# System Design Guide for Master

Motor Business Unit
Appliances Company

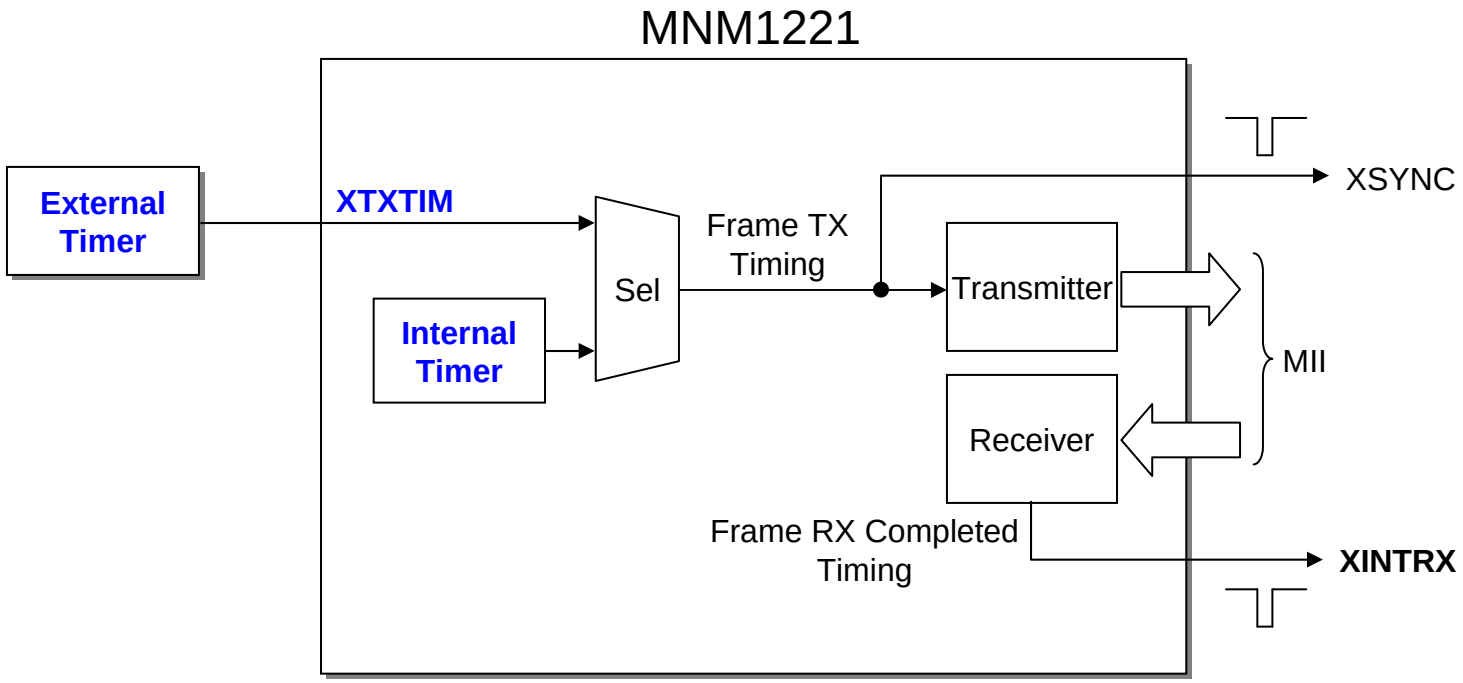
# Revision History

Panasonic
ideas for life

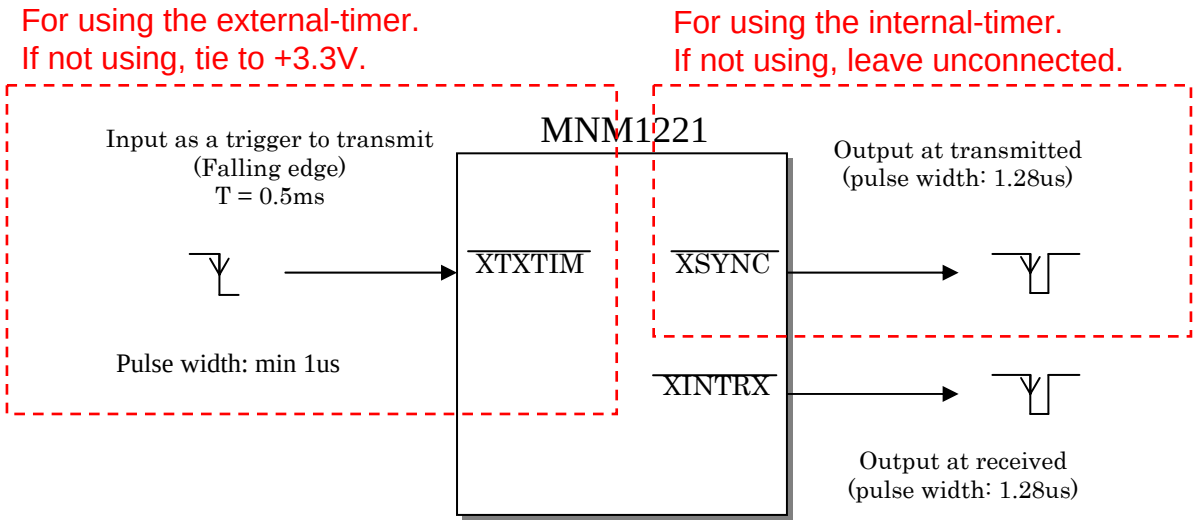
| Revision | Date | Change Description |
|---|---|---|
| 1 | 2005/7/14 | Initial Release |
| 2 | 2012/1/31 | P1 Changed title from “A Guide for Firmware Development”.<br>P3 Added introduction.<br>P7 Added SH7216 example.<br>P19 Added SH7145 example.<br>P25 Added TMS320F28335 example.<br>Deleted SH7065 example.<br>Deleted TMS320VC33-120 example.<br>P58 Added overview of profile position I/F.<br>P72 Added internal-timer using system.<br>Minor edits. |
| | | |

# 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.

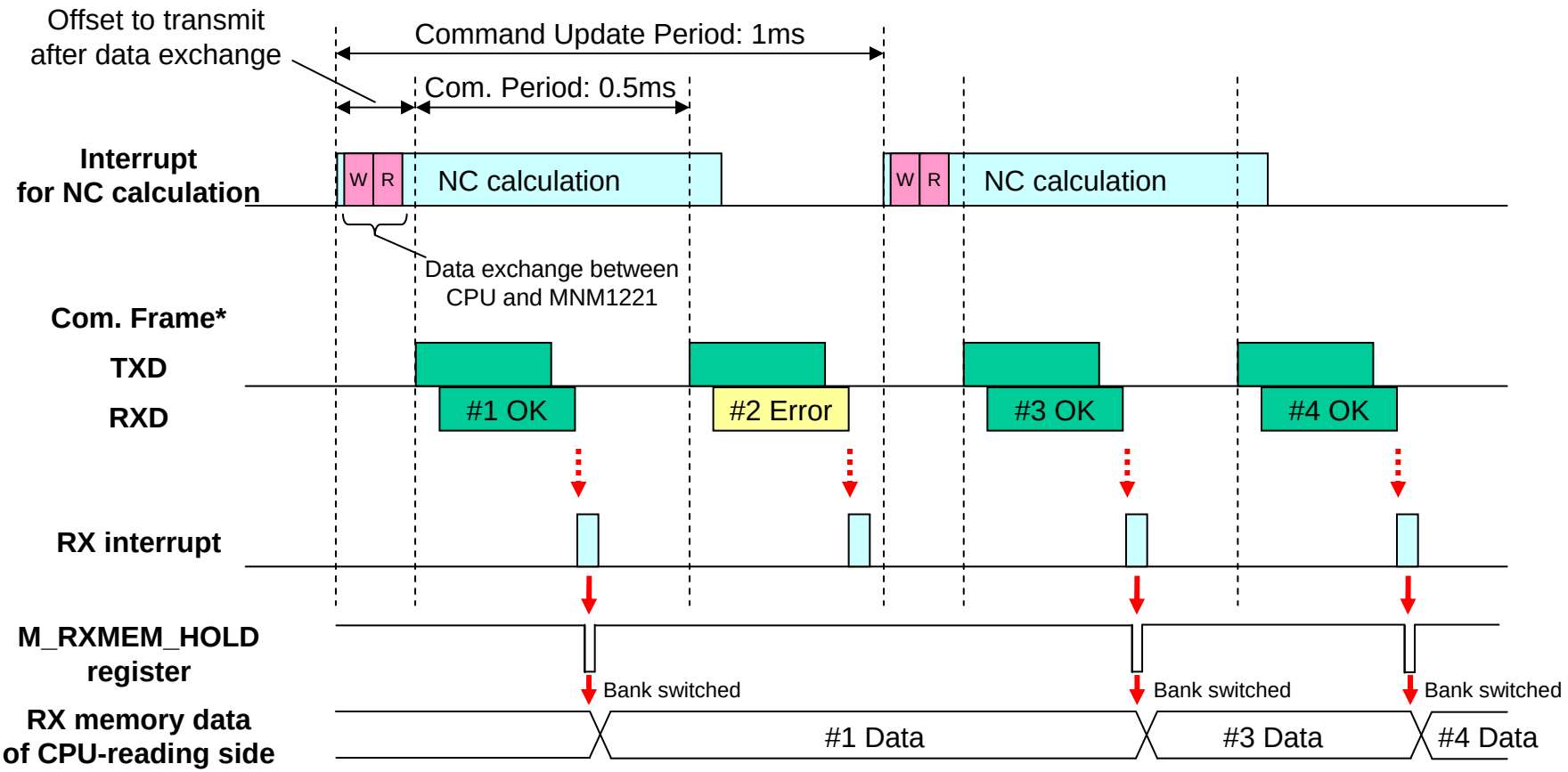
# Chapter 1

## External-Timer Using System

# Goal of Timing Control

Panasonic
ideas for life

The goal is to make the following timing:



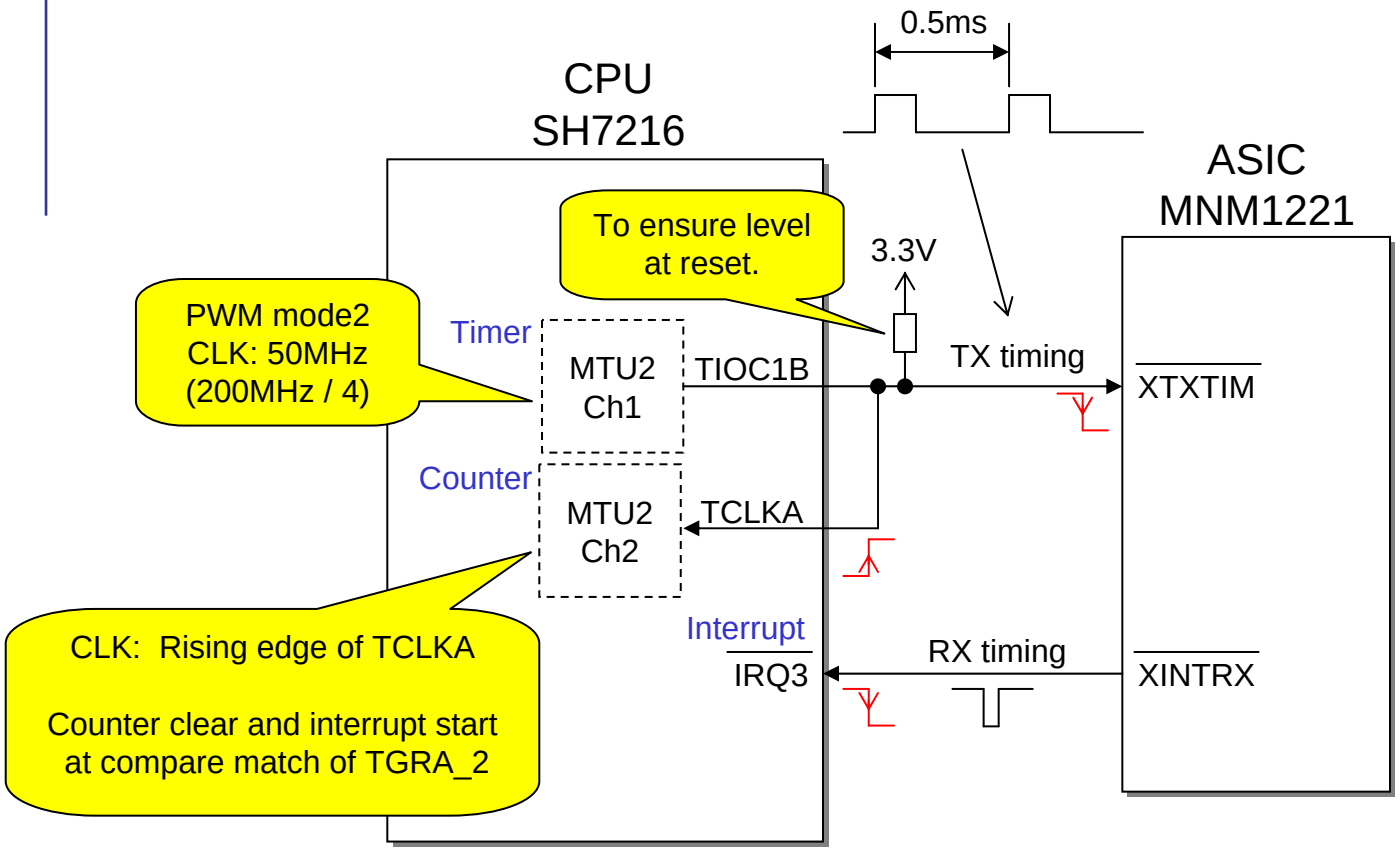

If a CRC error is detected in RX data, the RX bank is not switched in RX interrupt routine.
In this case, previous RX data is read in the data exchange at the beginning of NC calculation.

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

Panasonic
ideas for life

# Example for SH7216 (Renesas)

# Timing Circuit

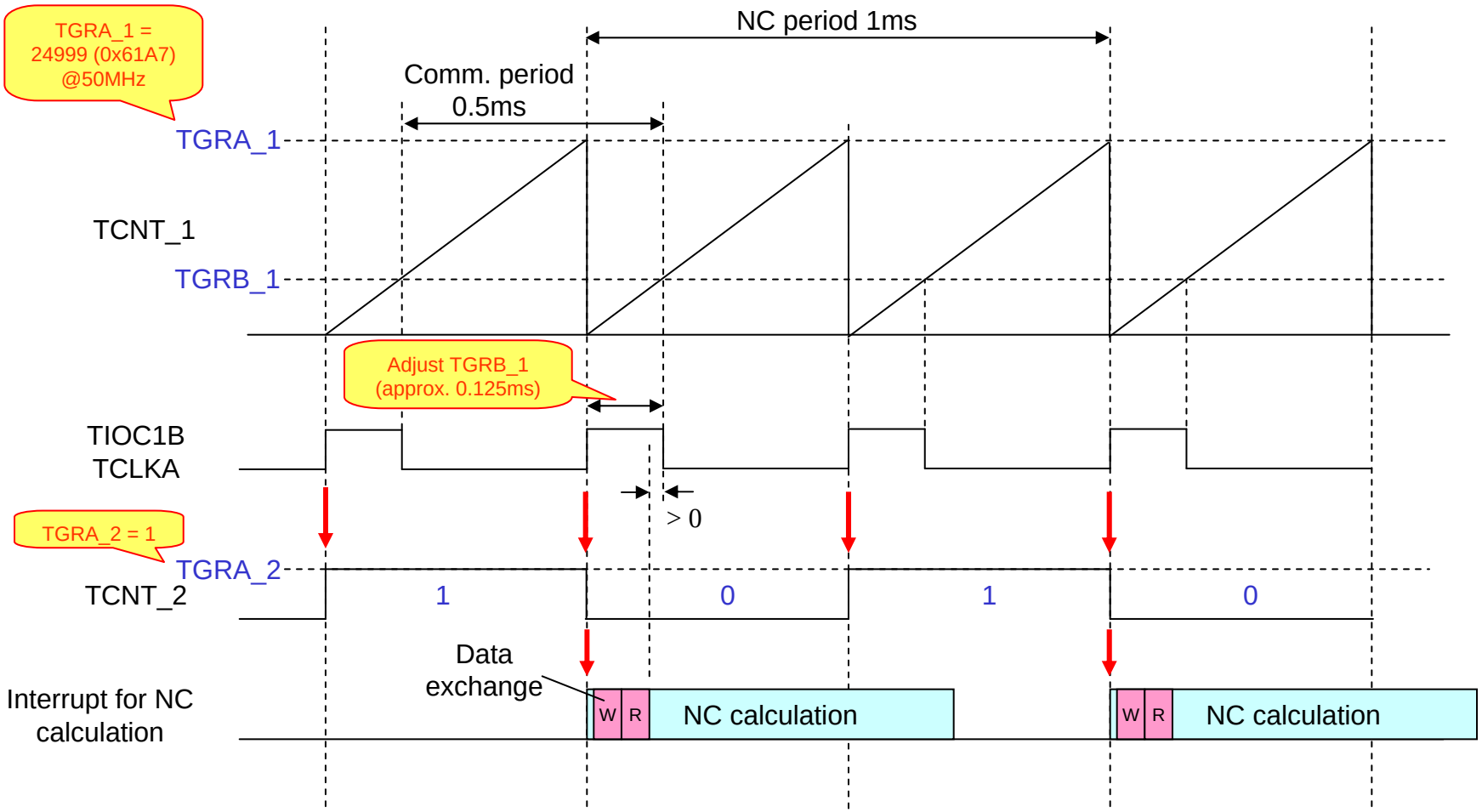



- MTU2-Ch1 generates TX timing signal. For 0.5ms, **TGRA_1 = 24999(0x61A7)@50MHz**
- MTU2-Ch2 divides this signal, and generates the start signal for NC calculating interrupt. For 1ms, **TGRA_2 = 1**
- IRQ3 by XINTRX of MNM1221 causes RX interrupt.

