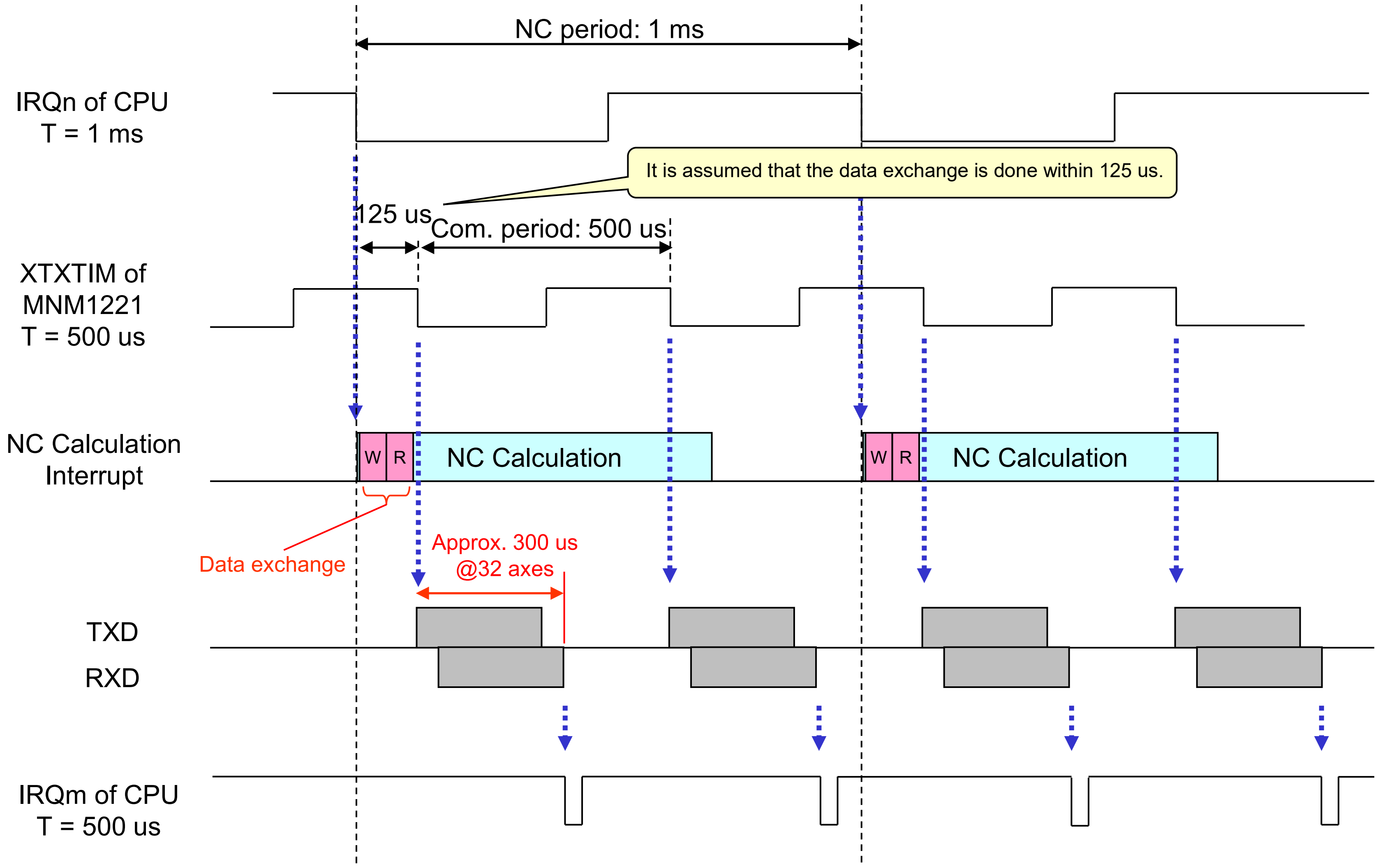
At system reset,
CLR* = “L” and PRE* = “H”
should be inputted
to clear Q output.

Priority: IRQm > IRQn

Timing Chart 1



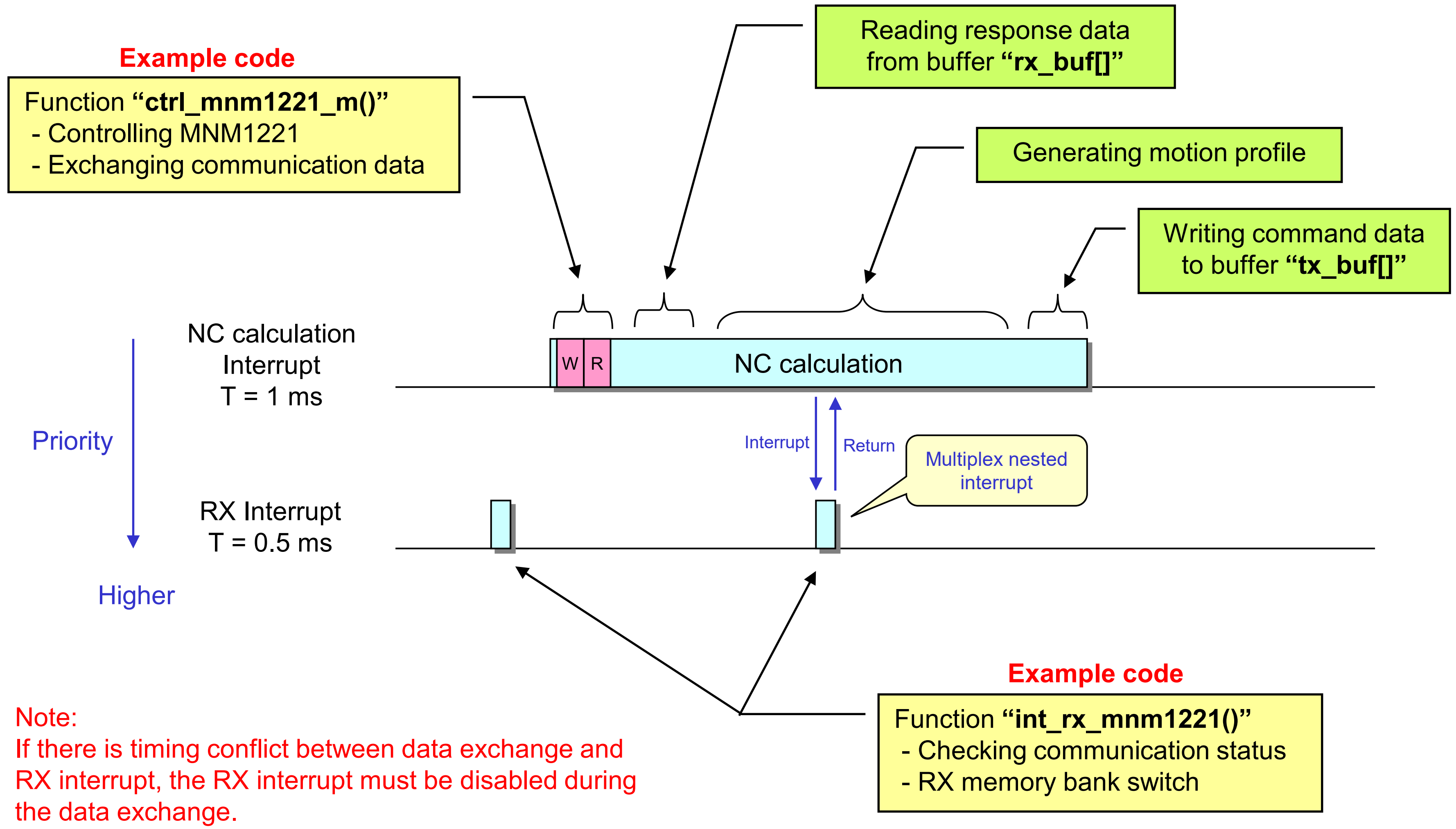
Timing Chart 2



Location of Example Codes

Location of Example Codes

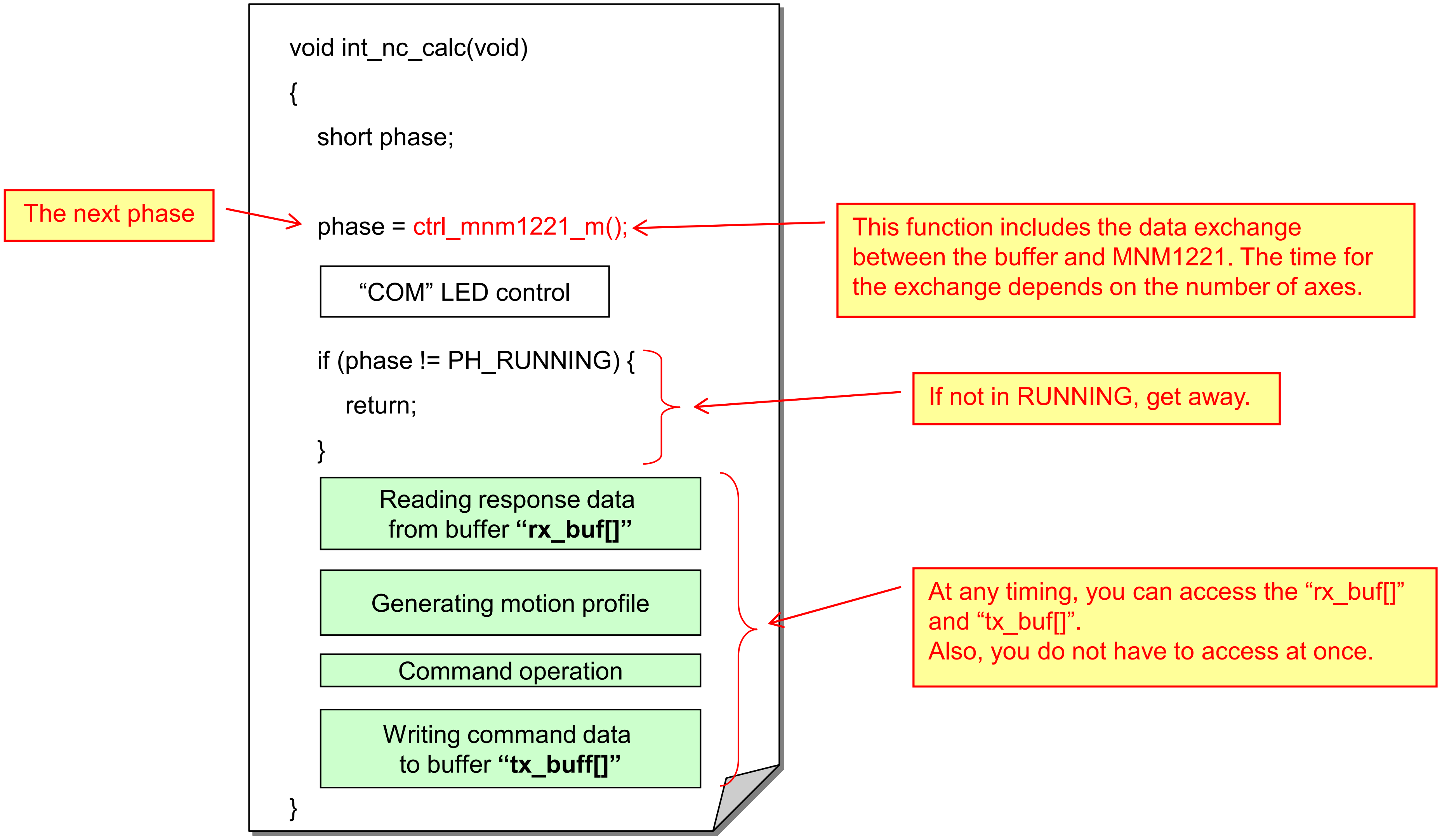
These functions must be made by yourself.



Example code
Function "ctrl_mnm1221_m()"
- Controlling MNM1221
- Exchanging communication data
Reading response data from buffer "rx_buf[]"
Generating motion profile
Writing command data to buffer "tx_buf[]"
NC calculation Interrupt T = 1 ms
W
R
NC calculation
Priority
Higher
Interrupt
Return
Multiplex nested interrupt
RX Interrupt T = 0.5 ms
Example code
Function "int_rx_mnm1221()"
- Checking communication status
- RX memory bank switch

Note:
If there is timing conflict between data exchange and RX interrupt, the RX interrupt must be disabled during the data exchange.

An Example of NC Calculation



```
void int_nc_calc(void)
{
    short phase;

    phase = ctrl_mnm1221_m();
    [ "COM" LED control ]

    if (phase != PH_RUNNING) {
        return;
    }
    [ Reading response data
      from buffer "rx_buf[]" ]
    [ Generating motion profile ]
    [ Command operation ]
    [ Writing command data
      to buffer "tx_buff[]" ]
}
```

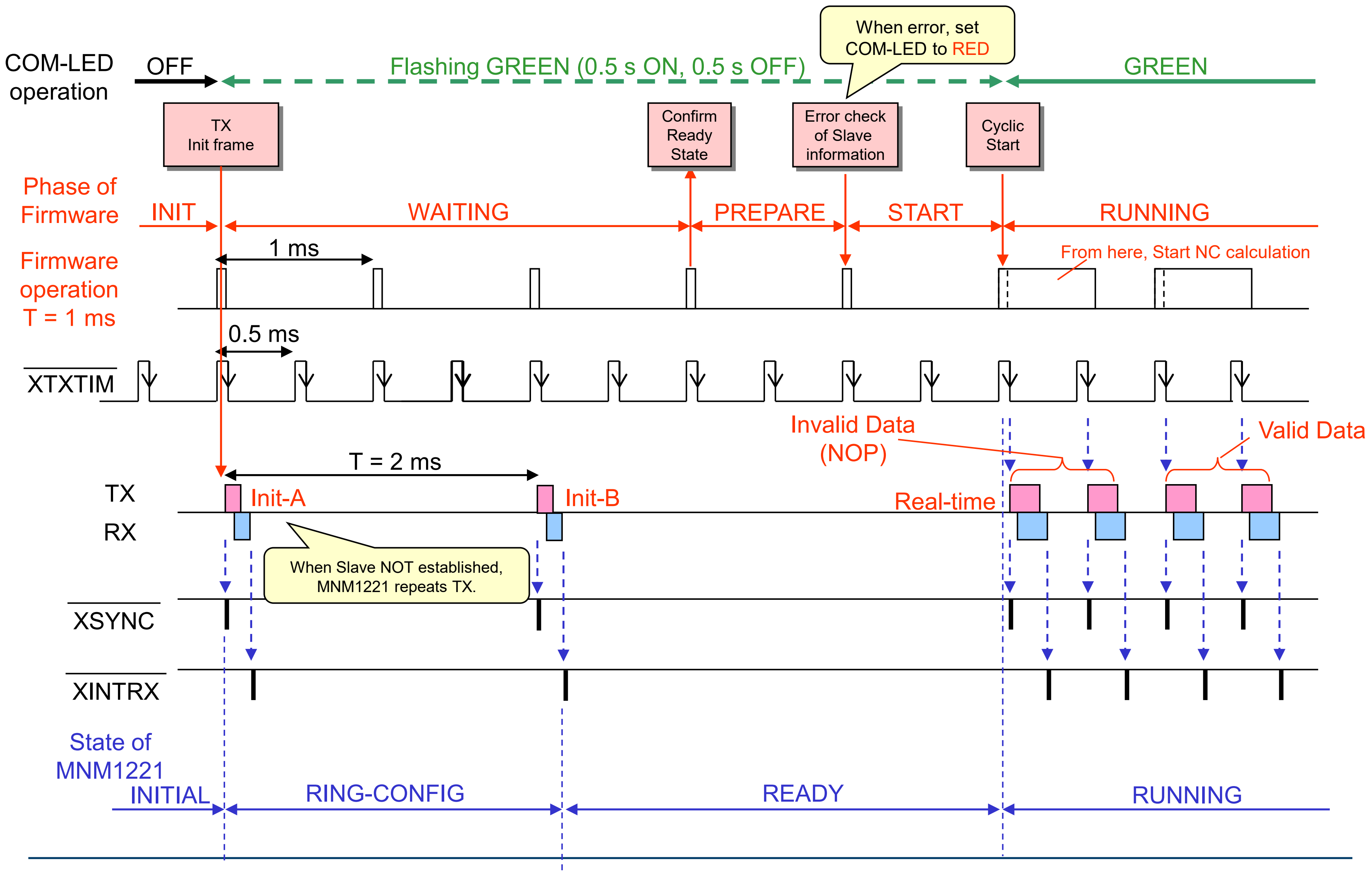
The next phase

This function includes the data exchange between the buffer and MNM1221. The time for the exchange depends on the number of axes.

