マスタ用システム設計ガイド

アプライアンス社
モータビジネスユニット

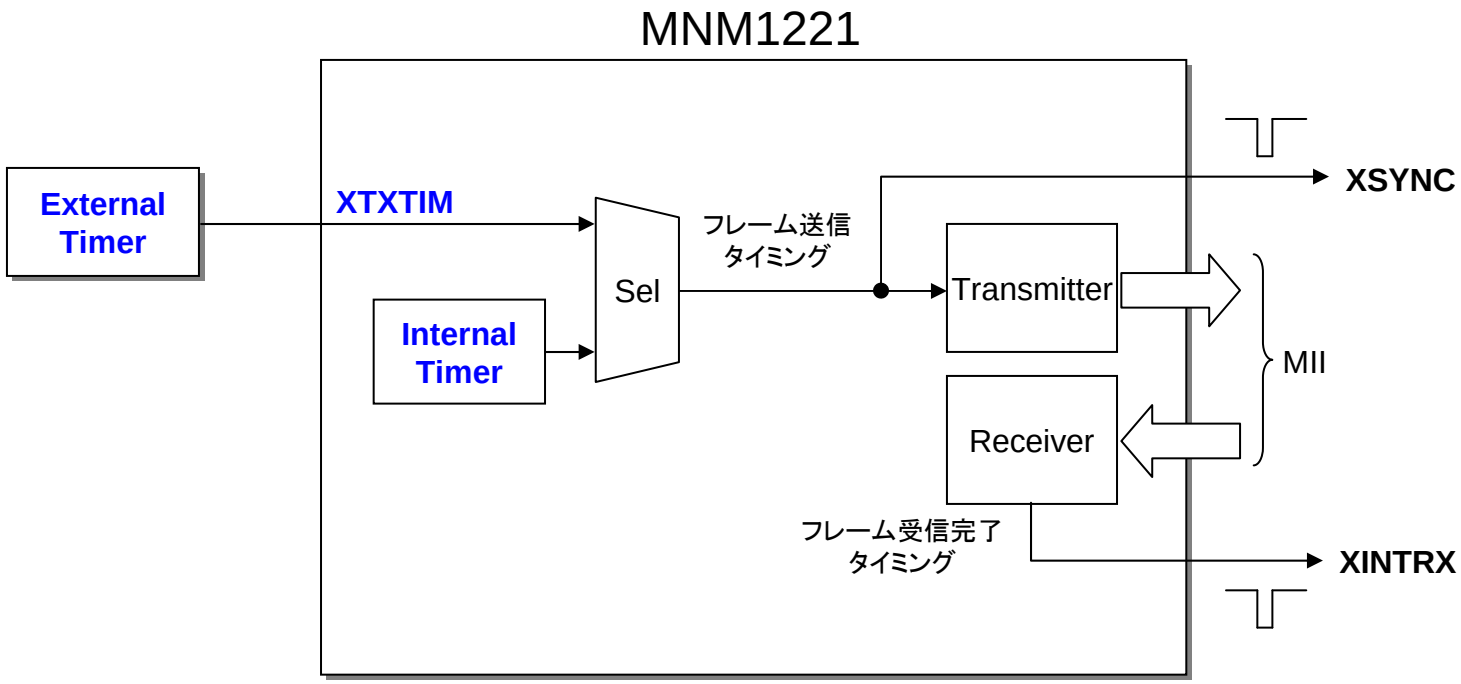
変更履歴

Panasonic
ideas for life

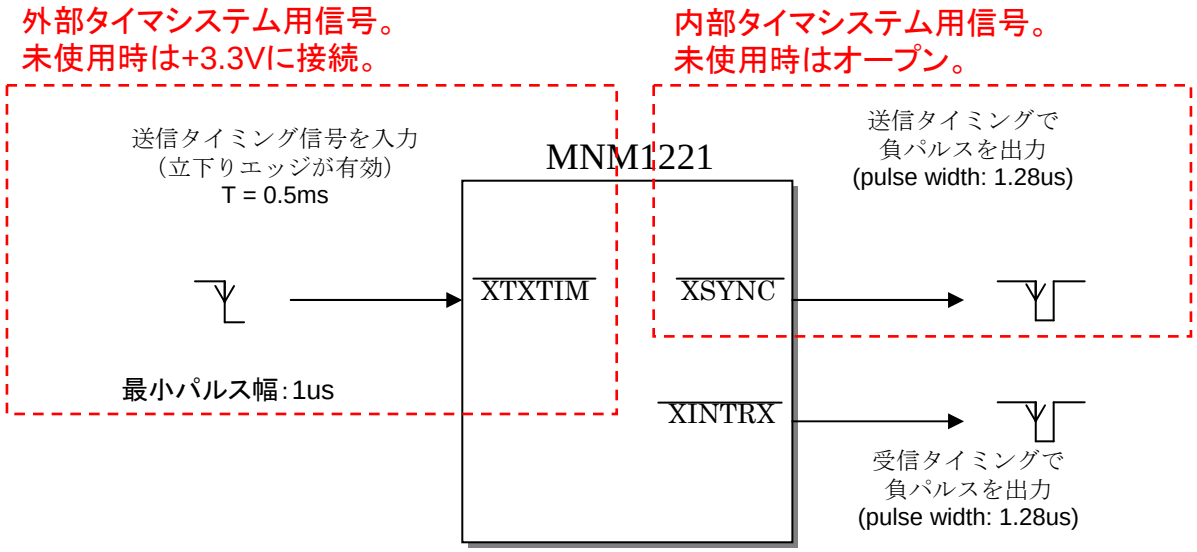
Revision	日付	変更内容
1	2005/7/14	初版
2	2012/2/7	P1 タイトルを「ソフトウェア（ファームウェア）開発のガイド」から変更。 P3 「はじめに」を追加。 P7 SH7216の例を追加。 P19 SH7145の例を追加。 P25 TMS320F28335の例を追加。 SH7065の例を削除。 TMS320VC33-120の例を削除。 P58 プロファイルポジションI/Fの概要を追加。 P72 「内部タイマを用いたシステム」を追加。 文言等を小変更。

はじめに

送信タイミングを制御するために、MNM1221は外部タイマと内部タイマの2つのタイマソースを持っています。本書では外部タイマを用いたシステムをChapter 1で、また、内部タイマを用いたシステムをChapter 2で説明します。



MNM1221のタイミング信号端子



XTXTIM入力はRUNNING状態において有効で、
それ以外の状態では無視されます。
RING CONFIG状態でのInit-A, Init-B フレームの送信は、
MNM1221が内部のタイマを用いて行います。

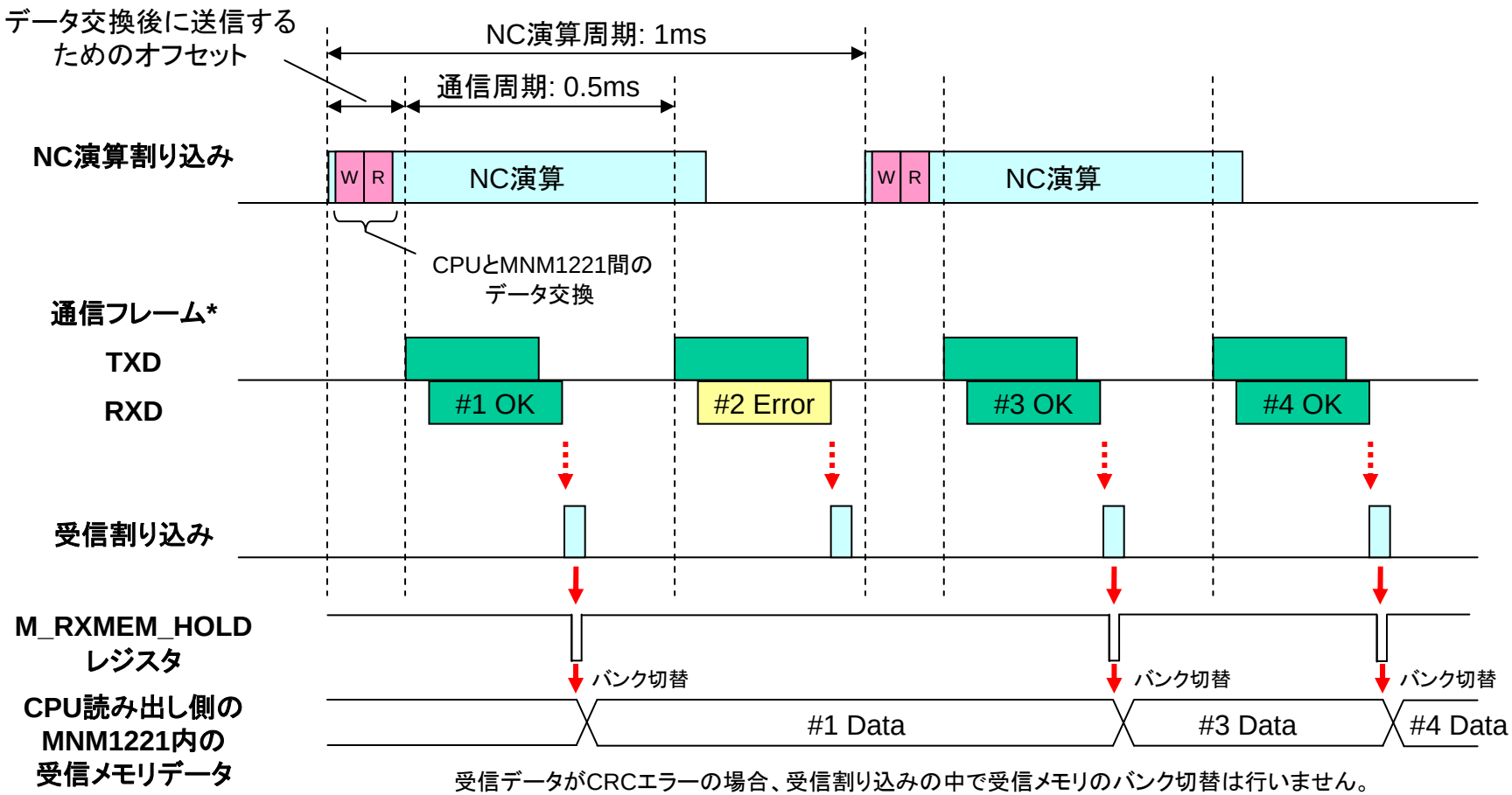
Chapter 1

外部タイマを用いたシステム

タイミング制御の目的

Panasonic
ideas for life

下図のタイミング作成が目的

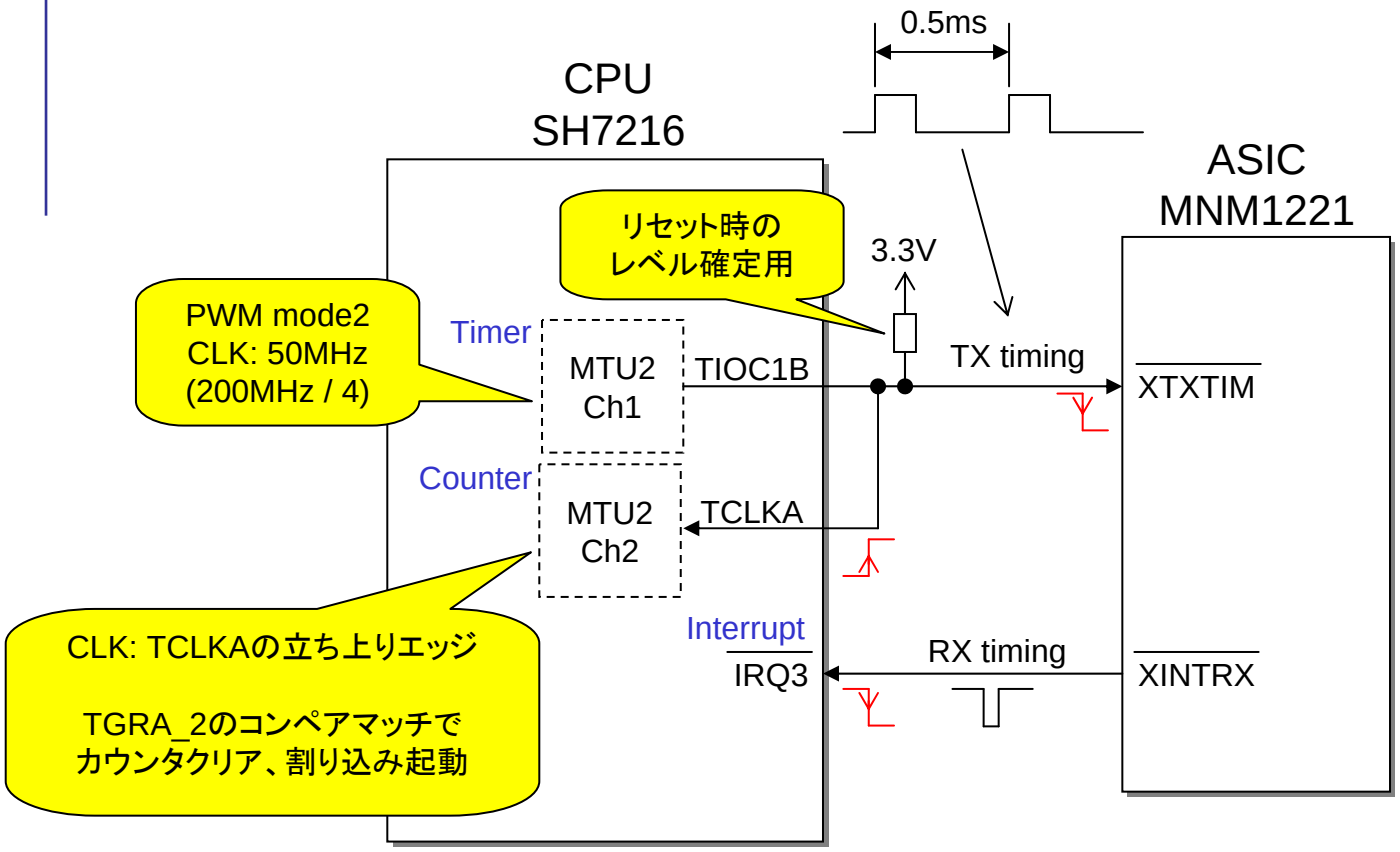


受信データがCRCエラーの場合、受信割り込みの中で受信メモリのバンク切替は行いません。
この場合、NC演算割り込み先頭のデータ交換では前回の受信データが読み出されます。

•1つのフレームには全スレーブノードのデータが含まれており、フレームの長さは接続されているノードの数に依存します。

SH7216 (Renesas) との接続例

タイミング回路



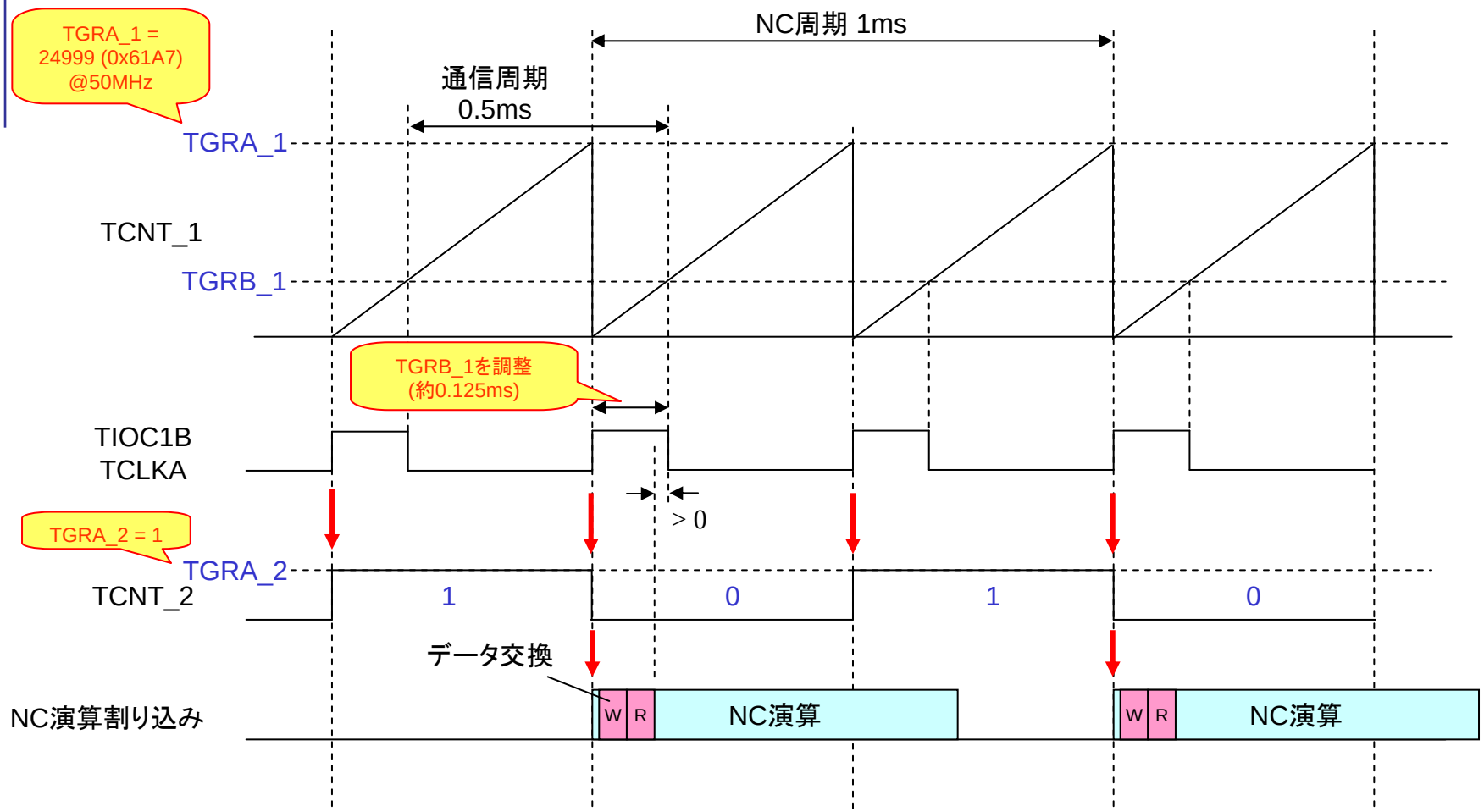
- MTU2-Ch1で送信タイミング信号を生成。周期0.5msの場合、**TGRA_1 = 24999(0x61A7)@50MHz**
- この信号をMTU2-Ch2で分周しNC演算割り込みの起動信号を生成。周期1msの場合、**TGRA_2 = 1**
- MNM1221のXINTRXをIRQ3で受けて受信割り込みを起動。

多重割り込みの設定

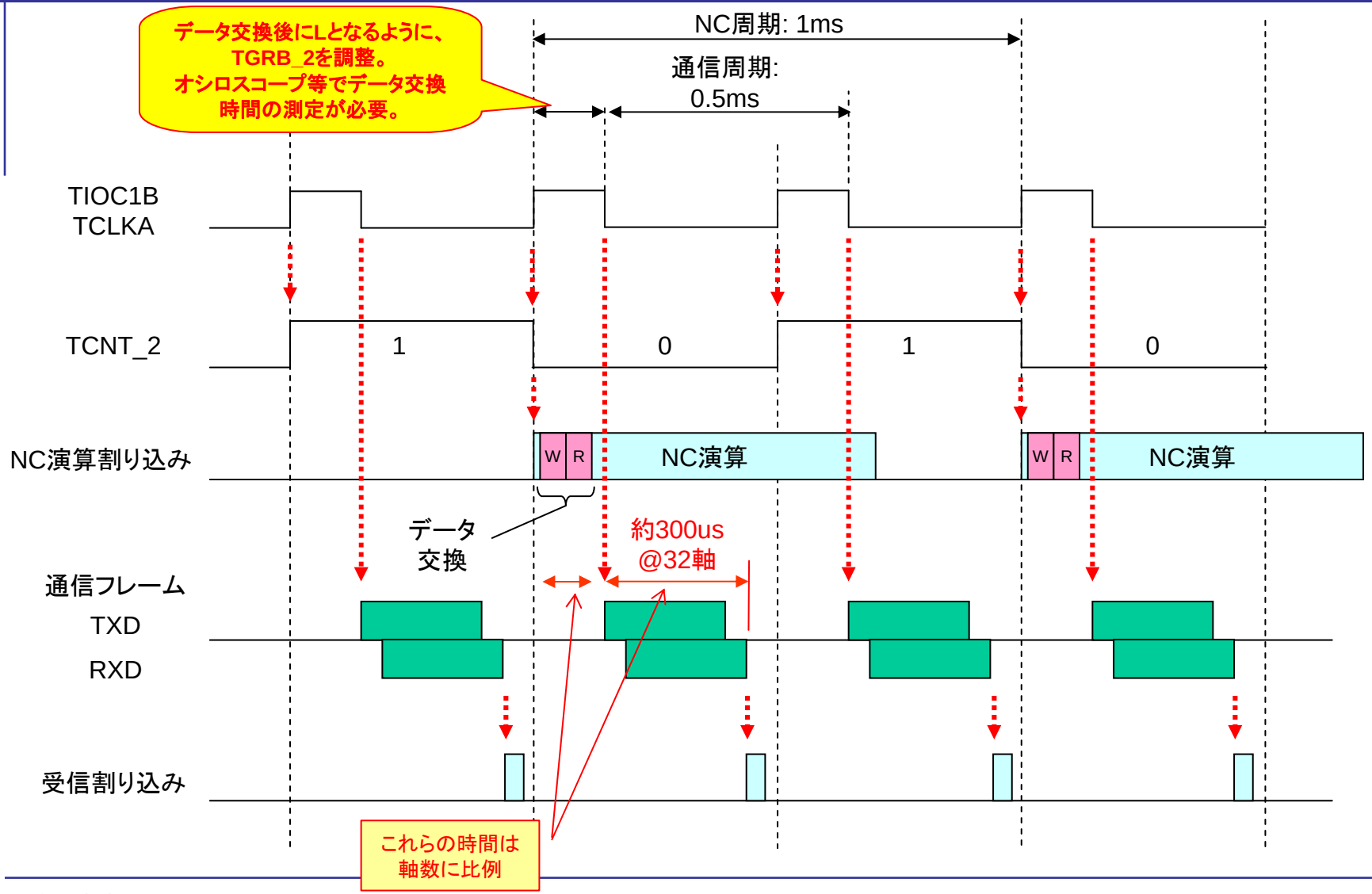
Panasonic
ideas for life

信号ソース	トリガ	優先順位	周期	処理
TGIA_2	MTU2 Ch2 コンペアマッチ	-	1ms	- 通信データ交換 - NC演算
/IRQ3	受信完了	TGIA_2 より高く 設定	0.5ms	- 通信ステータス確認 - 受信メモリバンク切替