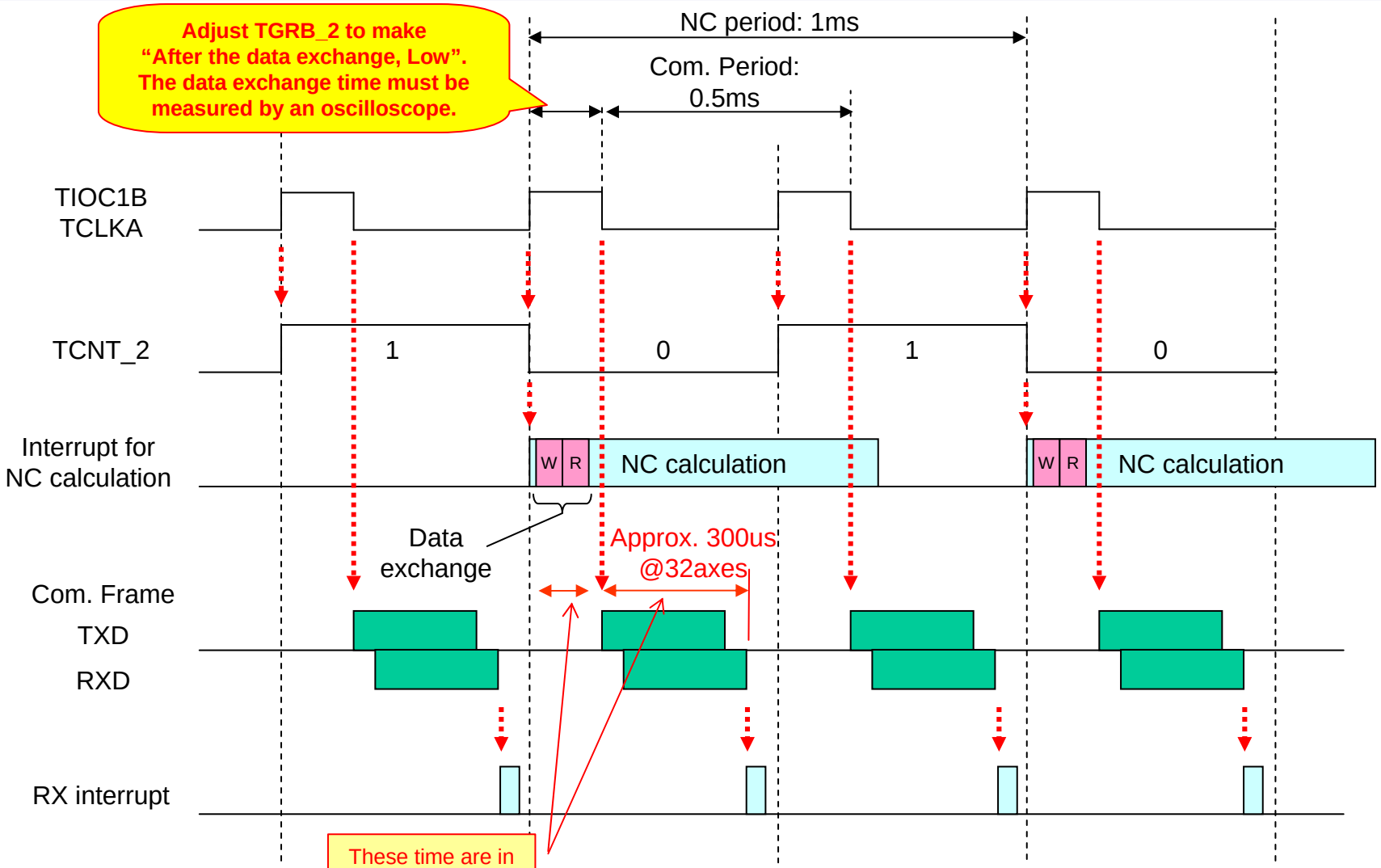
# Multiplex Interrupt Setting

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

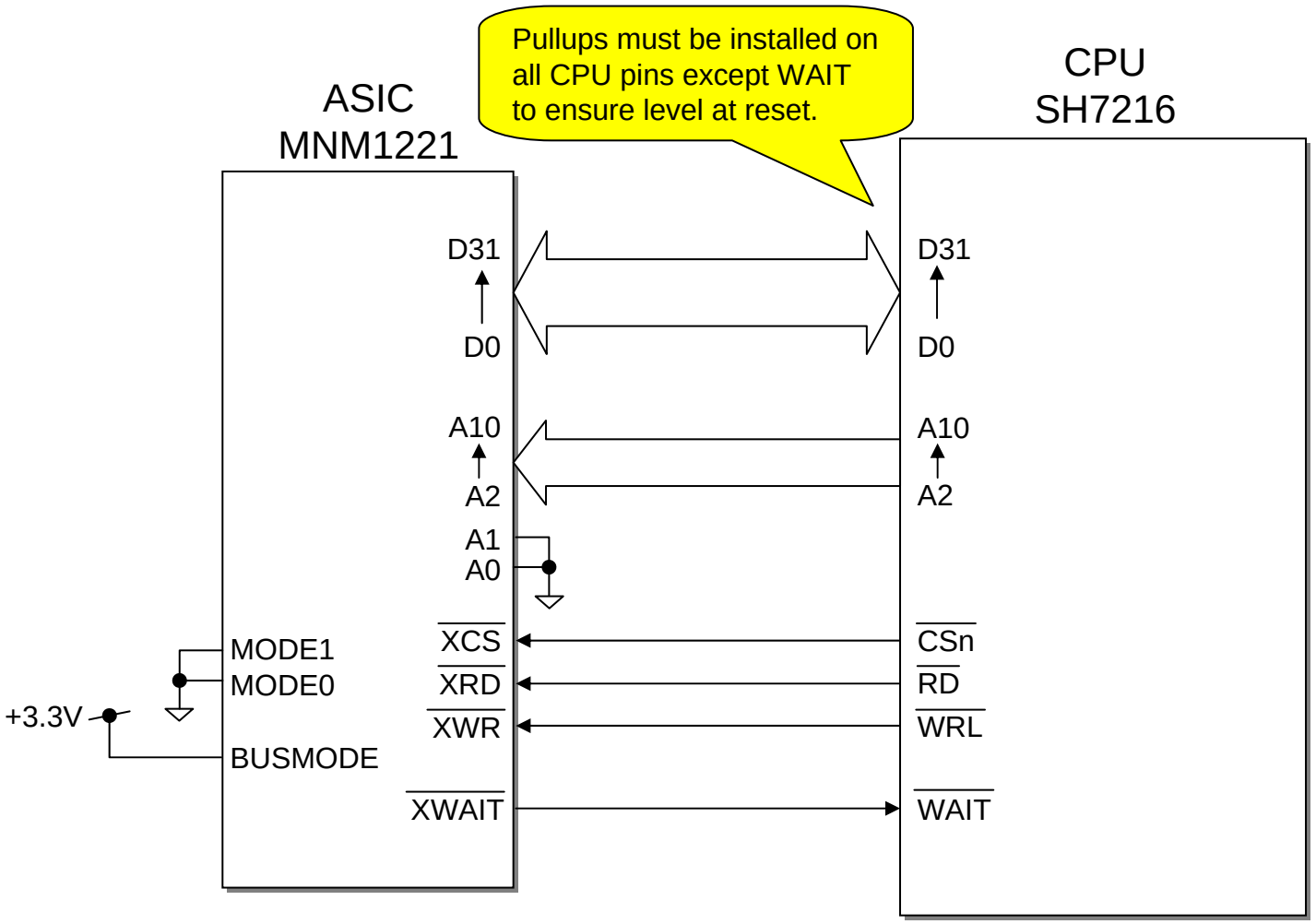
# Timing Chart 1



# Timing Chart 2

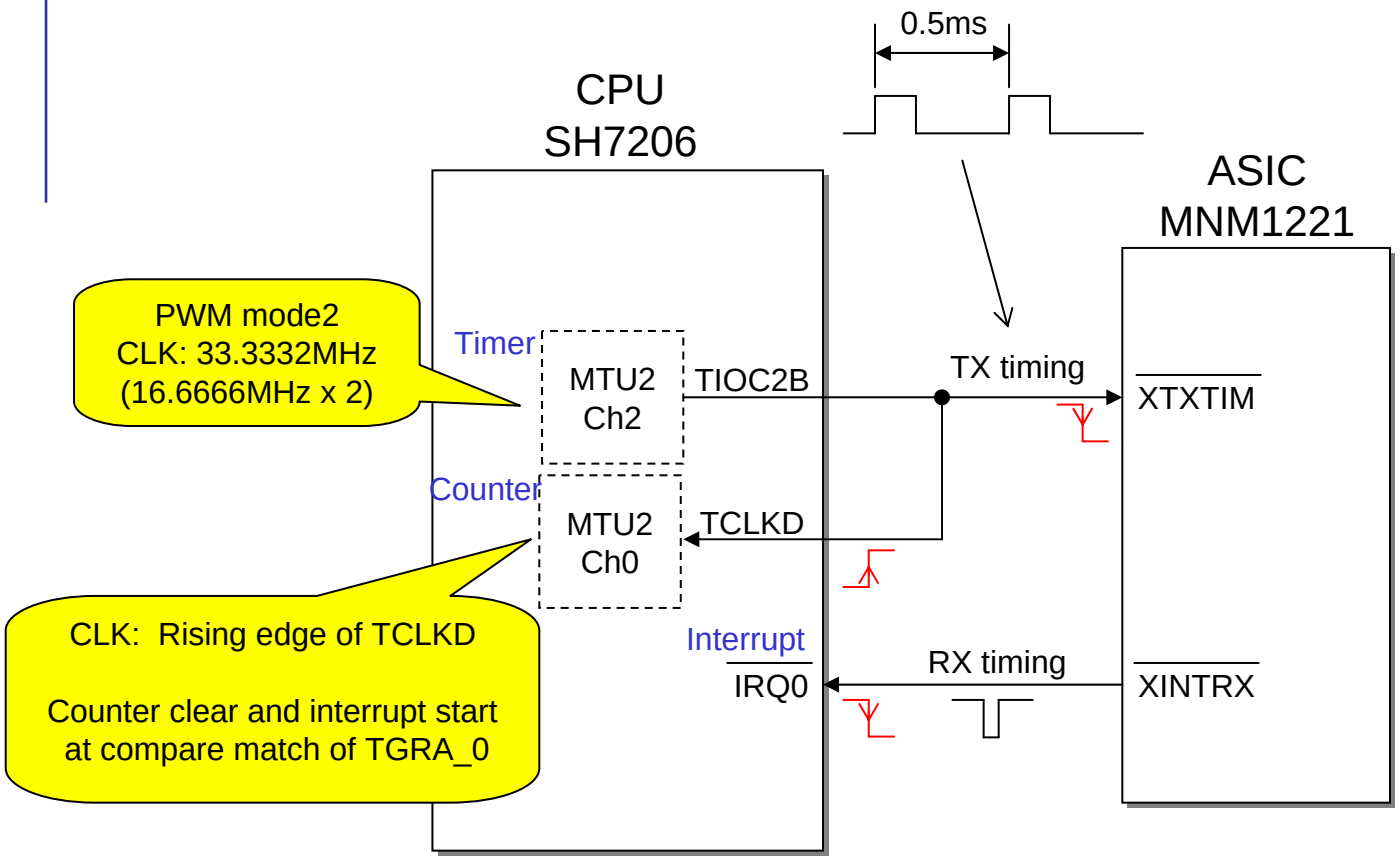


# Bus Connection



# Example for SH7206 (Renesas)

# Timing Circuit

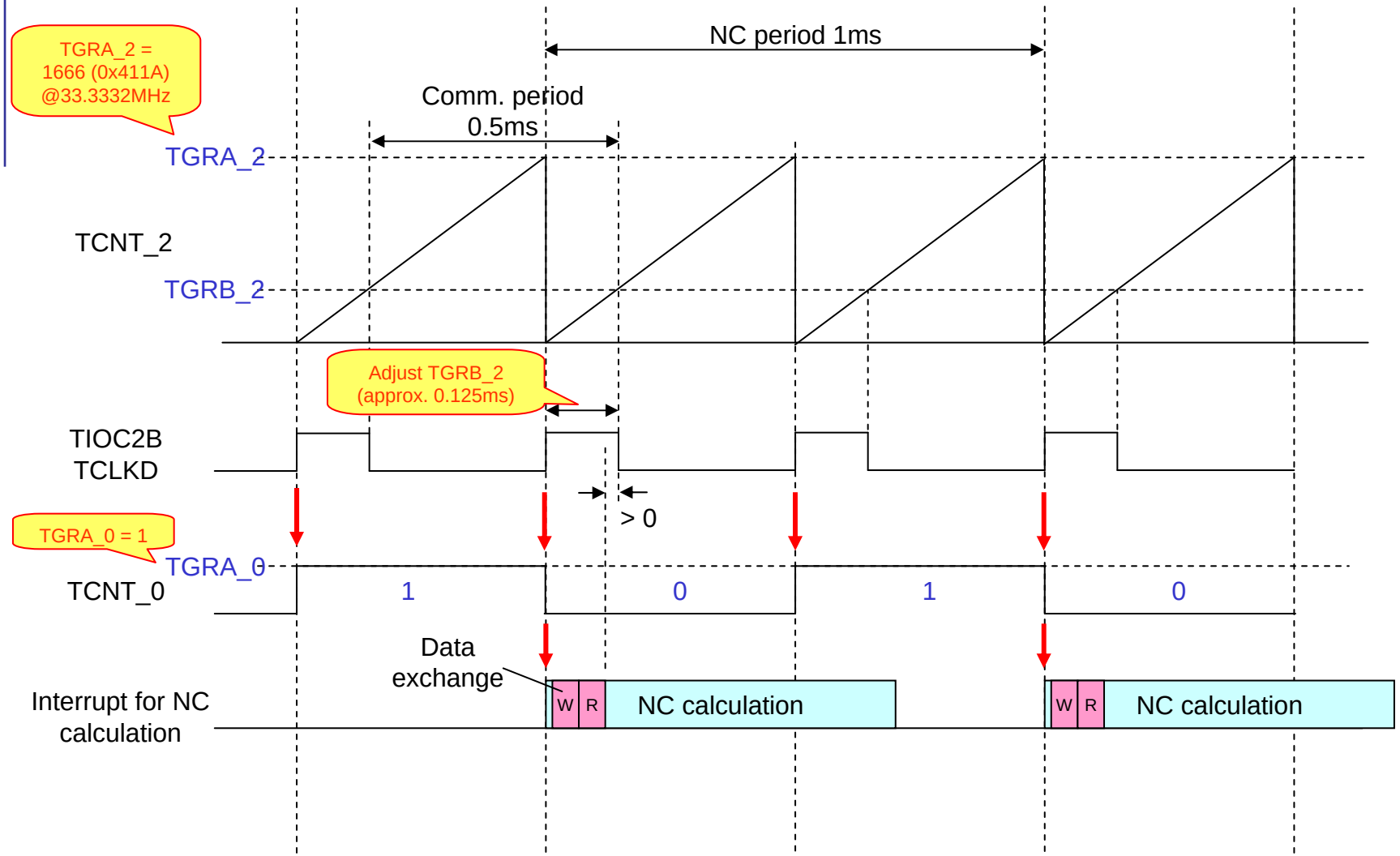



- MTU2-Ch2 generates TX timing signal. For 0.5ms, **TGRA_2 = 16666(0x411A)@33.3332MHz**
- MTU2-Ch0 divides this signal, and generates the start signal for NC calculating interrupt. For 1ms, **TGRA_0 = 1**
- IRQ2 by XINTRX of MNM1221 causes RX interrupt.

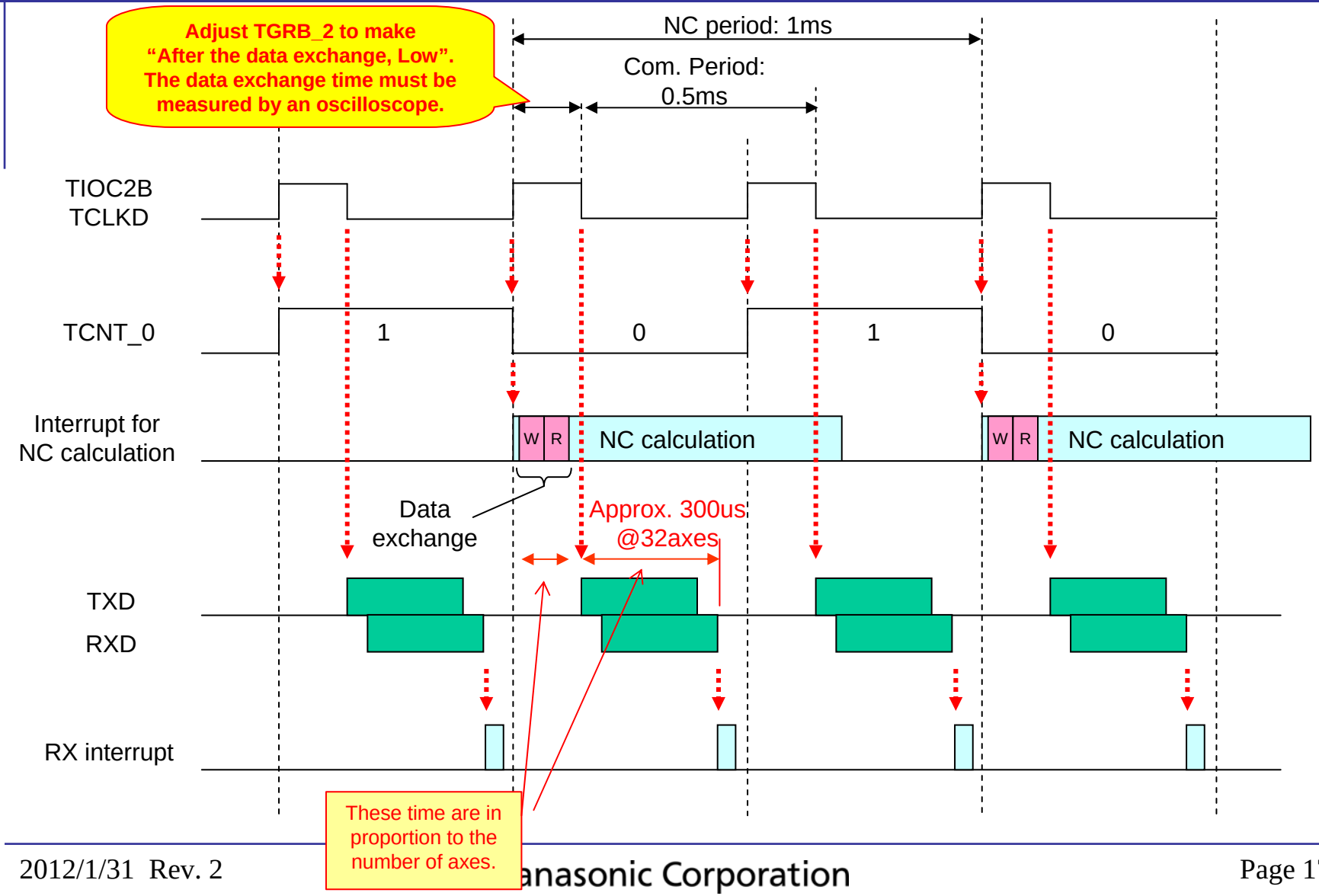
# Multiplex Interrupt Setting

| Source | Trigger | Priority | Period | Operation |
|---|---|---|---|---|
| TGIA_0 | Compare match of MTU2 Ch0 | - | 1ms | - Communication data exchange<br>- NC calculation |
| /IRQ0 | RX complete | Higher than TGIA_0 | 0.5ms | - Communication status check<br>- RX memory bank switch |

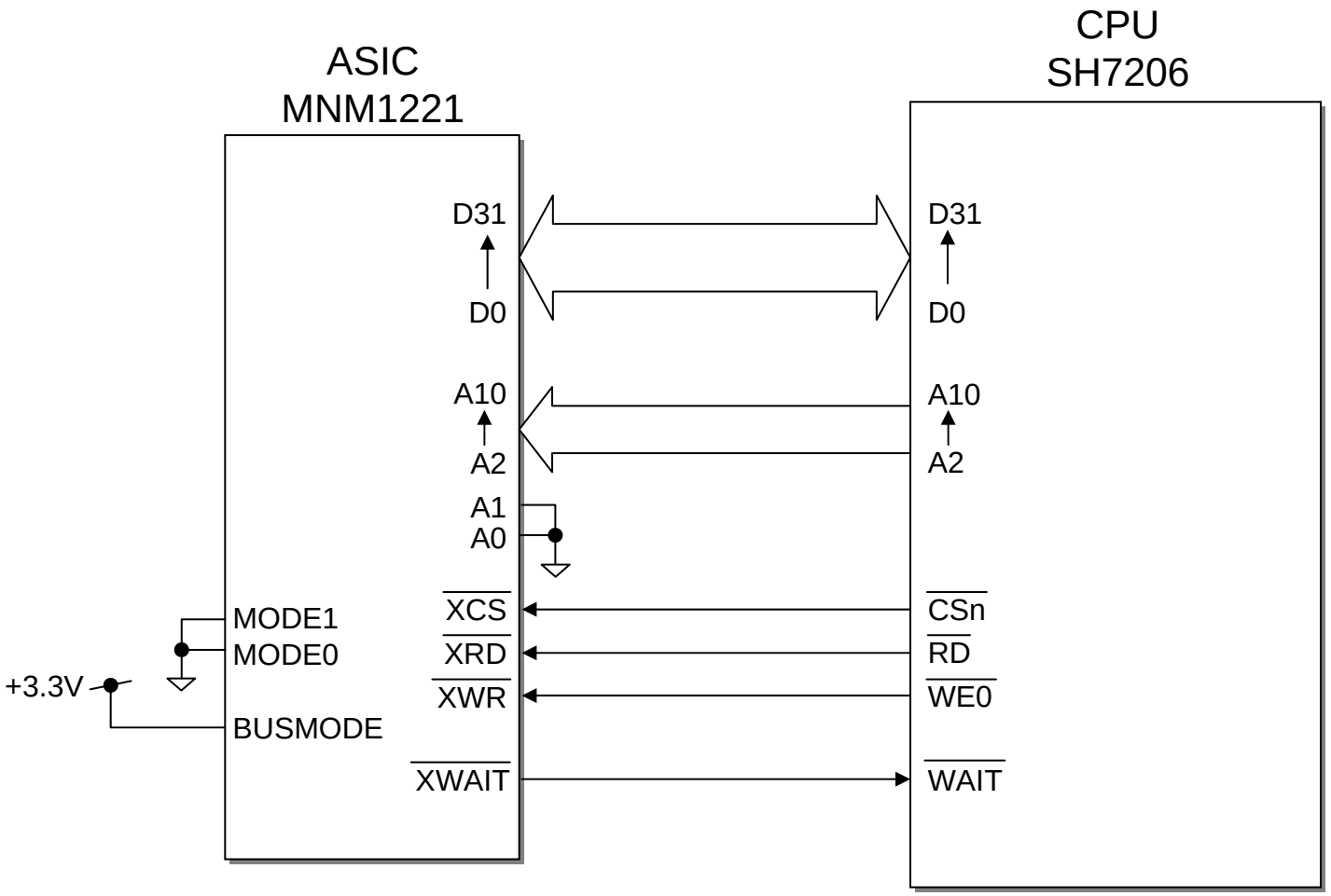
# Timing Chart 1



# Timing Chart 2



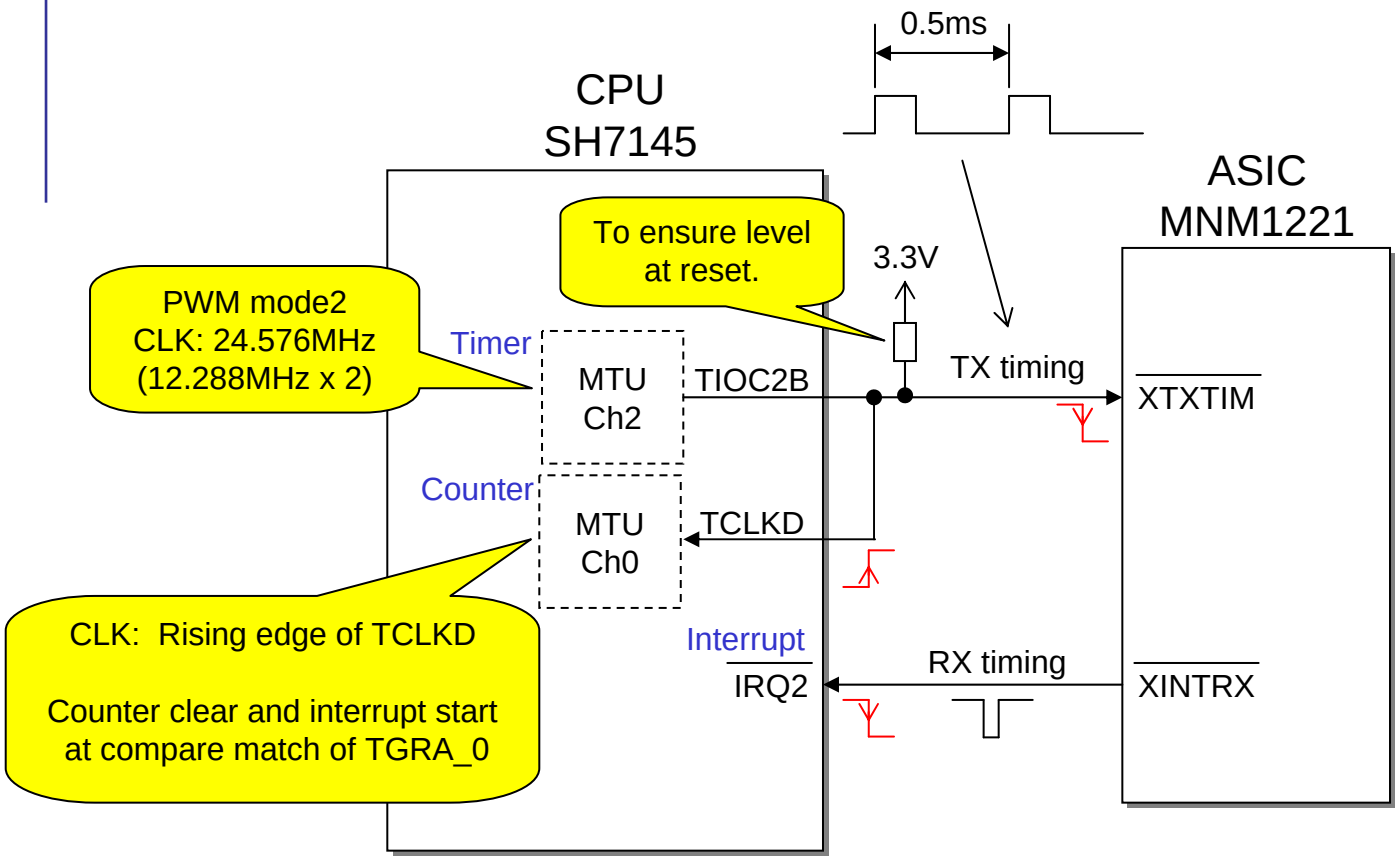
# Bus Connection




Note: Because each pin of SH7206 has “weak-keeper”, it is not necessary to install external pullups on bus-interface pins.