If not in RUNNING, get away.

At any timing, you can access the "rx_buf[]" and "tx_buf[]".
Also, you do not have to access at once.

Timing Chart at Start-up



Notes of Using Example Codes

CPU with “Write Buffer” or “Cache”

- When using a CPU with “Write Buffer”, it causes unstable timing for writing data into MNM1221, and the example codes cannot perform the operation normally. Therefore, the measures must be taken according to the CPU manual.
- “Cache” must be disabled for the address area where MNM1221 is placed.

Data Structure (32-bit)

The following shows the structure of one element of tx_buf[] or rx_buf[] array **when 32-bit bus access**.

One element of tx_buf[] or rx_buf[]

	Bit 31 Bit 24	Bit 23 Bit 16	Bit 15 Bit 8	Bit 7 Bit 0
data[0]	byte3	byte2	byte1	byte0
data[1]	byte7	byte6	byte5	byte4
data[2]	byteB	byteA	byte9	byte8
data[3]	byteF	byteE	byteD	byteC

“byte0 to F” is corresponding to contents of a data block consisting of 16 bytes.

Data Structure (16-bit)

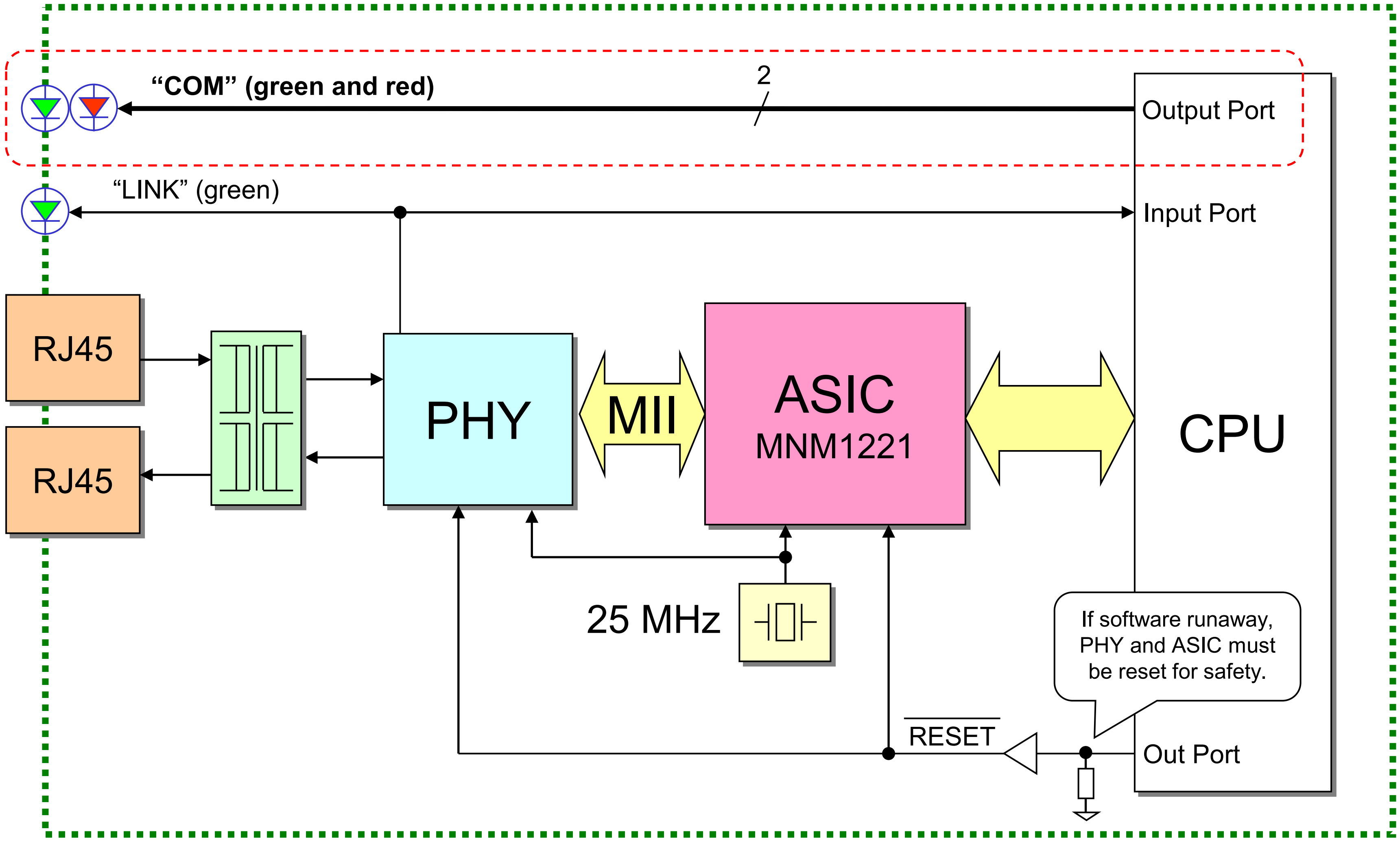
The following shows the structure of one element of tx_buf[] or rx_buf[] array **when 16-bit bus access**.

One element of tx_buf[] or rx_buf[]

	Bit 15 … Bit 8	Bit 7 … Bit 0
data[0]	byte1	byte0
data[1]	byte3	byte2
data[2]	byte5	byte4
data[3]	byte7	byte6
data[4]	byte9	byte8
data[5]	byteB	byteA
data[6]	byteD	byteC
data[7]	byteF	byteE

“byte0 to F” is corresponding to contents of a data block consisting of 16 bytes.

Status LEDs for Communication



“COM” LED Operation

“COM” LED which has red and green lights should be operated as follows:

Normally

Return value of ctrl_mnm1221_m()	“COM” LED operation
PH_INIT	Disappearance
PH_WAITING	Flashing Green (0.5 s ON, 0.5 s OFF)
PH_PREPARE	
PH_START	
PH_RUNNING	Solid Green

Error detected

Contents of error	“COM” LED operation
Timeout in RUNNING state	Flashing Red (0.5 s ON, 0.5 s OFF)
Mismatch of slave information (e.g. duplicate MAC-ID)	Solid Red

Notes:
- Solid Red means that a system reset is necessary to release the error.
- Either green or red must be lighted.

Overview of Cyclic Position I/F

Data Block

Command (TX)

Normally, set 0x20 (Position).

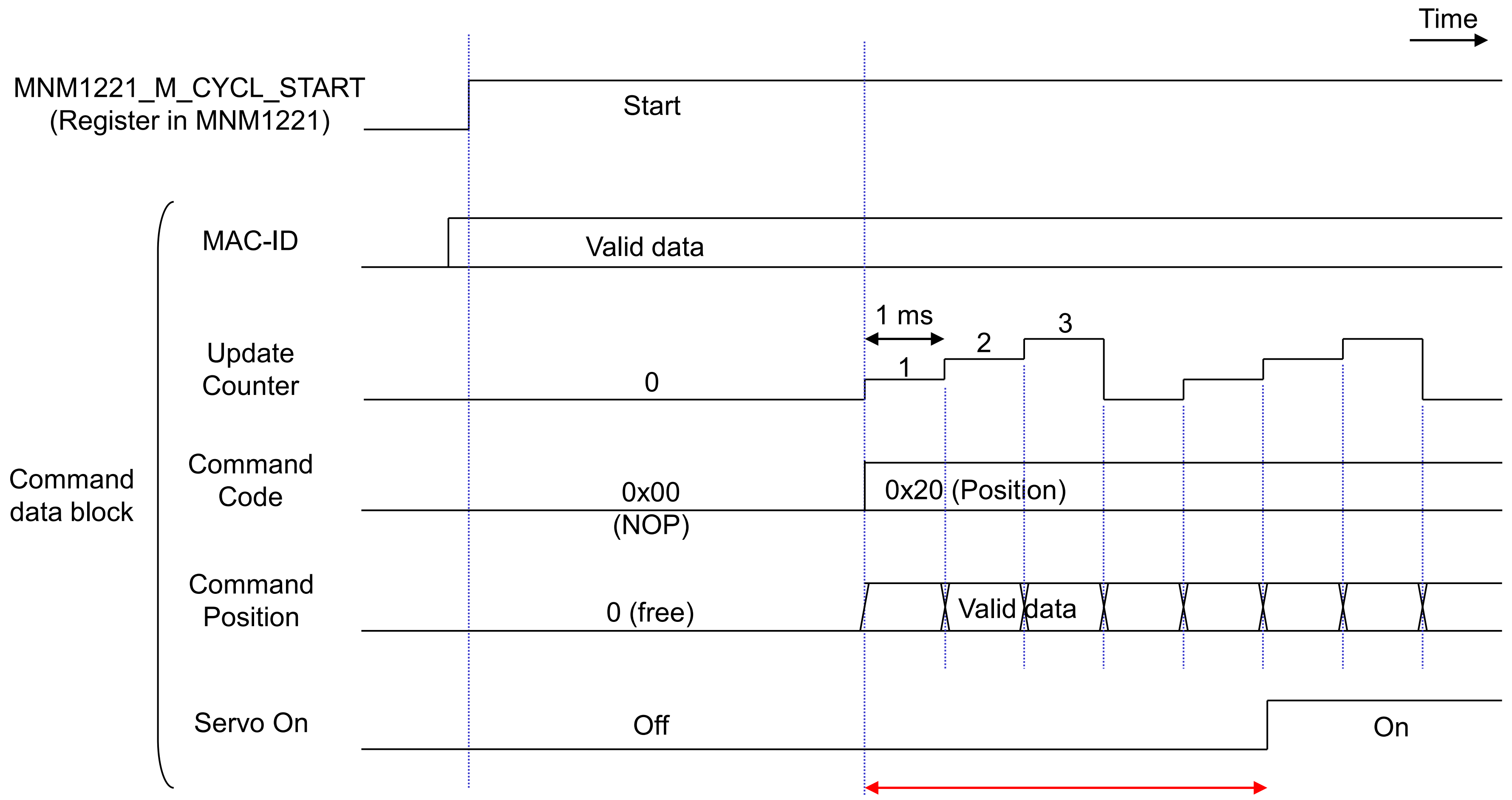
	bit7	bit6	bit5	bit4	bit3	bit2	bi	bit0
byte0	0 (CMD)	Update Counter		MAC-ID				
byte1	0	Command Code						
byte2	Servo On	0	0	Gain SW	TL SW	HM Ctrl	0	0
byte3	Hard Stop	SMT Stop	Pause	0	SL SW	0	EX-OUT2	EX-OUT1
byte4	Command Position							Low byte
byte5								Low Middle byte
byte6								High Middle byte
byte7								High byte
byte8	Command Data 2							Low byte
byte9								Low Middle byte
byteA								High Middle byte
byteB								High byte
byteC	Command Data 3							Low byte
byteD								Low Middle byte
byteE								High Middle byte
byteF								High byte

Response (RX)

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte0	1 (RSP)	Update Counter Echo		Actual MAC-ID				
byte1	CMD Error	Command Code Echo						
byte2	Servo Act.	Servo Ready	Alarm	Warn.	TL	HM Comp.	In Prog.	In Pos.
byte3	SI-MO5	SI-MO4	EXT 3	EXT 2	SI-MO1	Home	POT	NOT
byte4	Actual Position							Low byte
byte5								Low Middle byte
byte6								High Middle byte
byte7								High byte
byte8	Response Data 2							Low byte
byte9								Low Middle byte
byteA								High Middle byte
byteB								High byte
byteC	Response Data 3							Low byte
byteD								Low Middle byte
byteE								High Middle byte
byteF								High byte

Note: In cyclic position I/F, at least red portions must be supported.

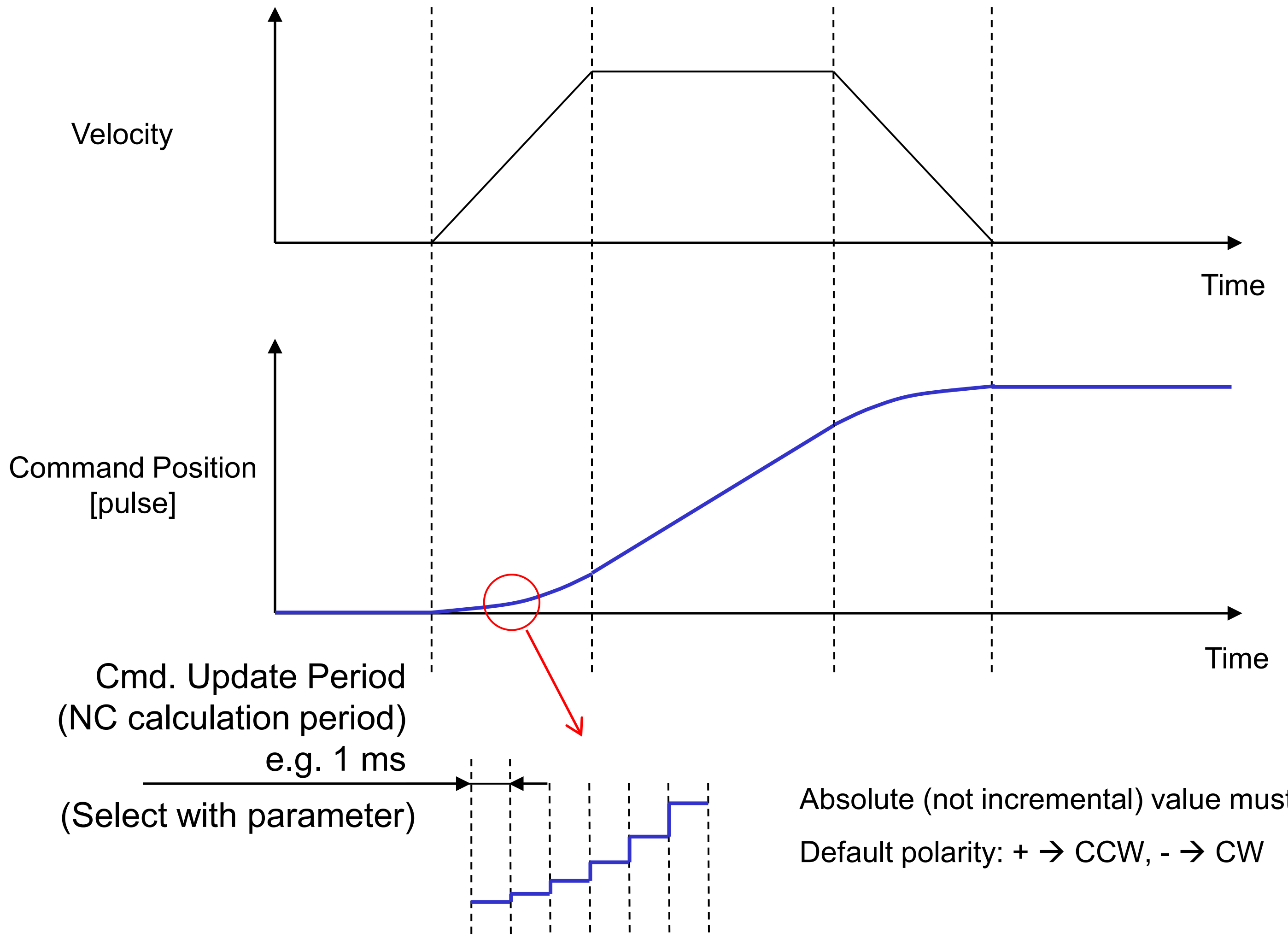
Command at Start-up



During Servo Off, Command Position should be set with Actual Position value of Response.

Note: This time chart shows an example of cyclic position I/F.