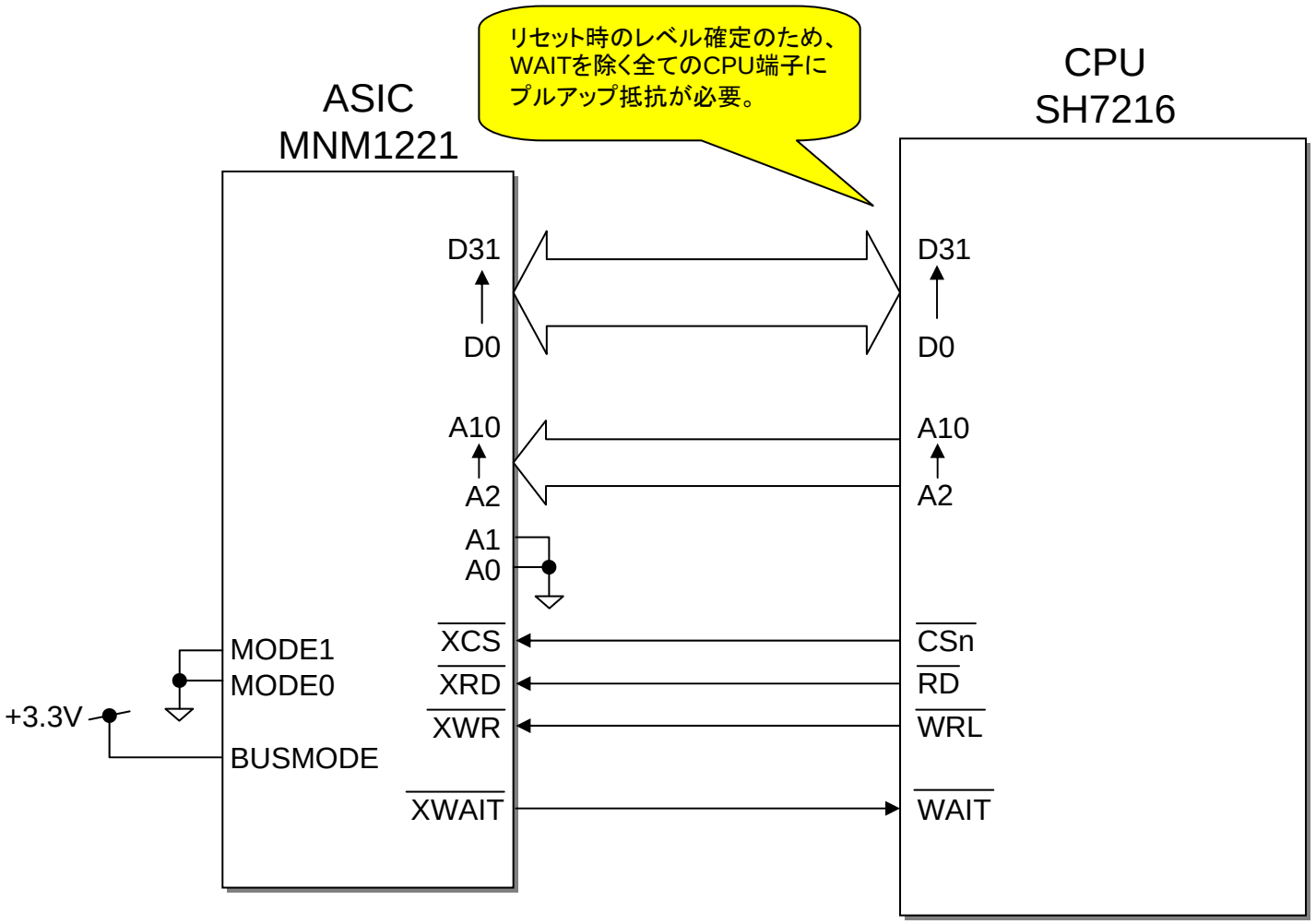
タイムチャート 1


TGRA_1 =
24999 (0x61A7)
@50MHz
NC周期 1ms
通信周期
0.5ms
TGRA_1
TCNT_1
TGRB_1
TGRB_1を調整
(約0.125ms)
TIOC1B
TCLKA
> 0
TGRA_2 = 1
TGRA_2
TCNT_2
1
0
1
0
データ交換
NC演算割り込み
W
R
NC演算
W
R
NC演算

タイムチャート 2

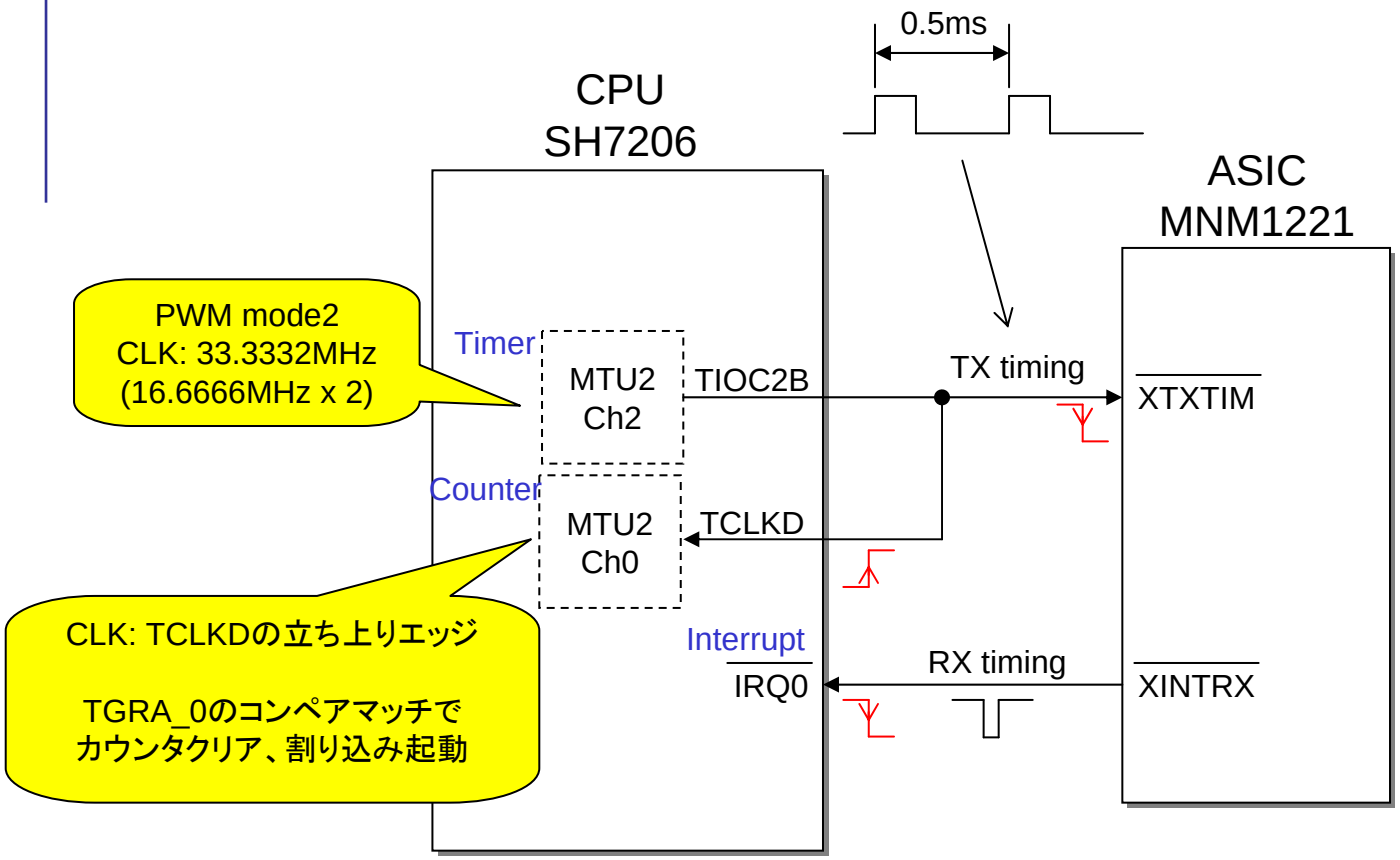

データ交換後にLとなるように、
TGRB_2を調整。
オシロスコープ等でデータ交換
時間の測定が必要。
NC周期: 1ms
通信周期:
0.5ms
TIOC1B
TCLKA
TCNT_2
1
0
1
0
NC演算割り込み
W
R
NC演算
W
R
NC演算
データ
交換
約300us
@32軸
通信フレーム
TXD
RXD
受信割り込み
これらの時間は
軸数に比例

バス接続


リセット時のレベル確定のため、
WAITを除く全てのCPU端子に
プルアップ抵抗が必要。
ASIC
MNM1221
CPU
SH7216
D31
D0
A10
A2
A1
A0
MODE1
MODE0
+3.3V
BUSMODE
XCS
XRD
XWR
XWAIT
CSn
RD
WRL
WAIT

SH7206 (Renesas) との接続例

タイミング回路



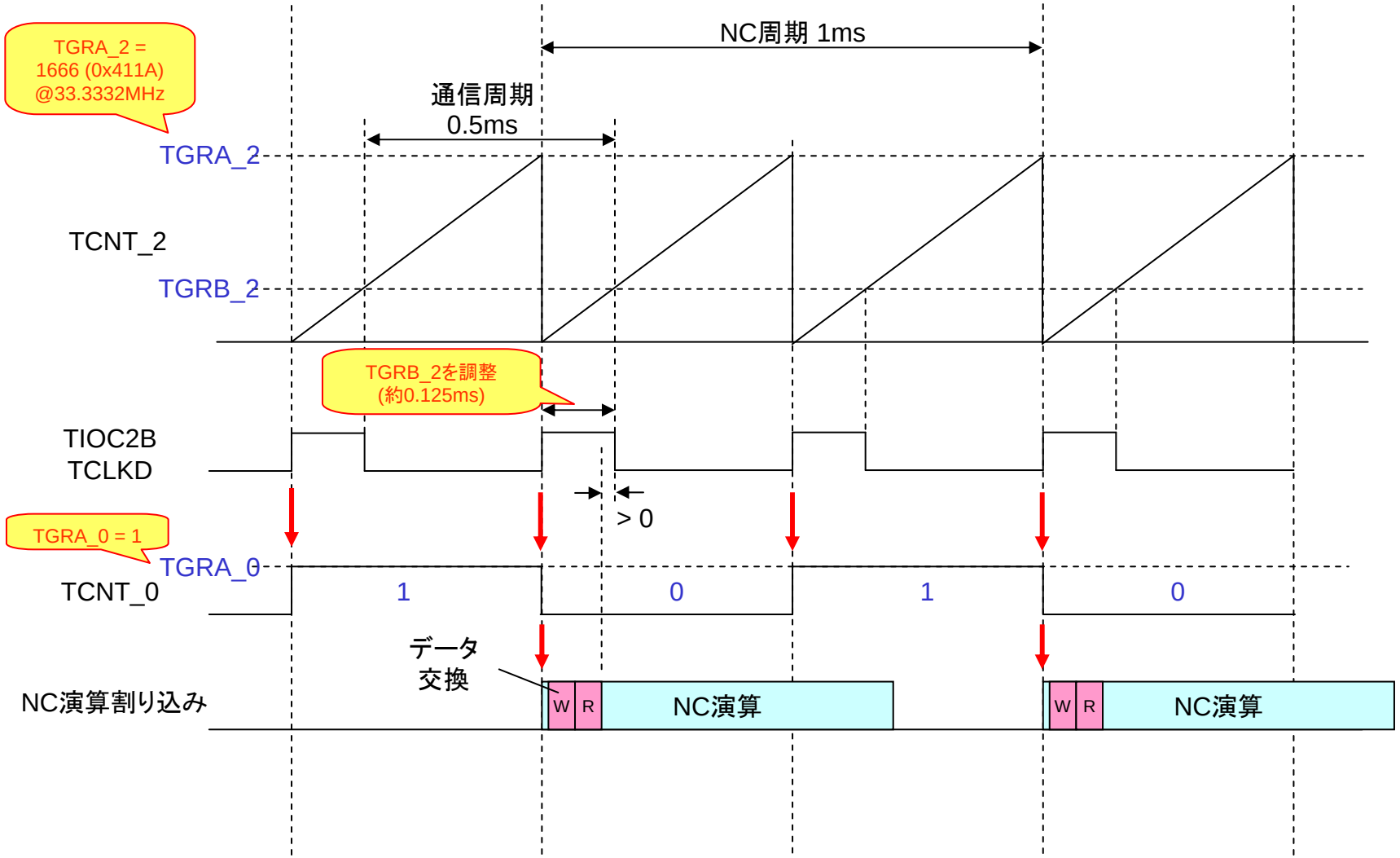
- MTU2-Ch2で送信タイミング信号を生成。周期0.5msの場合、**TGRA_2 = 16666(0x411A)@33.3332MHz**
- この信号をMTU2-Ch0で分周しNC演算割り込みの起動信号を生成。周期1msの場合、**TGRA_0 = 1**
- MNM1221のXINTRXをIRQ0で受けて受信割り込みを起動。

多重割り込みの設定

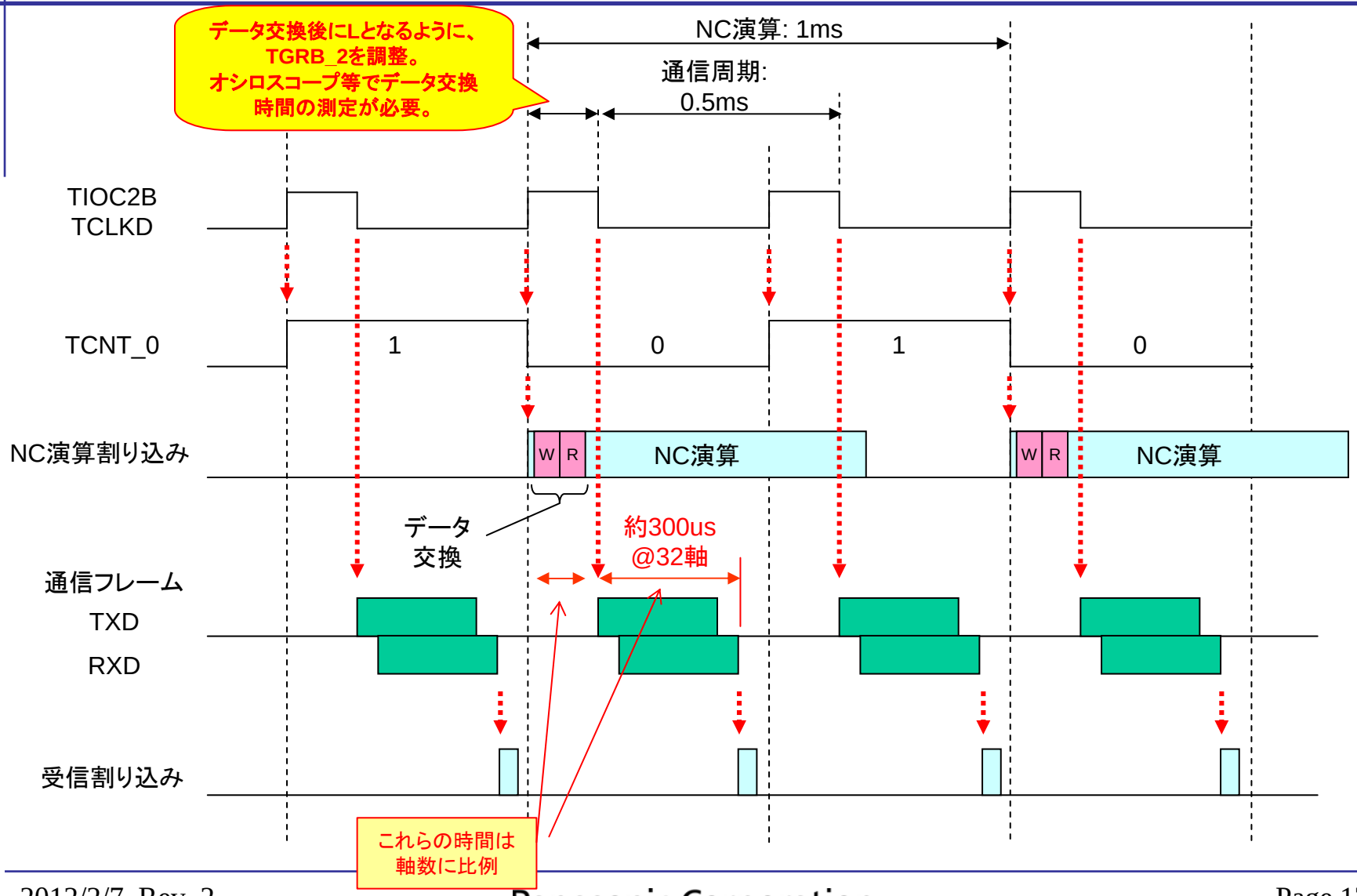
Panasonic
ideas for life

信号ソース	トリガ	優先順位	周期	処理
TGIA_0	MTU2 Ch0 コンペアマッチ	-	1ms	- 通信データ交換 - NC演算
/IRQ0	受信完了	TGIA_0 より高く 設定	0.5ms	- 通信ステータス確認 - 受信メモリバンク切替

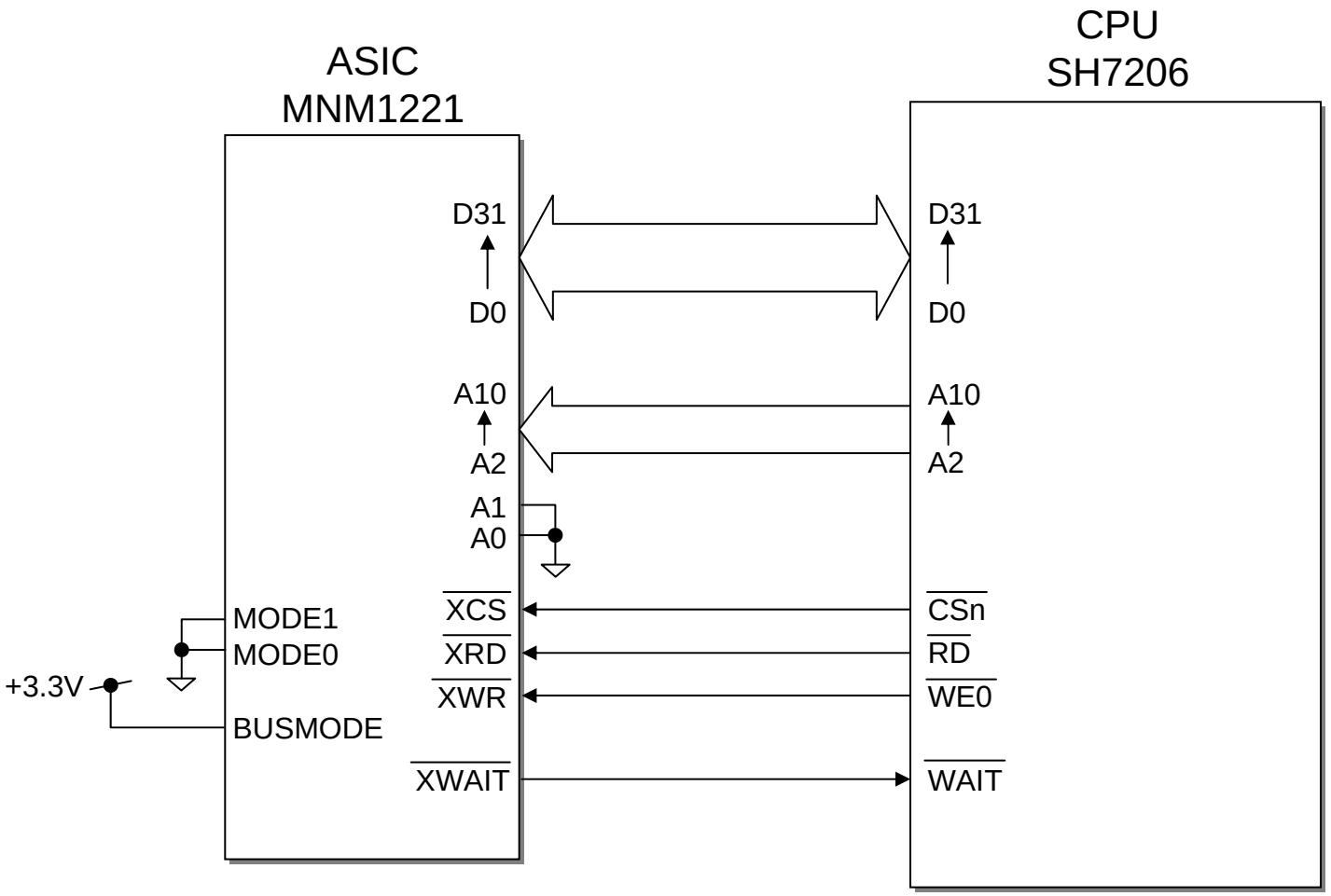
タイムチャート 1


TGRA_2 =
1666 (0x411A)
@33.3332MHz
NC周期 1ms
通信周期
0.5ms
TGRA_2
TCNT_2
TGRB_2
TGRB_2を調整
(約0.125ms)
TIOC2B
TCLKD
> 0
TGRA_0 = 1
TGRA_0
TCNT_0
1
0
1
0
データ
交換
NC演算割り込み
W
R
NC演算
W
R
NC演算

タイムチャート 2


データ交換後にLとなるように、
TGRB_2を調整。
オシロスコープ等でデータ交換
時間の測定が必要。
NC演算: 1ms
通信周期:
0.5ms
TIOC2B
TCLKD
TCNT_0
1
0
1
0
NC演算割り込み
W
R
NC演算
W
R
NC演算
データ
交換
約300us
@32軸
通信フレーム
TXD
RXD
受信割り込み
これらの時間は
軸数に比例