# Example for SH7145 (Renesas)

# Timing Circuit



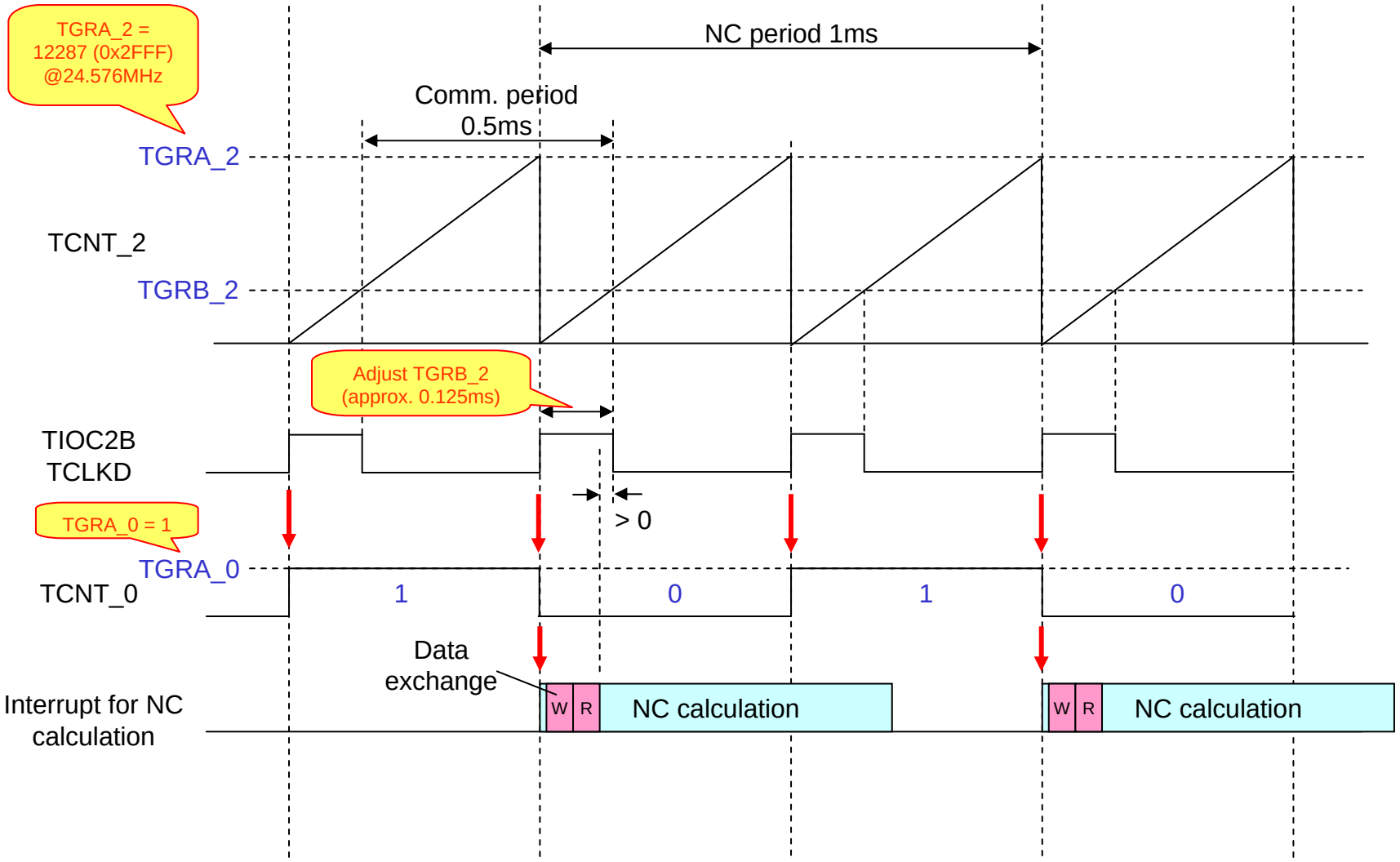

- MTU-Ch2 generates TX timing signal. For 0.5ms, **TGRA_2 = 12287(0x2FFF)@24.576MHz**
- MTU-Ch0 divides this signal, and generates the start signal for NC calculating interrupt. For 1ms, **TGRA_0 = 1**
- IRQ2 by XINTRX of MNM1221 causes RX interrupt.

# Multiplex Interrupt Setting

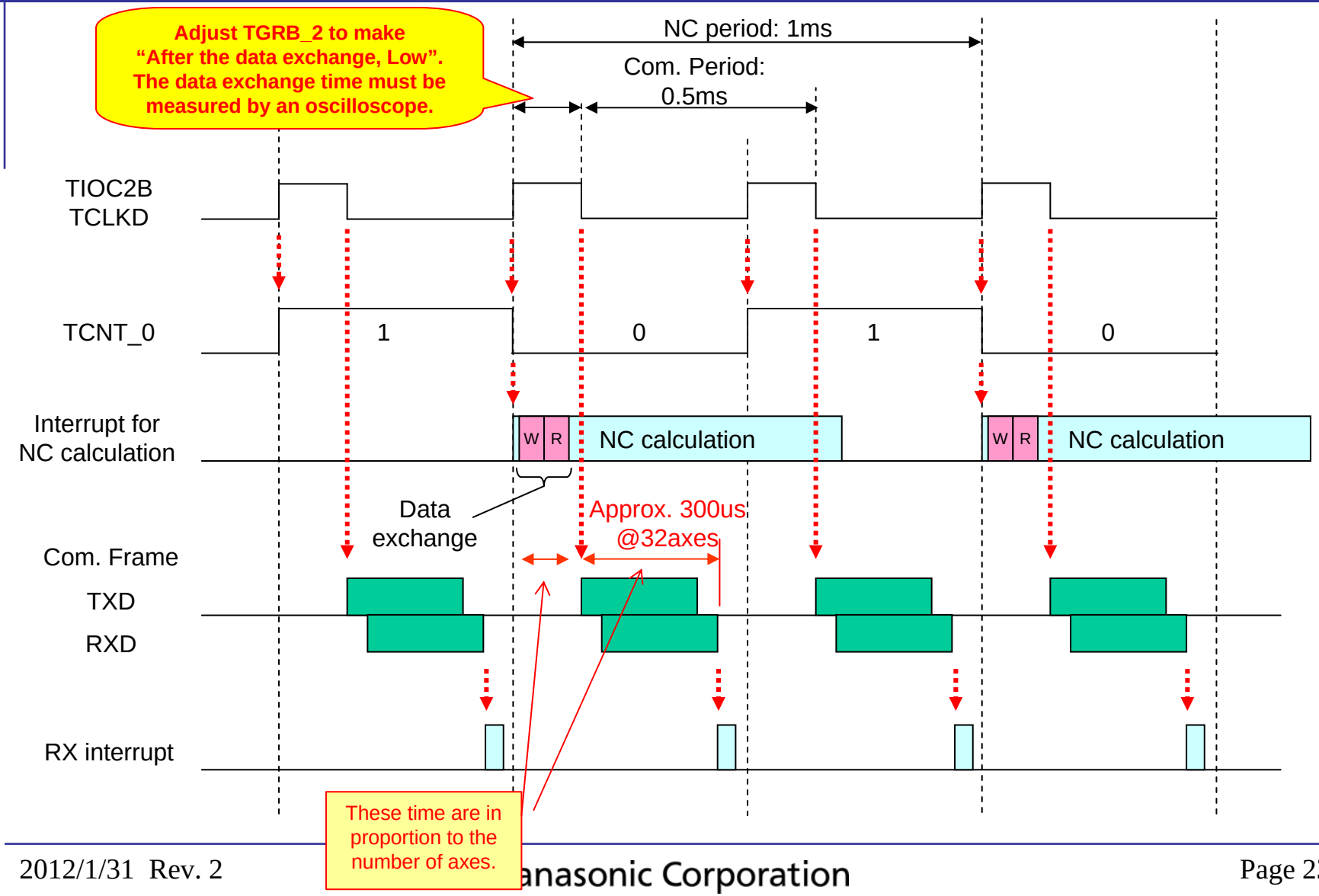
Panasonic
ideas for life

| Source | Trigger | Priority | Period | Operation |
|---|---|---|---|---|
| TGIA_0 | Compare match of MTU Ch0 | - | 1ms | - Communication data exchange<br>- NC Calculation |
| /IRQ2 | RX complete | Higher than TGIA_0 | 0.5ms | - Communication status check<br>- RX memory bank switch |

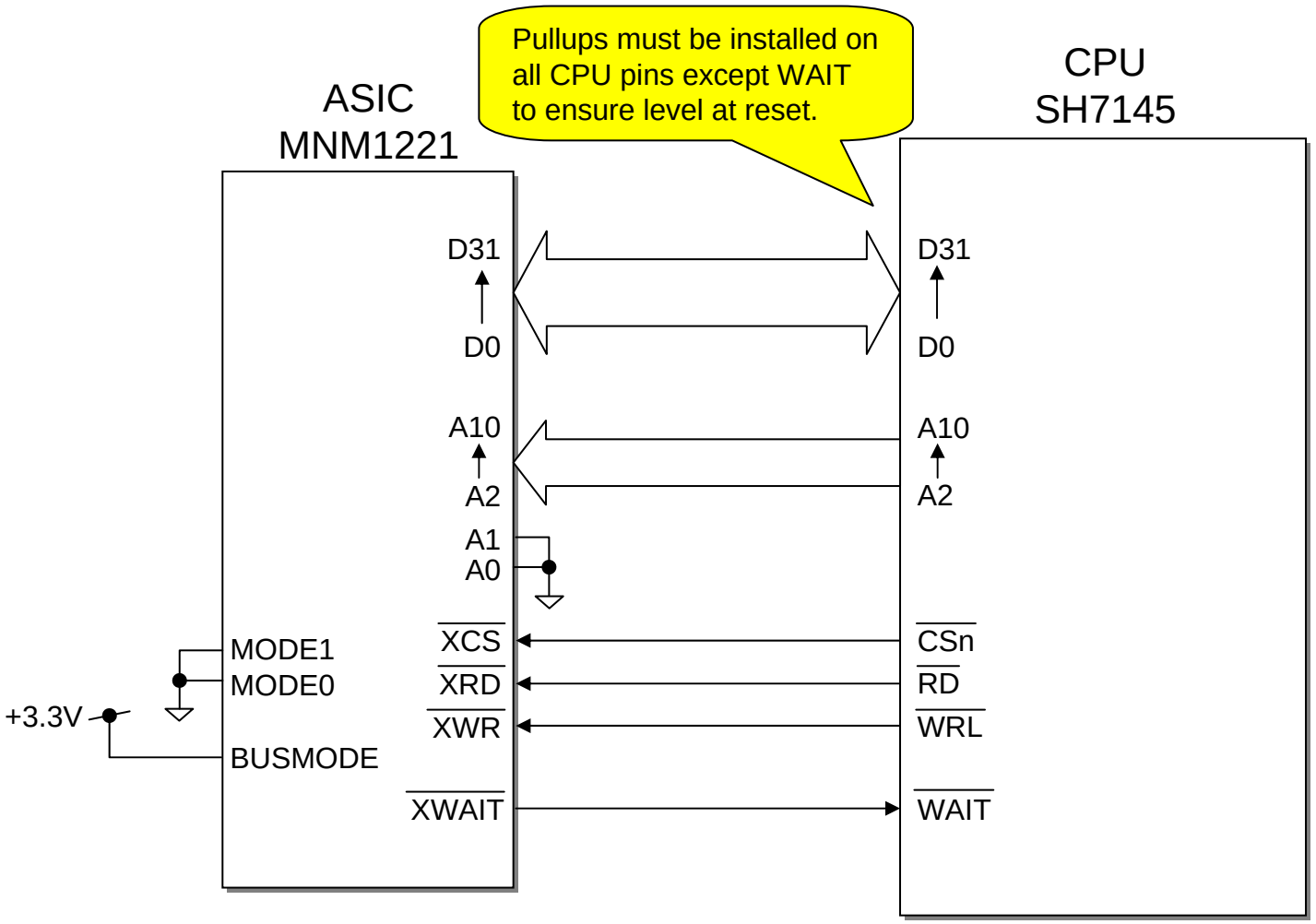
# Timing Chart 1



# Timing Chart 2

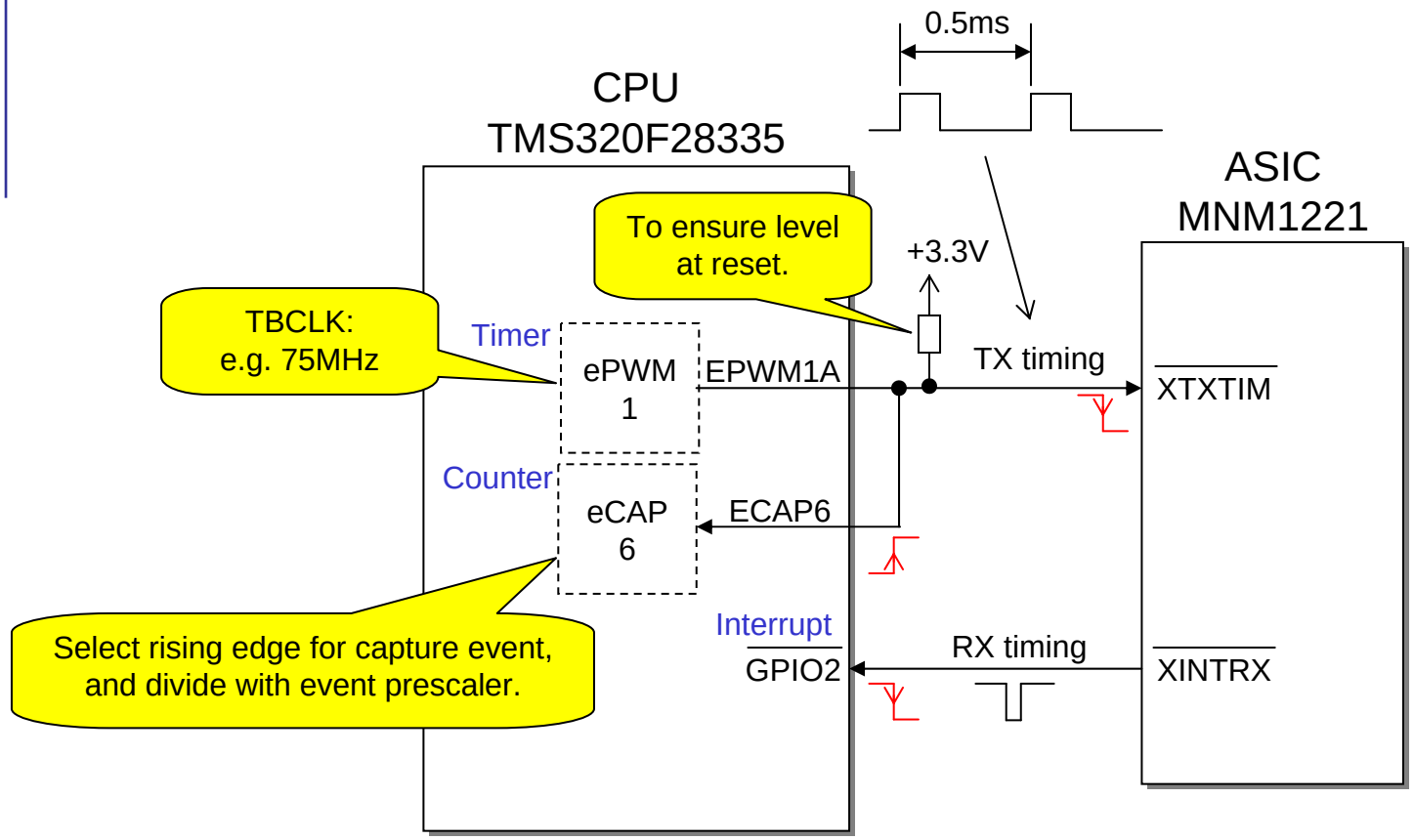


# Bus Connection



# Example for TMS320F28335 (TI)

# Timing Circuit




- ePWM1 generates TX timing signal. For 0.5ms, **TBPRD = 37499(0x927B)@TBCLK 75MHz**
- eCAP6 divides this signal, and generates the start signal for NC calculating interrupt. For 1ms, **PRESCALE = 1**
- External interrupt by XINTRX through GPIO2 causes RX interrupt.

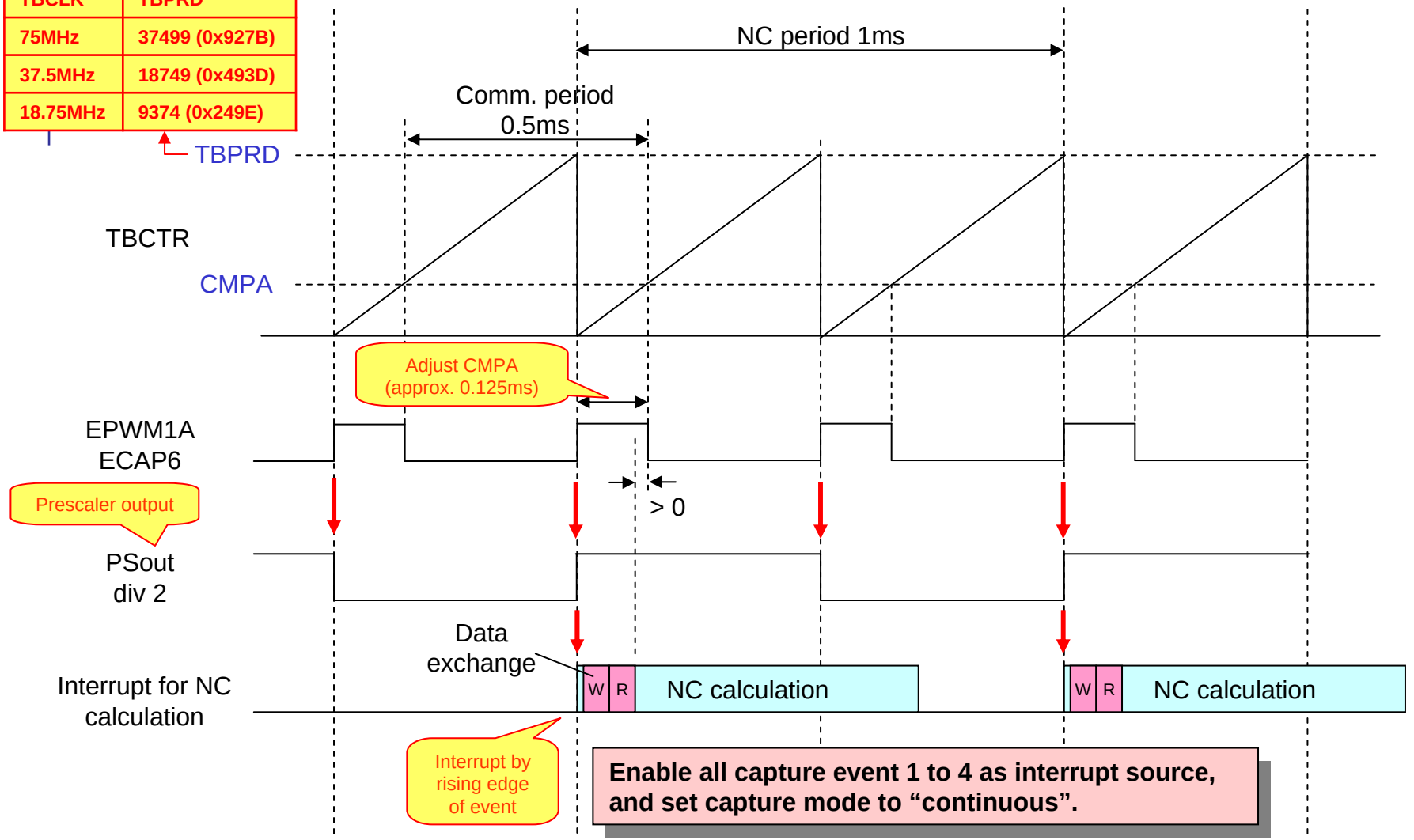
# Multiplex Interrupt Setting

Panasonic
ideas for life

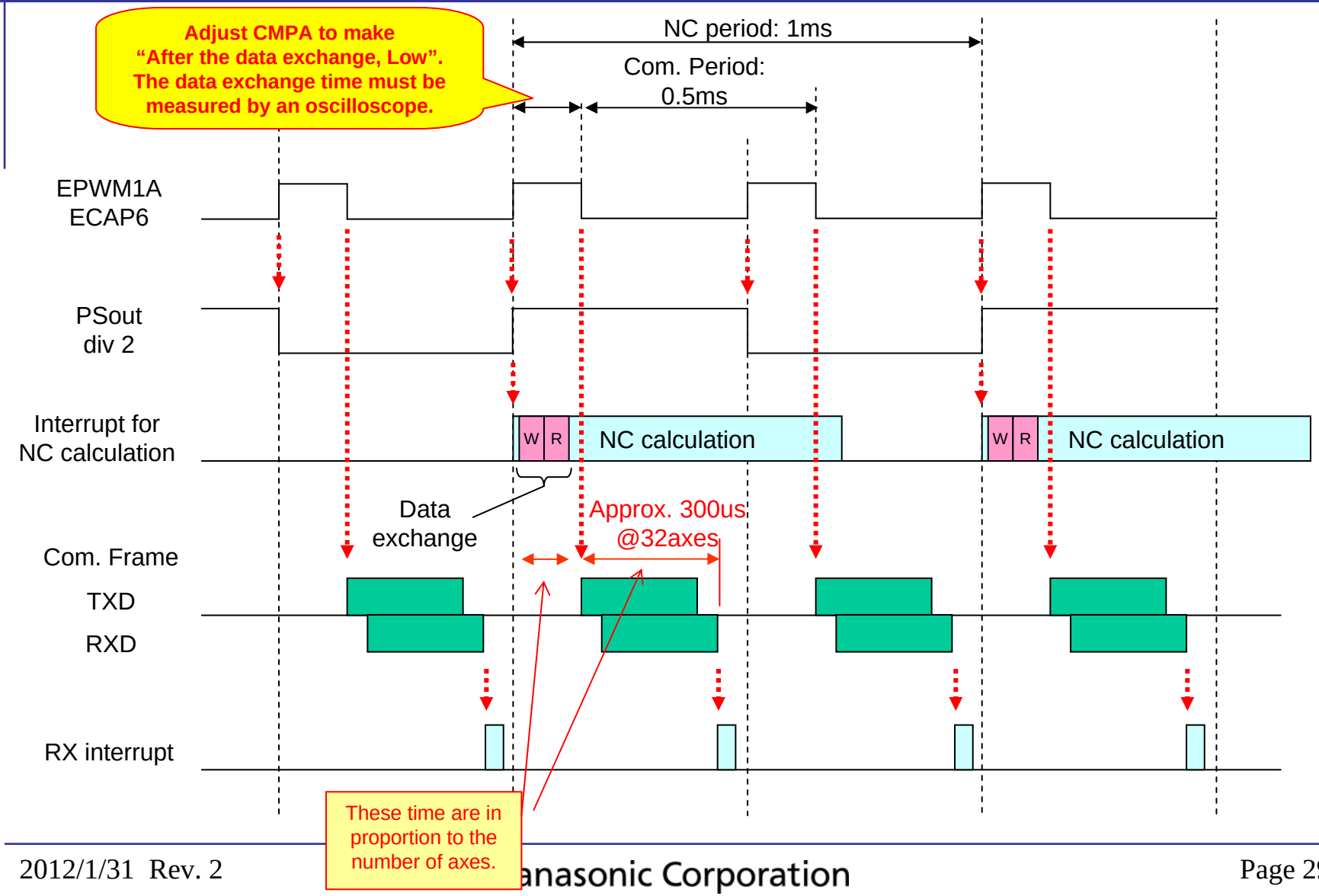
| Source | Trigger | Priority | Period | Operation |
|---|---|---|---|---|
| ECAP6 INT | Capture event of eCAP6 | - | 1ms | - Communication data exchange<br>- NC calculation |
| XINT through GPIO2 | RX complete | Higher than ECAP6 | 0.5ms | - Communication status check<br>- RX memory bank switch |