Cyclic Position I/F



Absolute (not incremental) value must be used.

Default polarity: + → CCW, - → CW

Overview of Profile Position I/F

Command

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte0	C/R (0)	Update Counter		MAC-ID (0 to 31)				
Byte1	TMG CNT	**17h (Command Code)**						
Byte2	Servo On	0	0	Gain SW	TL SW	Homing Ctrl	0	0
Byte3	**Hard Stop**	**Smooth Stop**	**Pause**	0	SL SW	0	EX-OUT2	EX-OUT1
Byte4	**Target Position**							
Byte5								
Byte6								
Byte7								
Byte8	**Type Code**							
Byte9	0							
Byte10	0							
Byte11	**Monitor Sel**							
Byte12	**Target Speed**							
Byte13								
Byte14								
Byte15								

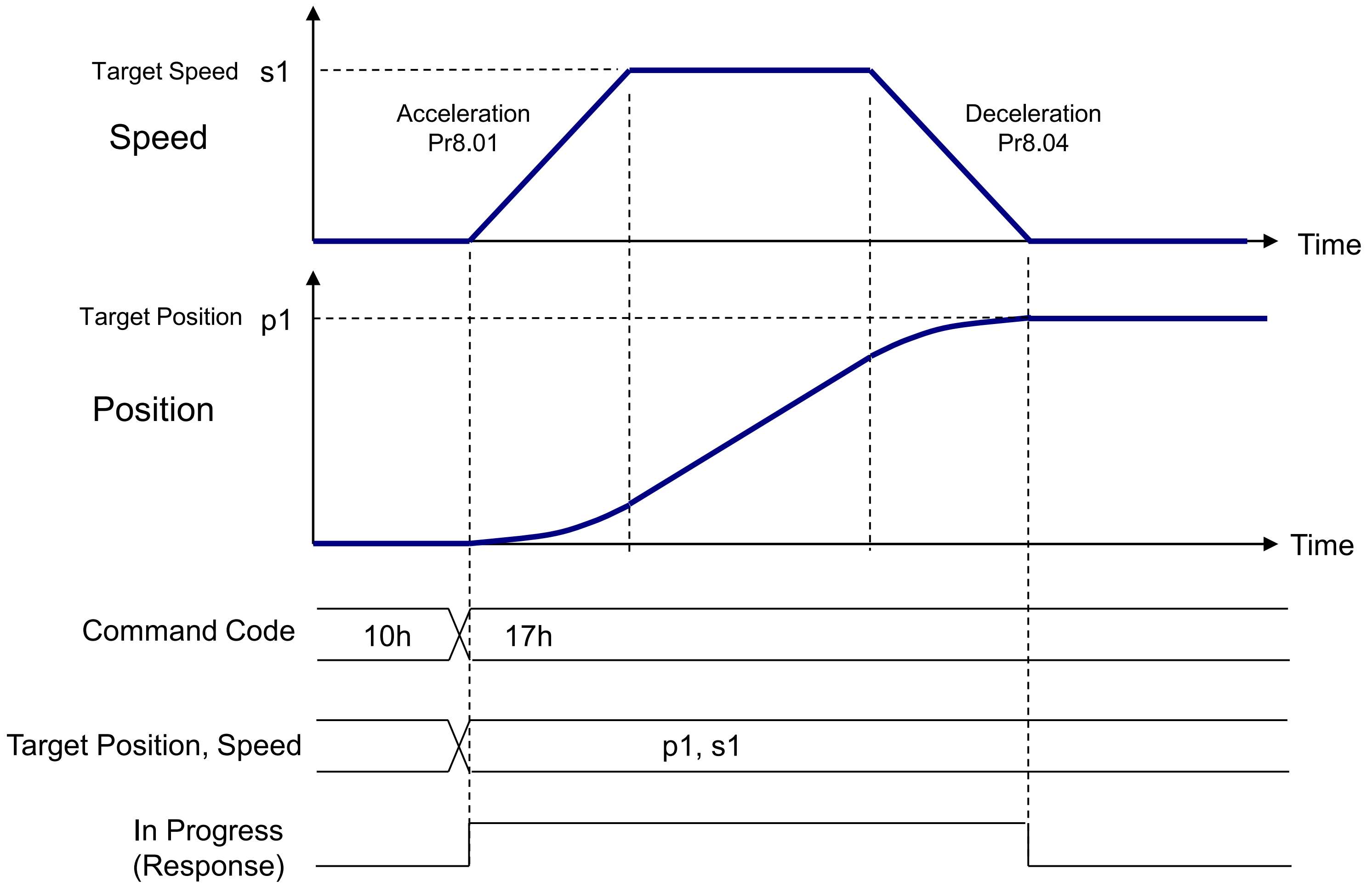
Mode, Inc/Abs

Response

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte0	C/R (1)	Update Counter Echo		Actual MAC-ID (0 to 31)				
Byte1	CMD Error	**17h (Command Code Echo)**						
Byte2	Servo Active	Servo Ready	Alarm	Warning	Torque Limited	Homing Complete	**In Progress**	In Position
Byte3	SI-MON5 /E-STOP	SI-MON4 /EX-SON	SI-MON3 /EXT3	SI-MON2 /EXT2	SI-MON1 /EXT1	Home	POT /NOT	NOT /POT
Byte4	**Actual Position**							
Byte5								
Byte6								
Byte7								
Byte8	**Type Code Echo**							
Byte9	ERR	WNG	0	BUSY	**PSL /NSL**	**NSL /PSL**	**NEAR**	**Latch Compl**
Byte10	0							
Byte11	**Monitor Sel Echo**							
Byte12	**Monitor Data**							
Byte13								
Byte14								
Byte15								

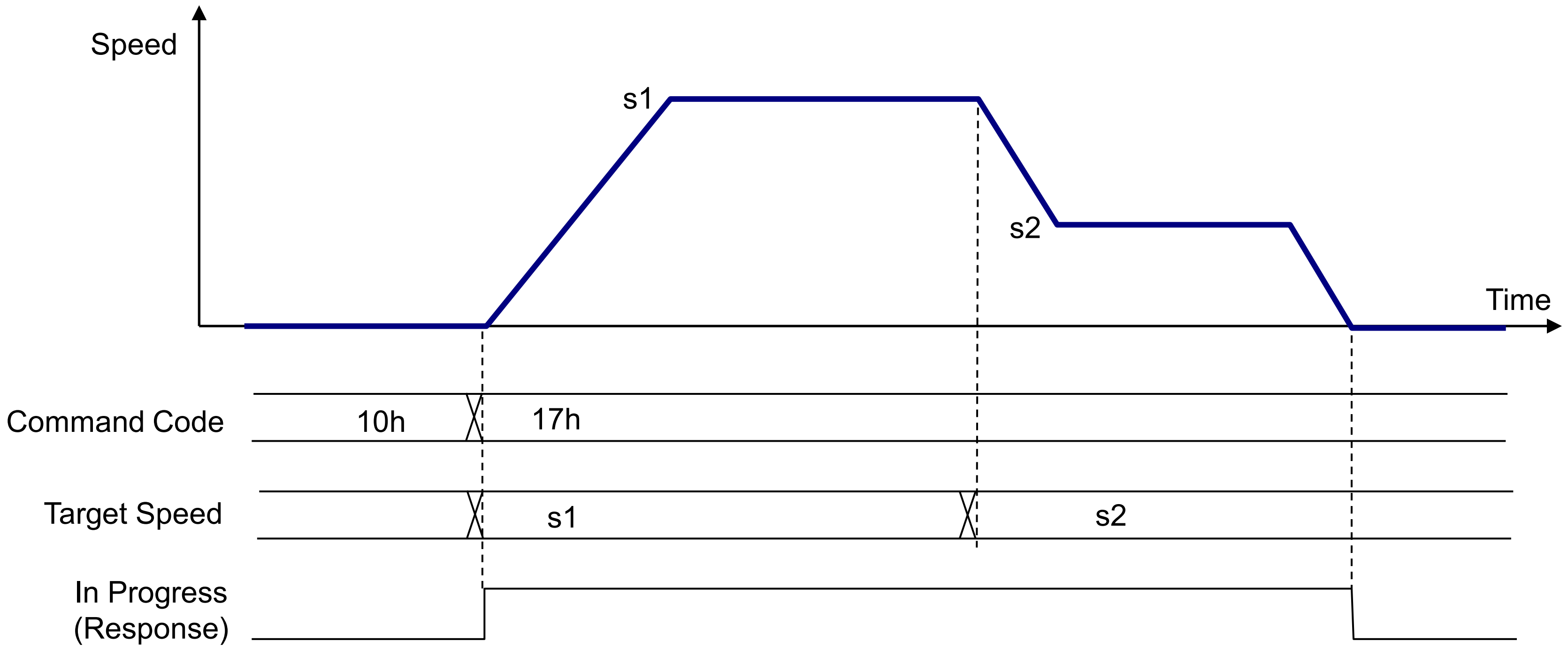
Start

When “In Progress” = 0, a change of Command Code 10h to 17h makes servo start motion.
Acceleration and deceleration are preset with parameter. Abs/Inc is set with Type Code at start.



Changing T Speed in Motion

When "In Progress" = 1, target speed can be changed.
Even if changing target speed to 0 or Pause to 1, "In Progress" keeps 1 during stop.



Modifying the Example Code

Definition for Bus Access

mnm1221_m.h

Modify this address value in order to suit to the located address of MNM1221.

```
/*** IMPORTANT!!! ***/
/* You must modify the following definition according to your system. */
/*-------------------------------------------------------------------------*/
/* Definition depend on your system                                        */
/*-------------------------------------------------------------------------*/
/* Located Address of MNM1221 */
#define ADDR_MNM1221        0x08000000      /* unit: byte address */

/* Data Bus Width to access to MNM1221 */
#define MASTER_16BIT_ACCESS
/* If NOT 16bits BUT 32bits, change this definition to comments or delete it. */
/*-------------------------------------------------------------------------*/
```

32-bit bus: Delete this.
16-bit bus: Leave this.

Definition of Variables

mnm1221_m.c

```
/*-------------------------------------------------------------------------*/
/* Declaration and definition of variables used in also other files        */
/*-------------------------------------------------------------------------*/
/*
* If your compiler does not allow that a file includes both declaration and
* definition of variables, the declaration should be moved to the other file.
*/

/* declaration */
extern Com_buf tx_buf[];        /* TX data buffer */
extern Com_buf rx_buf[];        /* RX data buffer */
extern Com_err com_err;         /* error status */
extern Mnm1221 mnm1221;         /* MNM1221 status */

/* definition */

/* Via the following buffer, the exchange of communication data should be done. */
Com_buf tx_buf[MAX_NS];         /* TX data buffer */
Com_buf rx_buf[MAX_NS];         /* RX data buffer */

Com_err com_err;                /* error status */
Mnm1221 mnm1221;                /* MNM1221 status */


/*-------------------------------------------------------------------------*/
```

For fast access,
locate this communication buffers
on the internal RAM of CPU
if possible.

Slave Information Table

mnm1221_m.c

```
static void set_slave_inf_base(void)
{
    short i;
/*
* The following is just for test,
* so should be replace with setting based on the user interface for configuration.
*/
    /* e.g.  4 Servo slaves */
    /*----------------------------------------------------------------*/
    /*                    active      mode        MACID     block N  */
    /*----------------------------------------------------------------*/
    slave_inf_base[0] = (1 << 15) | (1 << 13) | (1 << 8) | 1;
    slave_inf_base[1] = (1 << 15) | (1 << 13) | (2 << 8) | 1;
    slave_inf_base[2] = (1 << 15) | (1 << 13) | (3 << 8) | 1;
    slave_inf_base[3] = (1 << 15) | (1 << 13) | (4 << 8) | 1;
    /*----------------------------------------------------------------*/

    /* not presence (i.e. inactive, invalid) */
    /* MSB (active bit) must be set to zero. */
    for (i = 4; i < MAX_NS; i++) {
        slave_inf_base[i] = (0 << 15) | (1 << 13) | (31 << 8) | 1;
    }
}
```

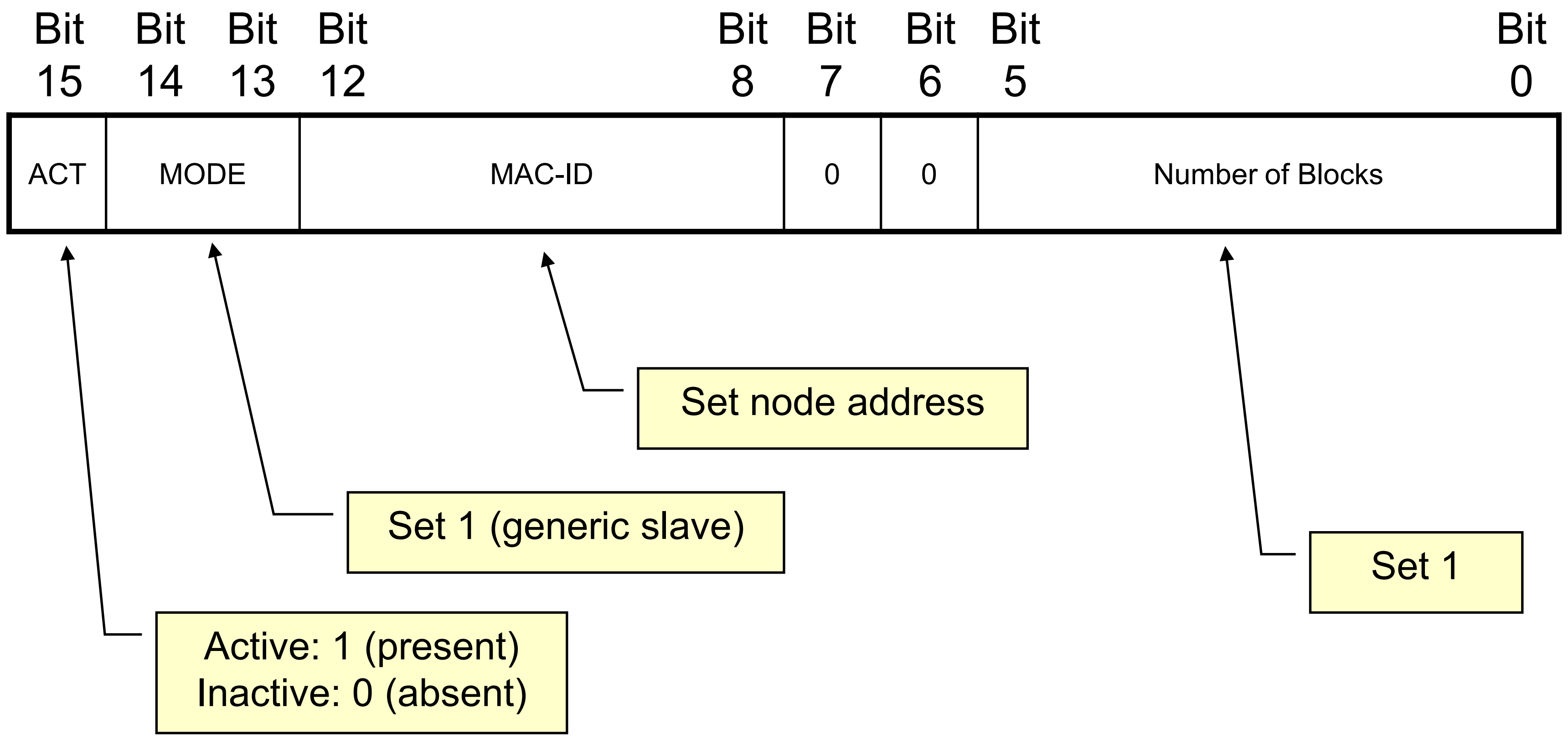
When test, modify these values according to actual your system.

In one-axis case, make these lines comment.

And change this number to 1.

Slave Information Table (Cont.)