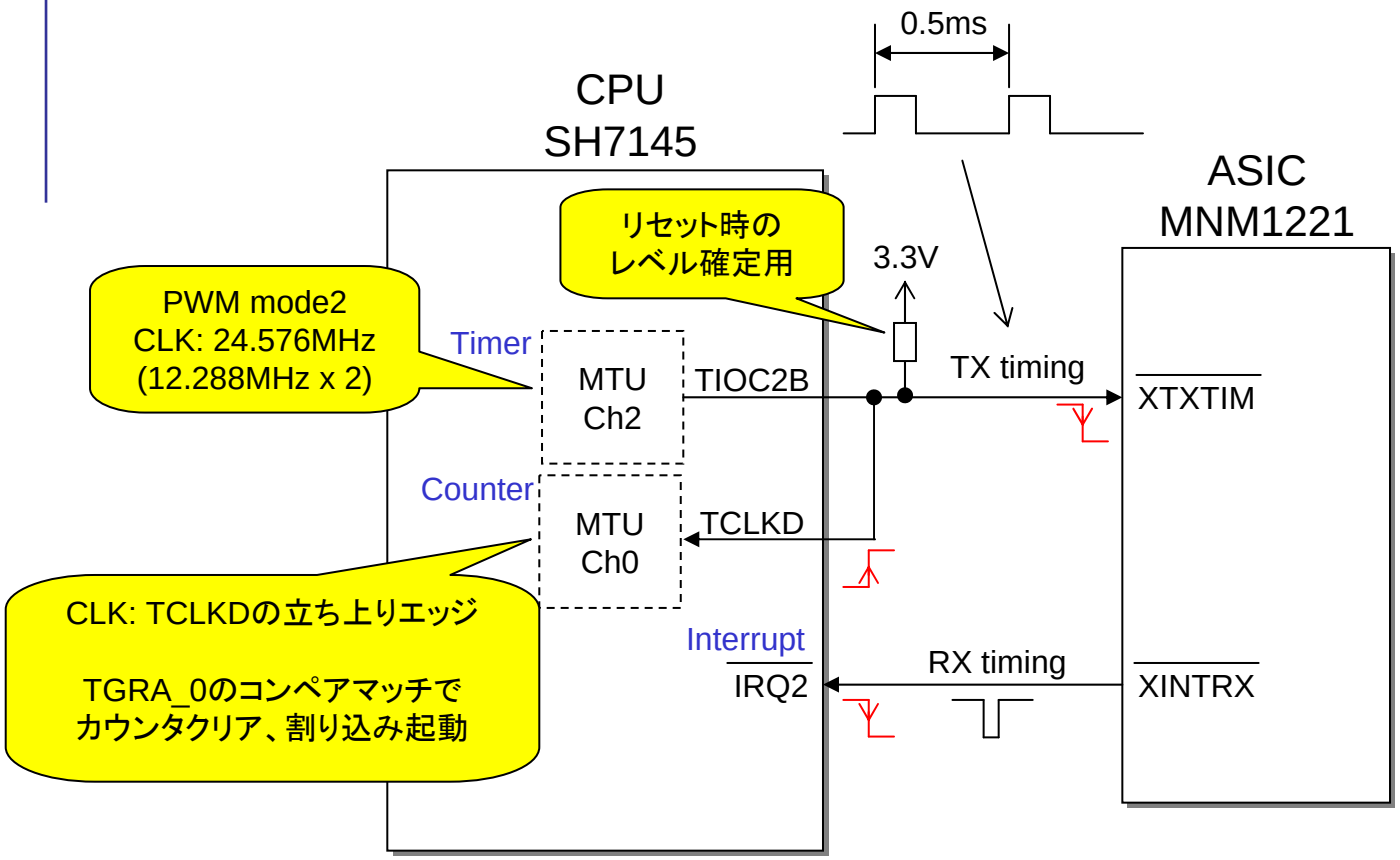
バス接続



注：SH7206の各端子には”ウィークキーパ”が付いているので、プルアップ抵抗は不要。

SH7145 (Renesas) との接続例

タイミング回路



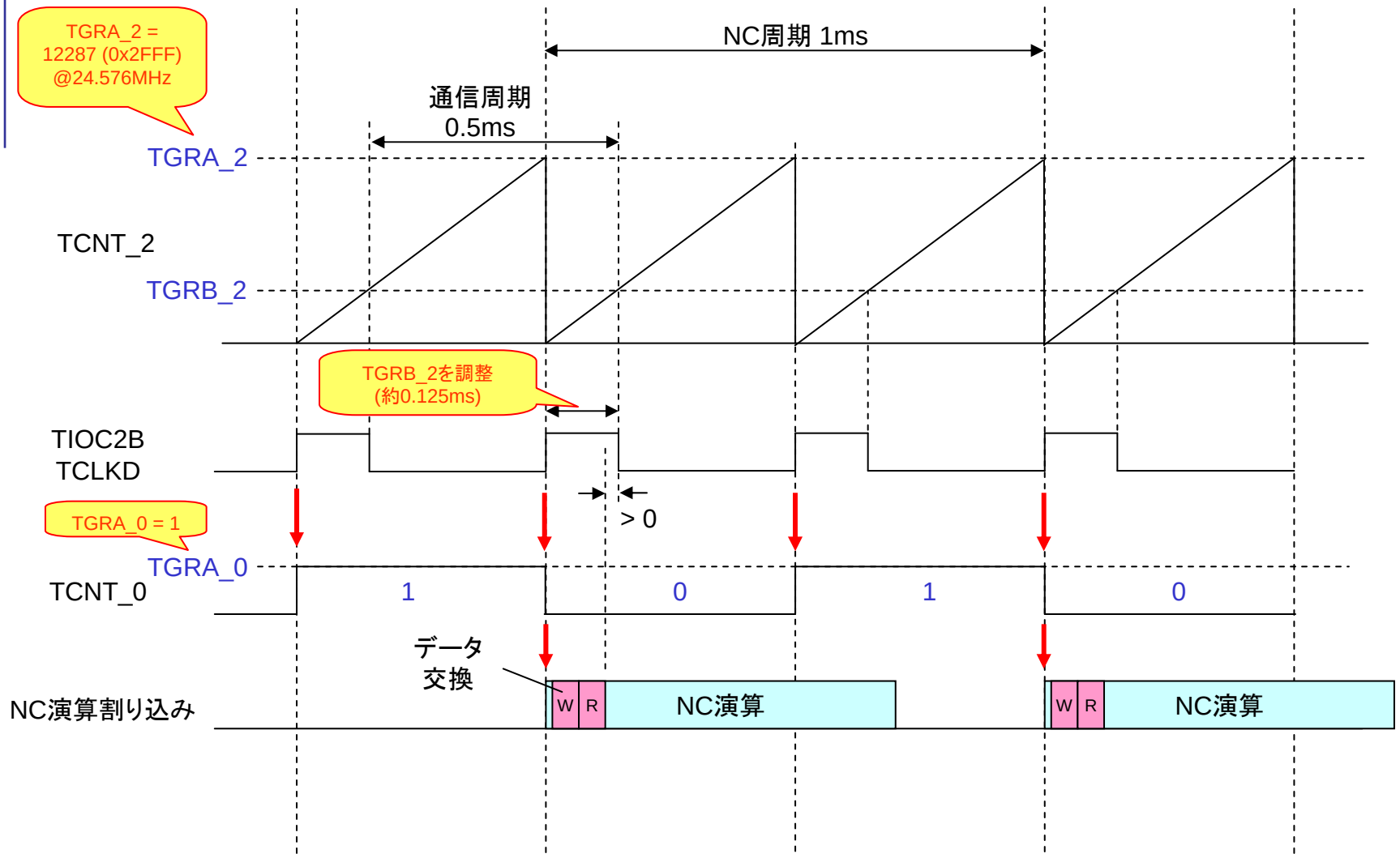
- MTU-Ch2で送信タイミング信号を生成。周期0.5msの場合、**TGRA_2 = 12287(0x2FFF)@24.576MHz**
- この信号をMTU-Ch0で分周しNC演算割り込みの起動信号を生成。周期1msの場合、**TGRA_0 = 1**
- MNM1221のXINTRXをIRQ2で受けて受信割り込みを起動。

多重割り込みの設定

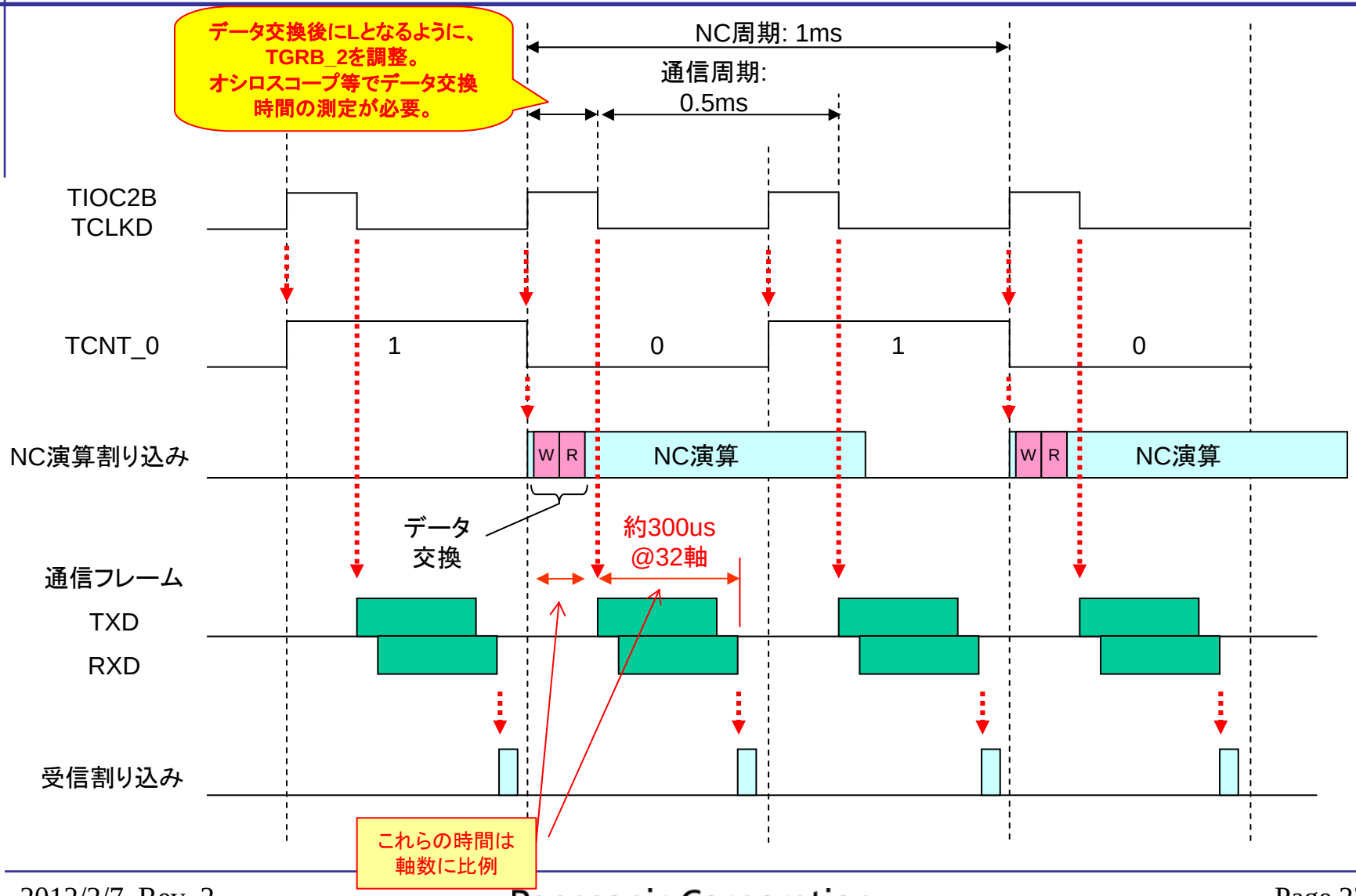
Panasonic
ideas for life

信号ソース	トリガ	優先順位	周期	処理
TGIA_0	MTU Ch0 コンペアマッチ	-	1ms	- 通信データ交換 - NC演算
/IRQ2	受信完了	TGIA_0 より高く 設定	0.5ms	- 通信ステータス確認 - 受信メモリバンク切替

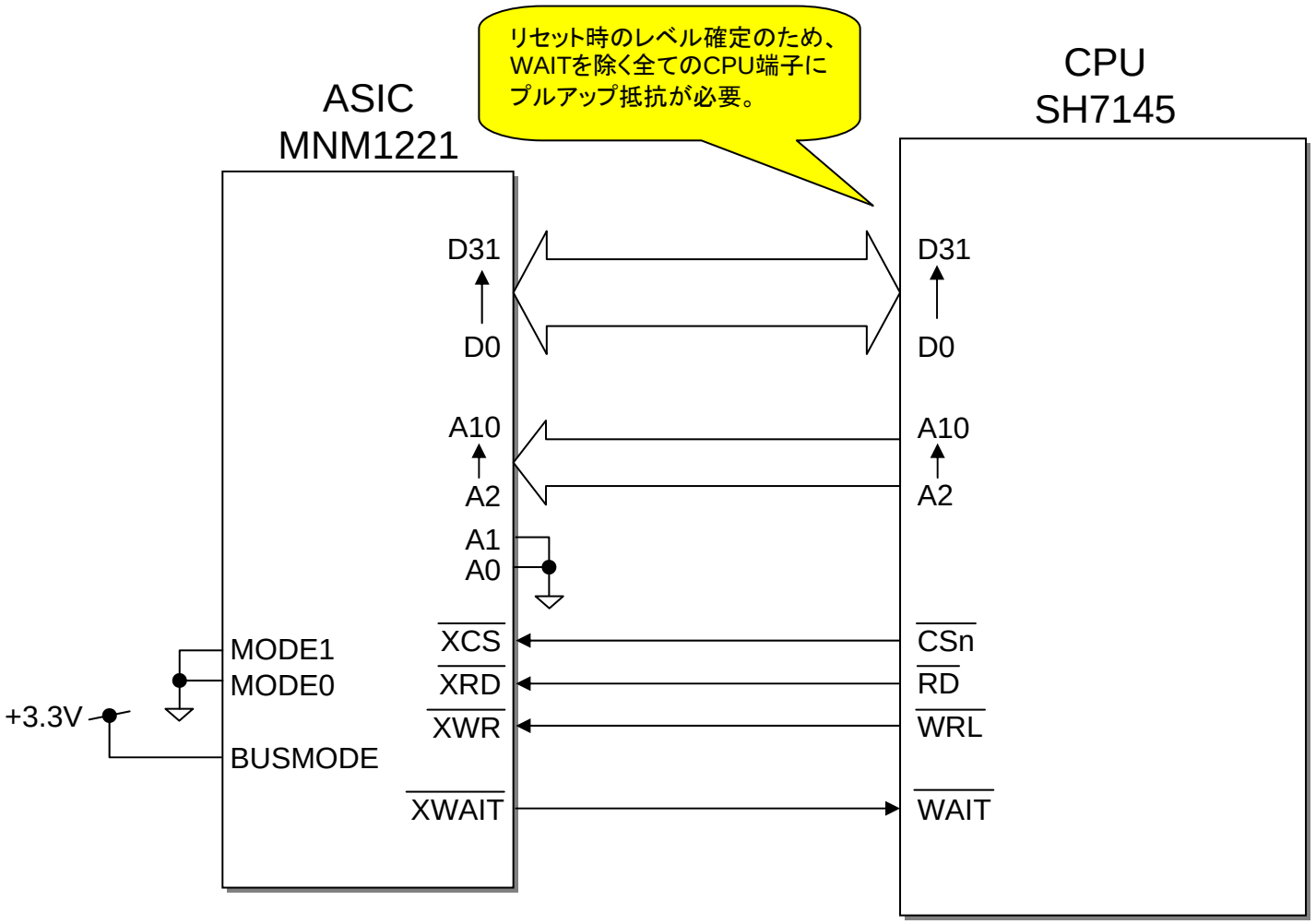
タイムチャート 1


TGRA_2 =
12287 (0x2FFF)
@24.576MHz
NC周期 1ms
通信周期
0.5ms
TGRA_2
TCNT_2
TGRB_2
TGRB_2を調整
(約0.125ms)
TIOC2B
TCLKD
> 0
TGRA_0 = 1
TGRA_0
TCNT_0
1
0
1
0
データ
交換
NC演算割り込み
W
R
NC演算
W
R
NC演算

タイムチャート 2

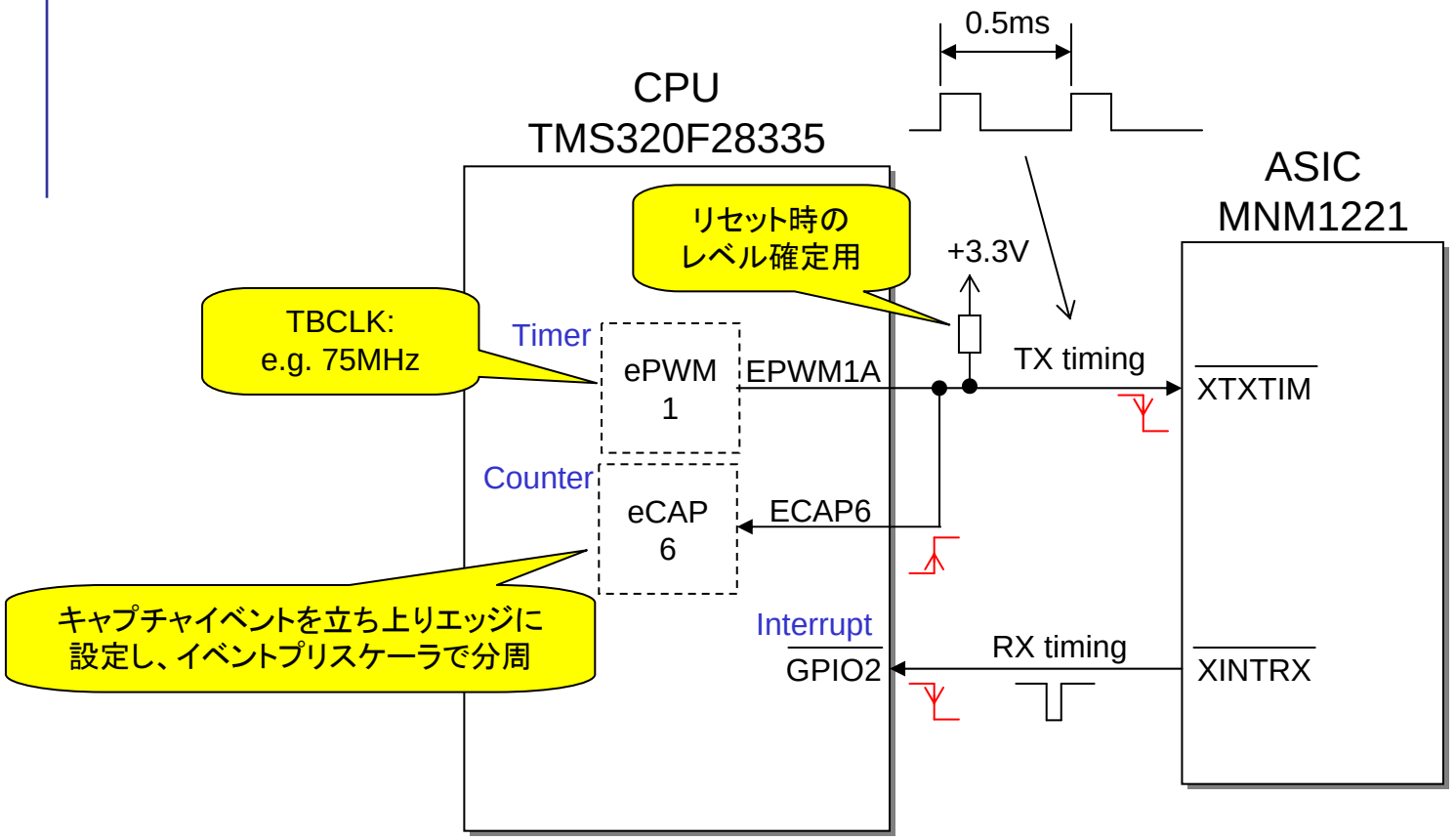

データ交換後にLとなるように、
TGRB_2を調整。
オシロスコープ等でデータ交換
時間の測定が必要。
NC周期: 1ms
通信周期:
0.5ms
TIOC2B
TCLKD
TCNT_0
1
0
1
0
NC演算割り込み
W
R
NC演算
W
R
NC演算
データ
交換
約300us
@32軸
通信フレーム
TXD
RXD
受信割り込み
これらの時間は
軸数に比例

バス接続


リセット時のレベル確定のため、
WAITを除く全てのCPU端子に
プルアップ抵抗が必要。
ASIC
MNM1221
CPU
SH7145
D31
D0
A10
A2
A1
A0
MODE1
MODE0
+3.3V
BUSMODE
XCS
XRD
XWR
XWAIT
CSn
RD
WRL
WAIT

TMS320F28335 (TI) との接続例

タイミング回路



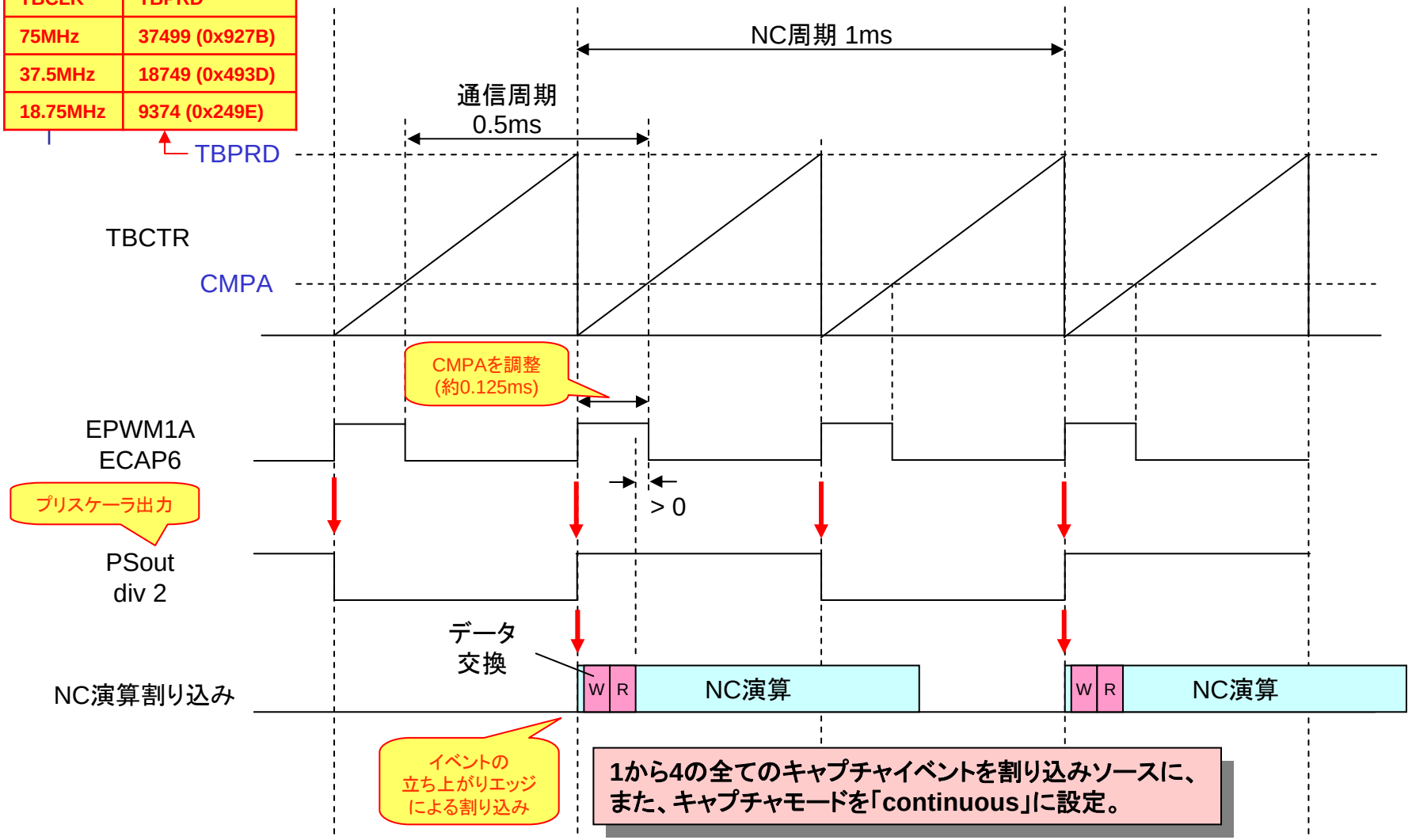
- ePWM1で送信タイミング信号を生成。周期0.5msの場合、**TBPRD = 37499(0x927B)@TBCLK 75MHz**
- この信号をeCAP6で分周しNC演算割り込みの起動信号を生成。周期1msの場合、**PRESCALE = 1**
- MNM1221のXINTRXをGPIO2による外部割り込みで受けて受信割り込みを起動。