# Timing Chart 1

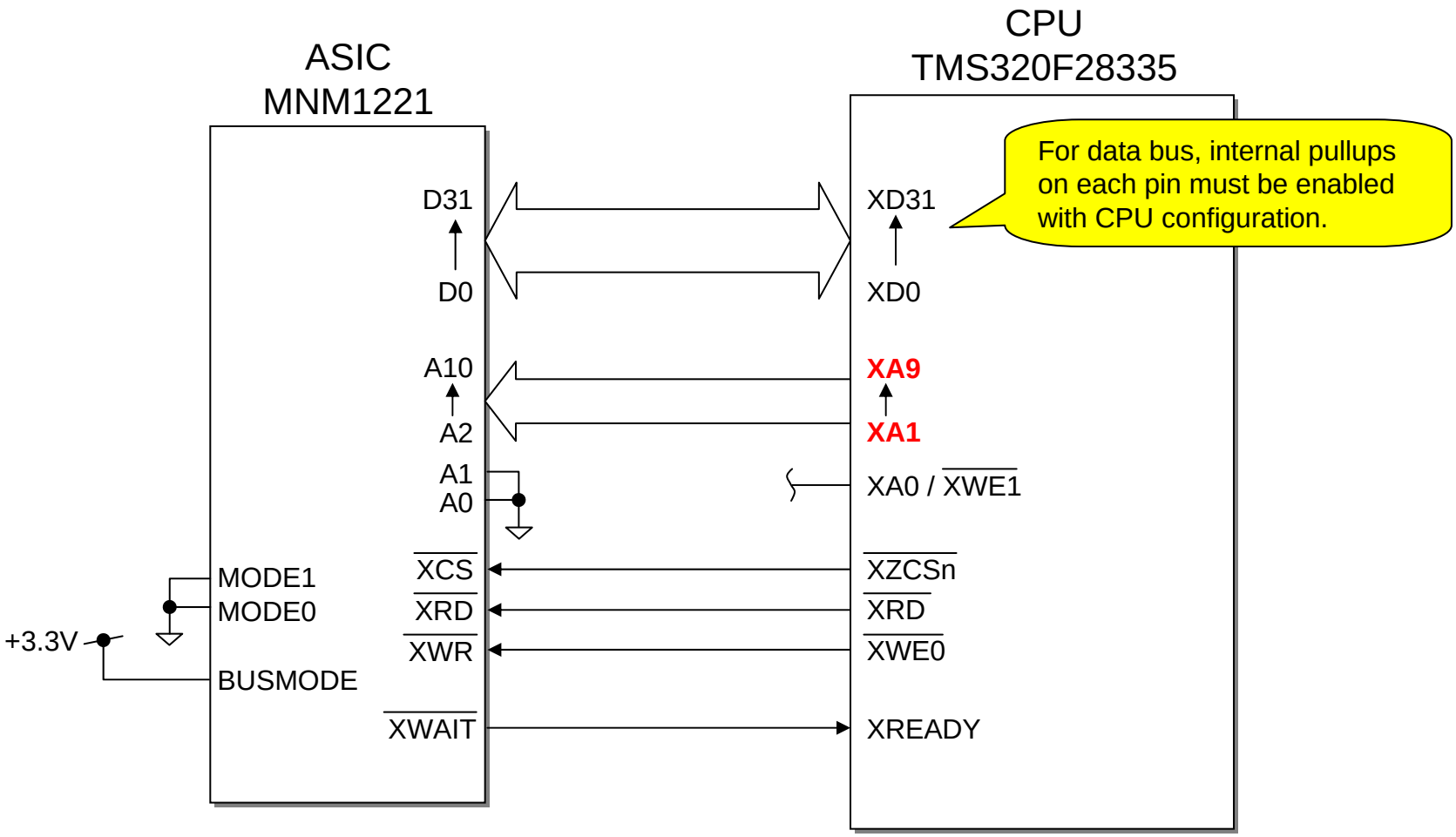
| TBCLK | TBPRD |
|---|---|
| 75MHz | 37499 (0x927B) |
| 37.5MHz | 18749 (0x493D) |
| 18.75MHz | 9374 (0x249E) |



# Timing Chart 2



# Bus Connection



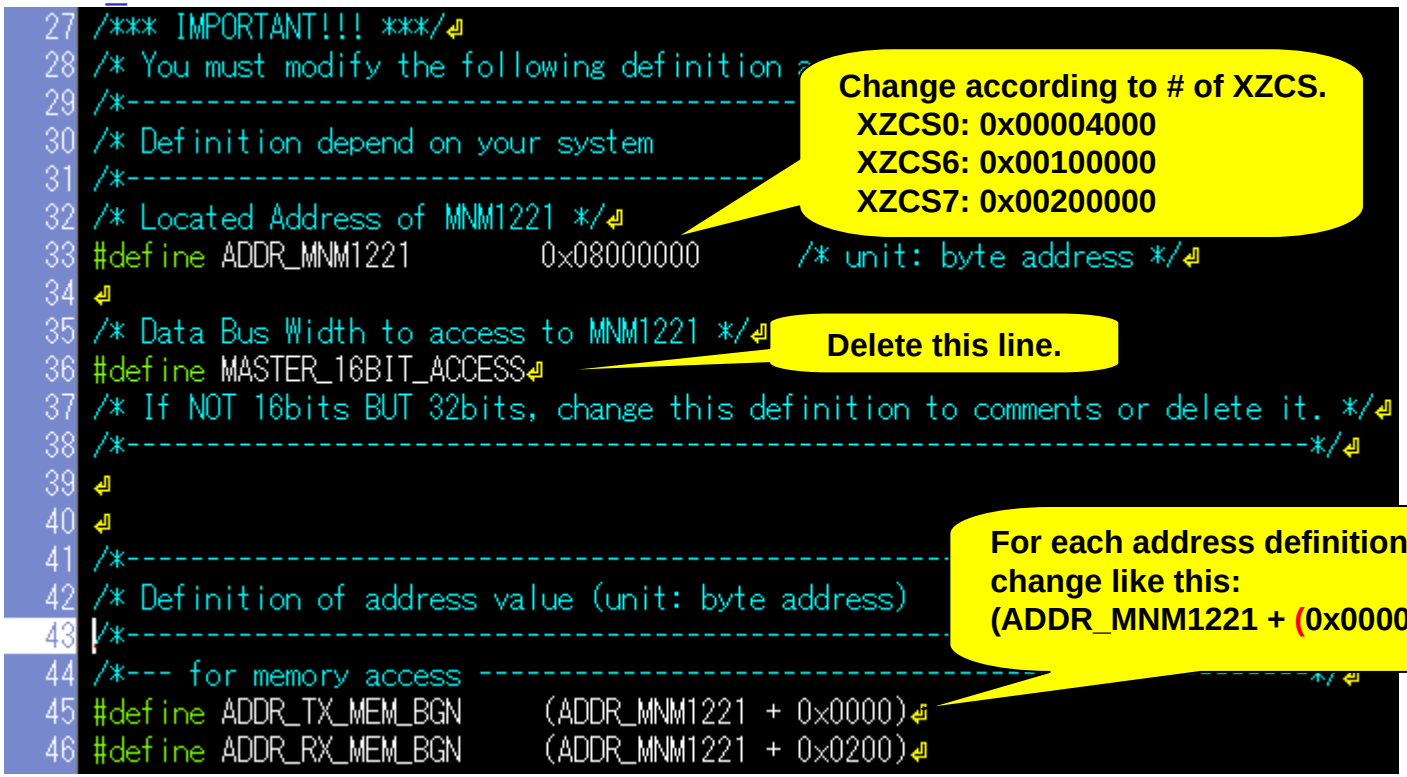

Note: TMS320F28335 has 16bit-unit address bus.

# Important Notes for Address

Although MNM1221 has 8bit-unit address bus, TMS320F28335 has 16bit-unit.
Therefore each address connection must be shifted 1,
such as A2(MNM1221) - XA1(TMS320F28335).

Also in the example code, all address definitions must be modified as follows:

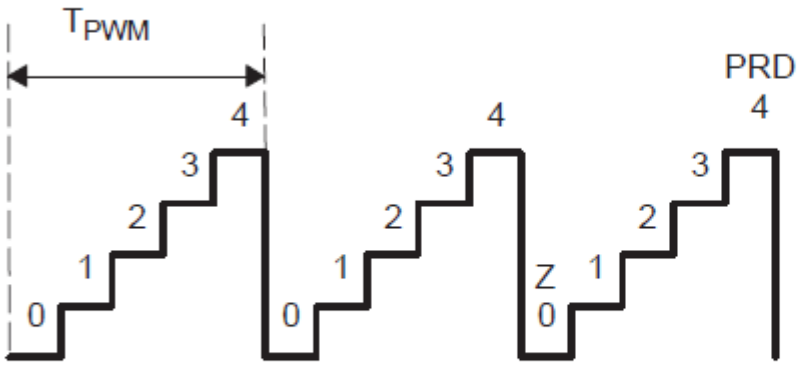
mnm1221_m.h



# Details of TMS320F28335 Configuration

# Period 0.5ms Setting for ePWM

Figure 2-3. Time-Base Frequency and Period




Setting for 0.5ms:

| TBCLK | TBPRD |
|---|---|
| 75MHz (150MHz / 2) | 37499 (0x927B) |
| 37.5MHz (150MHz / 4) | 18749 (0x493D) |
| 18.75MHz (150MHz / 8) | 9374 (0x249E) |

For Up Count and Down Count

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

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

TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)

# Interrupt by eCAP

Panasonic
ideas for life

4.5 Enhanced CAP Modules (eCAP1/2/3/4/5/6)

The 2833x/2823x device contains up to six enhanced capture (eCAP) modules. Figure 4-6 shows a functional block diagram of a module.

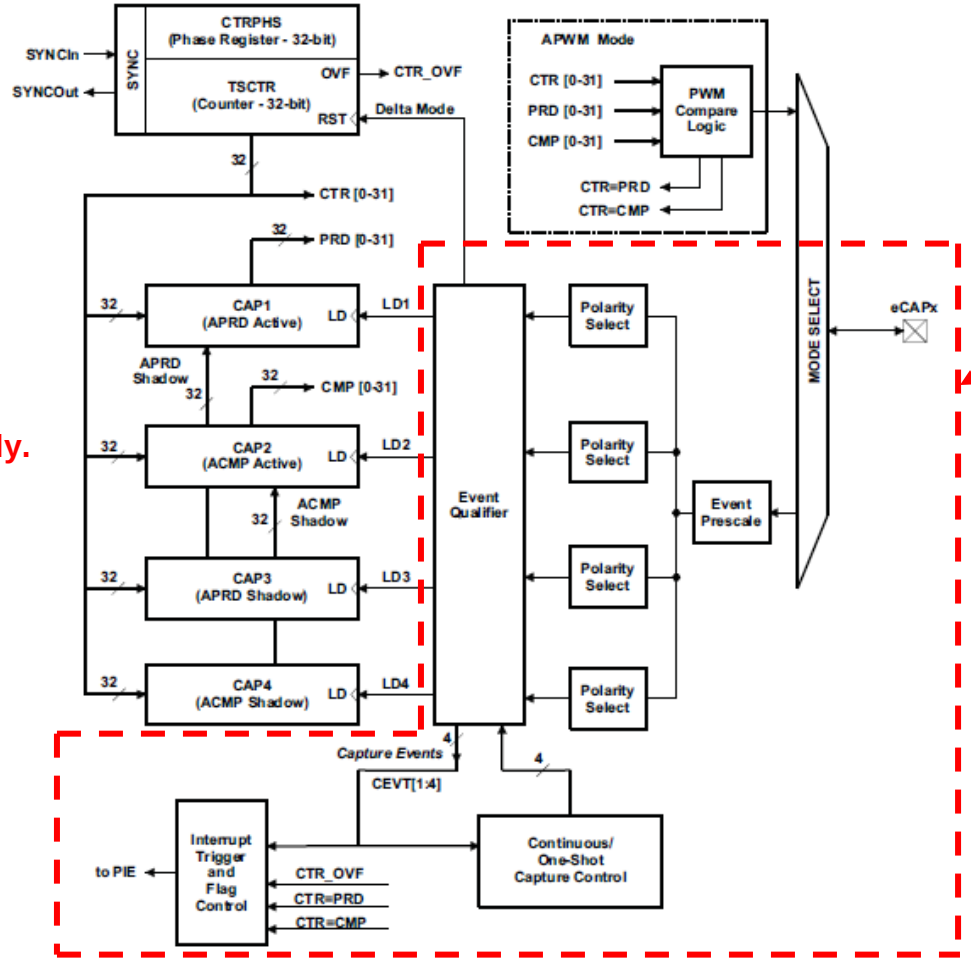


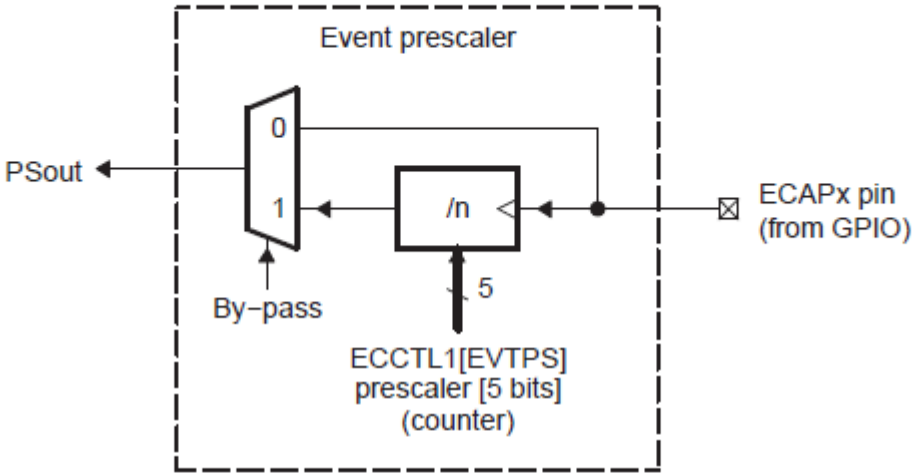

Figure 4-6. eCAP Functional Block Diagram

**Captured data is not used because of interrupt use only.**

**Using area for interrupt**

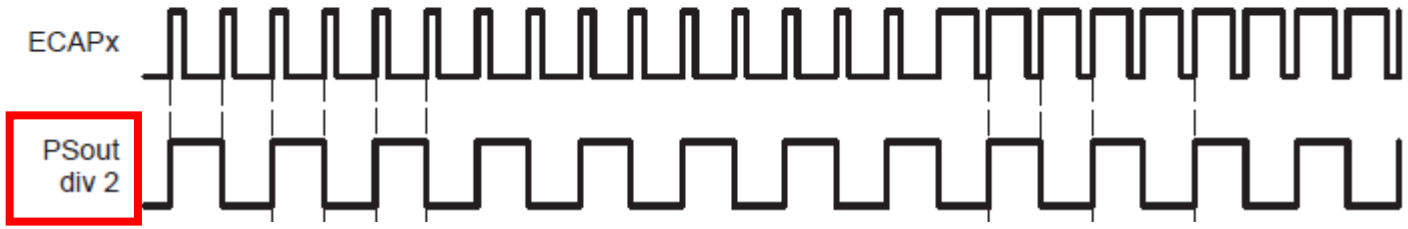
# Event Prescaler inside eCAP

Figure 4. Event Prescale Control




A When a prescale value of 1 is chosen (i.e. ECCTL1[13:9] = 0,0,0,0,0 ) the input capture signal by-passes the prescale logic completely.

Figure 5. Prescale Function Waveforms



# ECCTL1 Register

Panasonic
ideas for life

**Table 7. ECAP Control Register 1 (ECCTL1) Field Descriptions**

| Bit(s) | Field | Value | Description |
|---|---|---|---|
| 13:9 | PRESCALE | | Event Filter prescale select |
| | | 00000 | Divide by 1 (i.e,. no prescale, by-pass the prescaler) |
| | | 00001 | Divide by 2 |
| | | 00010 | Divide by 4 |
| | | 00011 | Divide by 6 |
| | | 00100 | Divide by 8 |
| | | 00101 | Divide by 10 |
| | | ... | |
| | | 11110 | Divide by 60 |
| | | 11111 | Divide by 62 |
| 6 | CAP4POL | | Capture Event 4 Polarity select |
| | | 0 | Capture Event 4 triggered on a rising edge (RE) |
| | | 1 | Capture Event 4 triggered on a falling edge (FE) |
| 4 | CAP3POL | | Capture Event 3 Polarity select |
| | | 0 | Capture Event 3 triggered on a rising edge (RE) |
| | | 1 | Capture Event 3 triggered on a falling edge (FE) |
| 2 | CAP2POL | | Capture Event 2 Polarity select |
| | | 0 | Capture Event 2 triggered on a rising edge (RE) |
| | | 1 | Capture Event 2 triggered on a falling edge (FE) |
| 0 | CAP1POL | | Capture Event 1 Polarity select |
| | | 0 | Capture Event 1 triggered on a rising edge (RE) |
| | | 1 | Capture Event 1 triggered on a falling edge (FE) |

# ECCTL2 Register

Panasonic
ideas for life

**Table 8. ECAP Control Register 2 (ECCTL2) Field Descriptions**

| Bit(s) | Field | | Description |
|---|---|---|---|
| 9 | CAP/APWM | | CAP/APWM operating mode select |
| | | 0 | ECAP module operates in capture mode. This mode forces the following configuration:<br>• Inhibits TSCTR resets via CTR = PRD event<br>• Inhibits shadow loads on CAP1 and 2 registers<br>• Permits user to enable CAP1-4 register load<br>• CAPx/APWMx pin operates as a capture input |
| | | 1 | ECAP module operates in APWM mode. This mode forces the following configuration:<br>• Resets TSCTR on CTR = PRD event (period boundary<br>• Permits shadow loading on CAP1 and 2 registers<br>• Disables loading of time-stamps into CAP1-4 registers<br>• CAPx/APWMx pin operates as a APWM output |
| 0 | CONT/ONESHT | | Continuous or one-shot mode control (applicable only in capture mode) |
| | | 0 | Operate in continuous mode |
| | | 1 | Operate in one-Shot mode |

# ECEINT Register

Panasonic
ideas for life

Table 9. ECAP Interrupt Enable Register (ECEINT) Field Descriptions

| Bits | Field | Value | Description |
|---|---|---|---|
| 15:8 | Reserved | | |
| 7 | CTR=CMP | | Counter Equal Compare Interrupt Enable |
| | | 0 | Disable Compare Equal as an Interrupt source |
| | | 1 | Enable Compare Equal as an Interrupt source |
| 6 | CTR=PRD | | Counter Equal Period Interrupt Enable |
| | | 0 | Disable Period Equal as an Interrupt source |
| | | 1 | Enable Period Equal as an Interrupt source |
| 5 | CTROVF | | Counter Overflow Interrupt Enable |
| | | 0 | Disabled counter Overflow as an Interrupt source |
| | | 1 | Enable counter Overflow as an Interrupt source |
| 4 | CEVT4 | | Capture Event 4 Interrupt Enable |
| | | 0 | Disable Capture Event 1 as an Interrupt source |
| | | 1 | Capture Event 4 Interrupt Enable |
| 3 | CEVT3 | | Capture Event 3 Interrupt Enable |
| | | 0 | Disable Capture Event 1 as an Interrupt source |
| | | 1 | Enable Capture Event 1 as an Interrupt source |
| 2 | CEVT2 | | Capture Event 2 Interrupt Enable |
| | | 0 | Disable Capture Event 1 as an Interrupt source |
| | | 1 | Enable Capture Event 1 as an Interrupt source |
| 1 | CEVT1 | | Capture Event 1 Interrupt Enable |
| | | 0 | Disable Capture Event 1 as an Interrupt source |
| | | 1 | Enable Capture Event 1 as an Interrupt source |
| 0 | Reserved | | |

Disable

Enable

# ECCLR Register

Panasonic
ideas for life

At the beginning of interrupt routine, interrupt flags must be cleared to prepare the next interrupt.