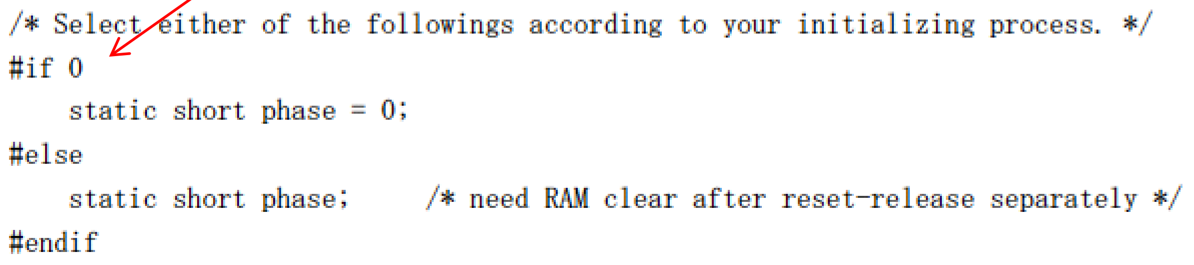
Structure of Slave Information data:



Style of Initializing Variables

ctrl_mnm1221_m() in mnm1221_m.c

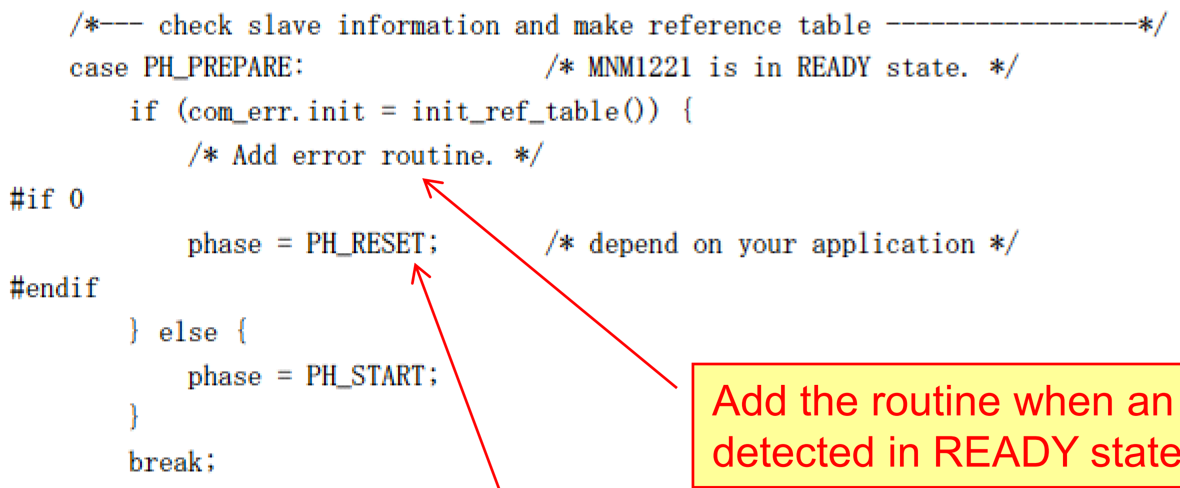
To set variable “phase” to 0 after reset, select this value (0 or 1) according to your initializing process.



```
/* Select either of the followings according to your initializing process. */
#if 0
    static short phase = 0;
#else
    static short phase;     /* need RAM clear after reset-release separately */
#endif
```

Checking Slave Information

ctrl_mnm1221_m() in mnm1221_m.c



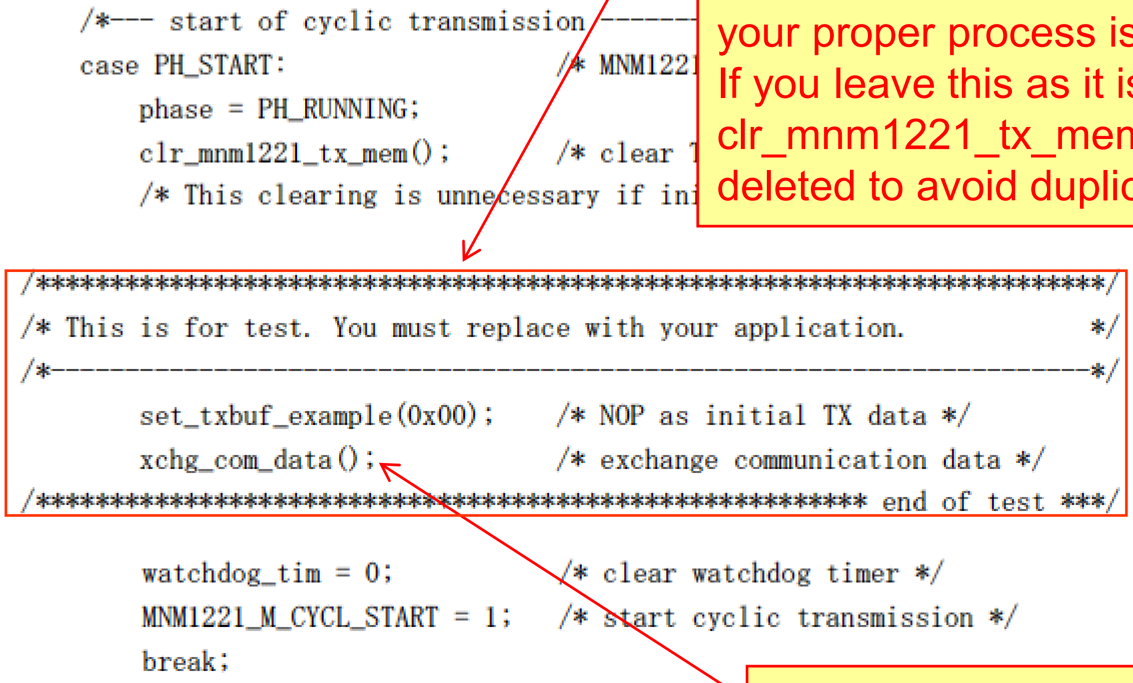
```
    /*--- check slave information and make reference table -----------------*/
    case PH_PREPARE:                /* MNM1221 is in READY state. */
        if (com_err.init = init_ref_table()) {
            /* Add error routine. */
#if 0
            phase = PH_RESET;       /* depend on your application */
#endif
        } else {
            phase = PH_START;
        }
        break;
```

Add the routine when an error is detected in READY state.

In actual application,
the phase should not be changed to PH_RESET until releasing errors.

Starting Cyclic Transmission

ctrl_mnm1221_m() in mnm1221_m.c



```
/*--- start of cyclic transmission -------
case PH_START:                    /* MNM1221
    phase = PH_RUNNING;
    clr_mnm1221_tx_mem();         /* clear T
    /* This clearing is unnecessary if ini

/***************************************************************************/
/* This is for test. You must replace with your application.             */
/*-----------------------------------------------------------------------*/
        set_txbuf_example(0x00);    /* NOP as initial TX data */
        xchg_com_data();            /* exchange communication data */
/******************************************************** end of test ***/

    watchdog_tim = 0;               /* clear watchdog timer */
    MNM1221_M_CYCL_START = 1;       /* start cyclic transmission */
    break;
```

This process is to set a initial transmit data including MACID.
Since this is for initial test,
your proper process is needed.
If you leave this as it is,
clr_mnm1221_tx_mem() should be deleted to avoid duplicate initializing.

The data exchange function is used provisionally. But the reading of the exchange is unnecessary.

In Running State

ctrl_mnm1221_m() in mnm1221_m.c

After test, you have to remove.

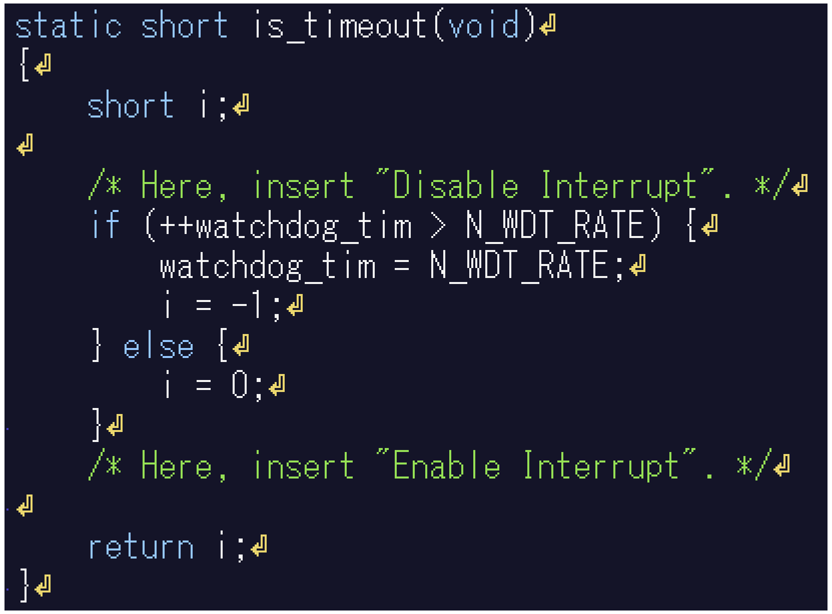
```
     /*--- running state (cyclic transmission) ----------
     case PH_RUNNING:               /* MNM1221 is in RUNNING state. */

/************************************************************************/
/* This is for test. You must replace with your application.           */
/*---------------------------------------------------------------------*/
/*
* In actual application, the routin setting to TX buffer will be placed
* after NC calculation. i.e. The routin is not in ctrl_mnm1221_m(), but
* at the end of the timer interrupt for NC calculation.
*/
        set_txbuf_example(0x20);    /* Position command */
/********************************************************* end of test ***/

        xchg_com_data();            /* exchange commun
        if (is_timeout()) {
            com_err.run |= B_TIMEOUT;
            /* Add error routine. */
#if 0
            phase = PH_RESET;       /* depend on your application */
#endif
        }
        break;
```

Add the routine when time-out error is detected.
At that time, Servo-OFF must be commanded to all servos for safety.

In actual application,
the phase should not be changed to PH_RESET until releasing errors.

Checking Timeout

is_timeout() in mnm1221_m.c



```
static short is_timeout(void)
{
    short i;

    /* Here, insert "Disable Interrupt". */
    if (++watchdog_tim > N_WDT_RATE) {
        watchdog_tim = N_WDT_RATE;
        i = -1;
    } else {
        i = 0;
    }
    /* Here, insert "Enable Interrupt". */

    return i;
}
```

Insert "Disable Interrupt".

Insert "Enable Interrupt".

Chapter 2

Internal-Timer System

The Point of System

- Set both “Command Update Period” and “Communication Period” to the same time.
- Start NC calculation interrupt with MNM1221 XSYNC.
- If not using RX interrupt, detect timeout error by software using “Update Counter Echo”.

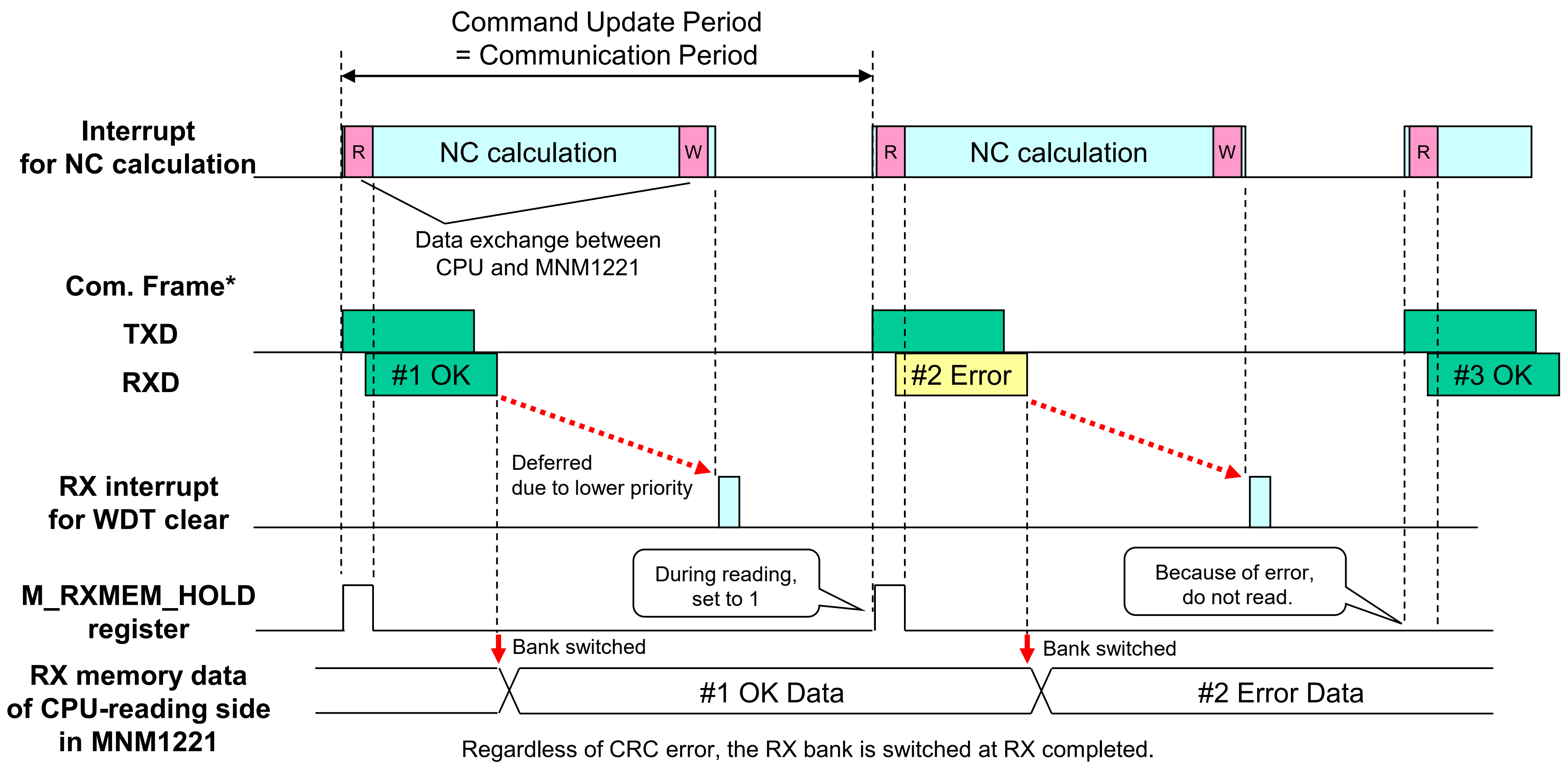
Period Setting of A6N Drive

Set both command update period and communication period to the same.
Default setting must be changed.