多重割り込みの設定

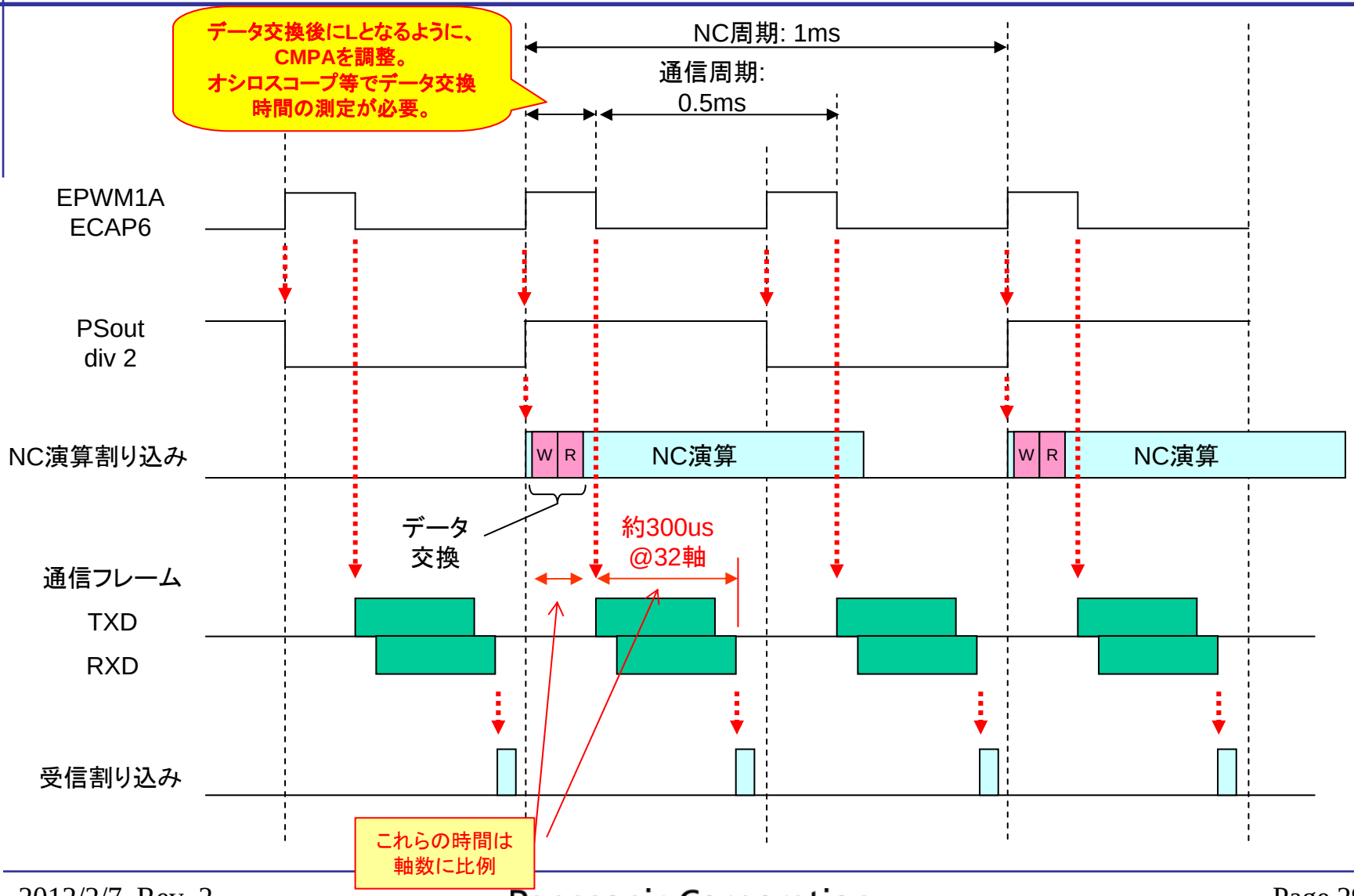
信号ソース	トリガ	優先順位	周期	処理
ECAP6 INT	eCAP6 キャプチャイベント	-	1ms	- 通信データ交換 - NC演算
GPIO2 経由 XINT	受信完了	ECAP6 より高く 設定	0.5ms	- 通信ステータス確認 - 受信メモリバンク切替

タイムチャート 1

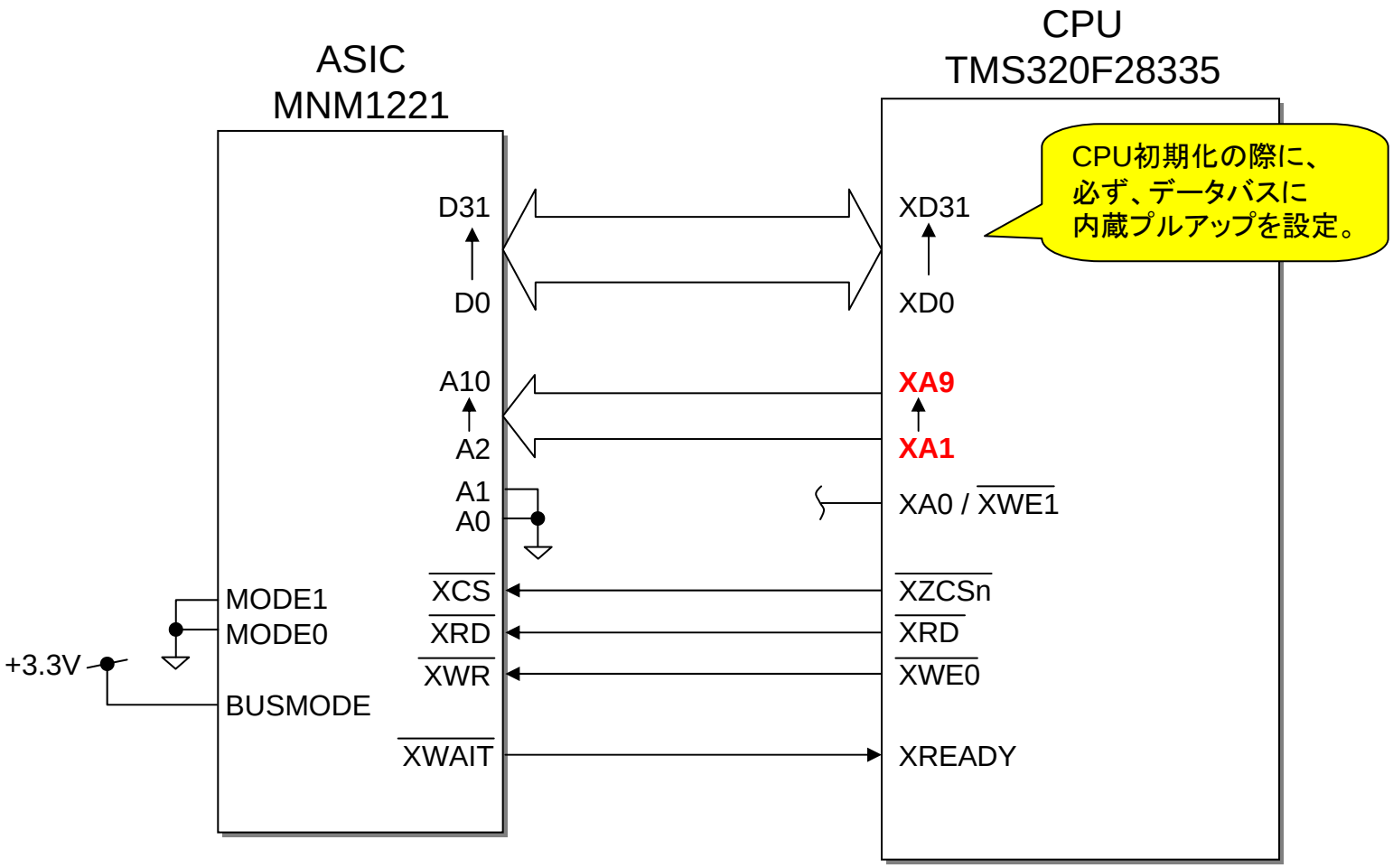
TBCLK	TBPRD
75MHz	37499 (0x927B)
37.5MHz	18749 (0x493D)
18.75MHz	9374 (0x249E)



タイムチャート 2


データ交換後にLとなるように、
CMPAを調整。
オシロスコープ等でデータ交換
時間の測定が必要。
NC周期: 1ms
通信周期:
0.5ms
EPWM1A
ECAP6
PSout
div 2
NC演算割り込み
W
R
NC演算
W
R
NC演算
データ
交換
約300us
@32軸
通信フレーム
TXD
RXD
受信割り込み
これらの時間は
軸数に比例

バス接続



注: TMS320F28335のアドレスバスは16bit単位。

アドレスに関する注意点

MNM1221が8bit単位アドレスなのに対して、TMS320F28335は16bit単位です。
このため、アドレスバスの接続の際は1bit分ずらす必要があります。
例えば、A2(MNM1221) は XA1(TMS320F28335)に接続してください。

また、サンプルコードにおいても、アドレス定義を下記のように変更してください。

mnm1221_m.h

```
/*** IMPORTANT!!! ***/
/* You must modify the following definition a
/*-----------------------------------------
/* 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)
```

使用するXZCSに応じて変更
XZCS0: 0x00004000
XZCS6: 0x00100000
XZCS7: 0x00200000

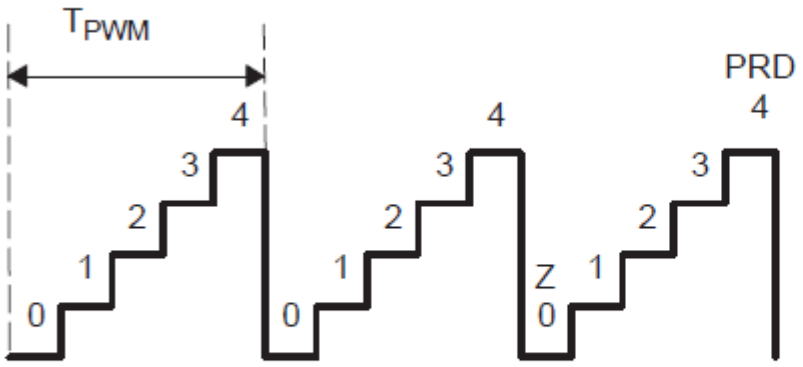
この行を削除

各アドレス定義を次のように変更
(ADDR_MNM1221 + (0x0000 >> 1))

TMS320F28335 設定の詳細

ePWMの周期0.5ms設定

Figure 2-3. Time-Base Frequency and Period



For Up Count and Down Count

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

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

TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)

0.5msにするための設定値:

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

eCAPの割り込み

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.

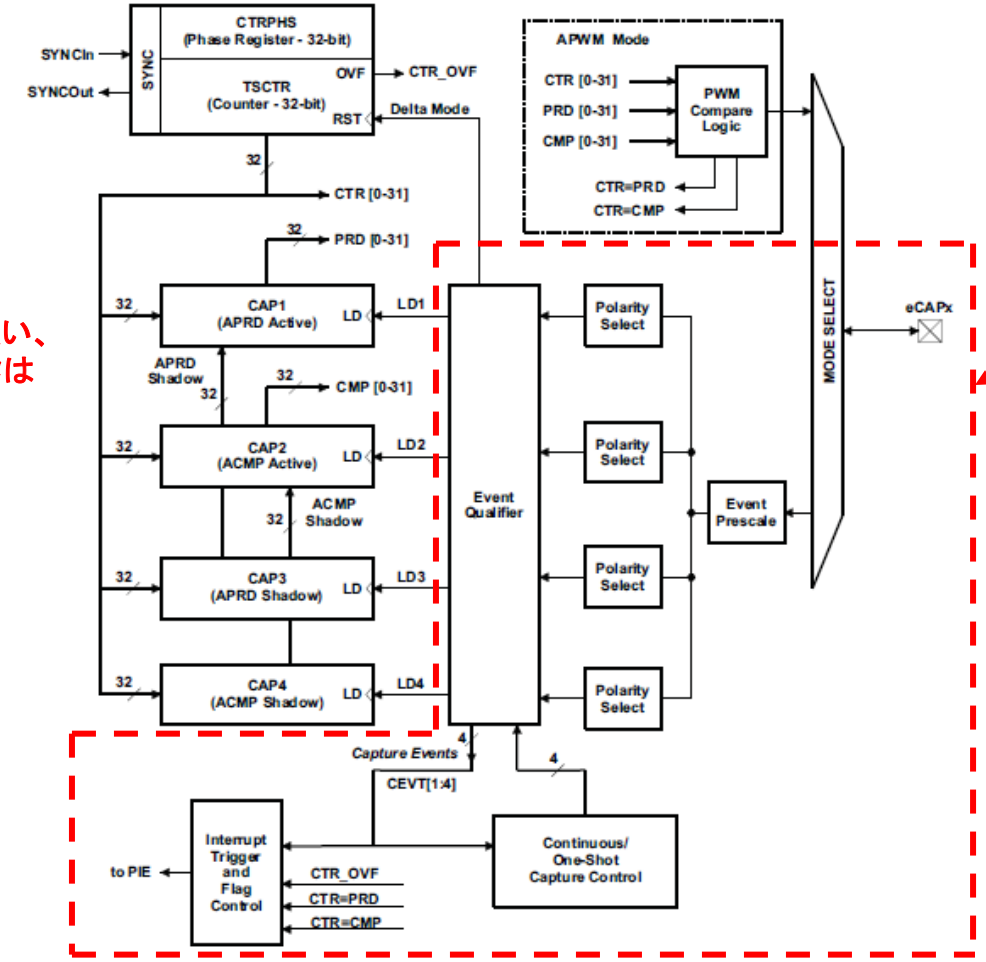


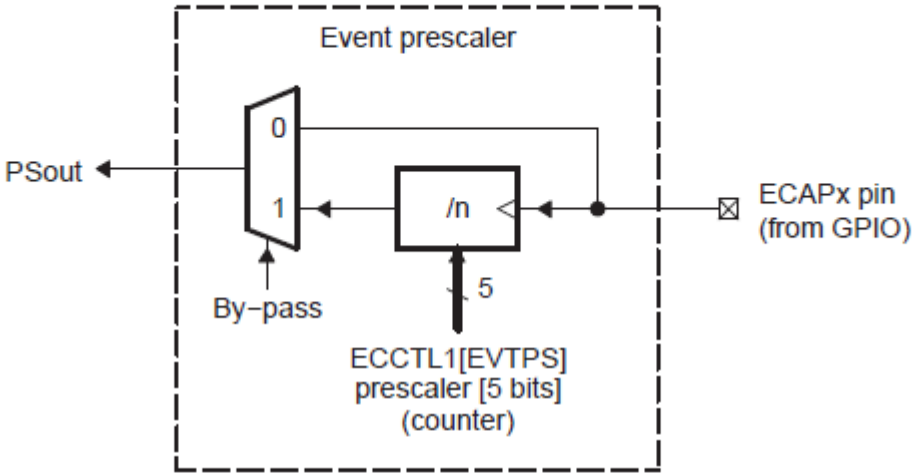
Figure 4-6. eCAP Functional Block Diagram

割り込みのみを使い、
キャプチャデータは
使いません。

使用する部分

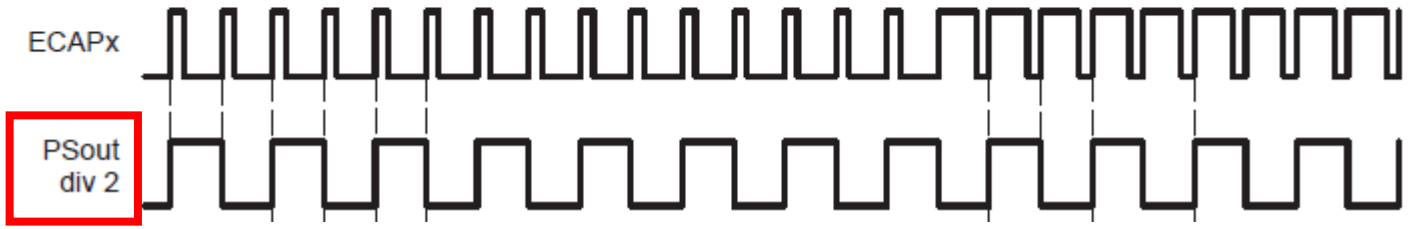
eCAP内のイベントプリスケーラ

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 レジスタ

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 レジスタ

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: • Inhibits TSCTR resets via CTR = PRD event • Inhibits shadow loads on CAP1 and 2 registers • Permits user to enable CAP1-4 register load • CAPx/APWMx pin operates as a capture input
		1	ECAP module operates in APWM mode. This mode forces the following configuration: • Resets TSCTR on CTR = PRD event (period boundary • Permits shadow loading on CAP1 and 2 registers • Disables loading of time-stamps into CAP1-4 registers • 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 レジスタ

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 レジスタ

Panasonic
ideas for life

次回の割り込みに備え、割り込みルーチンの先頭で割り込みフラグをクリアする必要があります。

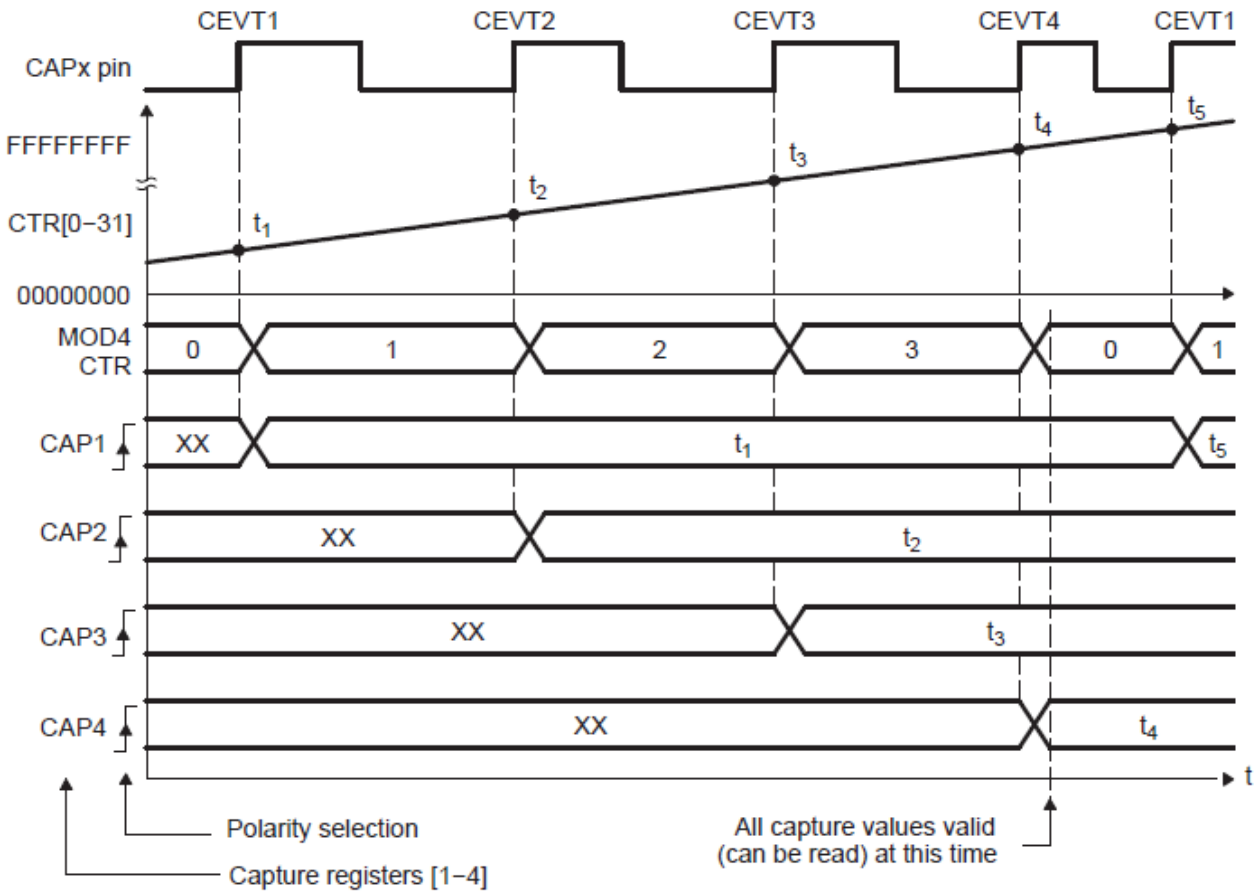
Table 11. ECAP Interrupt Clear Register (ECCLR) Field Descriptions

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.