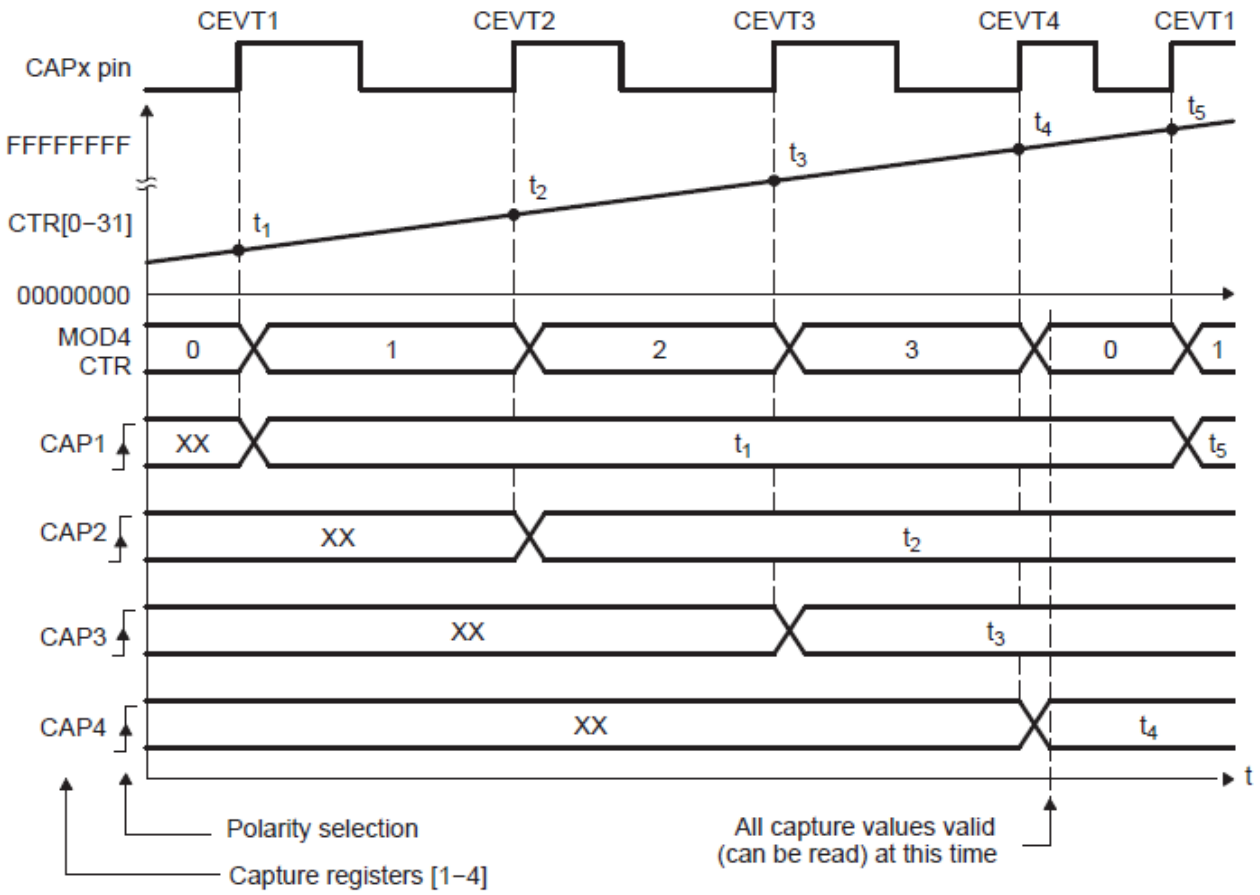
**Table 11. ECAP Interrupt Clear Register (ECCLR) Field Descriptions**

| Bits | Field | | Description |
|---|---|---|---|
| 15:8 | Reserved | | |
| 7 | CTR=CMP | | Counter Equal Compare Status Flag |
| | | 0 | Writing a 0 has no effect. Always reads back a 0 |
| | | 1 | Writing a 1 clears the CTR=CMP flag condition |
| 6 | CTR=PRD | | Counter Equal Period Status Flag |
| | | 0 | Writing a 0 has no effect. Always reads back a 0 |
| | | 1 | Writing a 1 clears the CTR=PRD flag condition |
| 5 | CTROVF | | Counter Overflow Status Flag |
| | | 0 | Writing a 0 has no effect. Always reads back a 0 |
| | | 1 | Writing a 1 clears the CTROVF flag condition |
| 4 | CEVT4 | | Capture Event 4 Status Flag |
| | | 0 | Writing a 0 has no effect. Always reads back a 0. |
| | | 1 | Writing a 1 clears the CEVT3 flag condition. |
| 3 | CEVT3 | | Capture Event 3 Status Flag |
| | | 0 | Writing a 0 has no effect. Always reads back a 0. |
| | | 1 | Writing a 1 clears the CEVT3 flag condition. |
| 2 | CEVT2 | | Capture Event 2 Status Flag |
| | | 0 | Writing a 0 has no effect. Always reads back a 0. |
| | | 1 | Writing a 1 clears the CEVT2 flag condition. |
| 1 | CEVT1 | | Capture Event 1 Status Flag |
| | | 0 | Writing a 0 has no effect. Always reads back a 0. |
| | | 1 | Writing a 1 clears the CEVT1 flag condition. |
| 0 | INT | | Global Interrupt Clear Flag |
| | | 0 | Writing a 0 has no effect. Always reads back a 0. |
| | | 1 | Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1. |

# Capture Interrupt Sequence

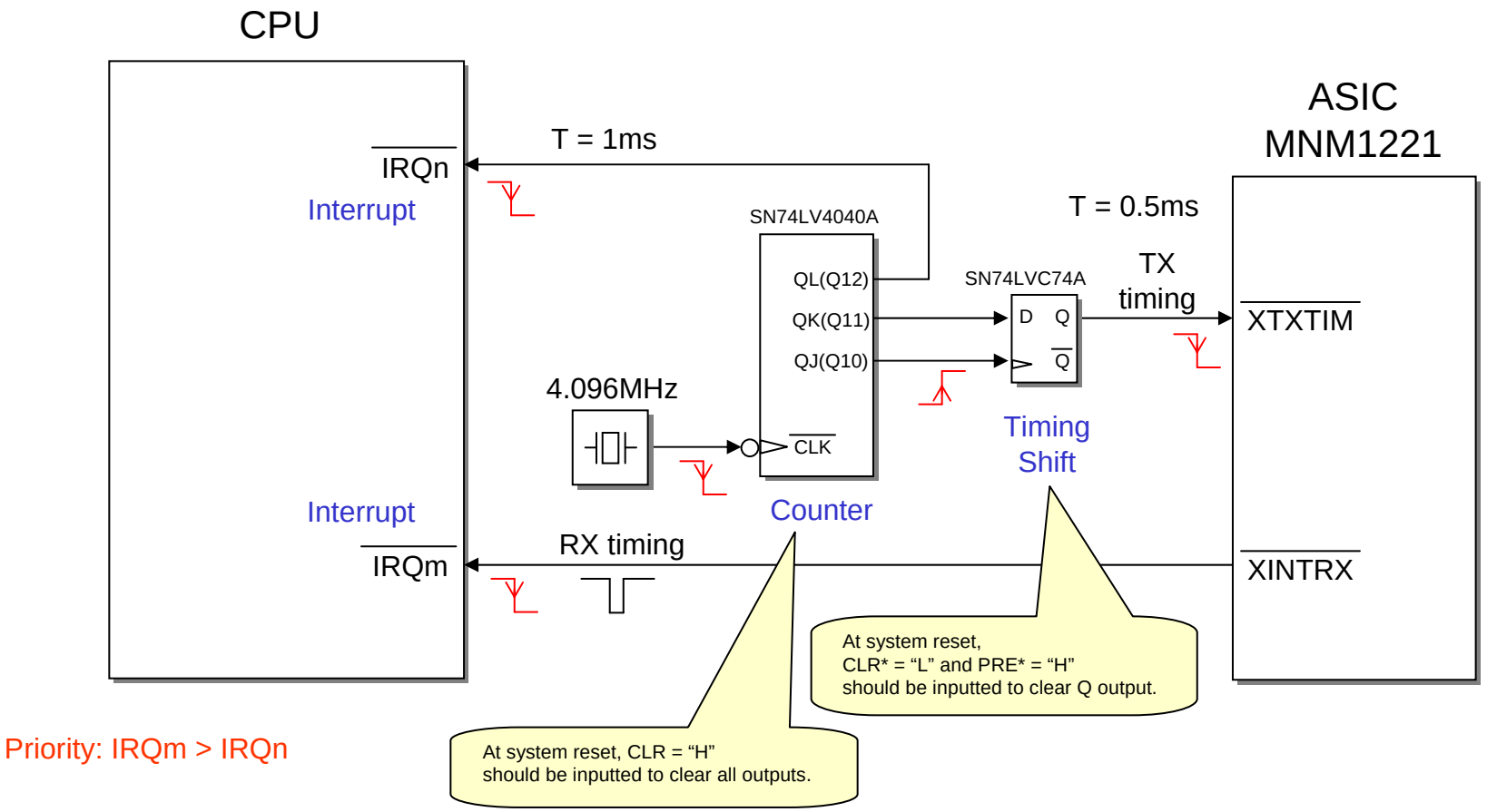
Interrupt occurring: CEVT1 → CEVT2 → CEVT3 → CEVT4 → CEVT1 → …

**Figure 22. Capture Sequence for Absolute Time-stamp and Rising Edge Detect**



# Example for
# CPU without Internal Timer

# Timing Circuit

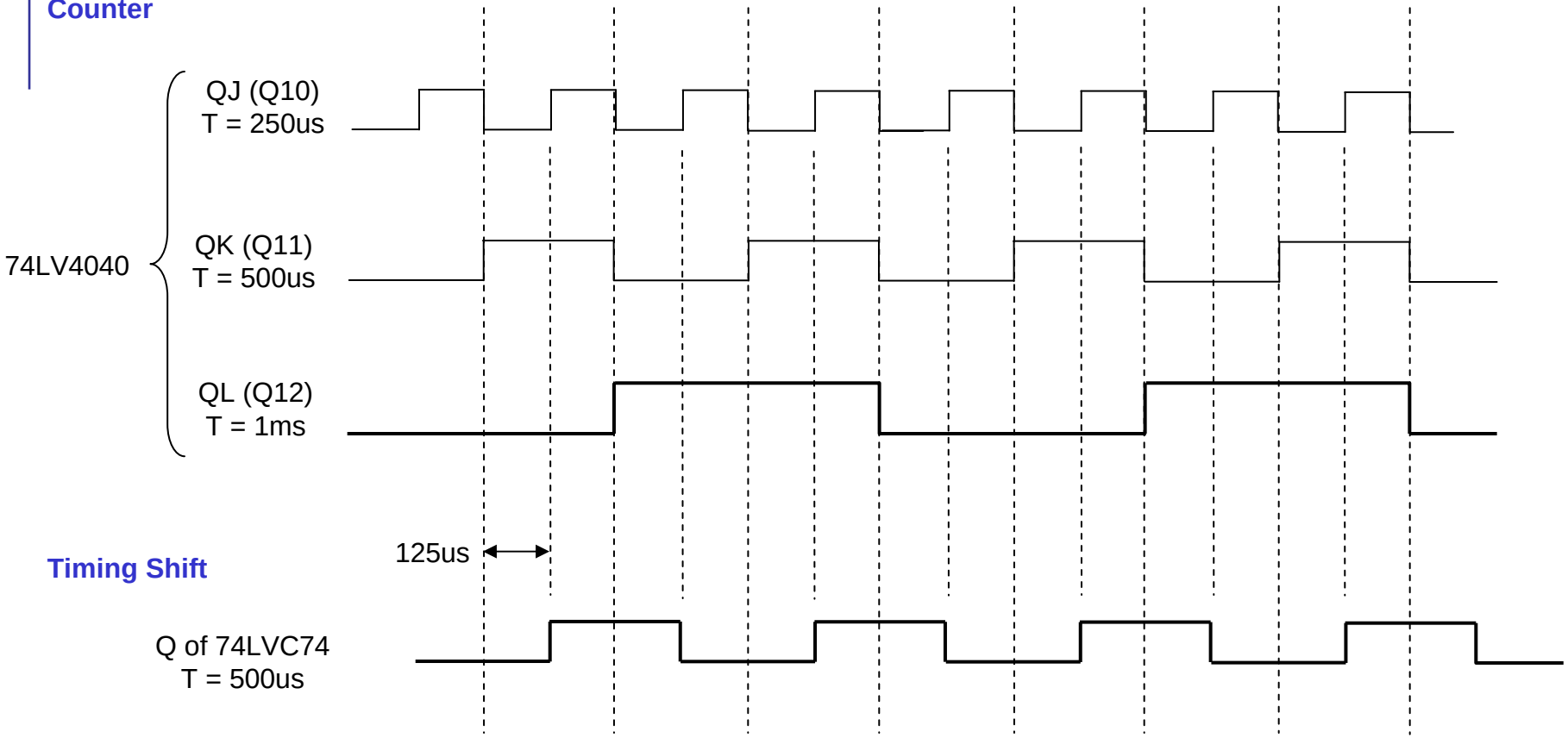



Priority: IRQm > IRQn

# Timing Chart 1

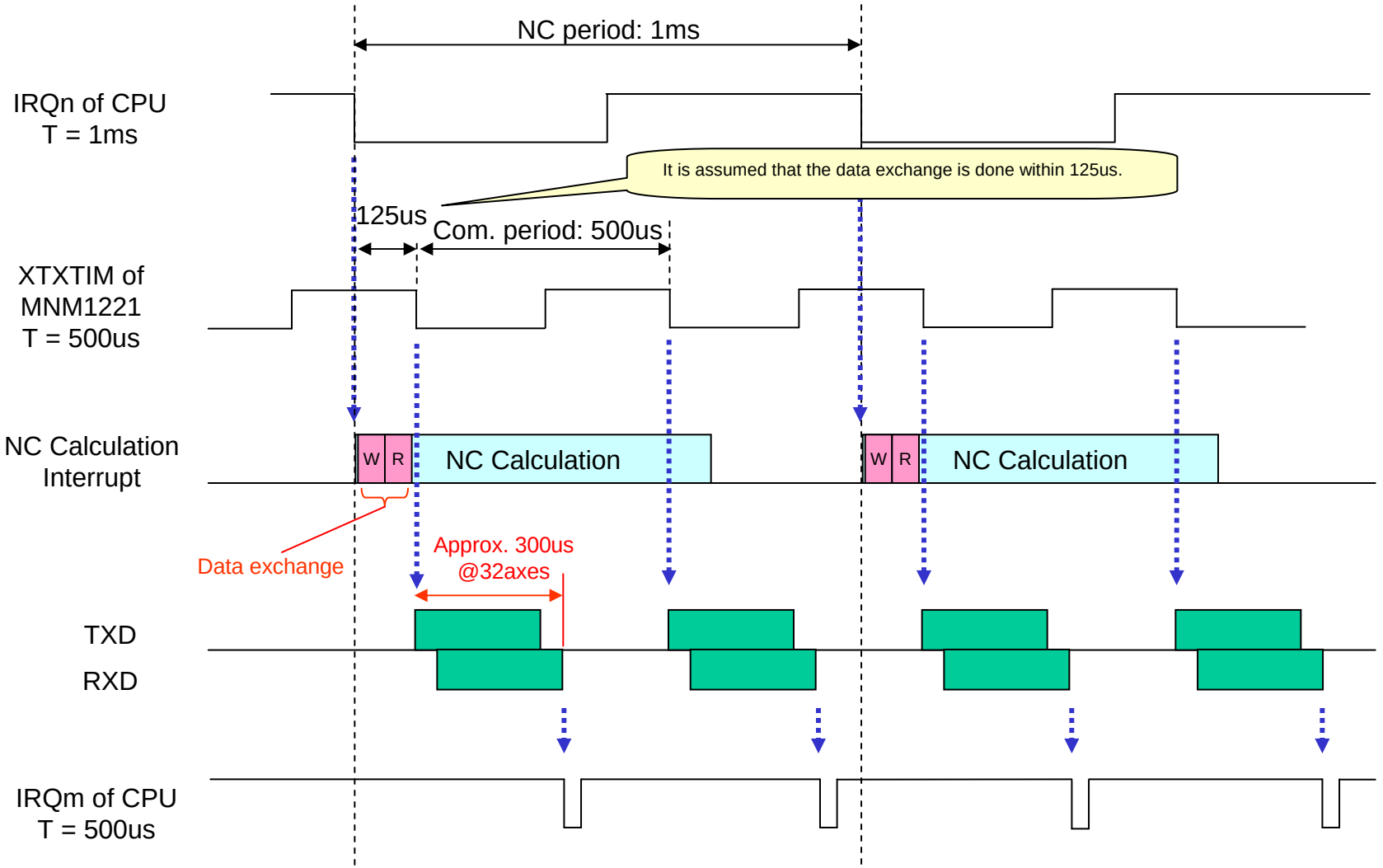
Panasonic
ideas for life

**Counter**




**Timing Shift**

# Timing Chart 2



# Location of Example Codes

# Location of Example Codes

Panasonic ideas for life

These functions have to be made by yourself.

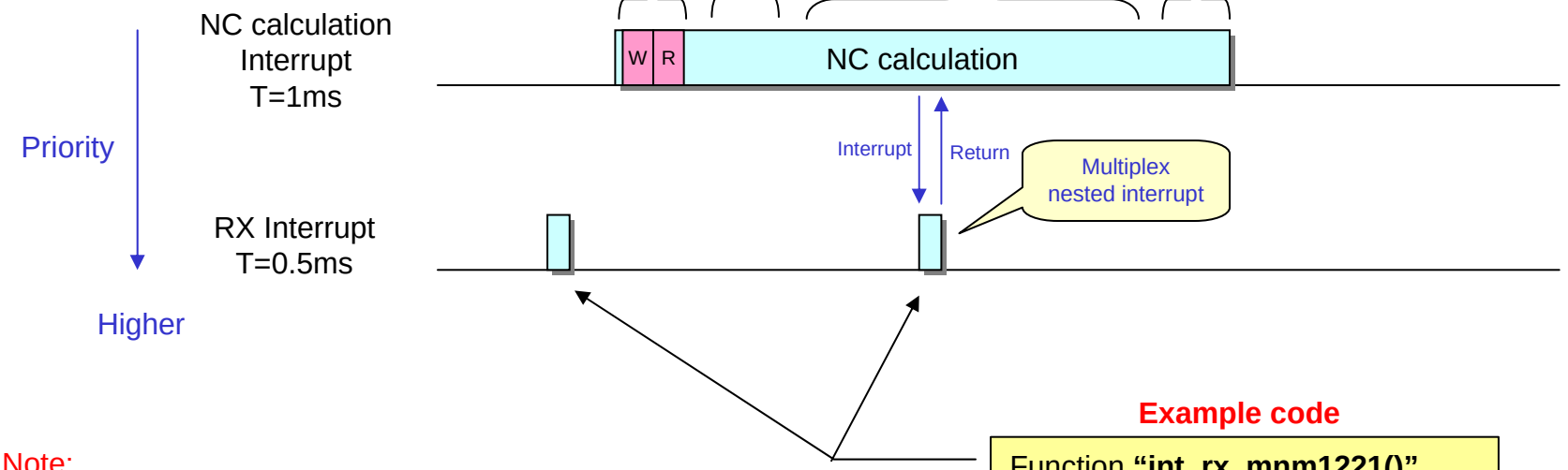
**Example code**

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[]”**

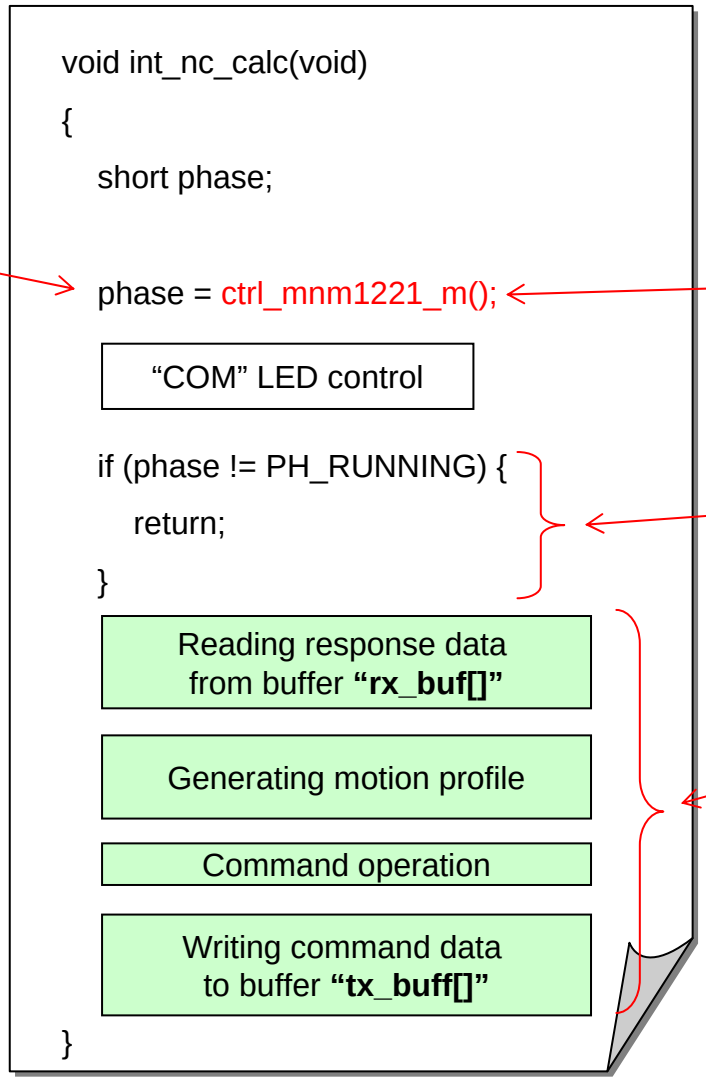



**Example code**

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



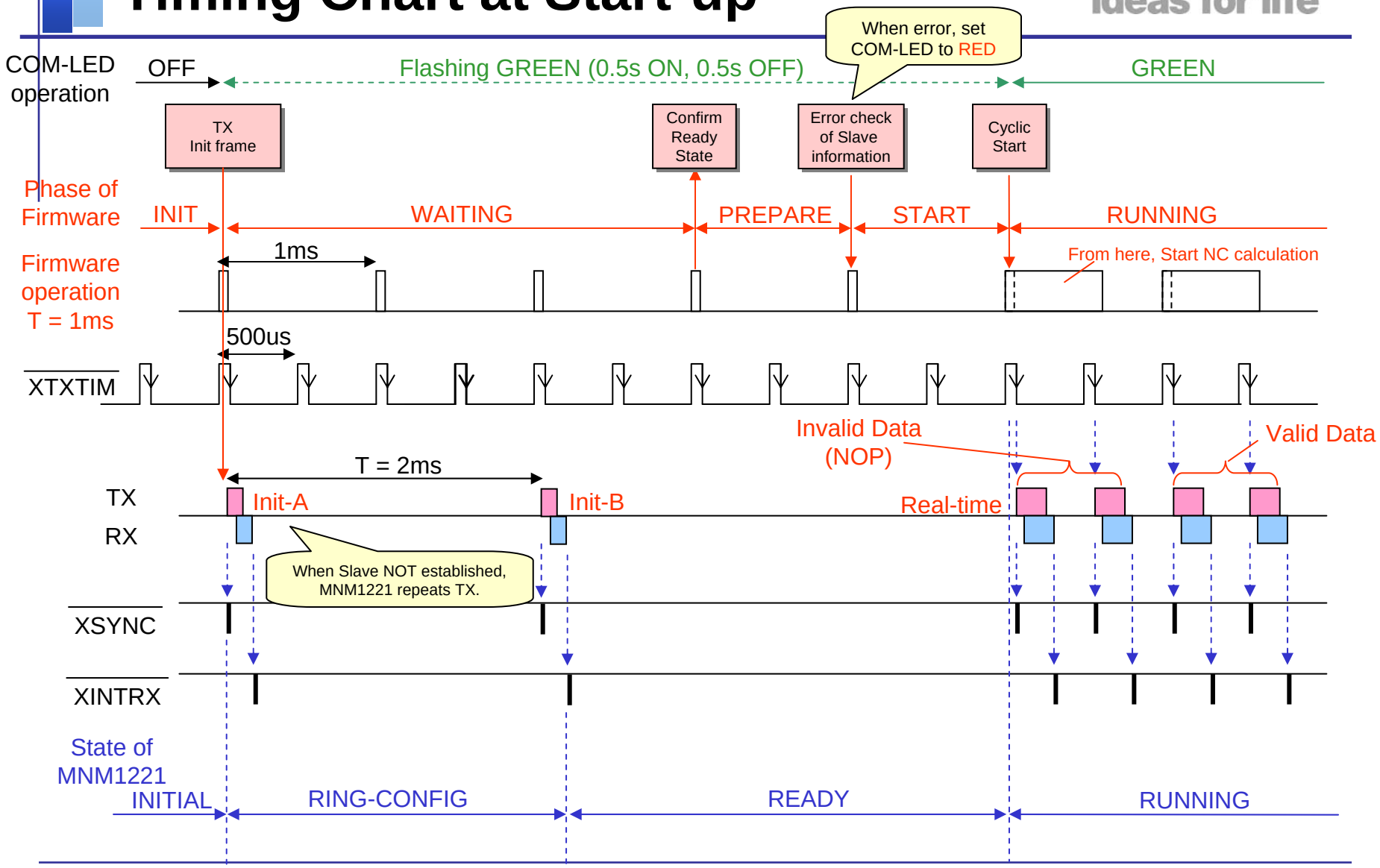

The next phase

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

# Data Structure (32bit)

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

| | | Bit 31 – 24 | Bit 23 – 16 | Bit 15 – 8 | Bit 7 – 0 |
|---|---|---|---|---|---|
| One element of tx_buf[] or rx_buf[] | 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 16bytes.