Update Period [ms]	Com. Period [ms]	Parameter Setting Value			Remark
		Pr7.20	Pr7.21	Pr7.91	
4.000	2.000	-1	2	2000000	
2.000	2.000	-1	1	2000000	
2.000	1.000	-1	2	1000000	
1.000	1.000	-1	1	1000000	Pr7.20 = 6, Pr7.21 = 1 also allowed.
1.000	0.500	-1	2	500000	Pr7.20 = 3, Pr7.21 = 2 also allowed.
0.500	0.500	-1	1	500000	Pr7.20 = 3, Pr7.21 = 1 also allowed.
0.500	0.250	-1	2	250000	
0.250	0.250	-1	1	250000	
0.250	0.125	-1	2	125000	
0.125	0.125	-1	1	125000	
0.125	0.0625	-1	2	62500	

Goal of Timing Control

The goal is to make the following timing:



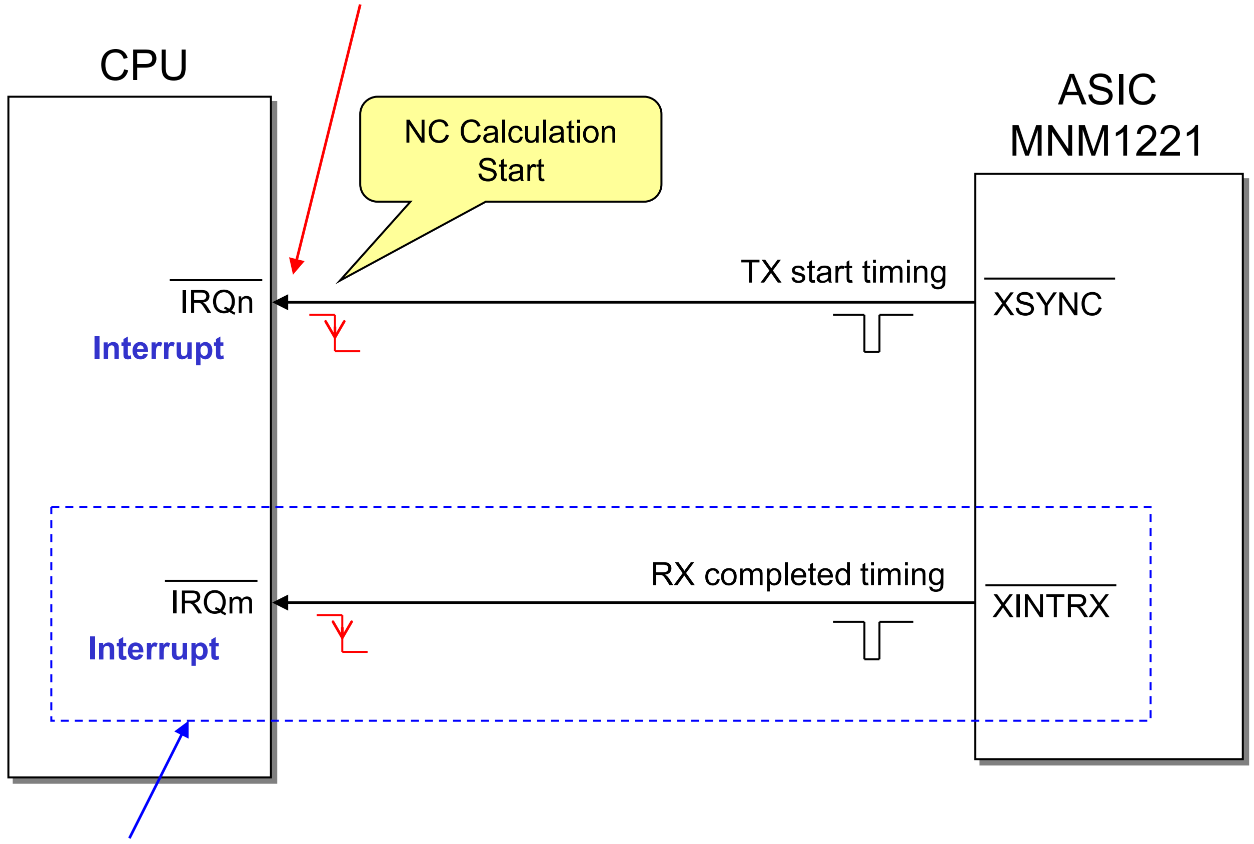
Regardless of CRC error, the RX bank is switched at RX completed.
In NC calculation routine, when RX data is OK, the M_RXMEM_HOLD is set to 1 and RX data is read.

* One frame contains data of all slave nodes, and its length depends on the number of connected nodes.

Example for an Embedded CPU

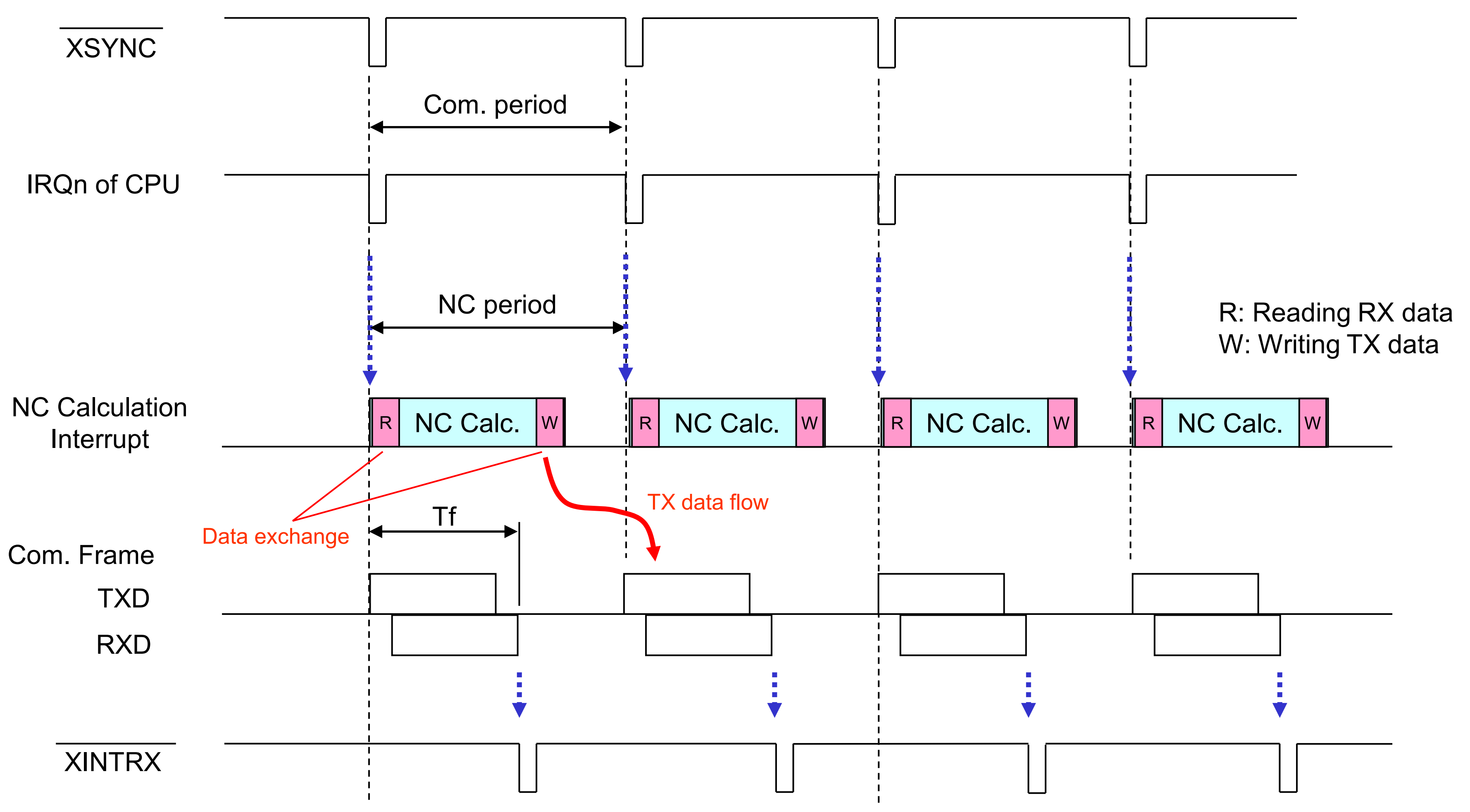
Timing Circuit

When not in RUNNING state, CPU interrupt should be disabled.



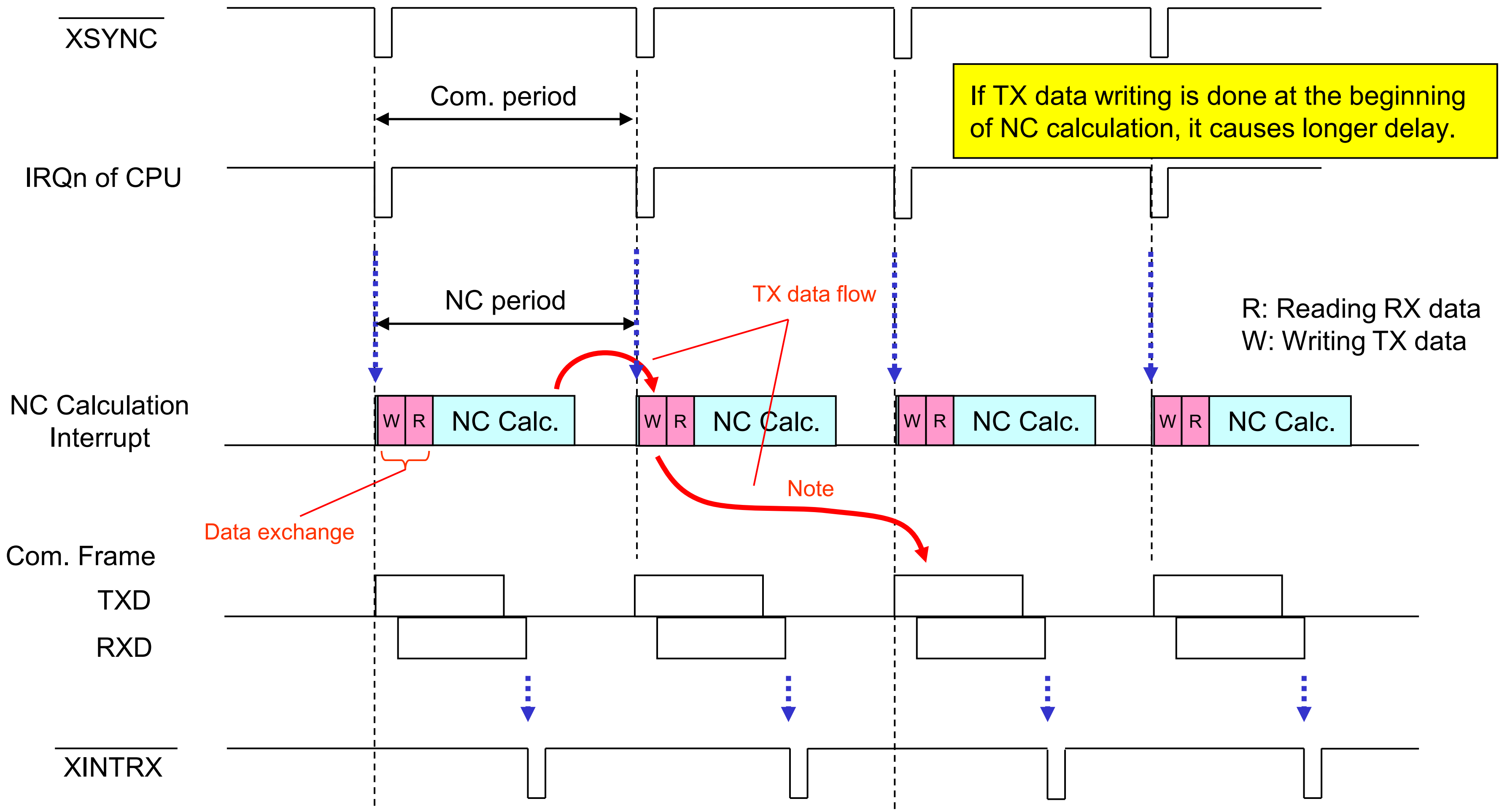
If not using RX interrupt, a timeout should be detected with the echo back of Update Counter in the data block.

Recommended Timing



Tf: approx. 11 us@1-axis to 300 us@32-axis

NOT Recommended Timing



Note:
At XSYNC falling edge, a frame has already been in transmitting, and the memory-bank switching after writing TX-data is deferred.
Therefore, the transmit timing is delayed for one communication period.

Timeout Detection Example for Not Using RX Interrupt

Update Counter

Command (TX)

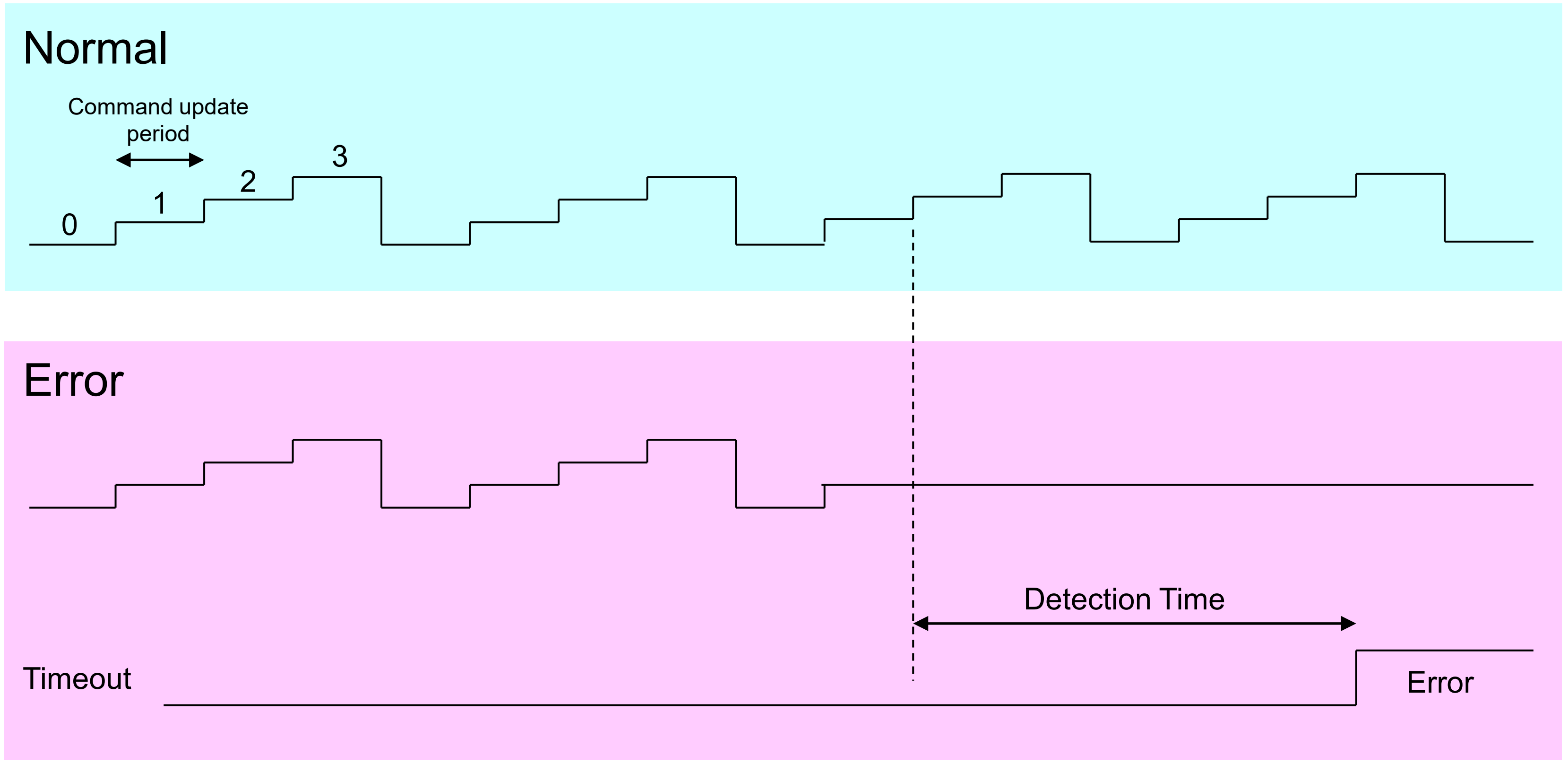
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte0	0 (CMD)	Update Counter		MAC-ID				
byte1	0	Command Code						
byte2	Servo On	0	0	Gain SW	TL SW	HM Ctrl	0	0
byte3	Hard Stop	SMT Stop	Pause	0	SL SW	0	EX-OUT2	EX-OUT1
byte4	Command Position						Low byte	
byte5							Low Middle byte	
byte6							High Middle byte	
byte7							High byte	
byte8	Command Data 2						Low byte	
byte9							Low Middle byte	
byteA							High Middle byte	
byteB							High byte	
byteC	Command Data 3						Low byte	
byteD							Low Middle byte	
byteE							High Middle byte	
byteF							High byte	

Response (RX)

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte0	1 (RSP)	Update Counter Echo		Actual MAC-ID				
byte1	CMD Error	Command Code Echo						
byte2	Servo Act.	Servo Ready	Alarm	Warn.	TL	HM Comp.	In Prog.	In Pos.
byte3	SI-MO5	SI-MO4	EXT 3	EXT 2	SI-MO1	Home	POT	NOT
byte4	Actual Position						Low byte	
byte5							Low Middle byte	
byte6							High Middle byte	
byte7							High byte	
byte8	Response Data 2						Low byte	
byte9							Low Middle byte	
byteA							High Middle byte	
byteB							High byte	
byteC	Response Data 3						Low byte	
byteD							Low Middle byte	
byteE							High Middle byte	
byteF							High byte	

Timeout Detection

If “Update Counter Echo” is not changed continuously for a certain time, timeout should be considered.