キャプチャ割り込みのシーケンス

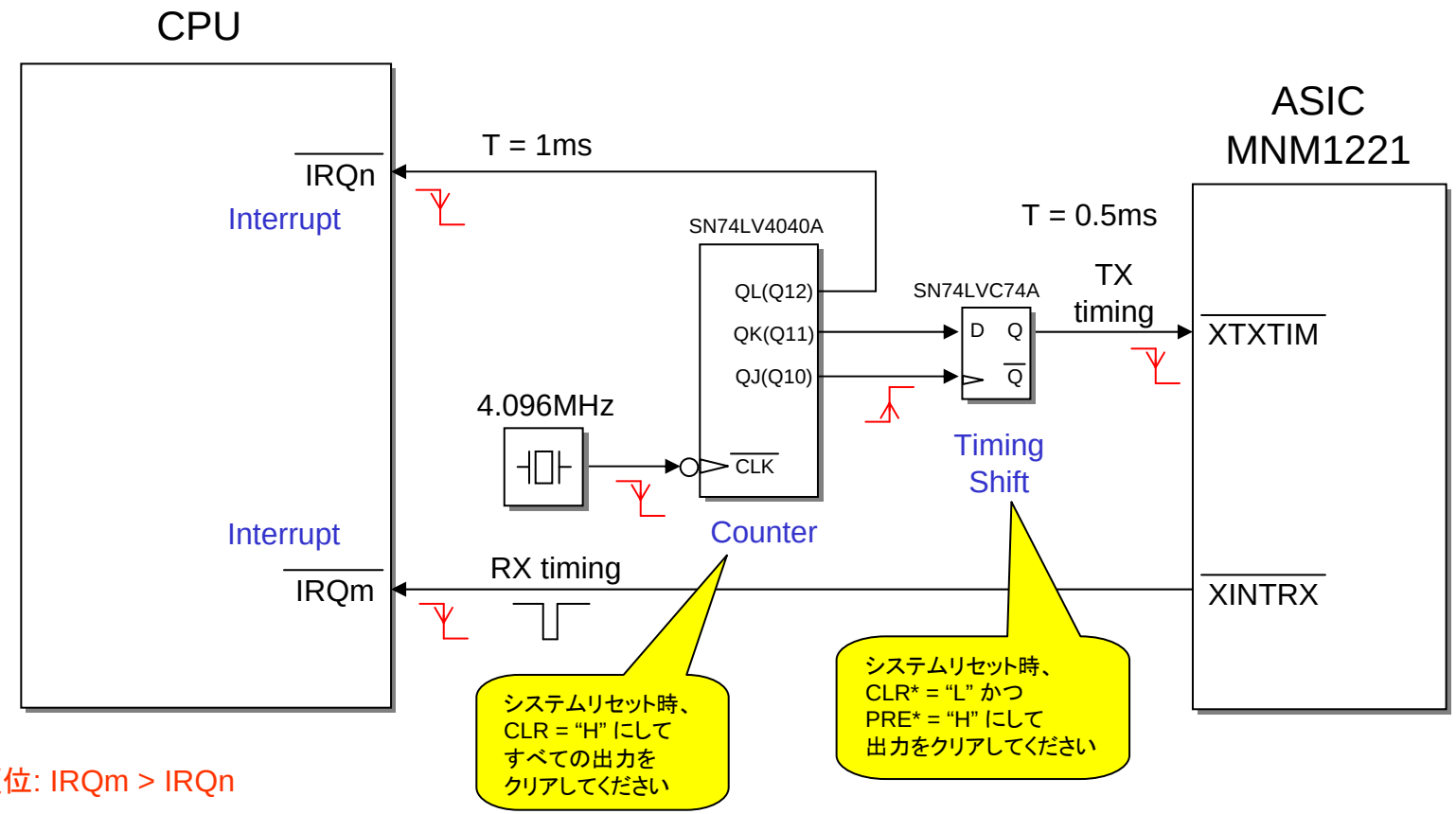
割り込み発生順序: CEVT1 → CEVT2 → CEVT3 → CEVT4 → CEVT1 → …

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



タイマを内蔵しないCPUとの接続例

タイミング回路

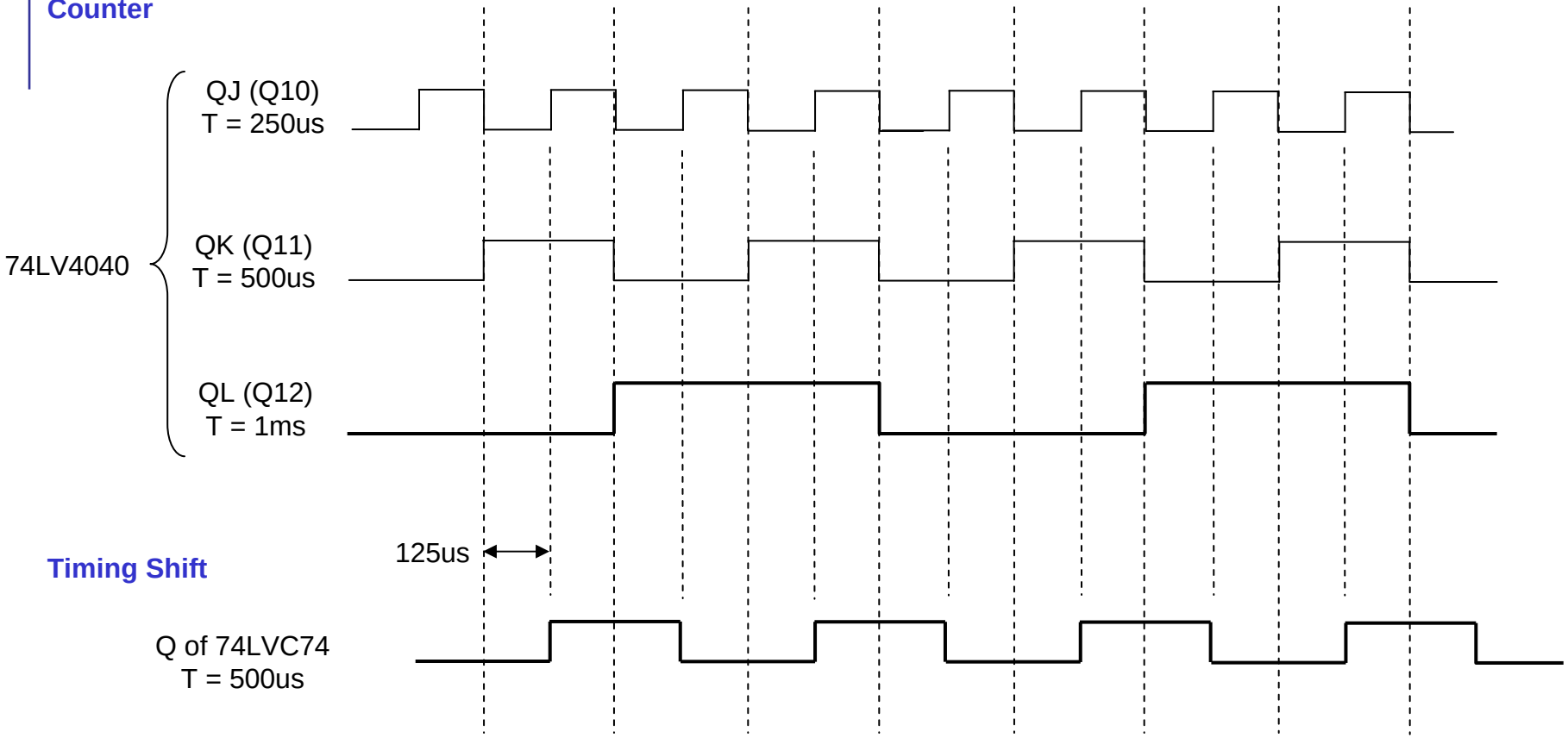


優先順位: IRQm > IRQn

タイムチャート 1

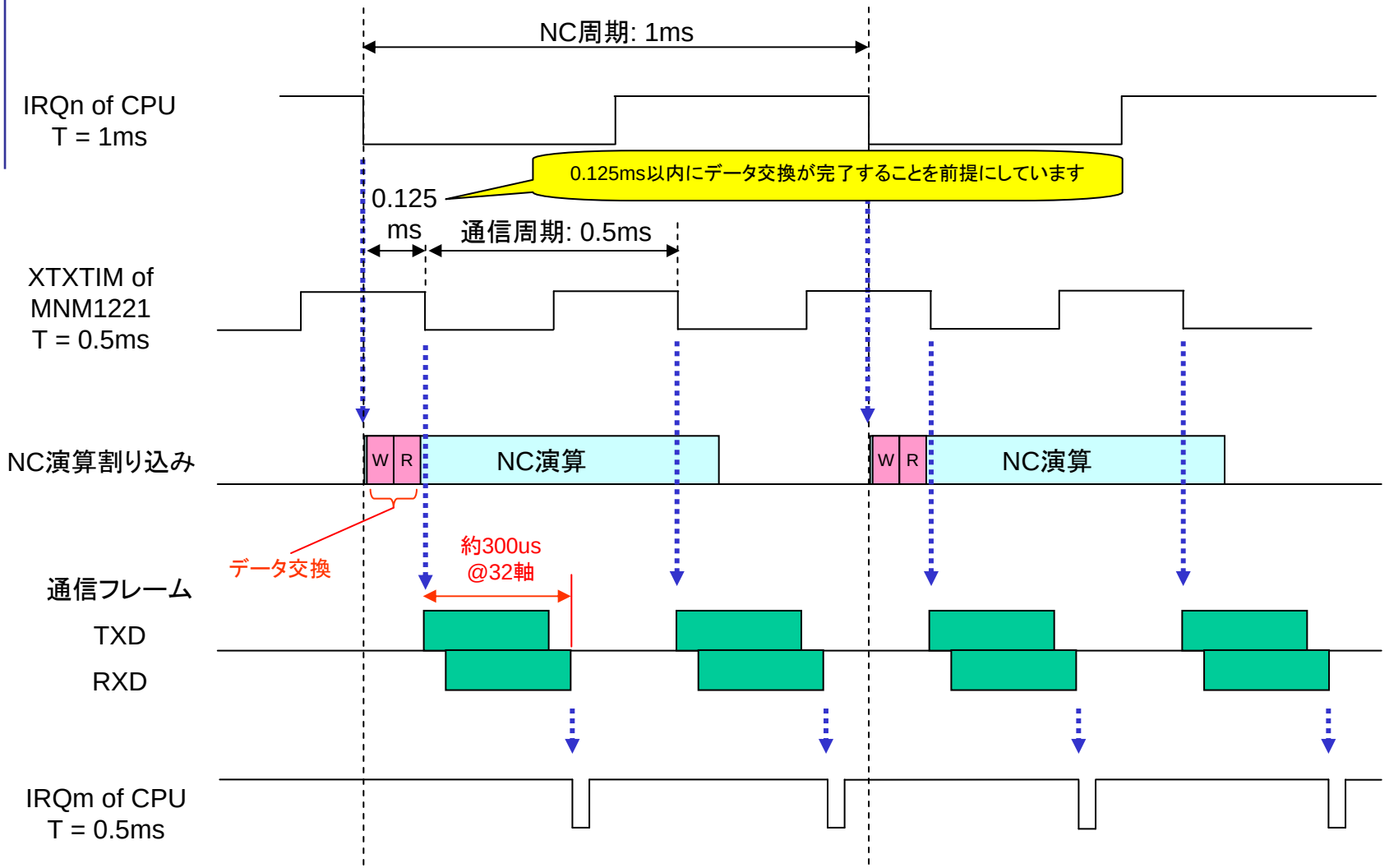
Panasonic
ideas for life

Counter



Timing Shift

タイムチャート 2


NC周期: 1ms
IRQn of CPU
T = 1ms
0.125ms以内にデータ交換が完了することを前提にしています
0.125
ms
通信周期: 0.5ms
XTXTIM of
MNM1221
T = 0.5ms
NC演算割り込み
W
R
NC演算
W
R
NC演算
データ交換
約300us
@32軸
通信フレーム
TXD
RXD
IRQm of CPU
T = 0.5ms

サンプルコードの配置

サンプルコードの配置

これらの関数はご自身で作成してください。

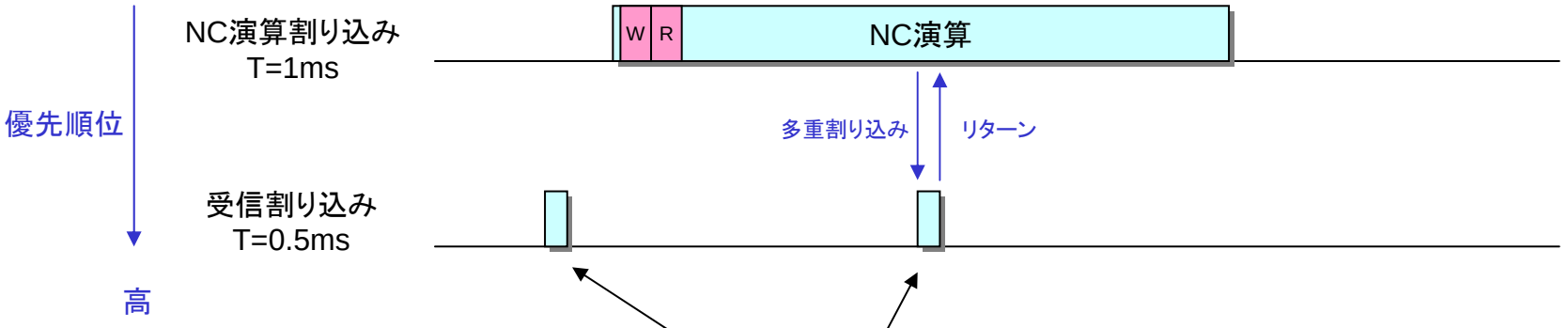
サンプルコード

関数 “ctrl_mnm1221_m()”
- MNM1221の動作制御
- 通信データ交換

バッファ “rx_buf[]”からの
レスポンスデータ読み出し

モーションプロファイル生成

バッファ “tx_buf[]”への
コマンドデータ書き込み

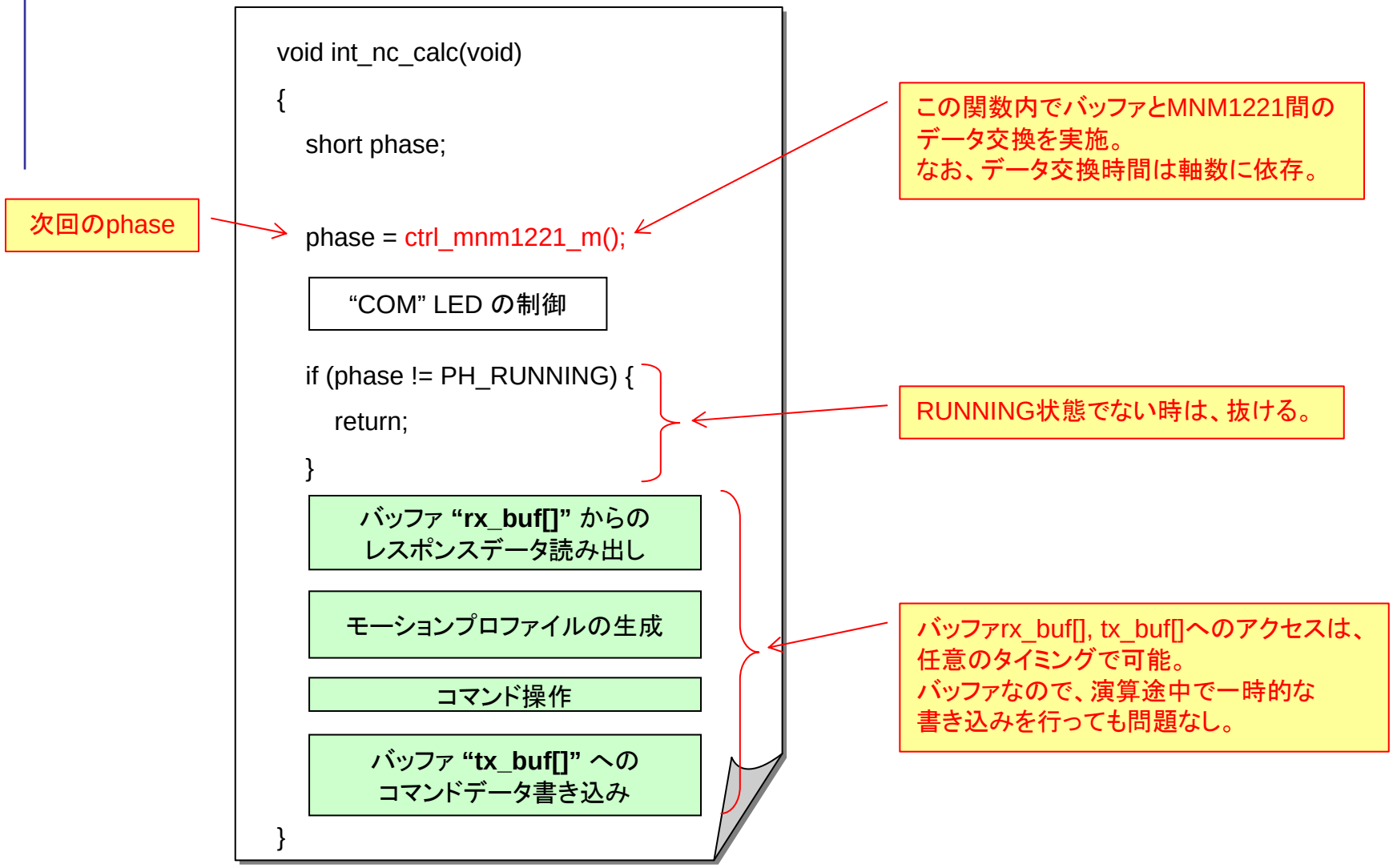


サンプルコード

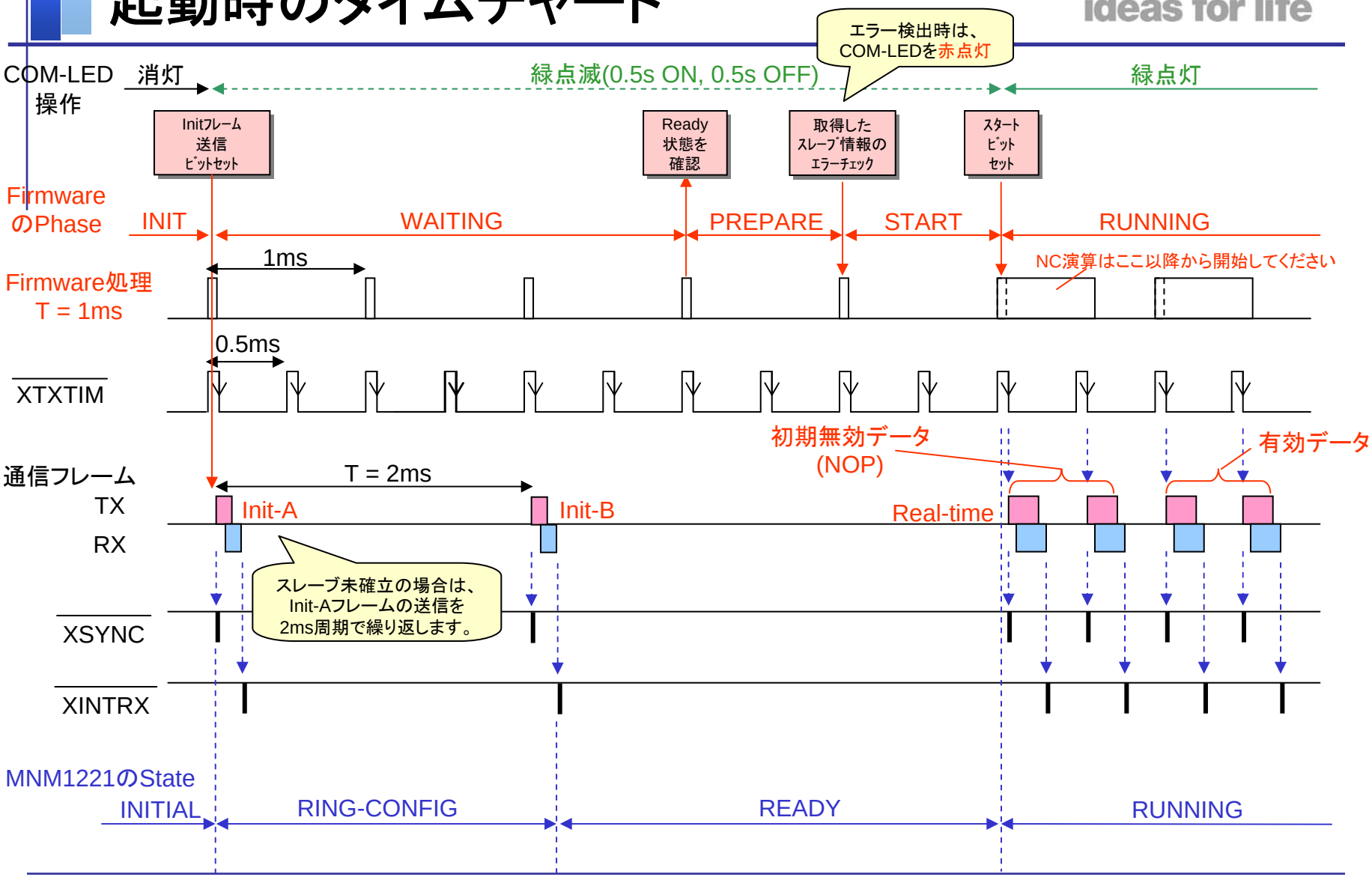
関数 “int_rx_mnm1221()”
- 通信ステータスの確認
- 受信メモリバンク切替

注意:
データ交換と受信割り込みが競合する可能性がある場合、
データ交換中は受信割り込みを禁止してください。

NC演算割り込みへの配置例



起動時のタイムチャート



サンプルコード使用上の注意点

送受信データバッファの構造(32bit)

Panasonic
ideas for life

下図は32ビットバスアクセス時における tx_buf[] もしくは rx_buf[] 構造体配列の1要素を示します。

		Bit 31 – Bit 24	Bit 23 – Bit 16	Bit 15 – Bit 8	Bit 7 – Bit 0
tx_buf[] もしくは rx_buf[] の1要素	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 ~ F」は、16バイトで構成されるデータブロックの内容と対応しています。

送受信データバッファの構造(16bit)

Panasonic
ideas for life

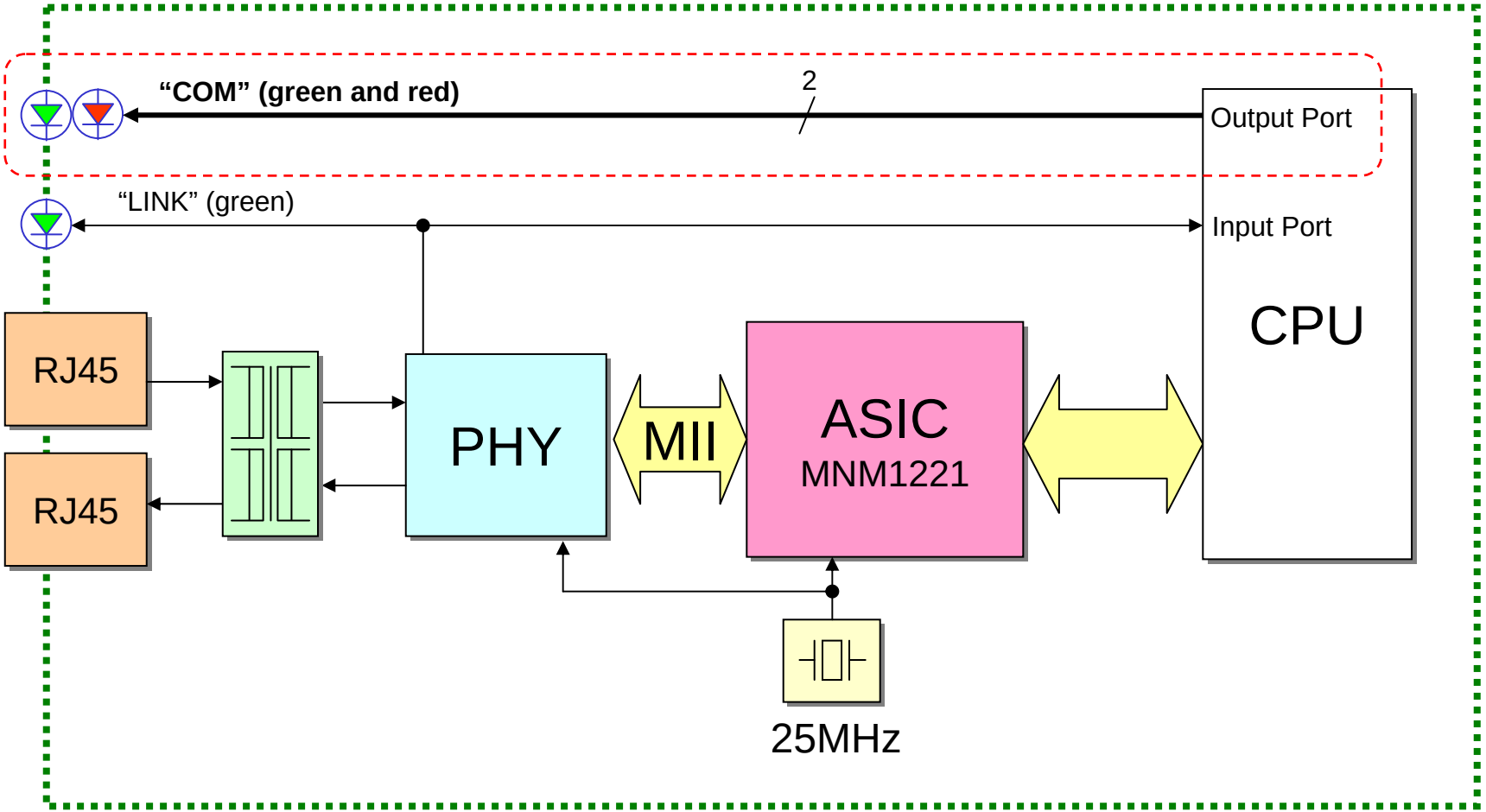
下図は16ビットバスアクセス時における tx_buf[] もしくは rx_buf[] 構造体配列の１要素を示します。

tx_buf[]
もしくは
rx_buf[]
の１要素

	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 ~ F」は、16バイトで構成されるデータブロックの内容と対応しています。