# Data Structure (16bit)

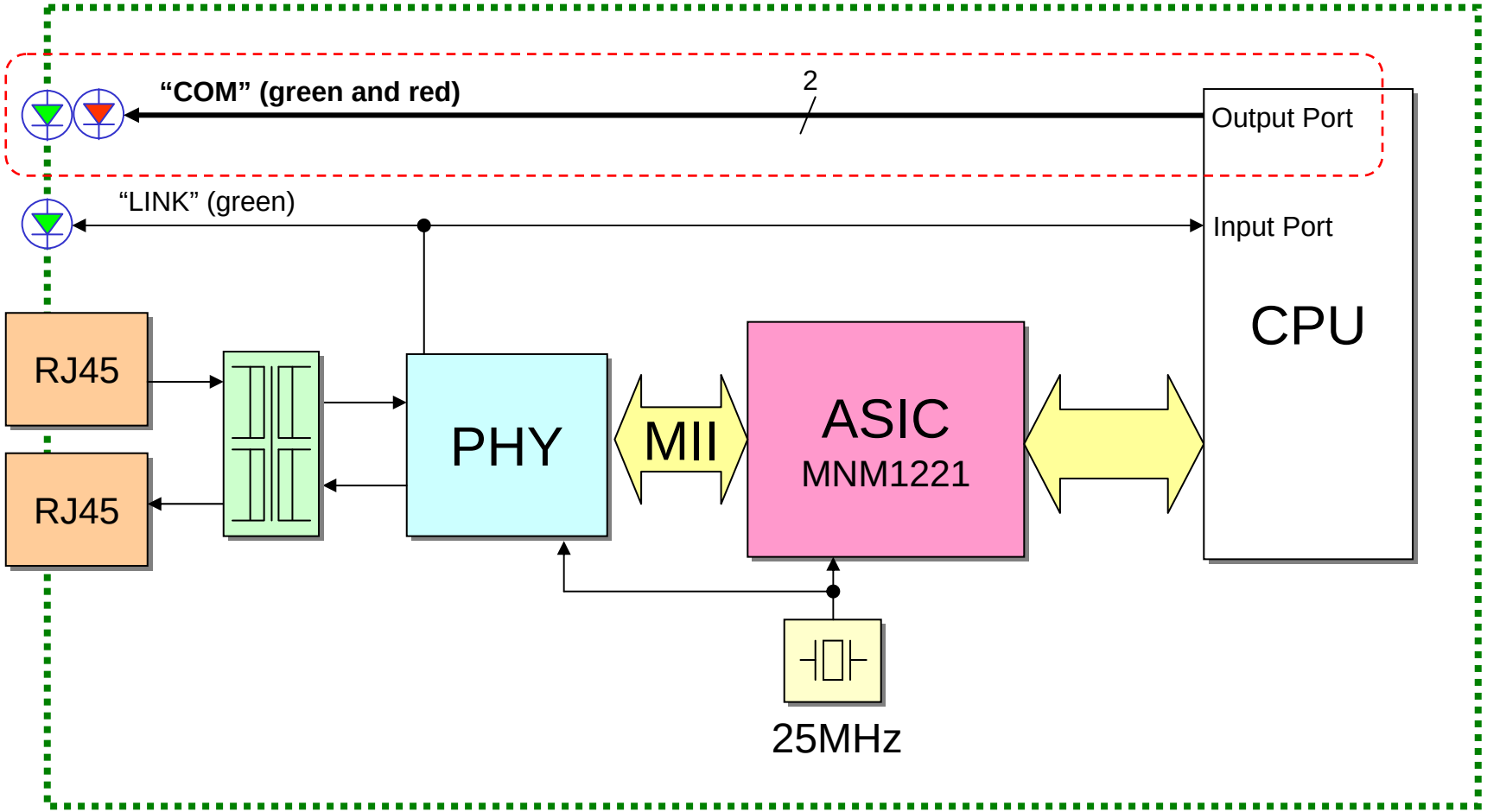
The following shows the structure of one element of tx_buf[] or rx_buf[] array **when 16bit bus access**.

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 16bytes.

# 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 (0.5s ON, 0.5s OFF) |
| PH_PREPARE | |
| PH_START | |
| PH_RUNNING | Solid Green |

Error detected

| Contents of error | “COM” LED operation |
|---|---|
| Timeout in RUNNING state | Flashing Red (0.5s ON, 0.5s 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

Panasonic ideas for life

## Command (TX)

Normally, set 0x20 (Position).

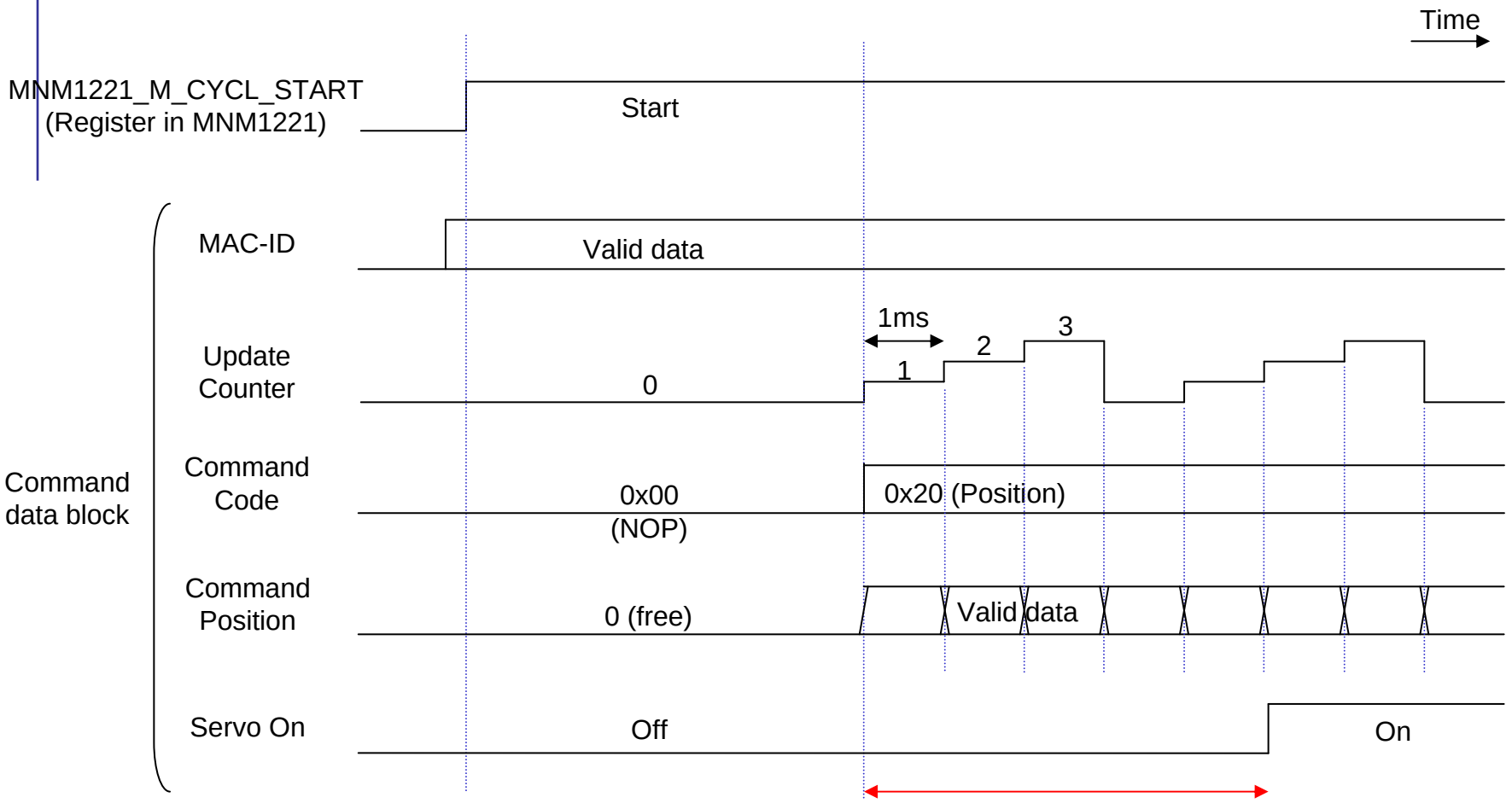
| | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|
| byte0 | 0 (CMD) | Update Counter | | MAC-ID | | | | |
| byte1 | 0 | Command Code | | | | | | |
| byte2 | Servo On | 0 | 0 | Gain SW | TL SW | HM Ctrl | 0 | 0 |
| byte3 | Hard Stop | SMT Stop | Pause | 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 |

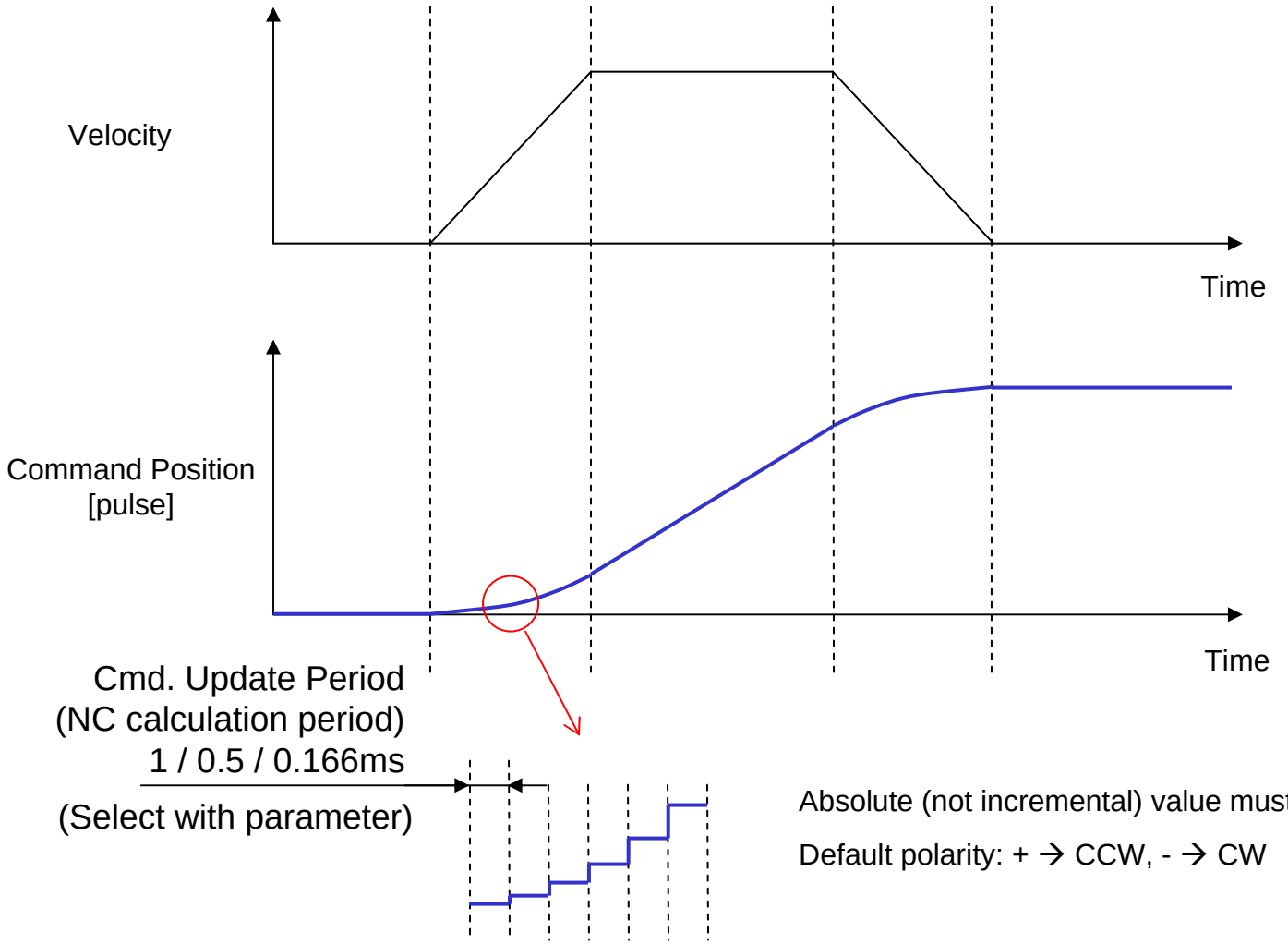
Note: In cyclic position I/F, at least red portions must be supported.

# Command at Start-up




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

Panasonic ideas for life

| | 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 | | | | | | | | |

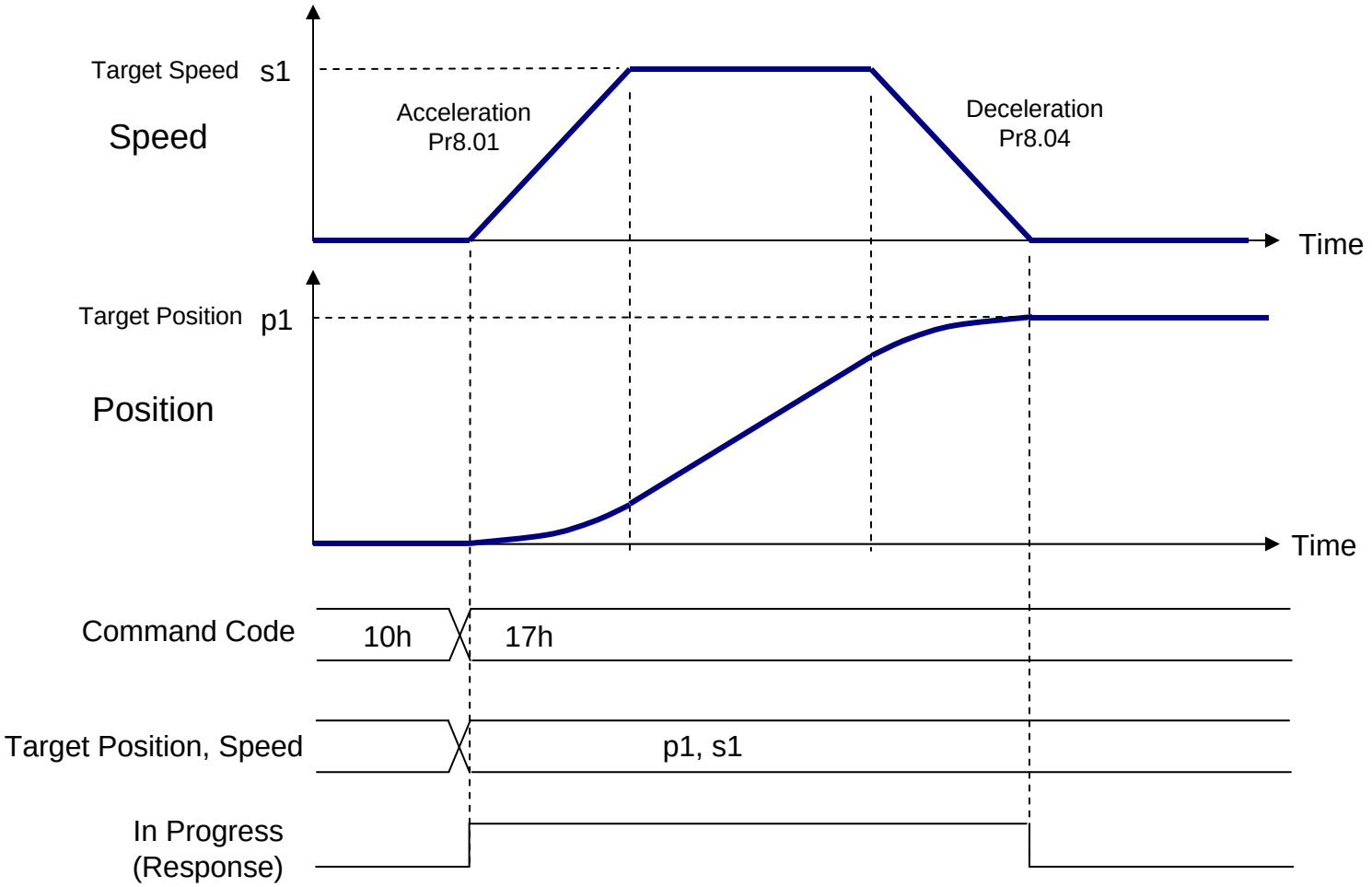
**Mode, Inc/Abs**

# 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 | | | | | | | | |

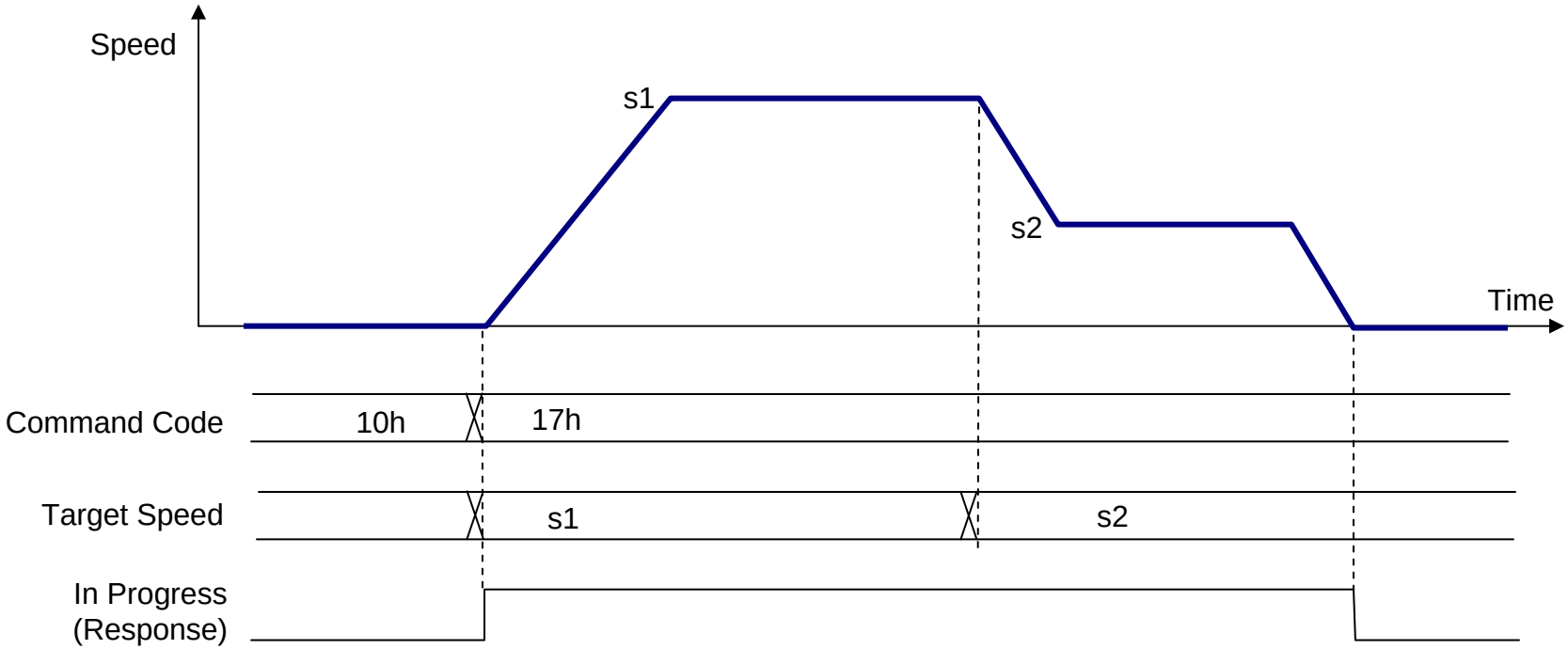
# Start

**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

Panasonic
ideas for life

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

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

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

# Definition of Variables

Panasonic
ideas for life

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

Panasonic
ideas for life

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;
    }
}
```

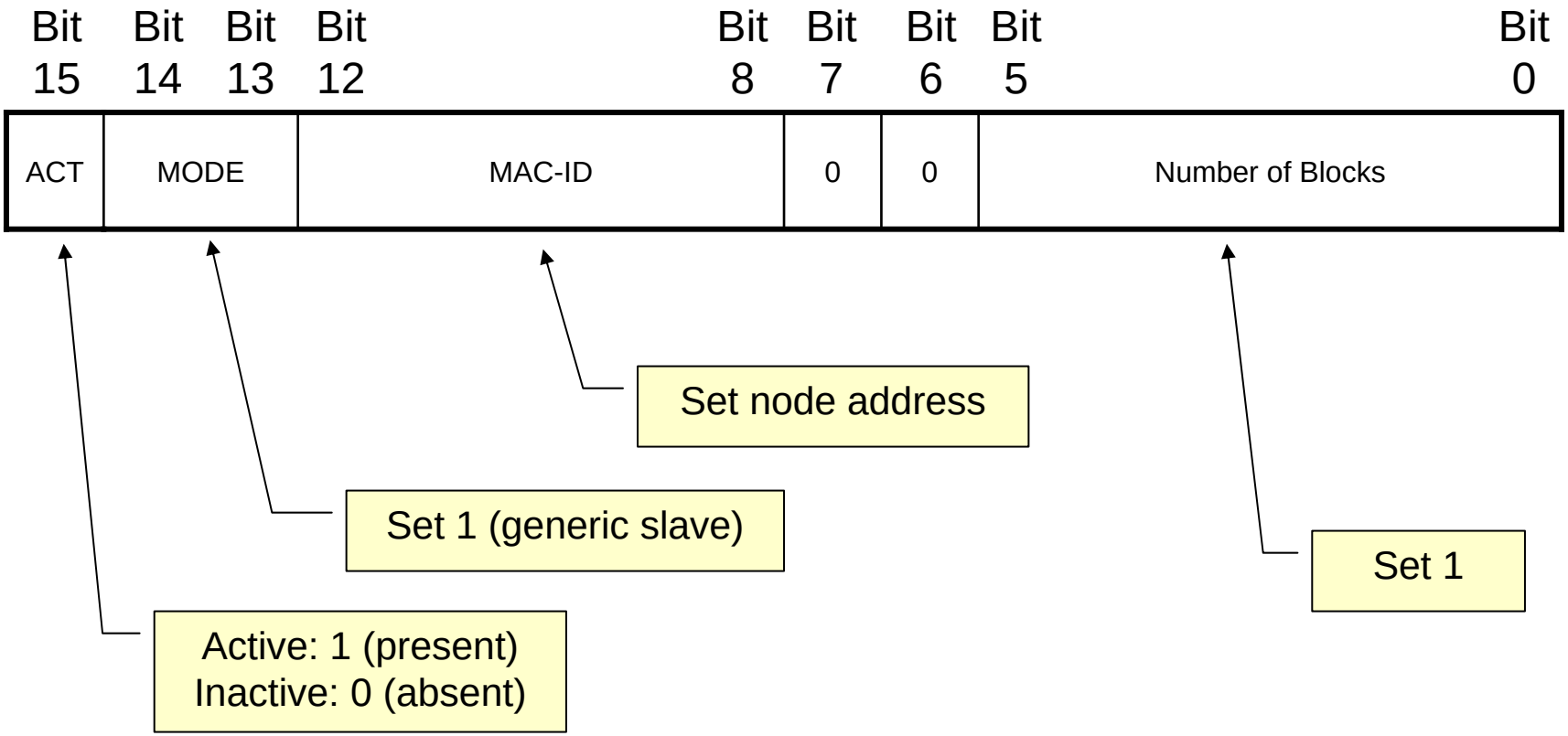
When test, modify these values according to actual your system.

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 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.