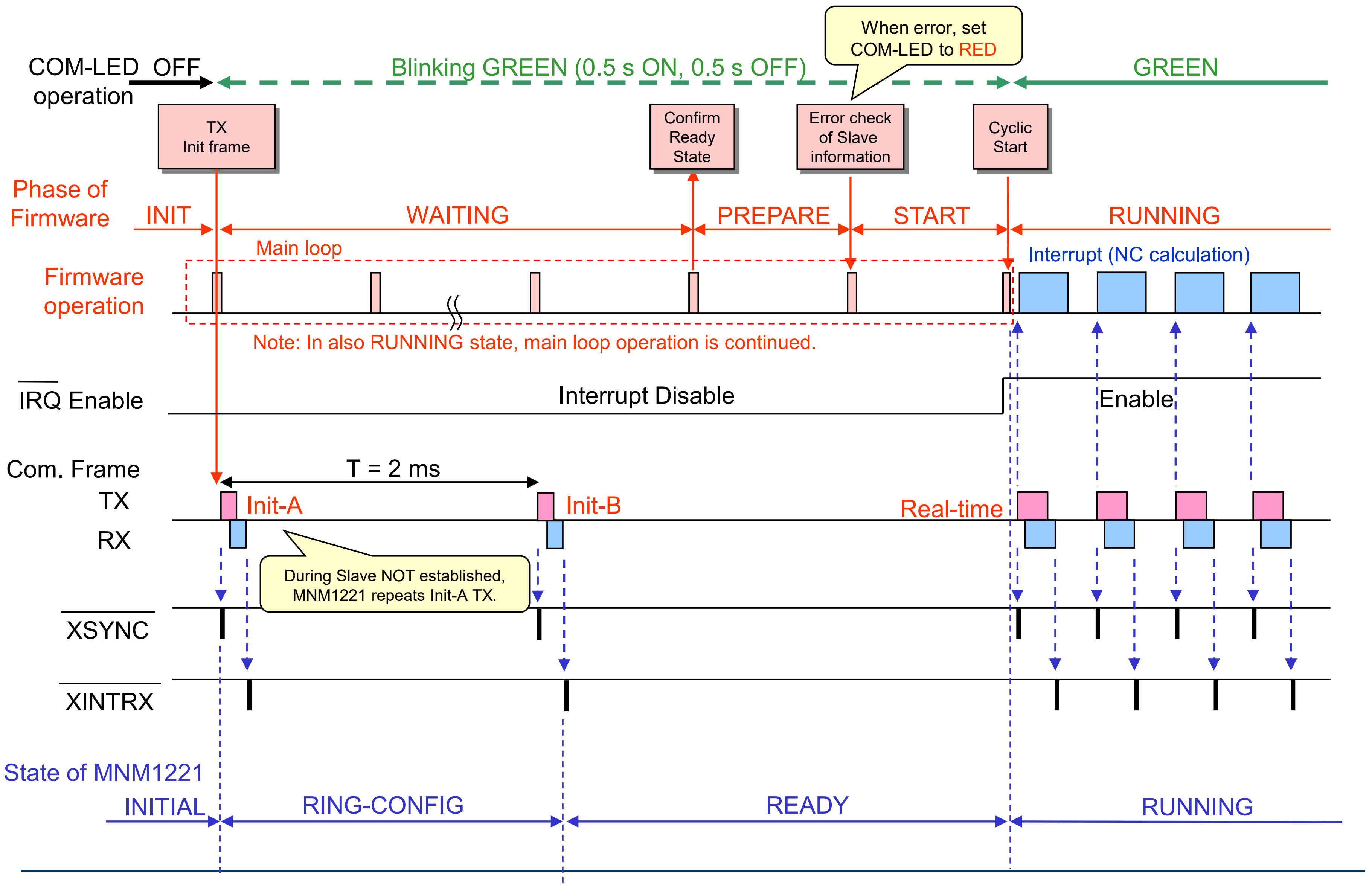


If timeout is detected, servo-off must be commanded to all axes for safety.

Modifying the Example Code

See also corresponding clause in chapter 1.
There are some abbreviations to prevent duplicate descriptions.

Timing at Start-up

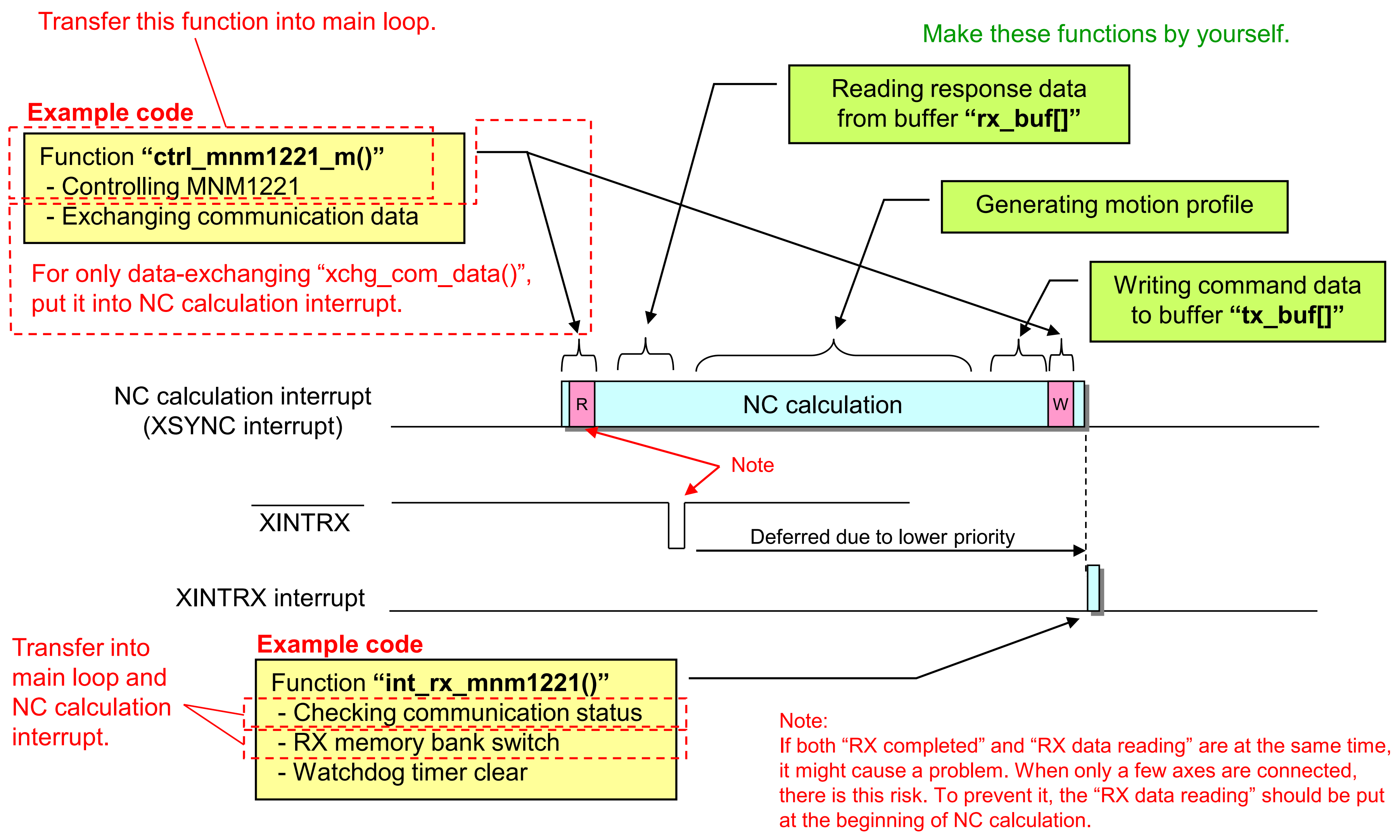


Task Assignment

Task	Trigger	Priority	Period	Operation
Main loop	-	-	-	- Controlling MNM1221 (including com-status check) - “COM” LED control
XSYNC Interrupt	TX start	-	e.g. 0.5 ms	- Communication data exchange - NC calculation - Timeout detection
XINTRX Interrupt	RX completed	Lower than XSYNC	Same as XSYNC	- Watchdog timer clear

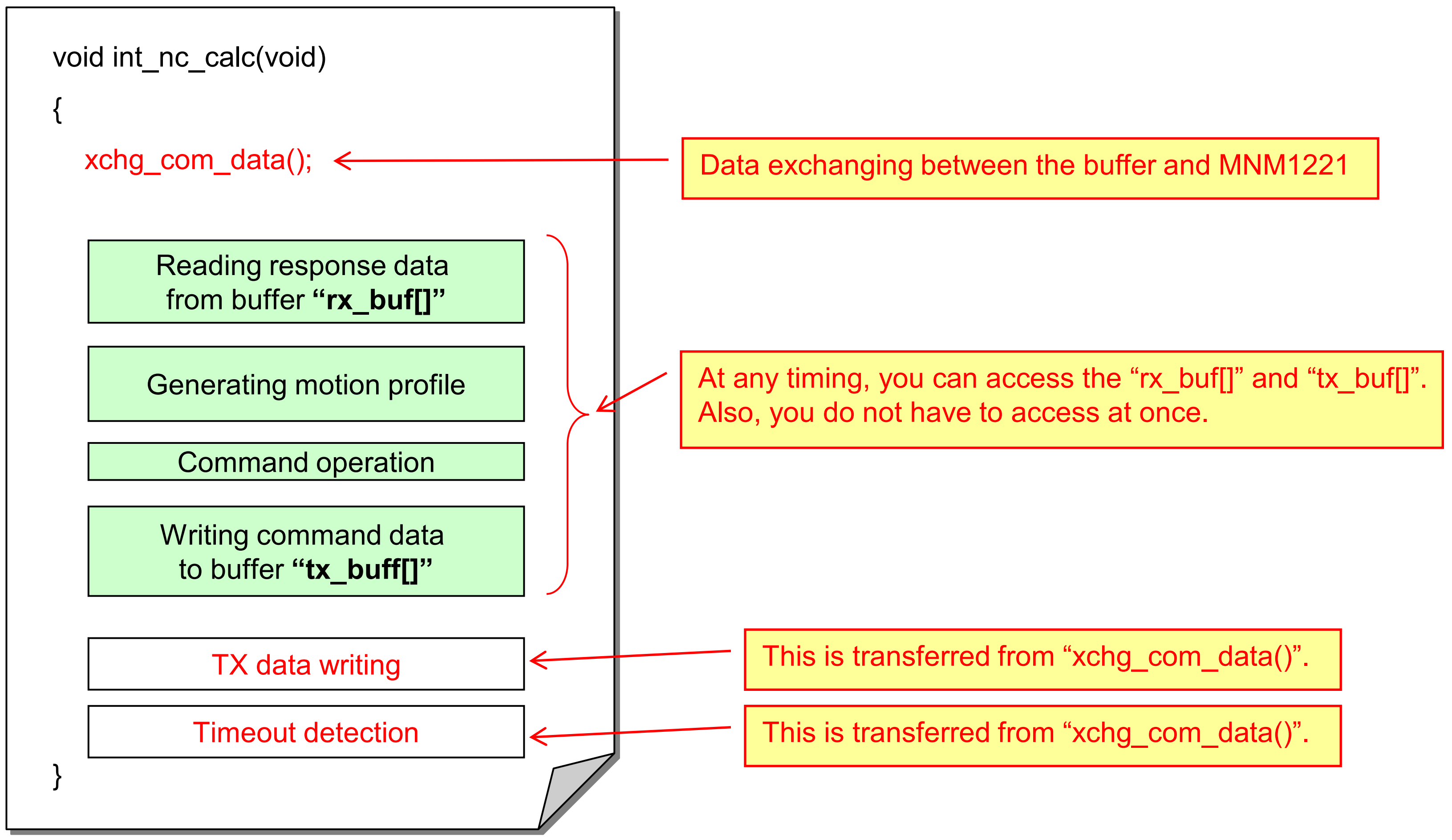
Notes:
- The interrupts must be disabled until RUNNING state.
- If not using XINTRX interrupt, a timeout detection with Update Counter Echo is necessary.

Location of Example Codes

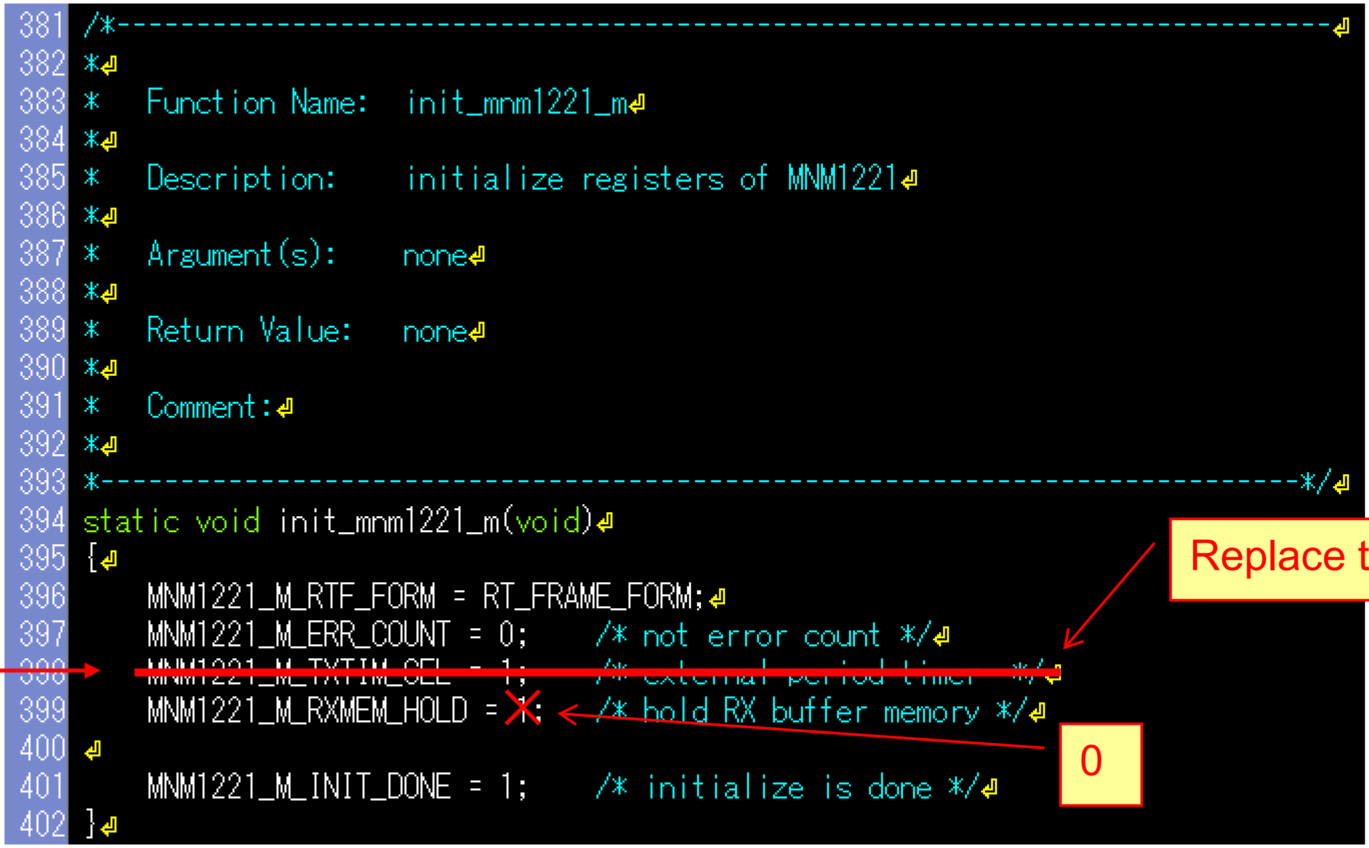

Transfer this function into main loop.
Make these functions by yourself.
Example code
Function "**ctrl_mnm1221_m()**"
- Controlling MNM1221
- Exchanging communication data
For only data-exchanging "xchg_com_data()", put it into NC calculation interrupt.
Reading response data from buffer "**rx_buf[]**"
Generating motion profile
Writing command data to buffer "**tx_buf[]**"
NC calculation interrupt (XSYNC interrupt)
R
NC calculation
W
Note
XINTRX
Deferred due to lower priority
XINTRX interrupt
Transfer into main loop and NC calculation interrupt.
Example code
Function "**int_rx_mnm1221()**"
- Checking communication status
- RX memory bank switch
- Watchdog timer clear

Note:
If both "RX completed" and "RX data reading" are at the same time, it might cause a problem. When only a few axes are connected, there is this risk. To prevent it, the "RX data reading" should be put at the beginning of NC calculation.