通信ステータスLED


“COM” (green and red)
2
Output Port
“LINK” (green)
Input Port
CPU
RJ45
RJ45
PHY
MII
ASIC
MNM1221
25MHz

“COM” LED操作

“COM” LED（赤、緑の2色発光）は下表のように制御してください。

通常時

ctrl_mnm1221_m() の戻り値	“COM” LED 操作
PH_INIT	消灯
PH_WAITING	緑点滅 (0.5s ON, 0.5s OFF)
PH_PREPARE	
PH_START	
PH_RUNNING	緑点灯

エラー検出時

エラーの内容	“COM” LED 操作
RUNNING状態でのタイムアウト	赤点滅 (0.5s ON, 0.5s OFF)
スレーブ情報のミスマッチ (重複MAC-IDなど)	赤点灯

注:
- 赤点灯は、リセットしないとエラー解除ができないことを示します。
- 赤と緑はいずれか一方のみがONし、同時にONすることはありません。

サイクリックポジションI/Fの概要

データブロック

Panasonic ideas for life

コマンド (TX)

通常 0x20 (Position)

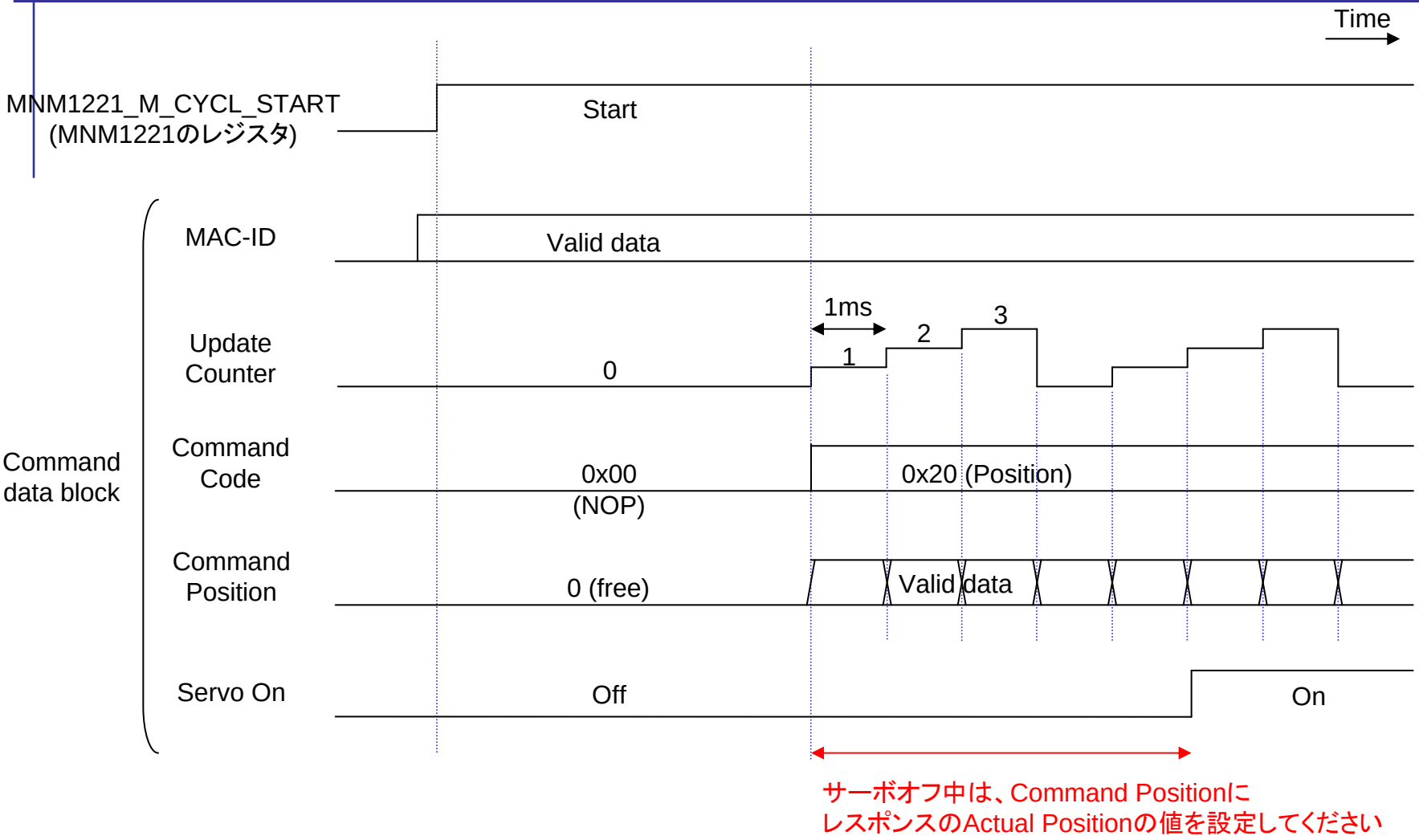
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte0	0 (CMD)	Update Counter		MAC-ID				
byte1	0	Command Code						
byte2	Servo On	0	0	Gain SW	TL SW	HM Ctrl	0	0
byte3	Hard Stop	SMT Stop	Pause	0	SL SW	0	EX-OUT2	EX-OUT1
byte4	Command Position							Low byte
byte5								Low Middle byte
byte6								High Middle 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

レスポンス (RX)

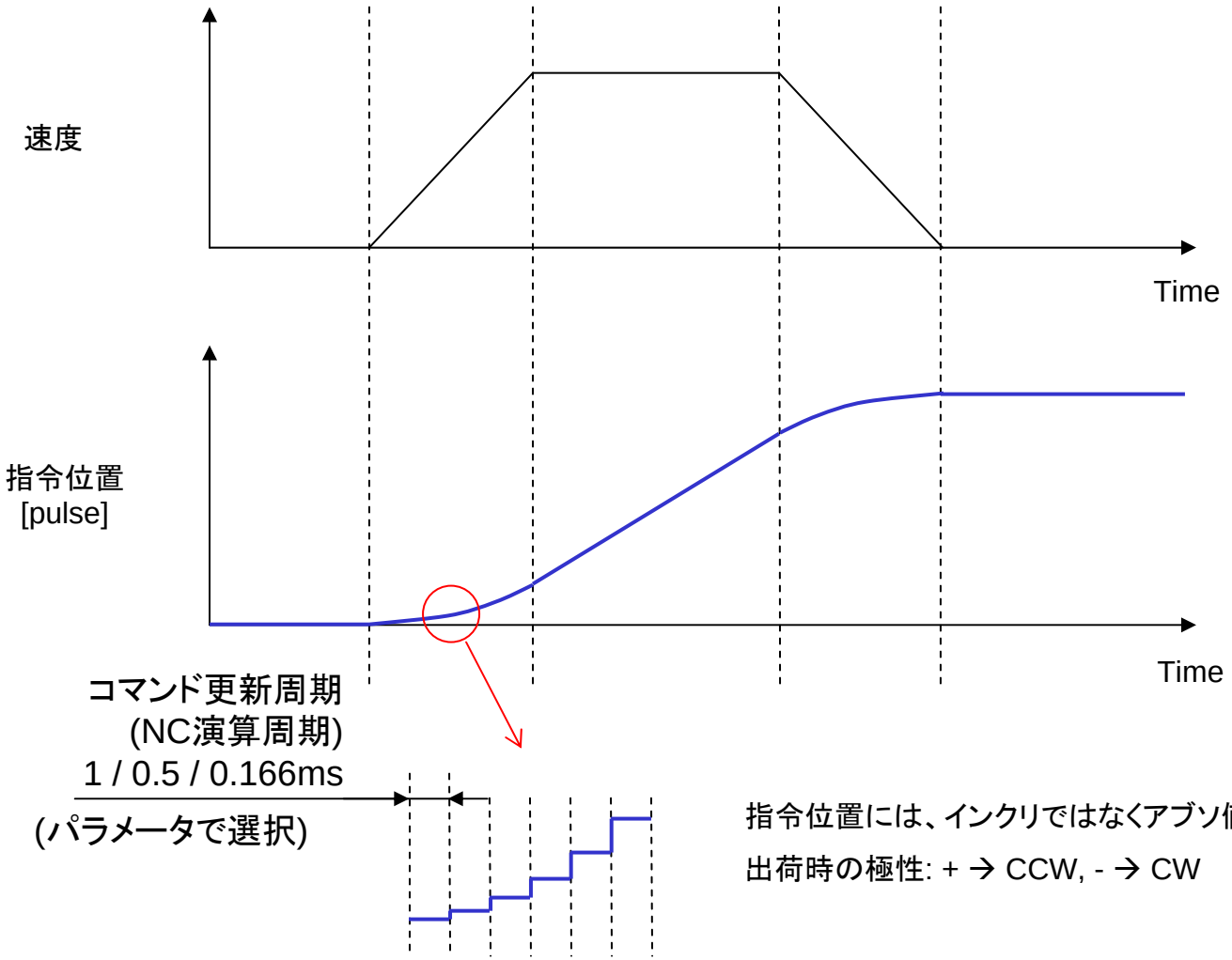
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte0	1 (RSP)	Update Counter Echo		Actual MAC-ID				
byte1	CMD Error	Command Code Echo						
byte2	Servo Act.	Servo Ready	Alarm	Warn.	TL	HM Comp.	In Prog.	In Pos.
byte3	SI-MO5	SI-MO4	EXT 3	EXT 2	SI-MO1	Home	POT	NOT
byte4	Actual Position							Low byte
byte5								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

注：少なくとも赤で示した部分のサポートが必要です。

起動時のコマンド


Time
MNM1221_M_CYCL_START
(MNM1221のレジスタ)
Start
MAC-ID
Valid data
Update
Counter
0
1ms
1
2
3
Command
data block
Command
Code
0x00
(NOP)
0x20 (Position)
Command
Position
0 (free)
Valid data
Servo On
Off
On
サーボオフ中は、Command Positionに
レスポンスのActual Positionの値を設定してください

サイクリックポジションI/F



指令位置には、インクリではなくアブソ値を用いてください。

出荷時の極性: + → CCW, - → CW

プロファイルポジションI/Fの概要

コマンド

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

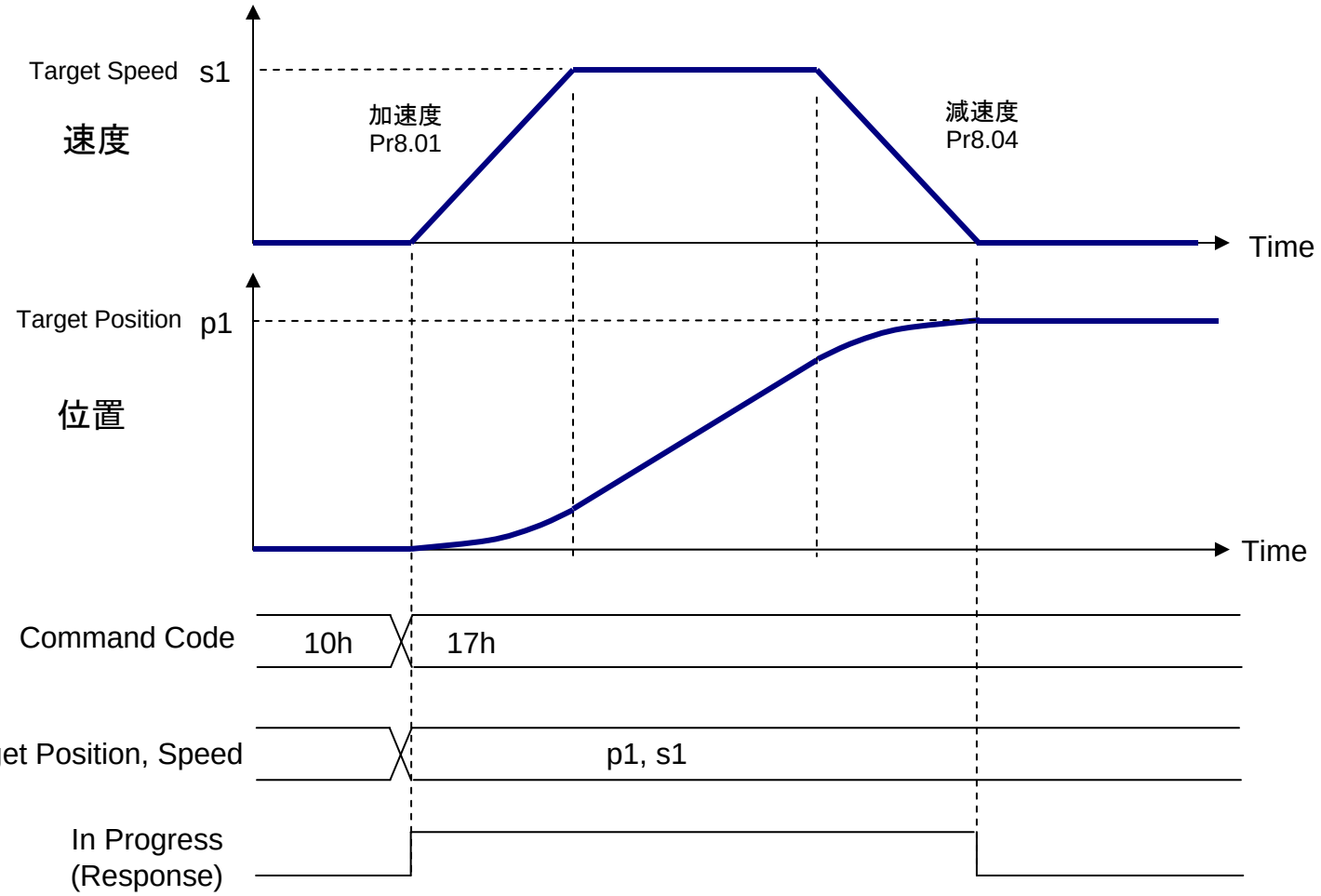
レスポンス

Panasonic
ideas for life

	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								

起動

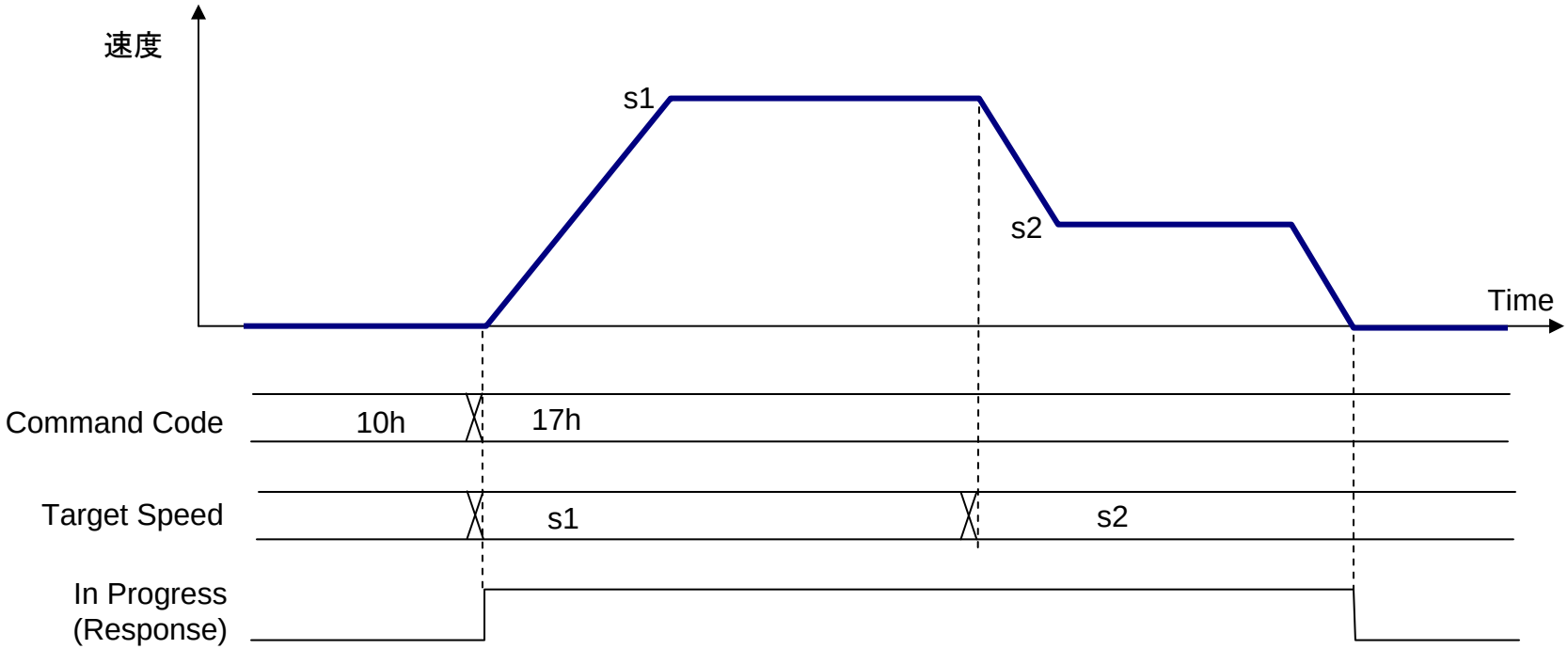
“In Progress”が0の時にコマンドコードを10hから17hに変更すると起動します。
加速度と減速度は事前にパラメータで設定しておき、Abs/Incは起動時にタイプコードで指定します。



動作中の目標速度変更

Panasonic
ideas for life

動作中（“In Progress”が1）に目標速度を変更できます。
動作中に目標速度を0に変更、もしくは、Pauseを1にした場合には、停止しても“In Progress”は1を保持します。



サンプルコードの変更

バスアクセスの定義

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

MNM1221が配置されているアドレス（バイト単位のアドレス値）に変更。

データバス幅が32bitの場合は削除。
16bitの場合は、そのまま残す。

変数の定義

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 */


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

高速アクセスのために、
この通信バッファはできるだけ
CPU内蔵RAMに配置して
ください。

スレーブ情報テーブル

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;
    }
}
```

テストの際、実際のシステムに応じてこれらの値を変更

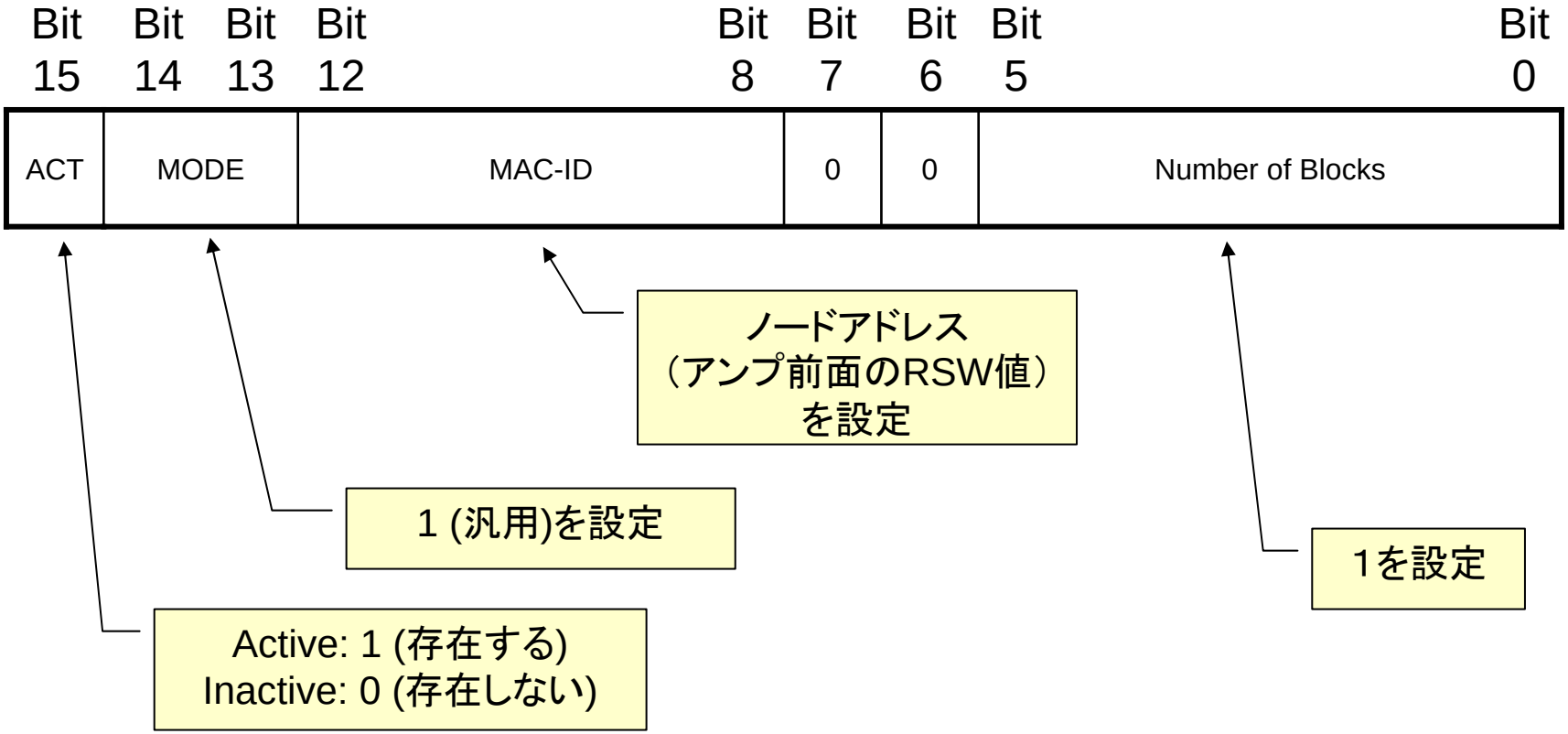
1軸で実験する場合は、これらの行をコメントに変更し、

更に、この値を1に変更

スレーブ情報テーブル (続き)

Panasonic
ideas for life

Slave Information dataの構造:



変数初期化のスタイル

ctrl_mnm1221_m() in mnm1221_m.c

変数“phase”には、必ず初期値として0を設定。
いずれかの初期化方法を選択。