```
/*************************************************************************/
/* 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;
```

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

# In Running State

Panasonic
ideas for life

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

After test, you have to remove.

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.

# Chapter 2

## Internal-Timer Using 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 A5N Drive

Panasonic ideas for life

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

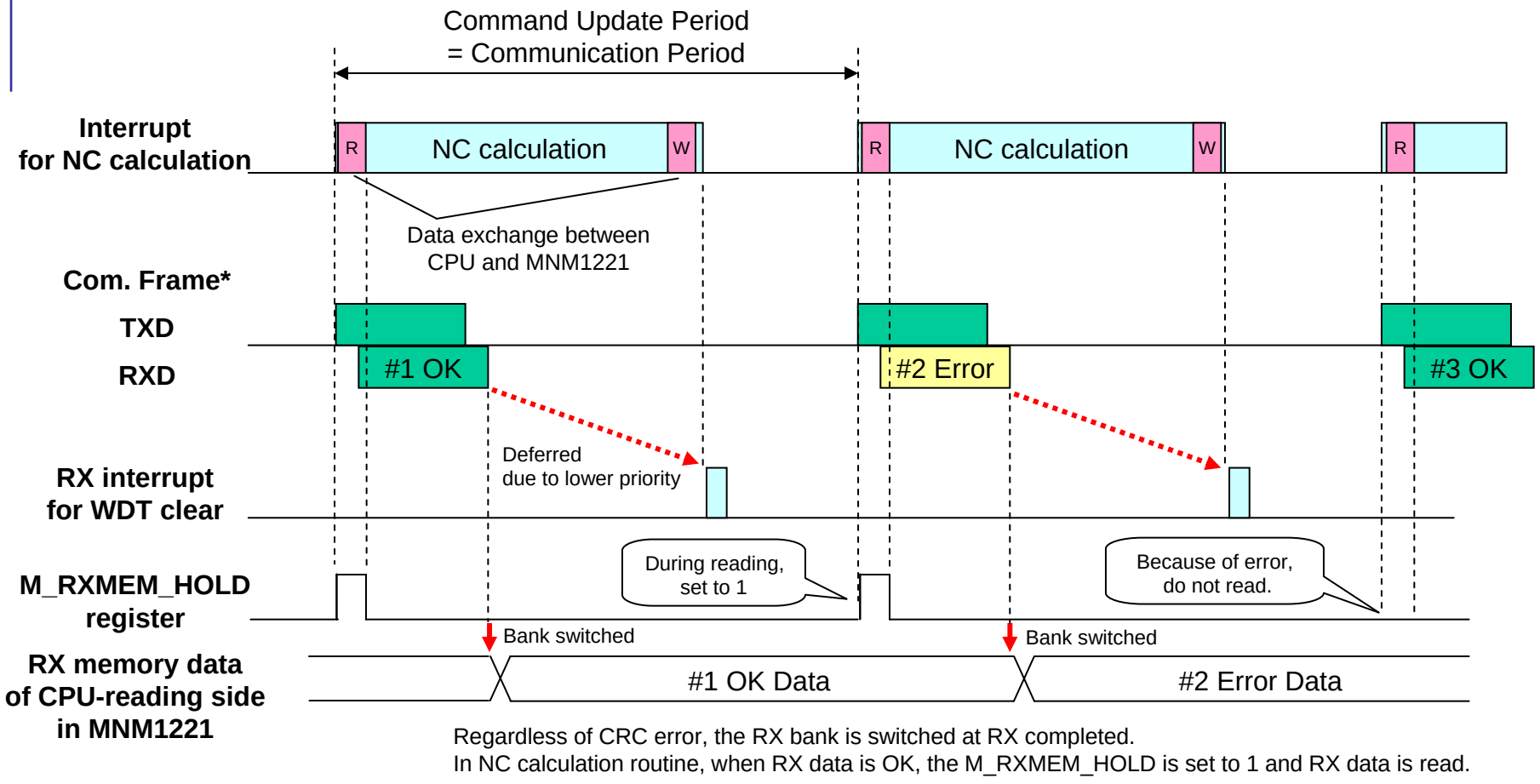
| Update Period | Com. Period | Setting | |
|---|---|---|---|
| | | Pr7.20 | Pr7.21 |
| 1.000ms | 1.000ms | 6 | 1 |
| 1.000ms | 0.500ms | 3 | 2 |
| 0.500ms | 0.500ms | 3 | 1 |
| 0.166ms | 0.166ms | 1 | 1 |
| 0.166ms | 0.083ms | 0 | 2 |

| | Name | Range | Description |
|---|---|---|---|
| Pr7.20 | Communication Period | 0 to 12 | 0: 0.083ms<br>1: 0.166ms<br>3: 0.5ms<br>6: 1.0ms<br>Else: Do not set. (Reserved) |
| Pr7.21 | Ratio of Command Update Period | 1 to 2 | Command Update / Communication Period<br>1: 1<br>2: 2 (Com.=0.5ms case only) |

# Goal of Timing Control

Panasonic
ideas for life

The goal is to make the following timing:

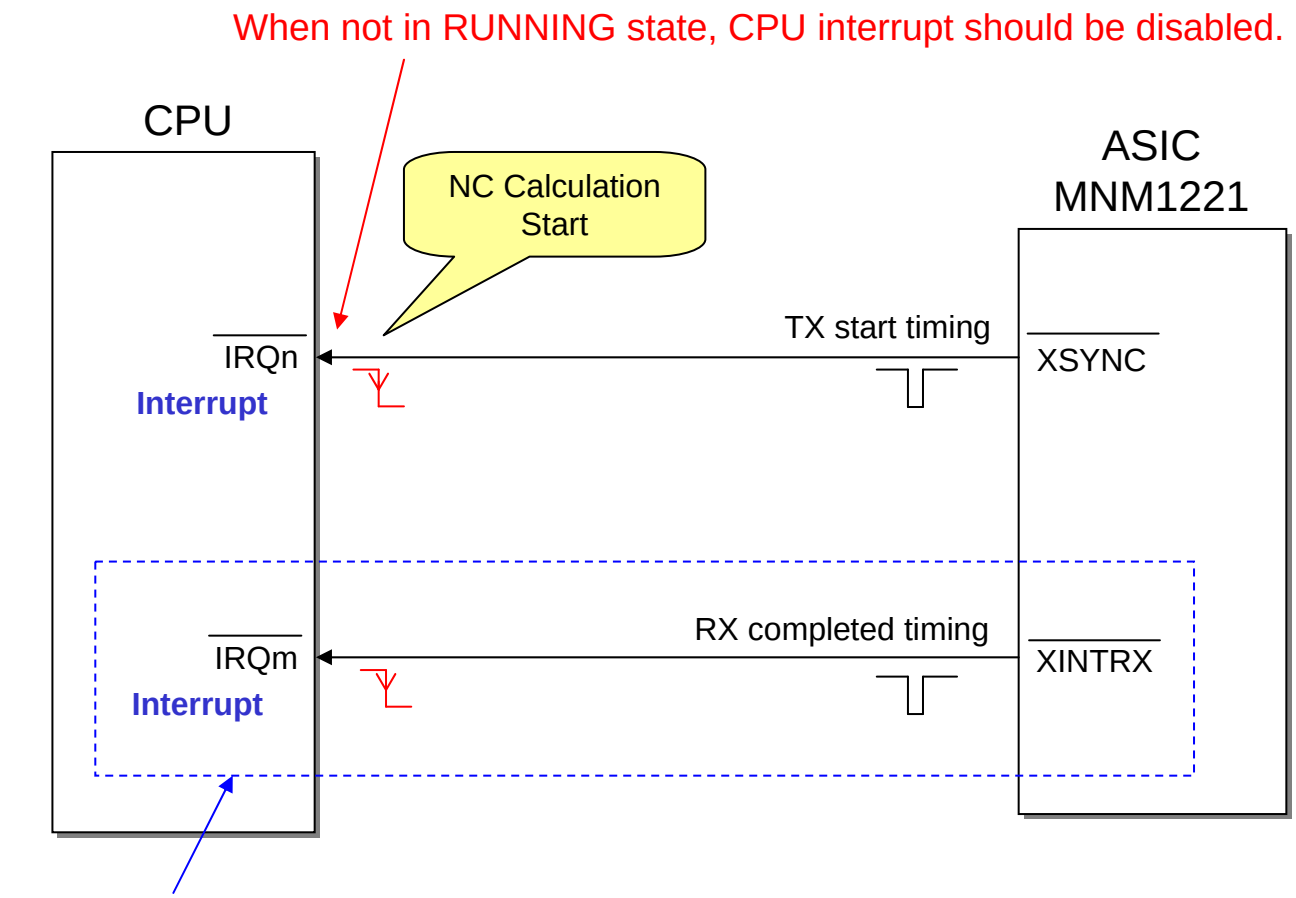



Regardless of CRC error, the RX bank is switched at RX completed.
In NC calculation routine, when RX data is OK, the M_RXMEM_HOLD is set to 1 and RX data is read.

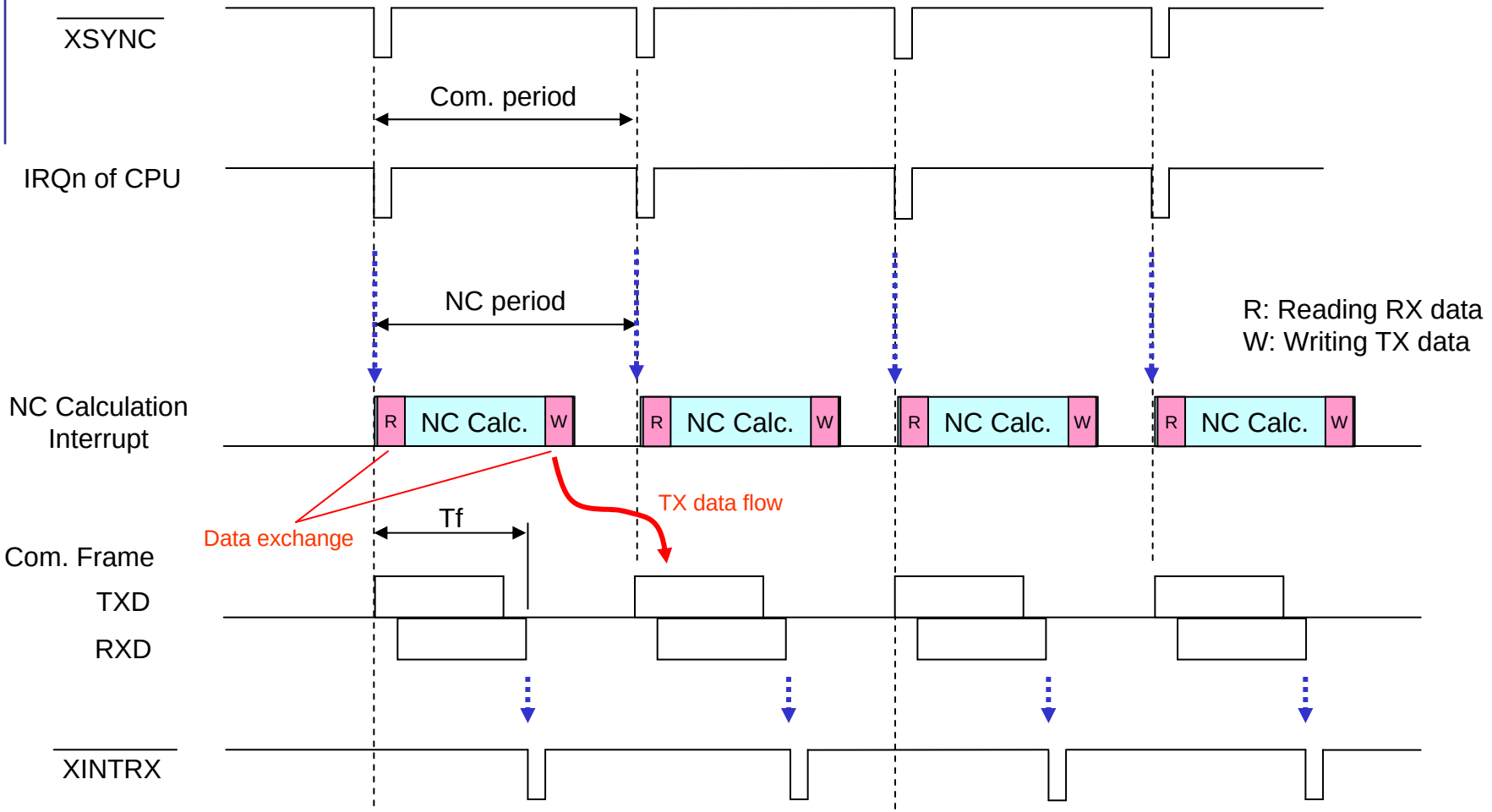
* One frame contains data of all slave nodes, and its length depends on the number of connected nodes.

# Example for an Embedded CPU

# Timing Circuit

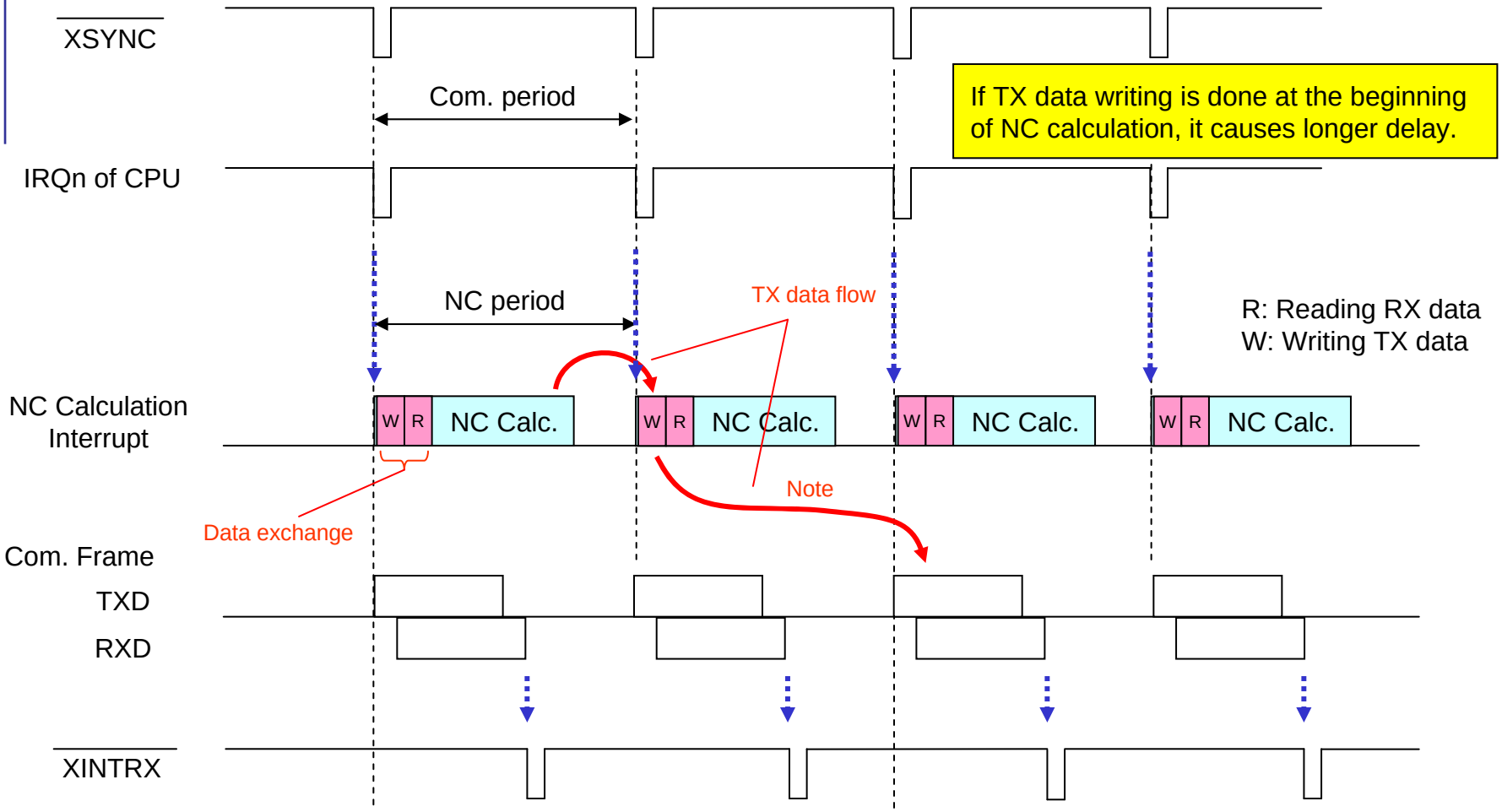


# Recommended Timing




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

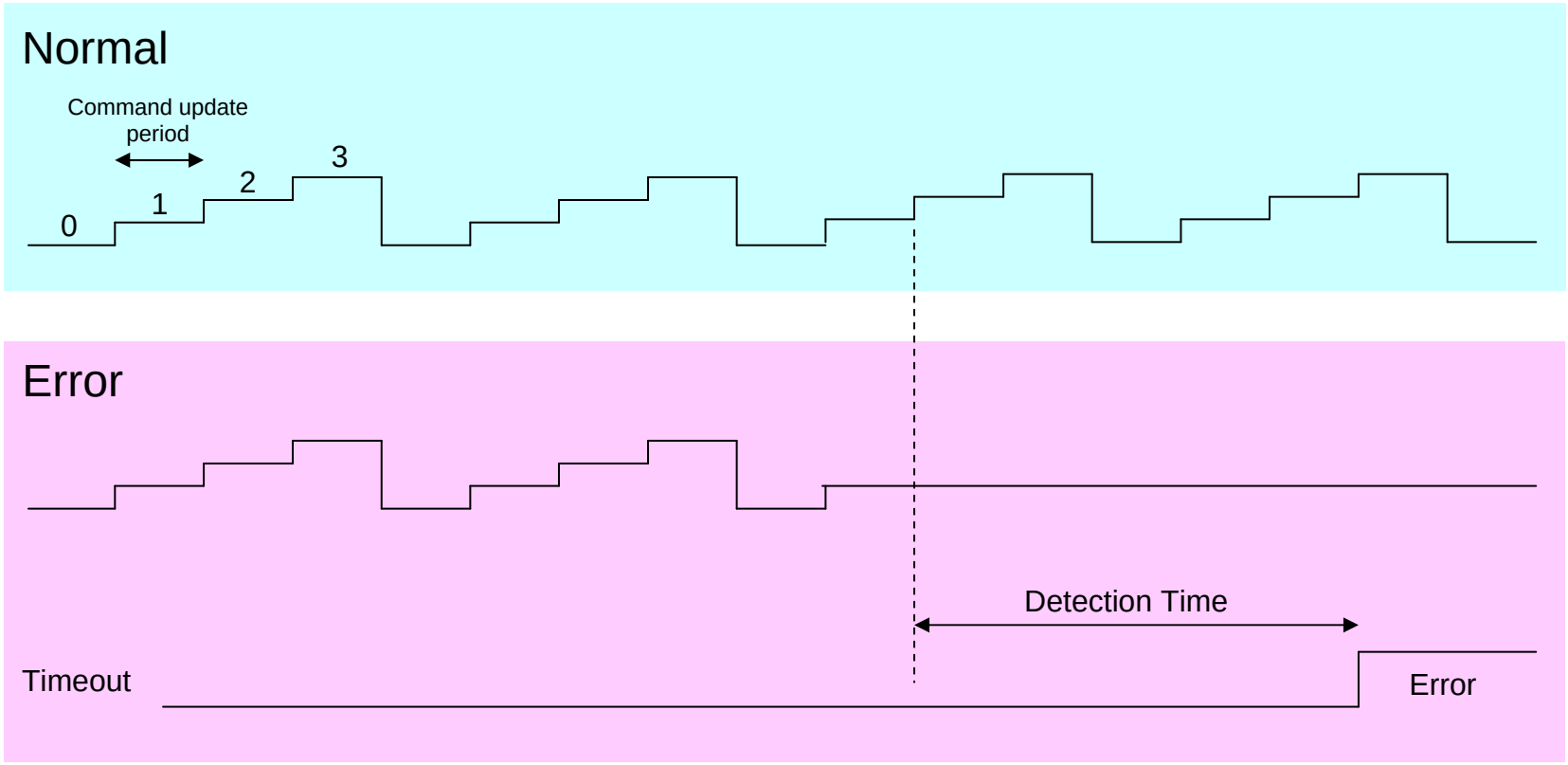
| | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|---|---|---|---|
| byte0 | 0 (CMD) | Update Counter | | MAC-ID | | | | |
| byte1 | 0 | Command Code | | | | | | |
| byte2 | Servo On | 0 | 0 | Gain SW | TL SW | HM Ctrl | 0 | 0 |
| byte3 | Hard Stop | SMT Stop | Pause | 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.

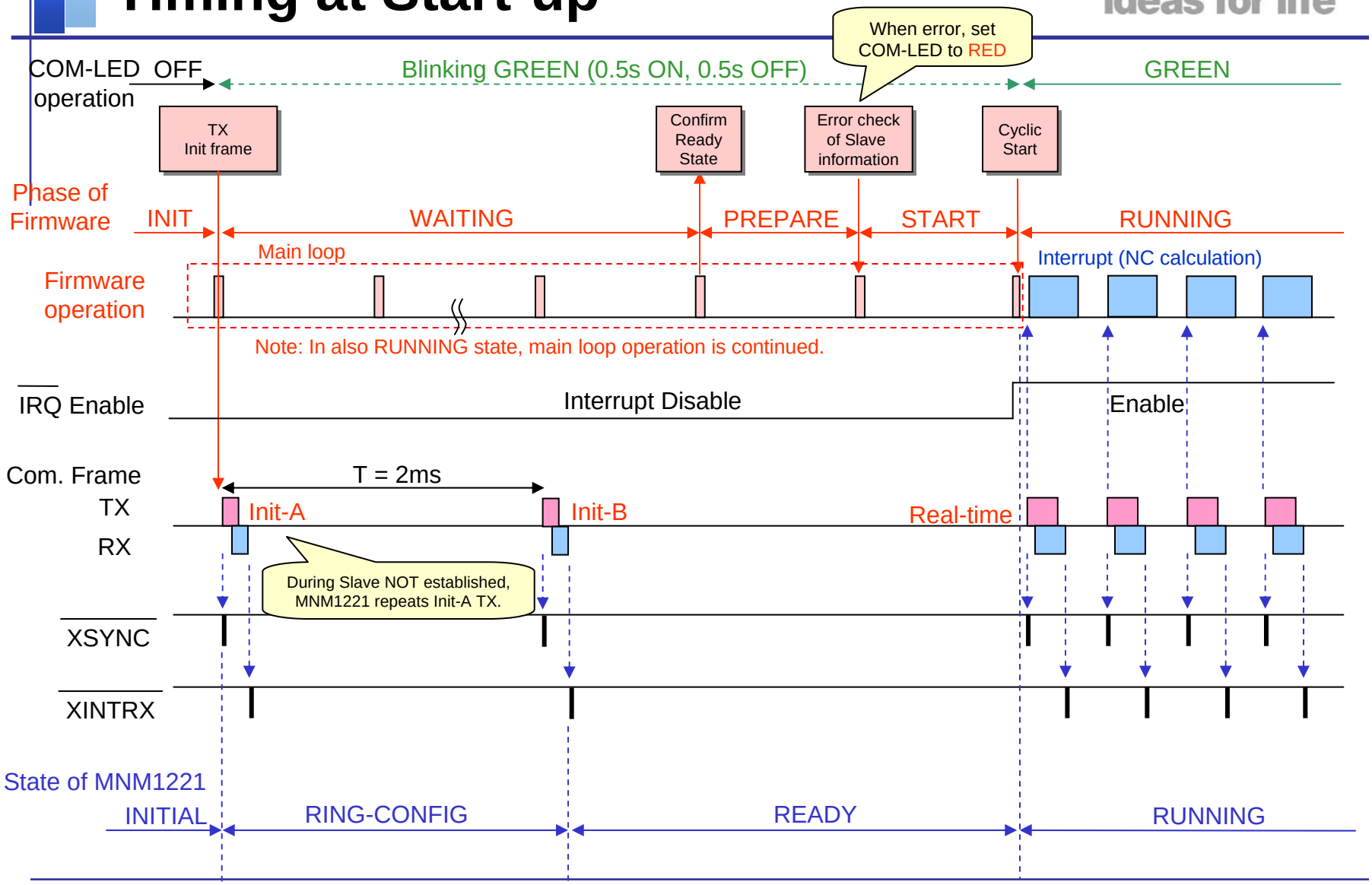



If timeout is detected, servo-off must be commanded to all axes for safety.

# Modifying the Example Code

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

# Timing at Start-up



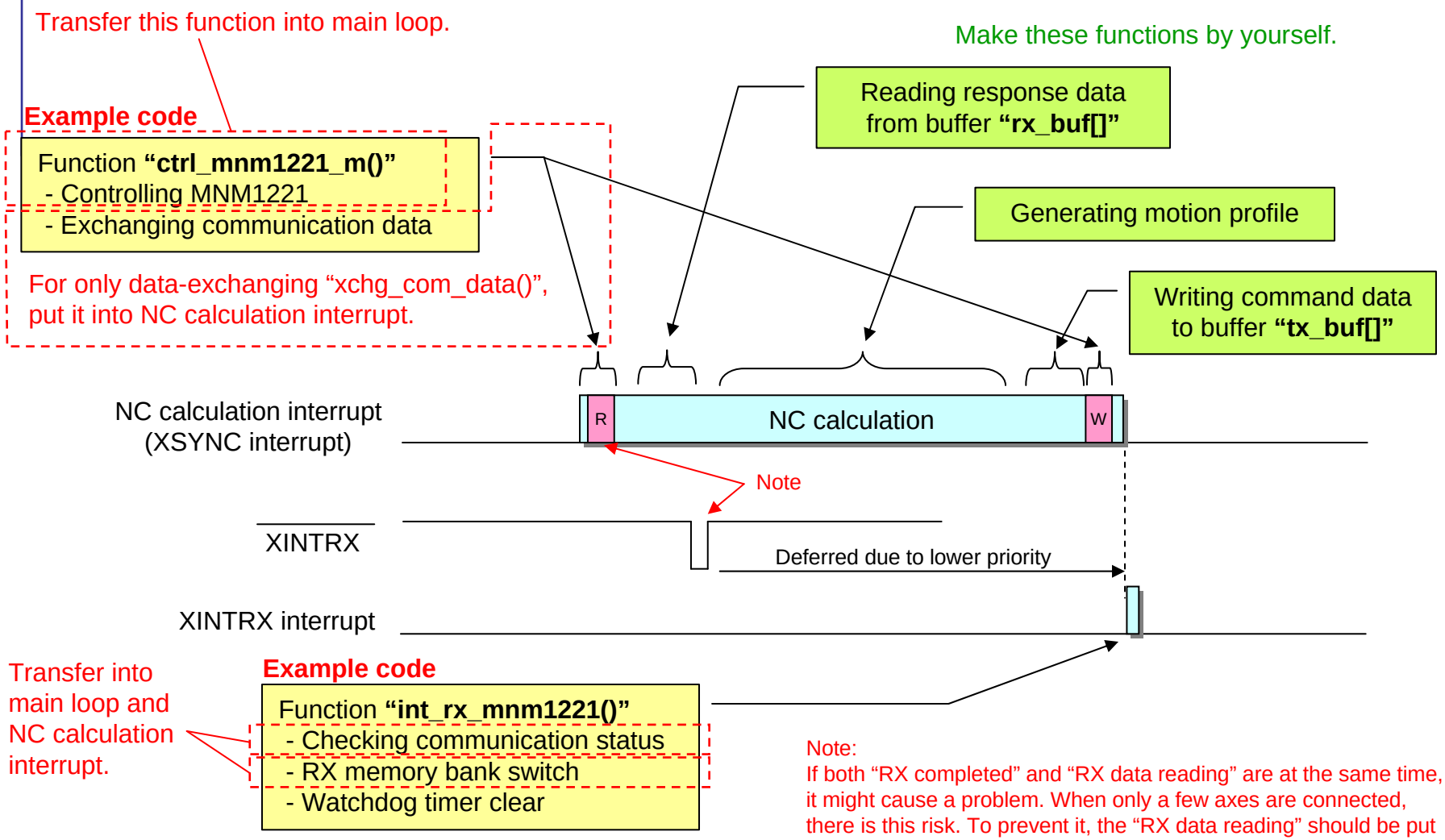
# Task Assignment

Panasonic
ideas for life

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

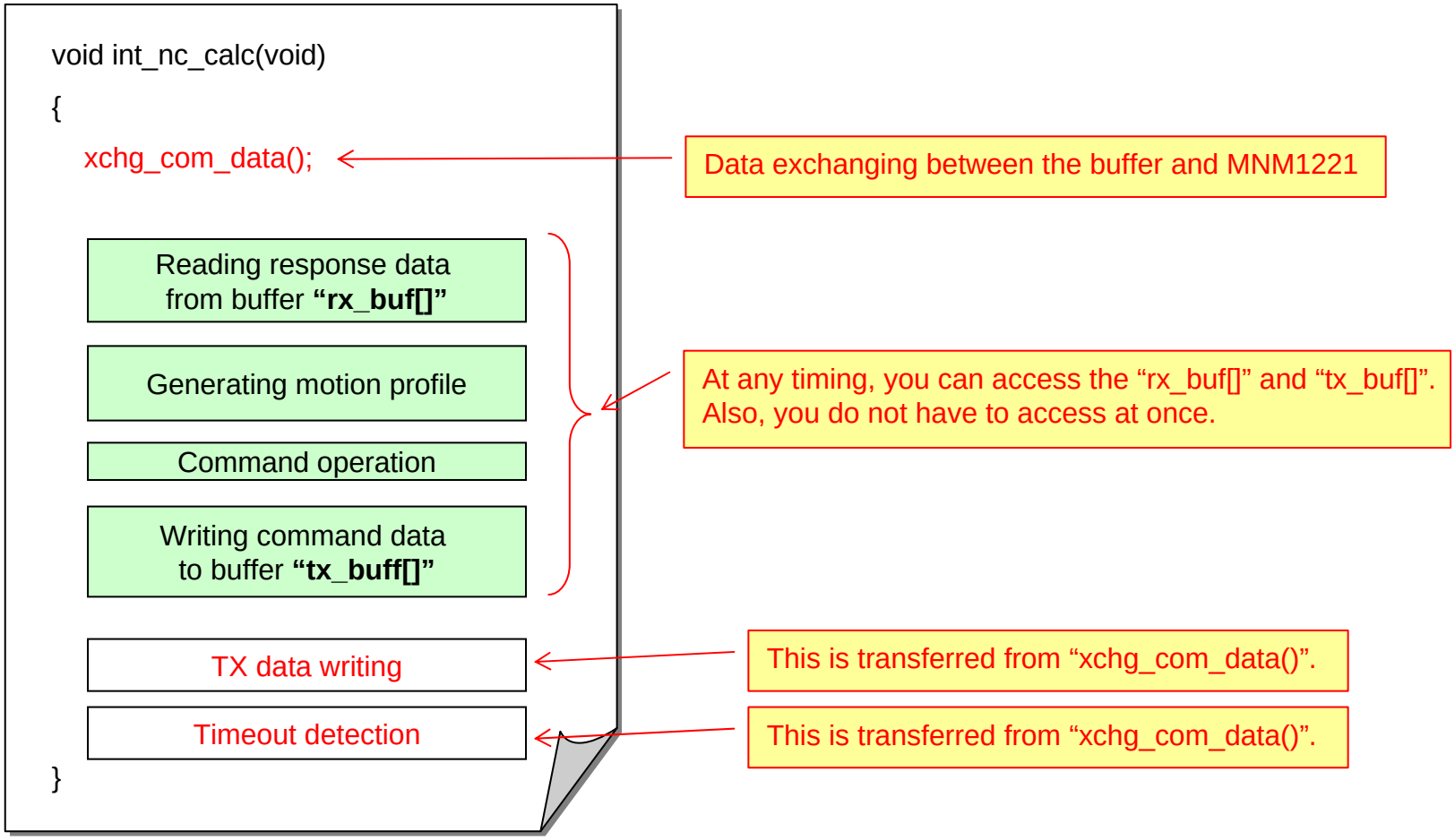
Notes:
- The interrupts must be disabled until RUNNING state.
- If not using XINTRX interrupt, a timeout detection with Update Counter Echo is necessary.

# Location of Example Codes




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



# MNM1221 initializing