An Example of Location


void int_nc_calc(void)
{
xchg_com_data();
Reading response data from buffer “**rx_buf[]**”
Generating motion profile
Command operation
Writing command data to buffer “**tx_buff[]**”
TX data writing
Timeout detection
}
Data exchanging between the buffer and MNM1221
At any timing, you can access the “rx_buf[]” and “tx_buf[]”. Also, you do not have to access at once.
This is transferred from “xchg_com_data()”.
This is transferred from “xchg_com_data()”.

MNM1221 initializing



```
/*-----------------------------------------------------------------------
*
*   Function Name:  init_mnm1221_m
*
*   Description:    initialize registers of MNM1221
*
*   Argument(s):    none
*
*   Return Value:   none
*
*   Comment:
*
*---------------------------------------------------------------------*/
static void init_mnm1221_m(void)
{
    MNM1221_M_RTF_FORM = RT_FRAME_FORM;
    MNM1221_M_ERR_COUNT = 0;    /* not error count */
~~    MNM1221_M_TXTIM_SEL = 1;    /* external period timer */~~
    MNM1221_M_RXMEM_HOLD = ~~1~~;  /* hold RX buffer memory */

    MNM1221_M_INIT_DONE = 1;    /* initialize is done */
}
```

Replace this line.

0

```
MNM1221_M_TX_PERIOD = 0x30D4;   /* T = 0.5 ms */
MNM1221_M_TXTIM_SEL = 0;        /* internal timer */
```

Communication Period	MNM1221_M_TX_PERIOD
1 ms	0x61A8
0.5 ms	0x30D4
0.25 ms	0x186A

Reading State

ctrl_mnm1221_m() in mnm1221_m.c

Put this function into main loop.

```
short ctrl_mnm1221_m(void)
{
/* Select either of the followings according to your initializing process.  */
#if 0
    static short phase = 0;
#else
    static short phase;     /* need RAM clear after reset-release separately */
#endif
    mnm1221.state = MNM1221_M_STATE;
    switch (phase) {
    /*--- initializing ----------------------------------------------------*/
    case PH_INIT:                    /* MNM1221 is in INITIAL state.  */
        phase = PH_WAITING;

        com_err.init = 0;
        com_err.run = 0;
        com_err.count = 0;

   /*    mnm1221.state = MNM1221_M_STATE;        */
        init_mnm1221_m();
        MNM1221_M_INITF_TX = 1;     /* transmit initial frame */
        break;

    /*--- waiting for ready state ----------------------------------------*/
```

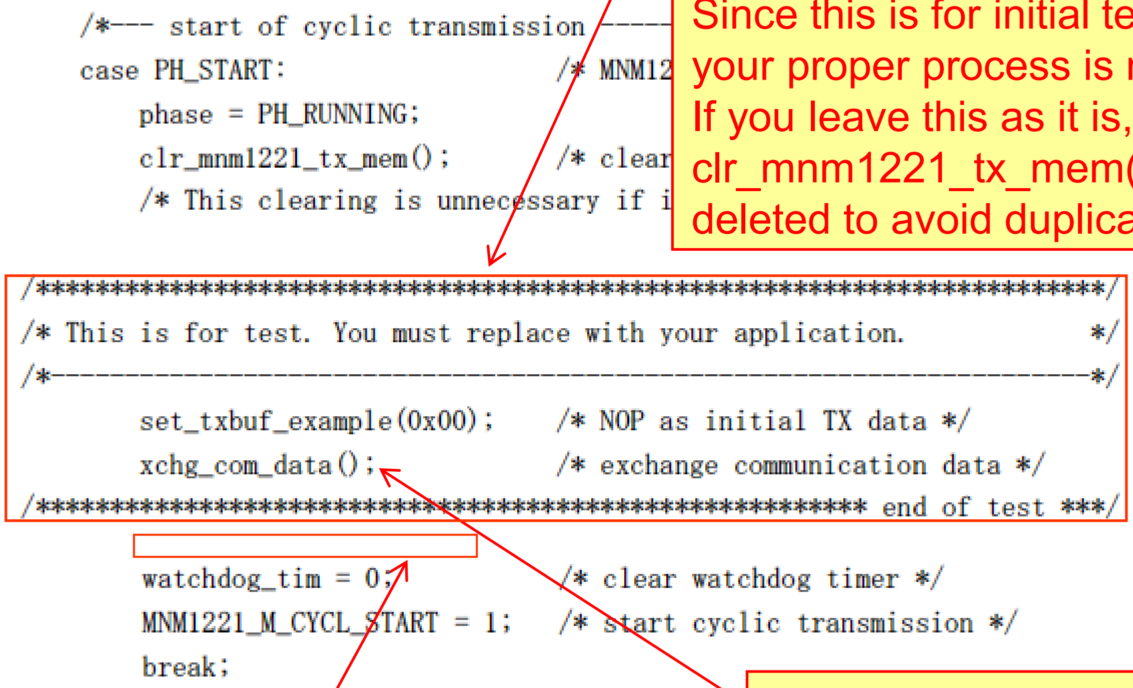
Insert this line.

Here, insert an interrupt disable function provided by yourself.

Delete this line.

Starting Cyclic Transmission

ctrl_mnm1221_m() in mnm1221_m.c



```
    /*--- start of cyclic transmission -----
    case PH_START:                    /* MNM12
        phase = PH_RUNNING;
        clr_mnm1221_tx_mem();         /* clear
        /* This clearing is unnecessary if i

/*************************************************************************/
/* This is for test.  You must replace with your application.             */
/*-----------------------------------------------------------------------*/
        set_txbuf_example(0x00);      /* NOP as initial TX data */
        xchg_com_data();              /* exchange communication data */
/*************************************************** end of test ***/

        watchdog_tim = 0;             /* clear watchdog timer */
        MNM1221_M_CYCL_START = 1;     /* start cyclic transmission */
        break;
```

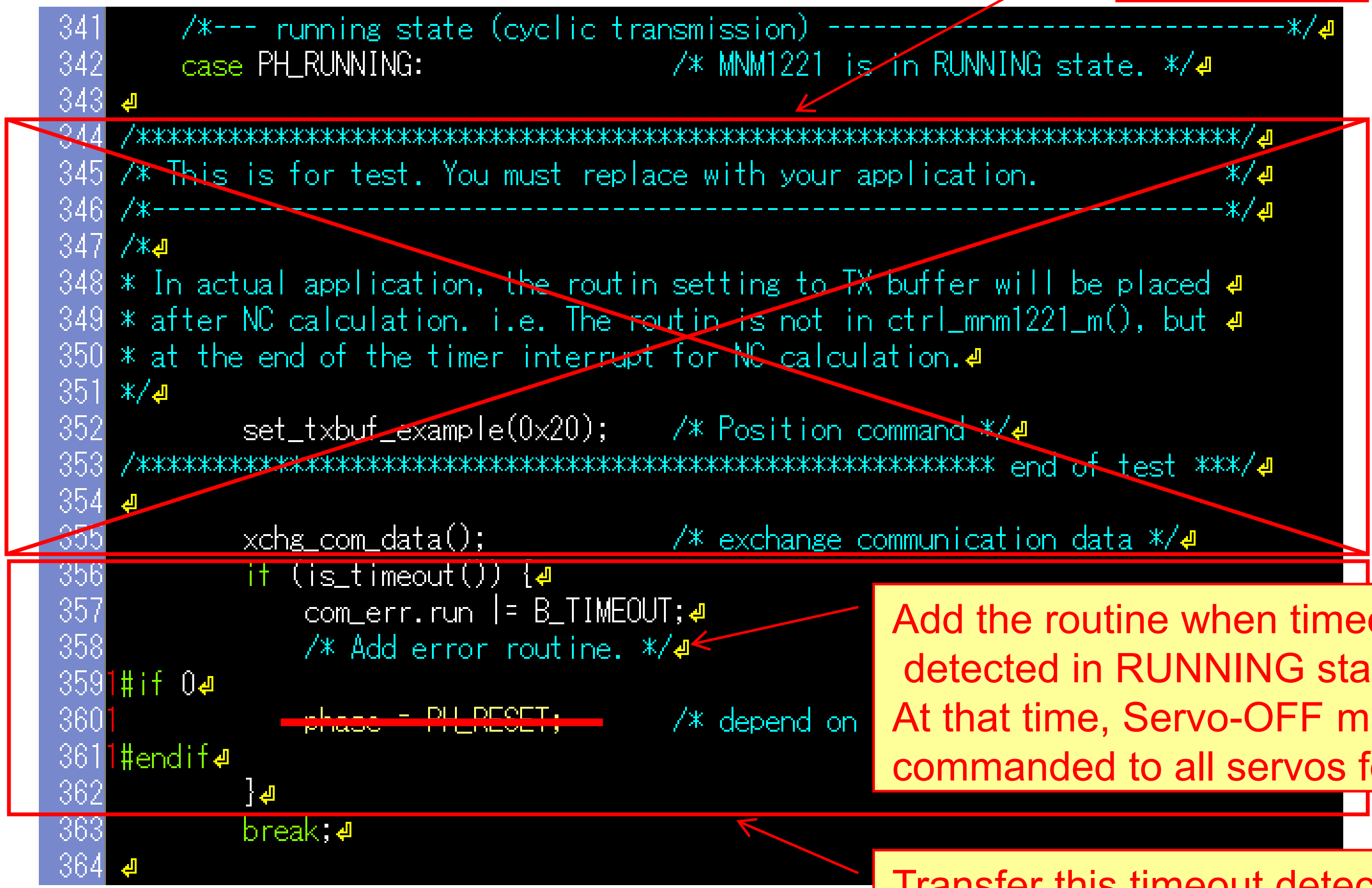
This process is to set a initial transmit data including MACID.
Since this is for initial test,
your proper process is needed.
If you leave this as it is,
clr_mnm1221_tx_mem() should be deleted to avoid duplicate initializing.

Replace with TX data writing extracted from xchg_com_data().

Here, insert an interrupt enable function provided by yourself.

In Running State

ctrl_mnm1221_m() in mnm1221_m.c



```
    /*--- running state (cyclic transmission) ------------------------------*/
    case PH_RUNNING:                    /* MNM1221 is in RUNNING state. */

/*************************************************************************/
/* This is for test. You must replace with your application.            */
/*-----------------------------------------------------------------------*/
/*
* In actual application, the routin setting to TX buffer will be placed
* after NC calculation. i.e. The routin is not in ctrl_mnm1221_m(), but
* at the end of the timer interrupt for NC calculation.
*/
        set_txbuf_example(0x20);    /* Position command */
/*********************************************************** end of test ***/

        xchg_com_data();            /* exchange communication data */
        if (is_timeout()) {
            com_err.run |= B_TIMEOUT;
            /* Add error routine. */
#if 0
        ~~phase = PH_RESET;~~       /* depend on
#endif
        }
        break;

```

Delete this.

Add the routine when timeout detected in RUNNING state.
At that time, Servo-OFF must be commanded to all servos for safety.

Transfer this timeout detection into NC calculation interrupt.

In RUNNING state, since the data exchanging is transferred to the interrupt, there is nothing to be done in main loop.

Data Exchanging

xchg_com_data() in mnm1221_m.c

Put this function to the beginning of NC calculation interrupt.

```
static void xchg_com_data(void)
{
#ifdef MASTER_16BIT_ACCESS   /* 16bits bus width */

    short i;
    volatile unsigned short *p;
```

unsigned long ul_temp;

Transfer this to the end of NC calculation interrupt. Note this is for 16-bit bus.

```
    /* TX */
    for (i = 0; i < mnm1221.blk_sum; i++) {
        p = P_MNM1221_TX_MEM_BGN;
        p += (order_ref_tbl[i] * N_DATA);
        mem_copy_s(p, &(tx_buf[i].data[0]), N_DATA);
    }
    MNM1221_M_TXMEM_SW = 1;           /* switch TX bank  */
```

TX data writing for 16-bit bus

```
ul_temp = MNM1221_M_DCRC_ERR_H;
ul_temp <<= 16;
ul_temp |= MNM1221_M_DCRC_ERR_L;
mnm1221.data_crc_err = ul_temp;

If (!mnm1221.data_crc_err) {
   MNM1221_M_RXMEM_HOLD = 1;
```

```
    /* RX */
    for (i = 0; i < mnm1221.blk_sum; i++) {
        p = P_MNM1221_RX_MEM_BGN;
        p += (order_ref_tbl[i] * N_DATA);
        mem_copy_s(&(rx_buf[i].data[0]), p, N_DATA);
    }
```

RX data reading for 16-bit bus

```
   MNM1221_M_RXMEM_HOLD = 0;
}
```

```
#else                         /* 32bits bus width */
```

Panasonic INDUSTRY

Your Committed Enabler

Data Exchanging (Cont.)

```
#else                           /* 32bits bus width */

    short i;
    volatile unsigned long *p;

    /* TX */
    for (i = 0; i < mnm1221.blk_sum; i++) {
        p = P_MNM1221_TX_MEM_BGN;
        p += (order_ref_tbl[i] * N_DATA);
        mem_copy_l(p, &(tx_buf[i].data[0]), N_DATA);
    }
    MNM1221_M_TXMEM_SW = 1;          /* switch TX bank  */

    /* RX */
    for (i = 0; i < mnm1221.blk_sum; i++) {
        p = P_MNM1221_RX_MEM_BGN;
        p += (order_ref_tbl[i] * N_DATA);
        mem_copy_l(&(rx_buf[i].data[0]), p, N_DATA);
    }
#endif
}
```

Transfer this to the end of NC calculation interrupt. Note this is for 32-bit bus.