```
/* 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
```

スレーブ情報のチェック処理

Panasonic
ideas for life

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;
```

READY状態でエラーが検出された場合の処理が必要。

実際のアプリケーションでは、エラーが解除されるまでは状態遷移しない処理とするのが一般的。

サイクリック伝送の開始

Panasonic
ideas for life

ctrl_mnm1221_m() in mnm1221_m.c

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

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

MNM1221の送信メモリに
初期送信データを設定するのが目的
（本来、読み出しは不要）。
この処理にてMACIDが設定されます。
初期テスト用の処理なので、基本的
にはテスト後に正規の初期化処理が
必要ですが、このまま使用する場合
には、処理の重複を避けるため、
直前のclr_mnm1221_tx_mem()を
削除してください。

データ交換関数を流用しているが、
本来、書き込みのみでよく、読み出しは不要

Running 状態

Panasonic ideas for life

ctrl_mnm1221_m() in mnm1221_m.c

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

/*****************************************************
/* This is for test. You must replace with your appli
/*-----------------------------------------------------
/*
* In actual application, the routin setting to TX but
* 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 your application */
#endif
        }
        break;
```

この部分は初期テスト用。
通常使用時は、この部分を削除し、
別途、これに相当する処理を
NC演算割り込み処理の末尾付近
（指令の演算結果が得られた後）に
配置してください。
すなわち、ctrl_mnm1221_m()内では
処理しないようにしてください。

必ず削除

データ交換処理

タイムアウトを検出した場合の処理を追加してください。また、同時に、全てのサーボにサーボオフを指令してください。（安全のため）

実際のアプリケーションでは、エラーが解除されるまでは状態遷移しない処理とするのが一般的。

Chapter 2

内部タイマを用いたシステム

システムのポイント

- 指令更新周期と通信周期を同じ時間に設定
- MNM1221のXSYNCでNC演算割り込みを起動
- RX割り込みを使わない場合には、
  “Update Counter Echo”を用いてソフトでタイムアウト検知

A5Nアンプの周期設定

Panasonic
ideas for life

指令更新周期と通信周期を同じに設定してください。
出荷設定を変更する必要があります。

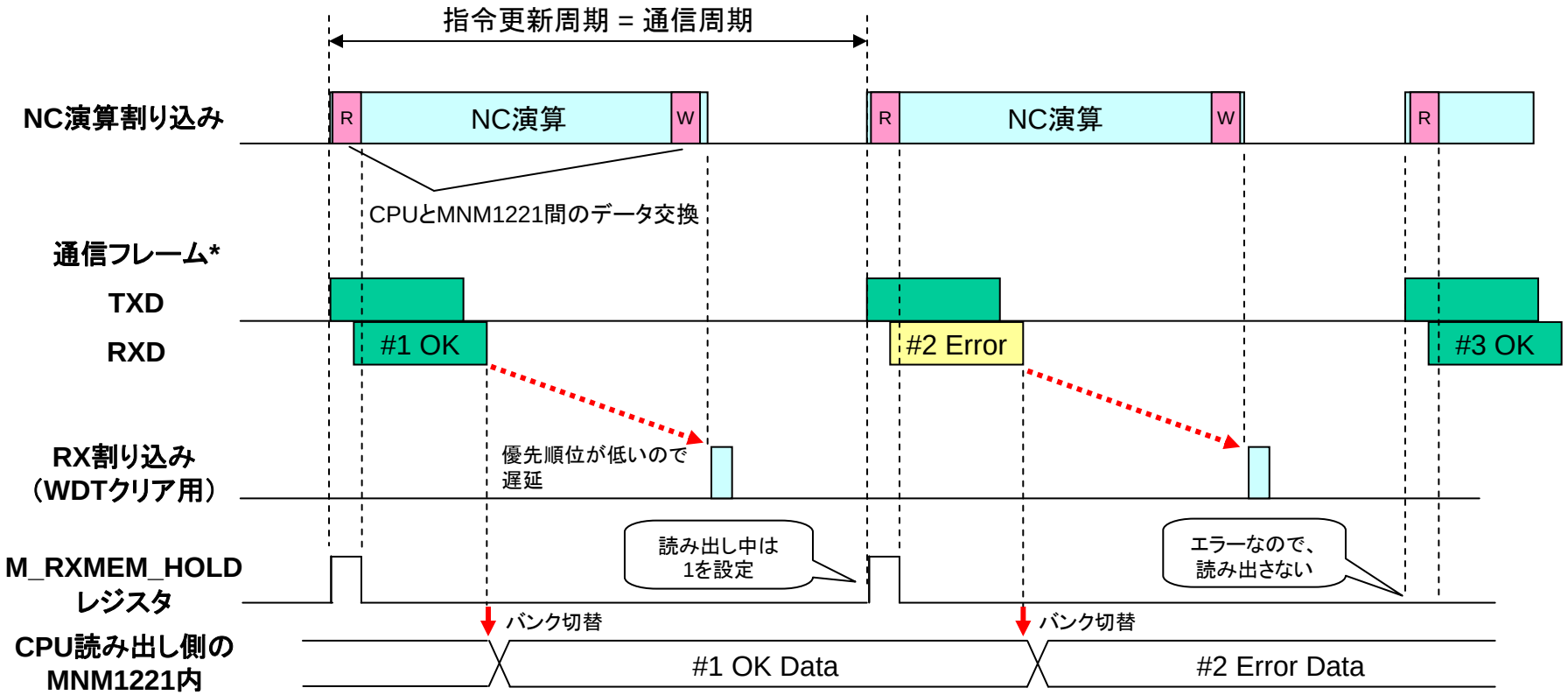
指令更新周期	通信周期	設定	
		Pr7.20	Pr7.21
1.000ms	1.000ms	6	1
1.000ms	0.500ms	3	2
0.500ms	0.500ms	3	1
0.166ms	0.166ms	1	1
0.166ms	0.083ms	0	2

	名称	設定範囲	内容
Pr7.20	RTEX通信周期	0～12	0: 0.083ms 1: 0.166ms 3: 0.5ms 6: 1.0ms その他: 設定禁止（予約）
Pr7.21	RTEX指令更新周期比	1～2	指令更新周期／通信周期 1: 1 2: 2

タイミング制御の目的

Panasonic ideas for life

下図のタイミング作成が目的

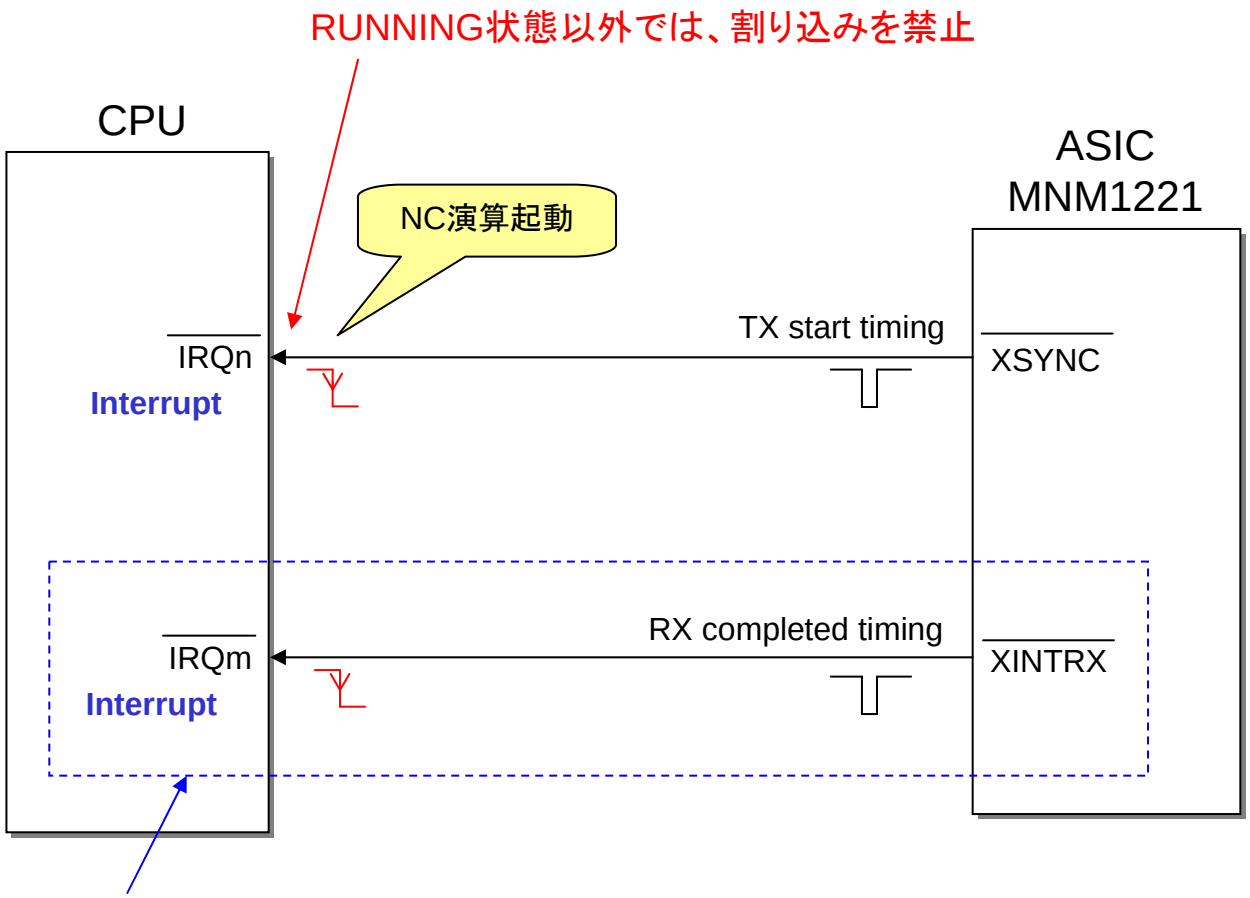


CRC異常の有無に関わらず、受信完了時にバンク切替。
NC演算では、受信データがOKの時にM_RXMEM_HOLDを1にして受信データを読み出し。

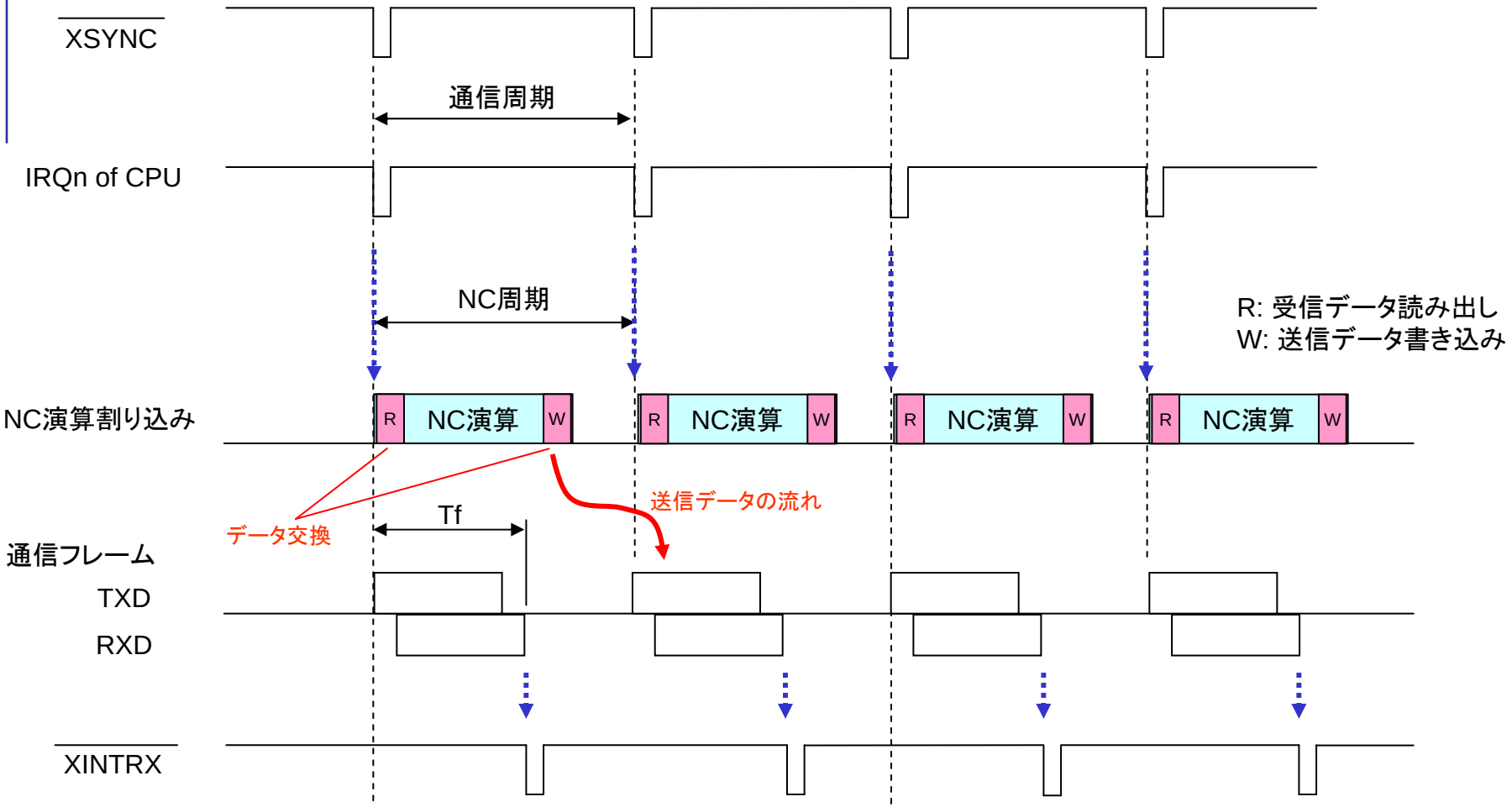
•1つのフレームには全スレーブノードのデータが含まれており、フレームの長さは接続されているノードの数に依存します。

組み込みCPUとの接続例

タイミング回路

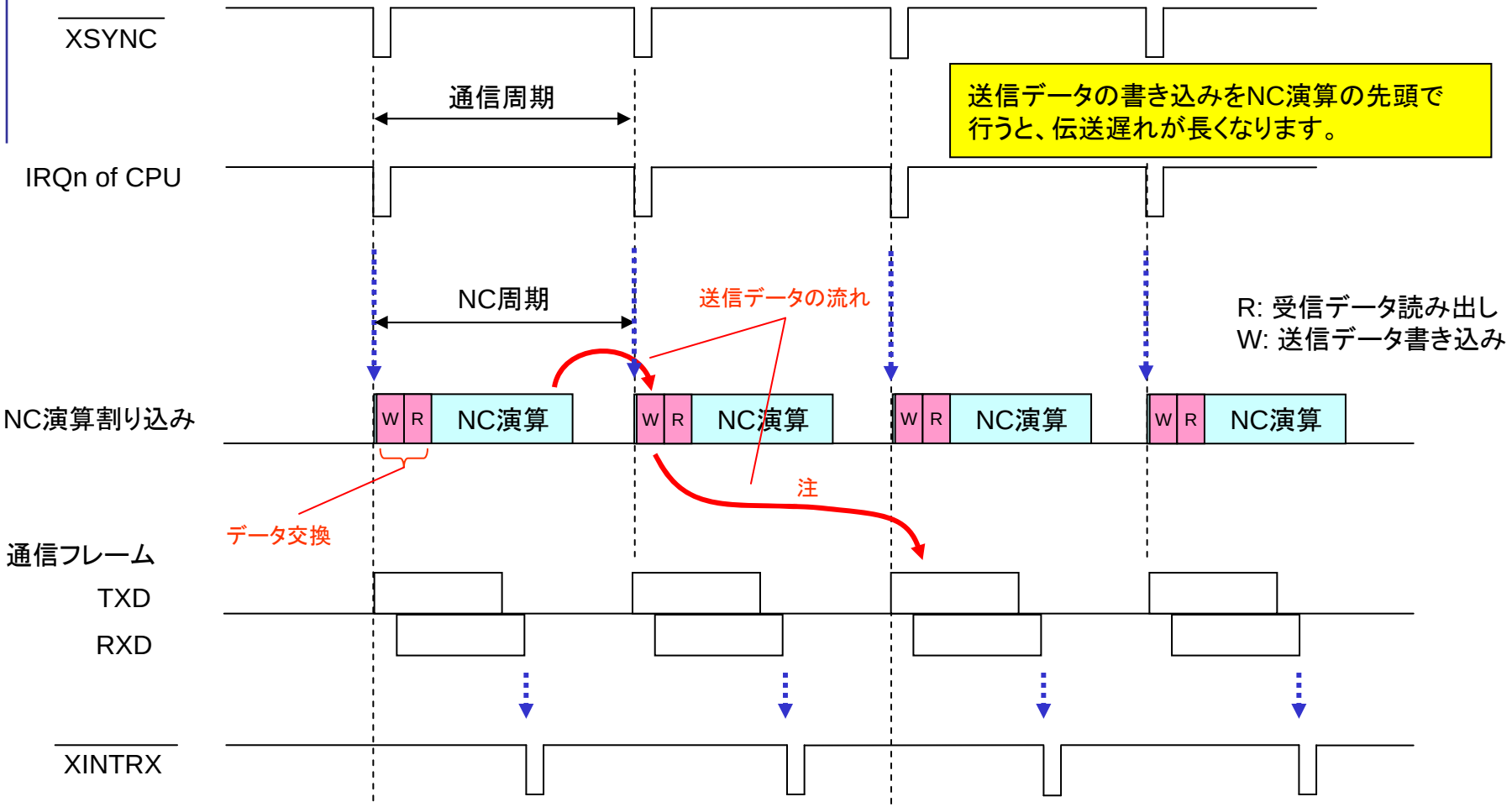

RUNNING状態以外では、割り込みを禁止
CPU
ASIC
MNM1221
NC演算起動
TX start timing
IRQn
Interrupt
XSYNC
RX completed timing
IRQm
Interrupt
XINTRX
受信割り込みを使わない場合は、Update Counterのエコーバックを用いてタイムアウト検知

推奨タイミング



Tf: 約11us@1軸 ～ 300us@32軸

好ましくないタイミング



注:
XSYNCの立ち下りエッジ時点ですでにフレーム送信を開始しているため、そこで送信メモリのバンク切り替えを指示しても
実際の切り替えは送信が完了するまで保留されます。このため、送信タイミングは 1通信周期分、遅れます。

RX割り込みを使わない場合の
タイムアウト検知例

Update Counter

Panasonic ideas for life

コマンド (TX)

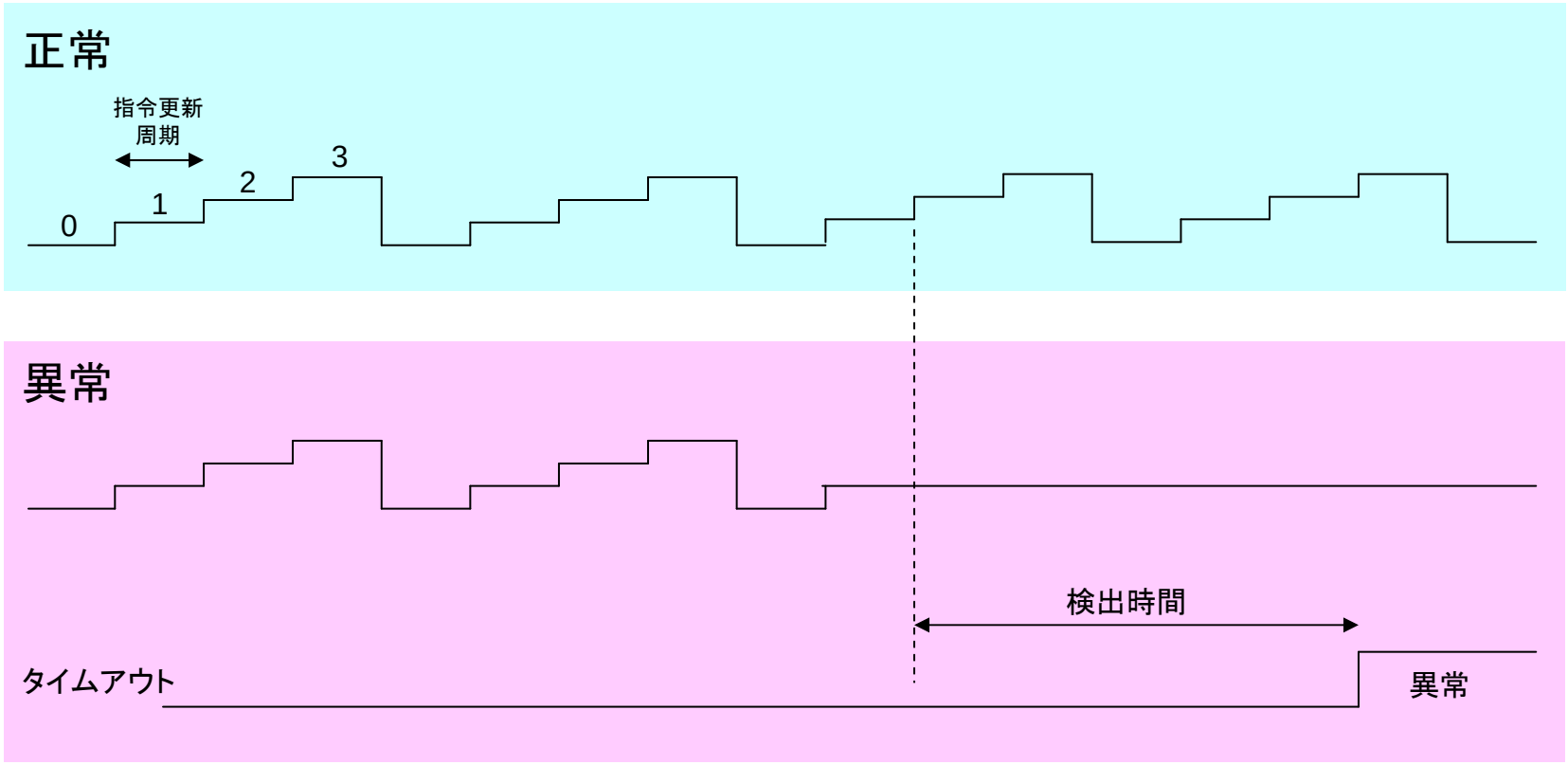
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte0	0 (CMD)	Update Counter		MAC-ID				
byte1	0	Command Code						
byte2	Servo On	0	0	Gain SW	TL SW	HM Ctrl	0	0
byte3	Hard Stop	SMT Stop	Pause	0	SL SW	0	EX-OUT2	EX-OUT1
byte4	Command Position							Low byte
byte5								Low Middle byte
byte6								High Middle 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

レスポンス (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

タイムアウト検知

“Update Counter Echo”が一定期間連続して変化しない場合にタイムアウトと判定

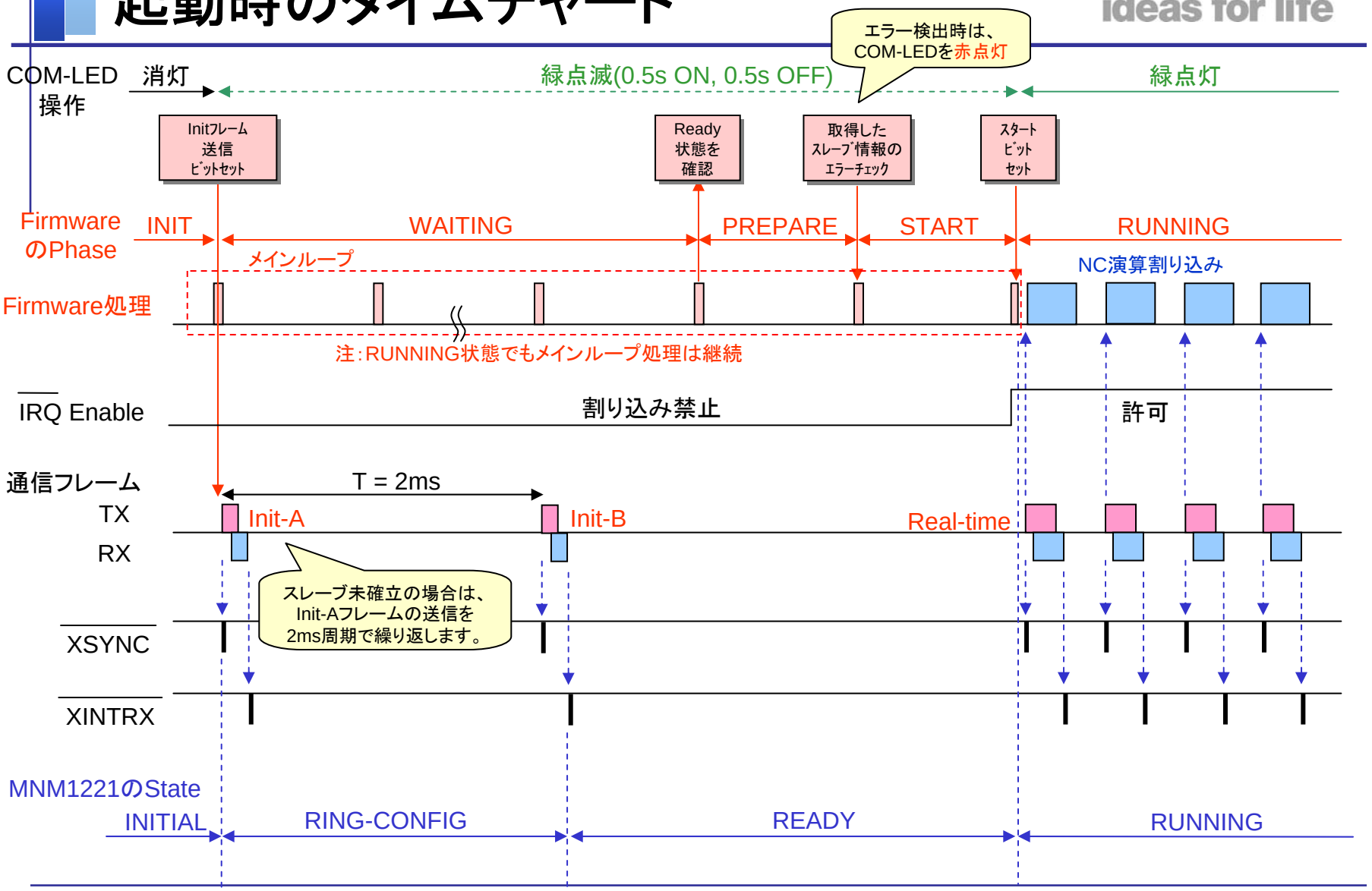


タイムアウトを検知した場合は、安全のため、全軸にサーボオフを指令してください。

サンプルコードの変更

Chapter 1 も参照してください。
重複を避けるために説明を省略している箇所があります。

起動時のタイムチャート



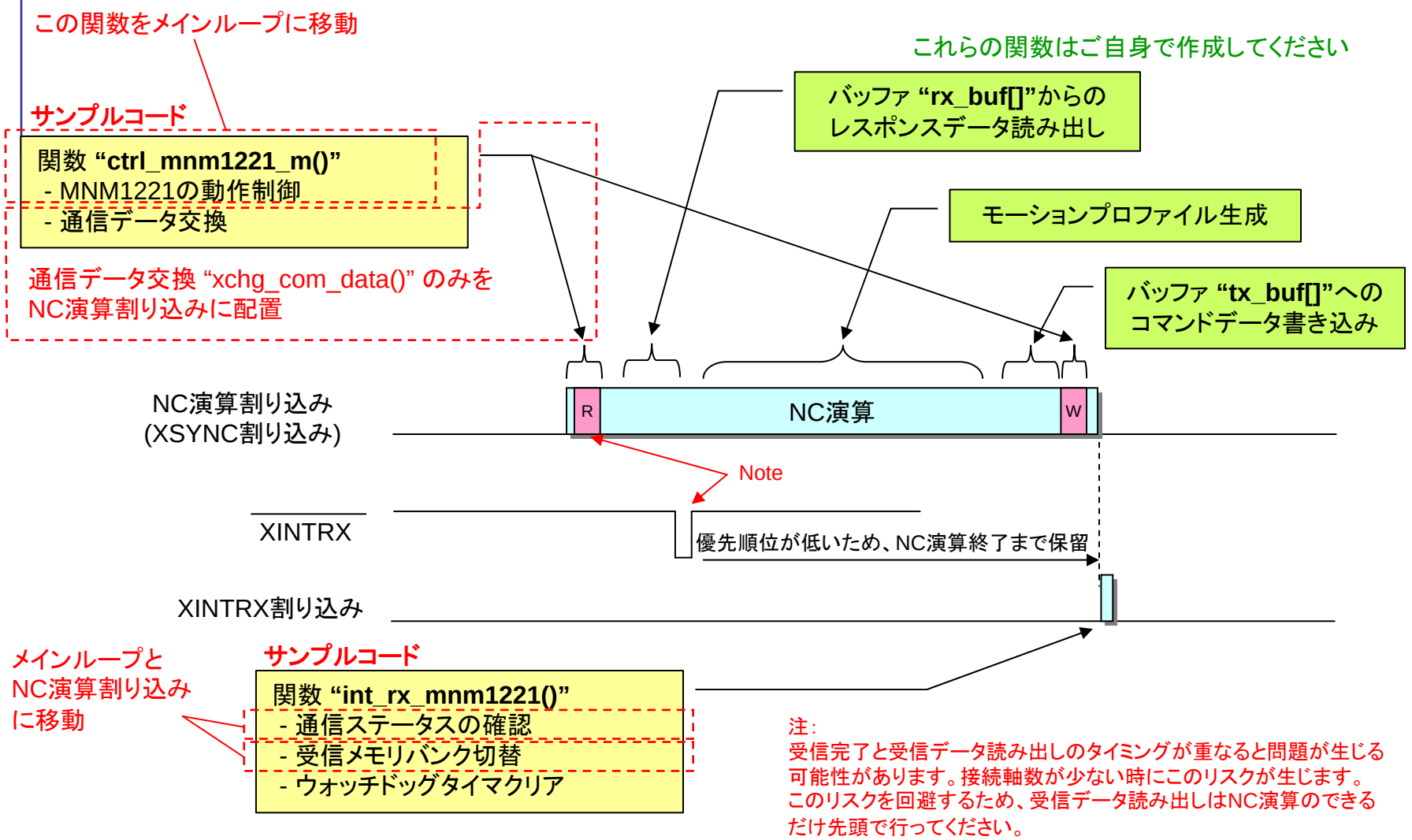
タスク割り付け

Panasonic
ideas for life

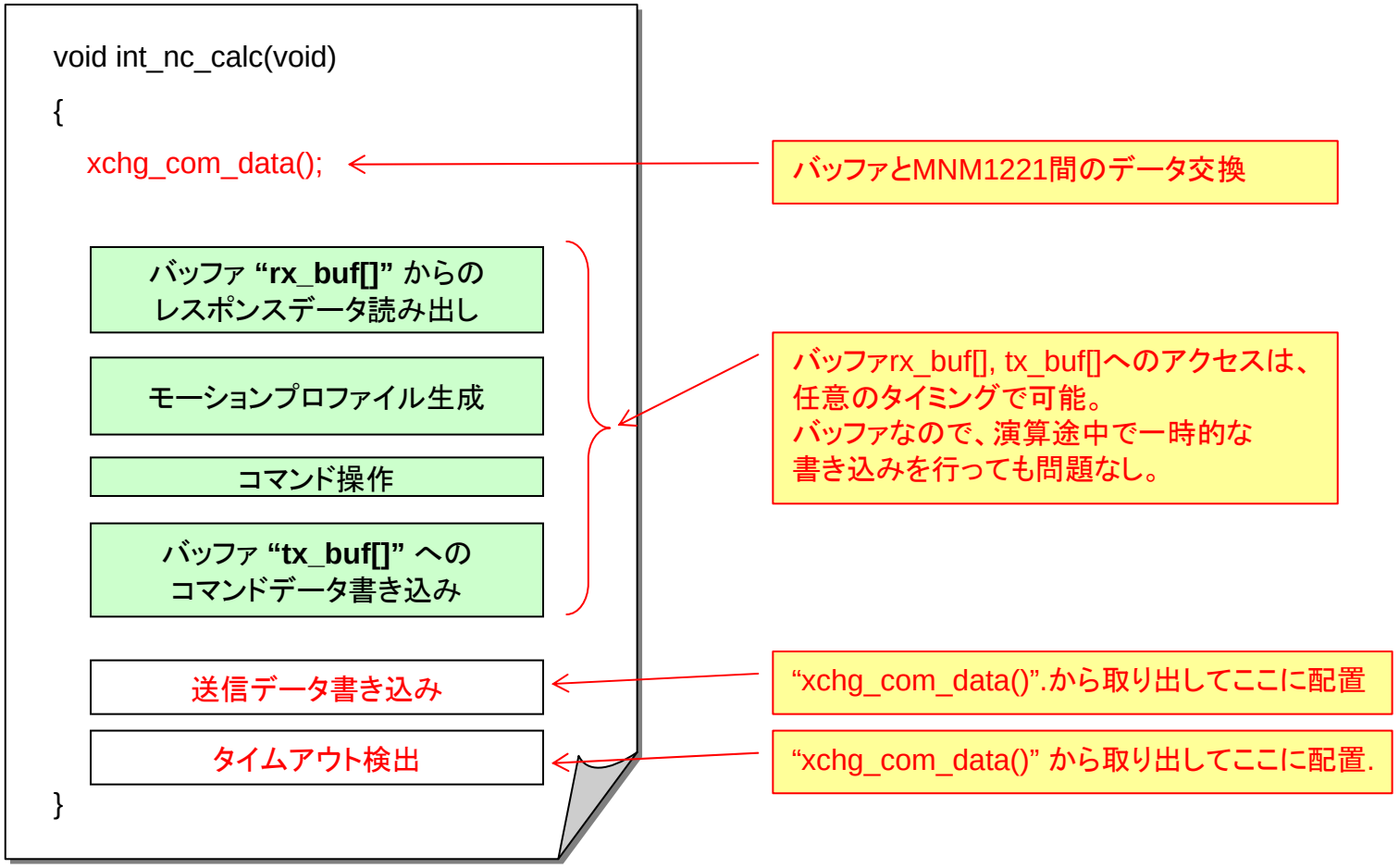
タスク	トリガ	優先順位	周期	処理
メイン ループ	-	-	-	- MNM1221の動作制御 （通信ステータスチェックを含む） - “COM” LED制御
XSYNC 割り込み	送信起動	-	（例） 0.5ms	- 通信データ交換 - NC演算 - タイムアウト検知
XINTRX 割り込み	受信完了	XSYNC より 低く設定	XSYNC と同じ	- ウォッチドッグタイマクリア

注:
- RUNNING状態になるまでは割り込み禁止。
- XINTRX割り込みを使わない場合は、Update Counter Echoを用いたタイムアウト検知が必要。

サンプルコードの配置


この関数をメインループに移動
これらの関数はご自身で作成してください
サンプルコード
関数 “ctrl_mnm1221_m()”
- MNM1221の動作制御
- 通信データ交換
通信データ交換 “xchg_com_data()” のみを
NC演算割り込みに配置
バッファ “rx_buf[]”からの
レスポンスデータ読み出し
モーションプロファイル生成
バッファ “tx_buf[]”への
コマンドデータ書き込み
NC演算割り込み
(XSYNC割り込み)
R
NC演算
W
Note
XINTRX
優先順位が低いため、NC演算終了まで保留
XINTRX割り込み
メインループと
NC演算割り込み
に移動
サンプルコード
関数 “int_rx_mnm1221()”
- 通信ステータスの確認
- 受信メモリバンク切替
- ウォッチドッグタイマクリア
注:
受信完了と受信データ読み出しのタイミングが重なると問題が生じる
可能性があります。接続軸数が少ない時にこのリスクが生じます。
このリスクを回避するため、受信データ読み出しはNC演算のできる
だけ先頭で行ってください。

NC演算割り込みへの配置例



MNM1221初期化

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

この行を置換

0

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

通信周期	MNM1221_M_TX_PERIOD
1ms	0x61A8
0.5ms	0x30D4
0.166ms	0x1047

状態の読み出し

Panasonic
ideas for life

ctrl_mnm1221_m() in mnm1221_m.c

この関数を
メインループに配置

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

この行を挿入

ここに割り込みを禁止する関数を挿入

この行を削除

サイクリック伝送の起動

ctrl_mnm1221_m() in mnm1221_m.c