```
381 /*-------------------------------------------------------------------------
382 *
383 *   Function Name:   init_mnm1221_m
384 *
385 *   Description:     initialize registers of MNM1221
386 *
387 *   Argument(s):     none
388 *
389 *   Return Value:    none
390 *
391 *   Comment:
392 *
393 *-------------------------------------------------------------------------*/
394 static void init_mnm1221_m(void)
395 {
396     MNM1221_M_RTF_FORM = RT_FRAME_FORM;
397     MNM1221_M_ERR_COUNT = 0;     /* not error count */
398     MNM1221_M_TXTIM_SEL = 1;     /* external period timer */
399     MNM1221_M_RXMEM_HOLD = 1;    /* hold RX buffer memory */
400
401     MNM1221_M_INIT_DONE = 1;     /* initialize is done */
402 }
```

Replace this line.

0

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

| Communication Period | MNM1221_M_TX_PERIOD |
|---|---|
| 1ms | 0x61A8 |
| 0.5ms | 0x30D4 |
| 0.166ms | 0x1047 |

# Reading State

Panasonic ideas for life

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 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.

```
/*************************************************************************/
/* 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;
```

Here, insert an interrupt enable function provided by yourself.

Replace with TX data writing extracted from xchg_com_data().

# In Running State

Panasonic
ideas for life

ctrl_mnm1221_m() in mnm1221_m.c

Delete this.

```
341     /*--- running state (cyclic transmission) ------------------------------*/
342     case PH_RUNNING:                /* MNM1221 is in RUNNING state. */
343
344 /*****************************************************************************/
345 /* This is for test. You must replace with your application.                 */
346 /*---------------------------------------------------------------------------*/
347 /*
348 * In actual application, the routin setting to TX buffer will be placed
349 * after NC calculation. i.e. The routin is not in ctrl_mnm1221_m(), but
350 * at the end of the timer interrupt for NC calculation.
351 */
352         set_txbuf_example(0x20);    /* Position command */
353 /************************************************** end of test ***/
354
355         xchg_com_data();            /* exchange communication data */
356         if (is_timeout()) {
357             com_err.run |= B_TIMEOUT;
358             /* Add error routine. */
3591#if 0
3601             phase = PH_RESET;      /* depend on
3611#endif
362         }
363         break;
364
```

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

Panasonic
ideas for life

## xchg_com_data() in mnm1221_m.c

Put this function to the beginning of NC calculation interrupt.

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

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

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

unsigned long ul_temp;

```
    /* 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

```
    /* 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

```
    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;


        MNM1221_M_RXMEM_HOLD = 0;
    }
#else                           /* 32bits bus width */
```

# Data Exchanging (Cont.)

Panasonic
ideas for life

```
#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  */

mnm1221.data_crc_err = MNM1221_M_DCRC_ERR;

    /* 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);
    }

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

   MNM1221_M_RXMEM_HOLD = 0;
}

#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

# XINTRX Interrupt

Panasonic
ideas for life

Delete this.

```
212 void int_rx_mnm1221_m(void)
213 {
214 #ifdef MASTER_16BIT_ACCESS      /* 16bits bus */
215     unsigned long ul_temp;
216 #endif
217
218     mnm1221.state = MNM1221_M_STATE;
219
220     /* If not RUNNING state (cyclic transmission), return. */
221     if (!(mnm1221.state & B_RUNNING)) {
222         mnm1221.err_flgs1 = MNM1221_M_ERR_FLAGS1;
223         return;
224     }
225
226     mnm1221.err_flgs2 = MNM1221_M_ERR_FLAGS2;
227 /*
228 * Normally, err_flgs1 and 2 are not used. They are monitored only.
229 * Because MNM1221 has error recovering function in RINNING state,
230 * the received data can be used as long as data_crc_err indicates OK.
231 */
```

# XINTRX Interrupt (Cont.)

Panasonic
ideas for life

Delete this.

```
232 ↵
233 #ifdef MASTER_16BIT_ACCESS      /* 16bits bus */↵
234     ul_temp = MNM1221_M_DCRC_ERR_H;↵
235     ul_temp <<= 16;↵
236     ul_temp |= MNM1221_M_DCRC_ERR_L;↵
237     mnm1221.data_crc_err = ul_temp;↵
238 #else                           /* 32bits bus */↵
239     mnm1221.data_crc_err = MNM1221_M_DCRC_ERR;↵
240 #endif↵
241 ↵
242     /* To read valid data only, if error, do not switch RX memory bank.*/↵
243     if (mnm1221.data_crc_err) {↵
244     /* error detected */↵
245         com_err.count++;↵
246     } else {↵
247     /* data is OK */↵
248         MNM1221_M_RXMEM_HOLD = 0;    /* switch bank */↵
249         MNM1221_M_RXMEM_HOLD = 1;    /* return hold state */↵
250     }↵
251 ↵
252     watchdog_tim = 0;       /* clear watchdog timer */↵
253 }↵
```

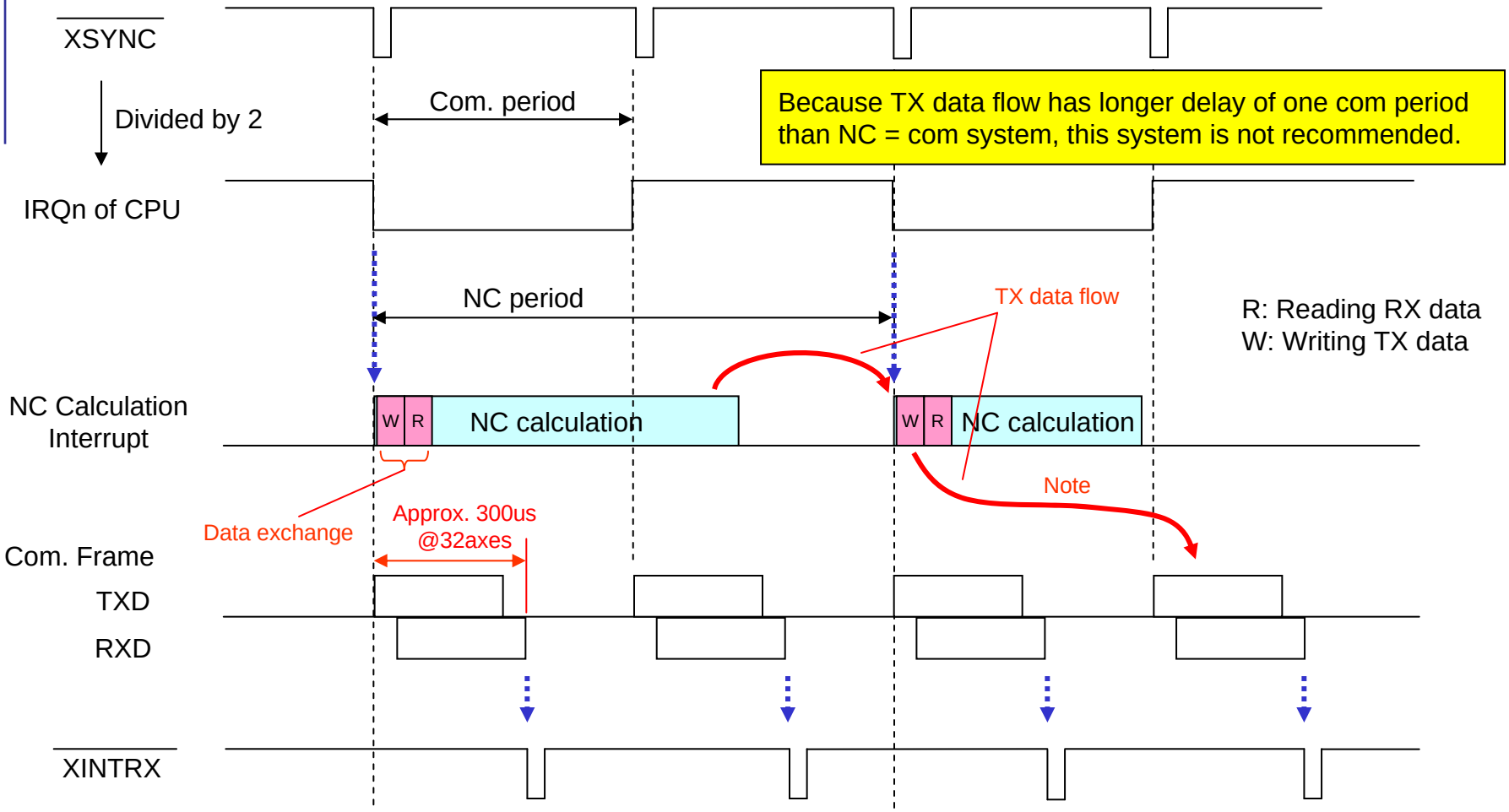
Leave “watchdog timer clear” only.

# Appendix

# System Consideration for 2 times communication one NC calculation

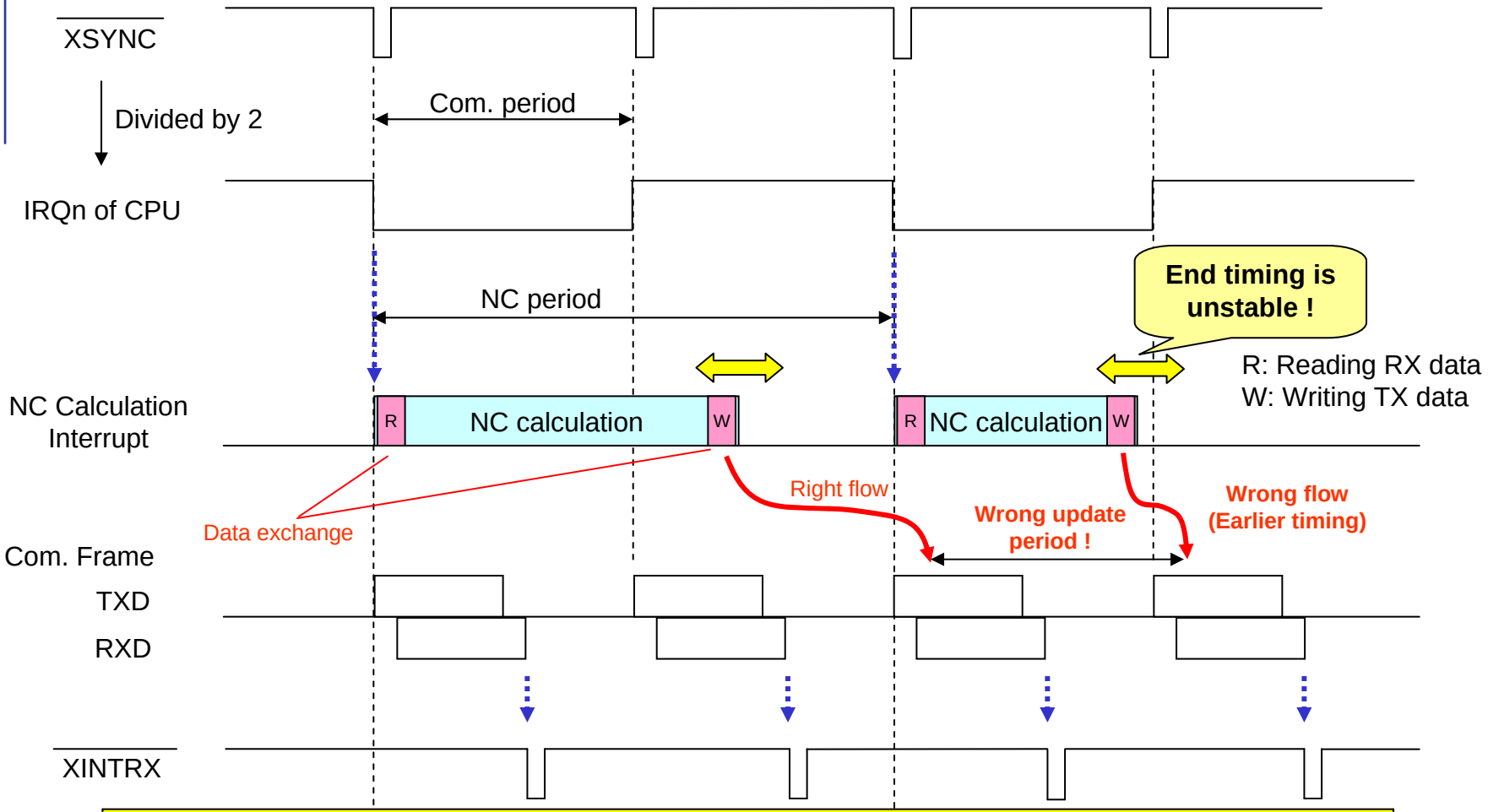
This system is not recommended because of longer delay in TX data flow.

# Timing Chart




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.**