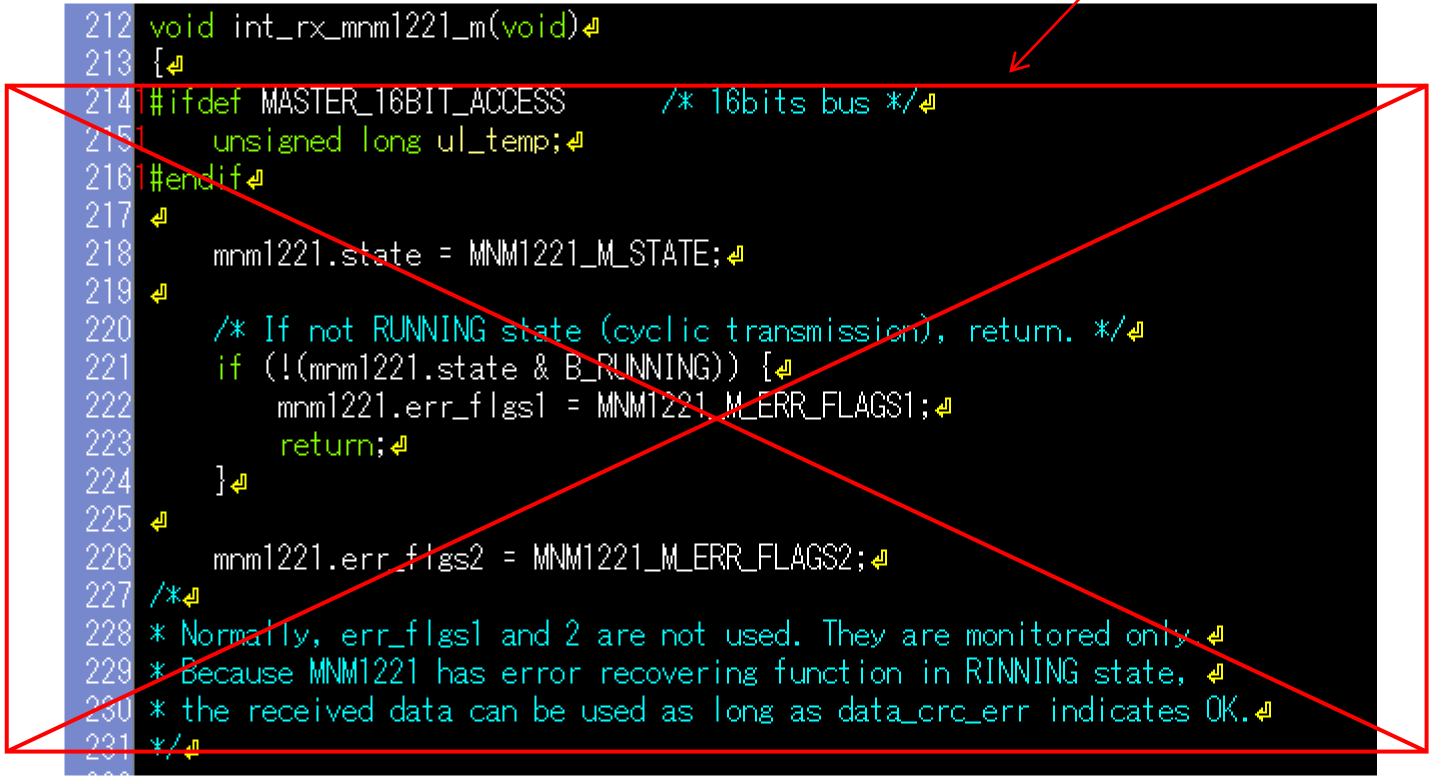
TX data writing for 32-bit bus

RX data reading for 32-bit bus

```
mnm1221.data_crc_err = MNM1221_M_DCRC_ERR;

If (!mnm1221.data_crc_err) {
   MNM1221_M_RXMEM_HOLD = 1;

   MNM1221_M_RXMEM_HOLD = 0;
}
```

XINTRX Interrupt

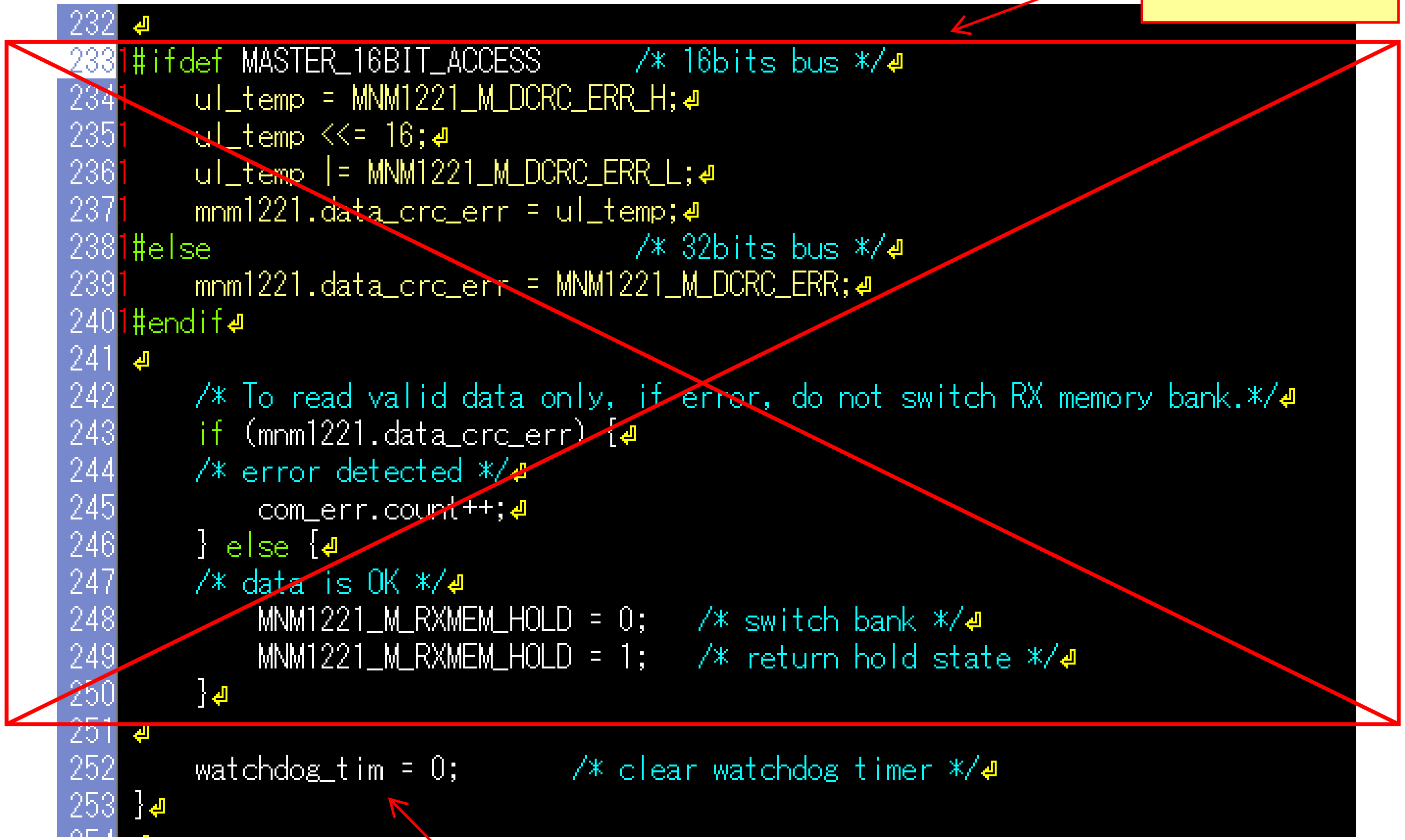
Delete this.



```
void int_rx_mnm1221_m(void)
{
#ifdef MASTER_16BIT_ACCESS        /* 16bits bus */
    unsigned long ul_temp;
#endif

    mnm1221.state = MNM1221_M_STATE;

    /* If not RUNNING state (cyclic transmission), return. */
    if (!(mnm1221.state & B_RUNNING)) {
        mnm1221.err_flgs1 = MNM1221_M_ERR_FLAGS1;
        return;
    }

    mnm1221.err_flgs2 = MNM1221_M_ERR_FLAGS2;
/*
* Normally, err_flgs1 and 2 are not used. They are monitored only
* Because MNM1221 has error recovering function in RINNING state,
* the received data can be used as long as data_crc_err indicates OK.
*/
```

XINTRX Interrupt (Cont.)

Delete this.



```

#ifdef MASTER_16BIT_ACCESS      /* 16bits bus */
    ul_temp = MNM1221_M_DCRC_ERR_H;
    ul_temp <<= 16;
    ul_temp |= MNM1221_M_DCRC_ERR_L;
    mnm1221.data_crc_err = ul_temp;
#else                           /* 32bits bus */
    mnm1221.data_crc_err = MNM1221_M_DCRC_ERR;
#endif

    /* To read valid data only, if error, do not switch RX memory bank.*/
    if (mnm1221.data_crc_err) {
    /* error detected */
        com_err.count++;
    } else {
    /* data is OK */
        MNM1221_M_RXMEM_HOLD = 0;   /* switch bank */
        MNM1221_M_RXMEM_HOLD = 1;   /* return hold state */
    }

    watchdog_tim = 0;        /* clear watchdog timer */
}
```

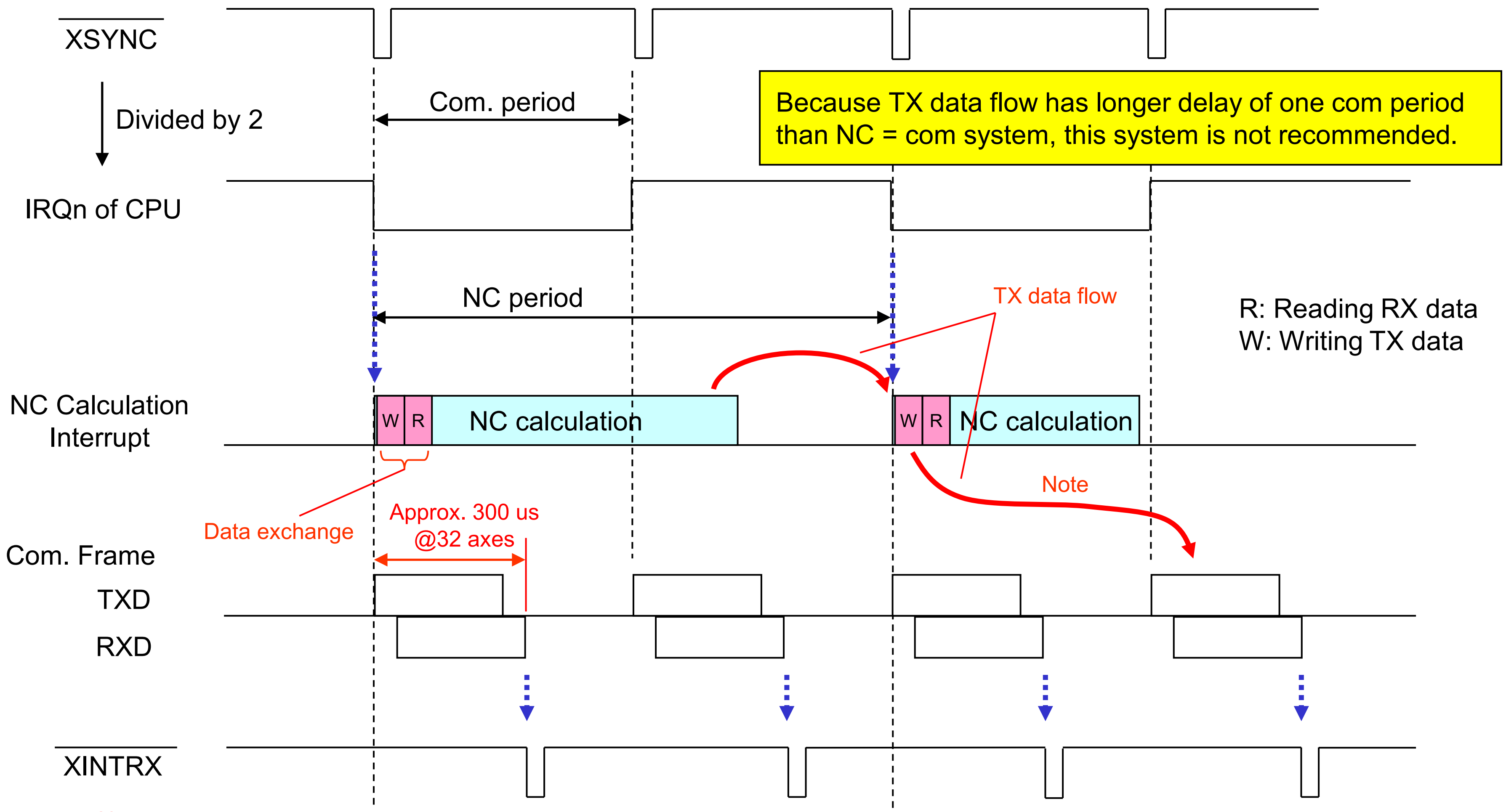
Leave “watchdog timer clear” only.

Appendix

System Consideration for 2 times communication one NC calculation

This system is not recommended because of longer delay in TX data flow.

Timing Chart

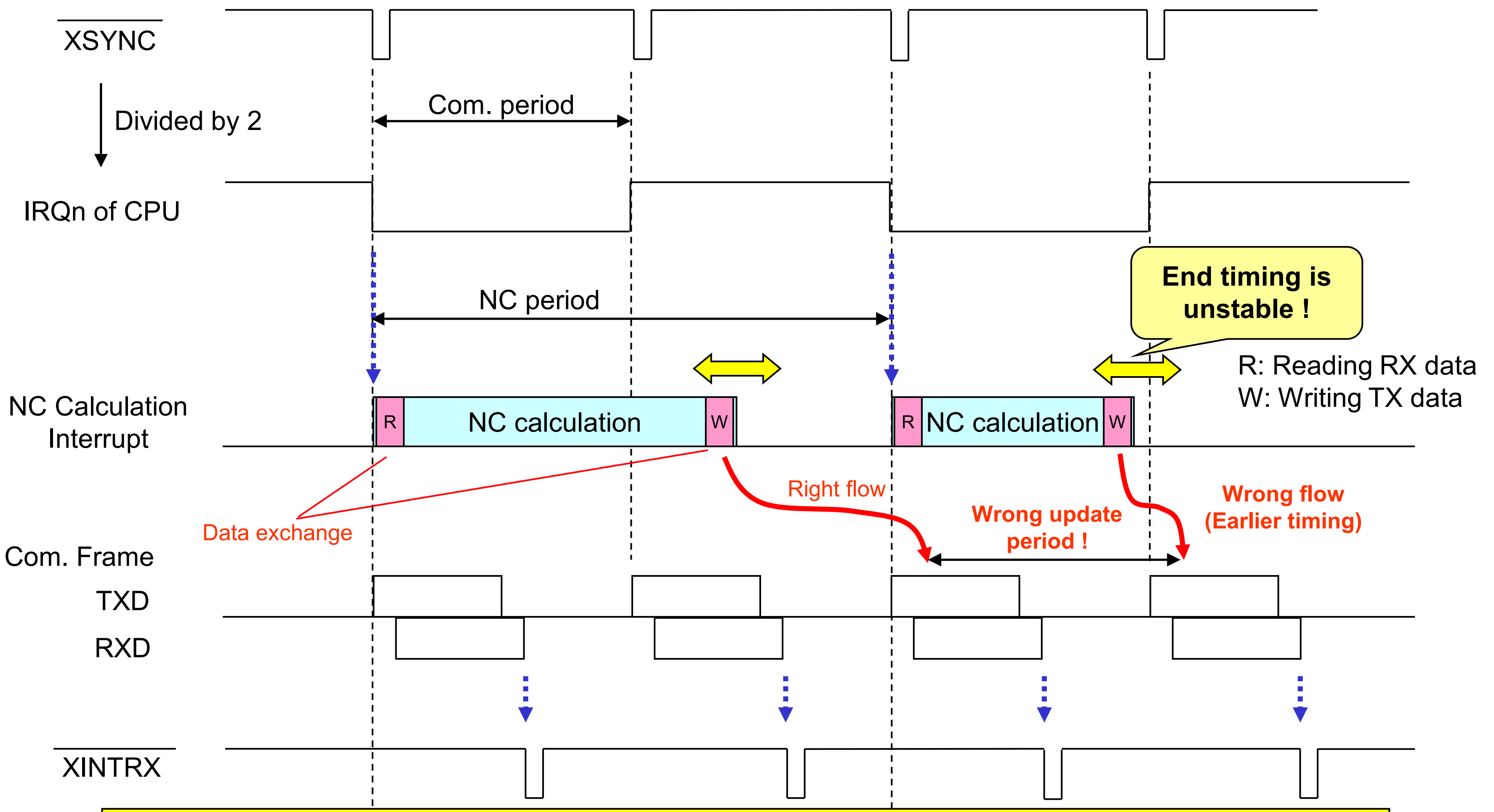


Because TX data flow has longer delay of one com period than NC = com system, this system is not recommended.

R: Reading RX data
W: Writing TX data

Note:
At XSYNC falling edge, a frame has already been in transmitting, and the memory-bank switching after writing TX-data is deferred.
Therefore, the transmit timing is delayed for one communication period.

BAD Example



If TX data writing is done at the end of NC calculation, the transmit timing can be shorter delay. However, the data update period cannot be constant because the end timing is unstable.

Panasonic
INDUSTRY