```
/*--- start of cyclic transmission ------------------
case PH_START:                  /* MNM1221 is in READ
    phase = PH_RUNNING;
    clr_mnm1221_tx_mem();       /* clear TX memory, j
    /* This clearing is unnecessary if initial data s

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

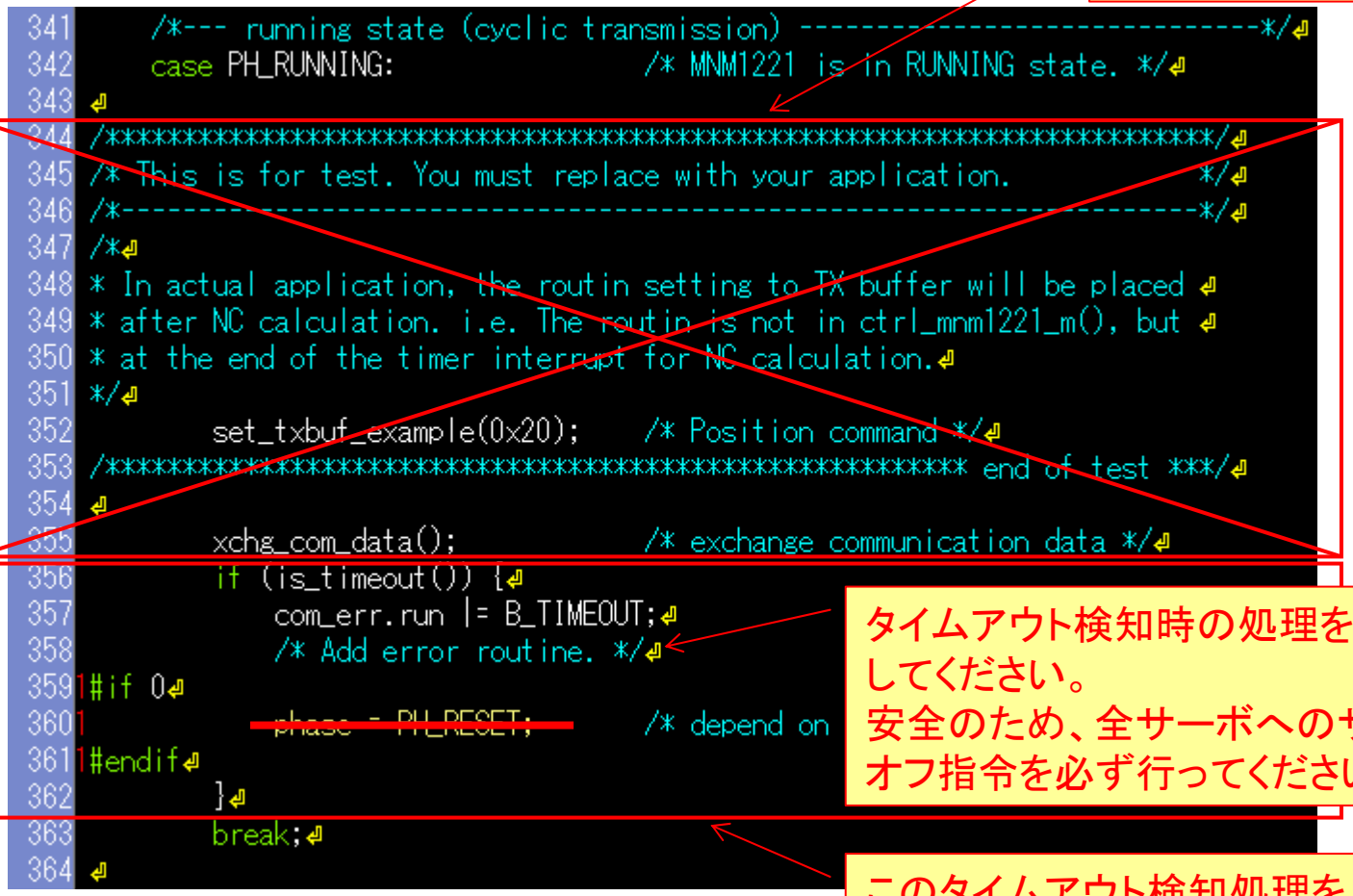
MNM1221の送信メモリに
初期送信データを設定します。
初期テスト後に正規の初期化
処理に置換してください。
このまま使用する場合には、
処理の重複を避けるため、
直前のclr_mnm1221_tx_mem()
を削除。

ここに割り込みを
許可する関数を挿入

この行を xchg_com_data() から
抜き出した送信データ書き込み
処理に置換

Running状態

ctrl_mnm1221_m() in mnm1221_m.c




```
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. */
359 #if 0
360             phase = PH_RESET;       /* depend on
361 #endif
362         }
363         break;
364
```

RUNNING状態においては、メインループでは何も行いません。

データ交換

Panasonic
ideas for life

xchg_com_data() in mnm1221_m.c

この関数をNC演算割り込みの先頭に配置

この部分をNC演算割り込みの末尾に移動。（16bitバス用）

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

    short i;
    volatile unsigned short *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_s(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_s(&(rx_buf[i].data[0]), p, N_DATA);
    }

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

unsigned long ul_temp;

16bitバス用
送信データ書き込み

```
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;
}
```

16bitバス用
受信データ読み出し

データ交換 (続き)

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
}
```

この部分をNC演算割り込みの末尾に移動。
（32bitバス用）

32bitバス用
送信データ書き込み

32bitバス用
受信データ読み出し

XINTRX割り込み

この部分を削除

```
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割り込み (続き)

Panasonic
ideas for life

この部分を削除

```
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 }↵
```

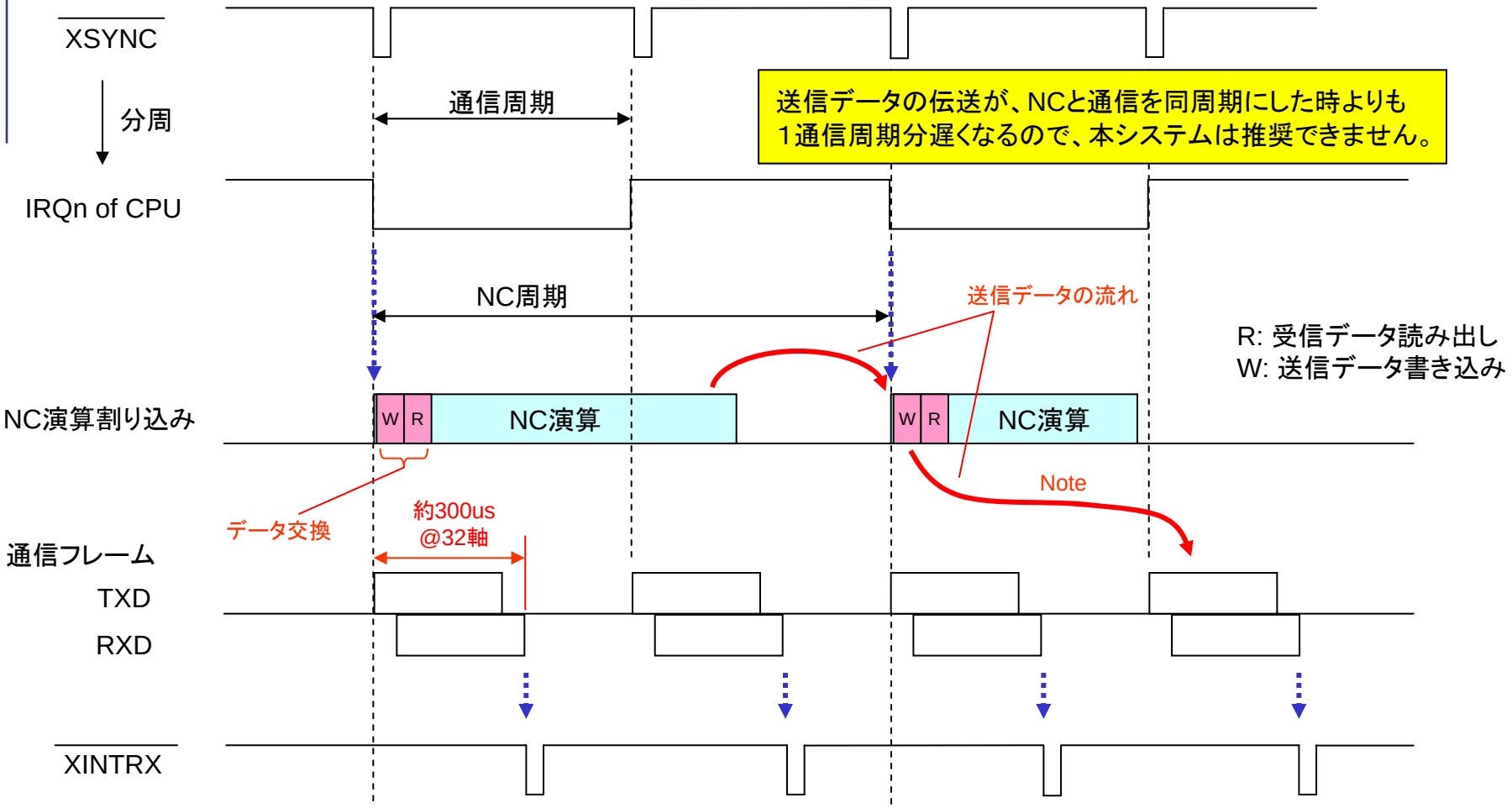
このウォッチドッグタイマクリアのみ残す

Appendix

NC演算1回あたり通信2回のシステム検討

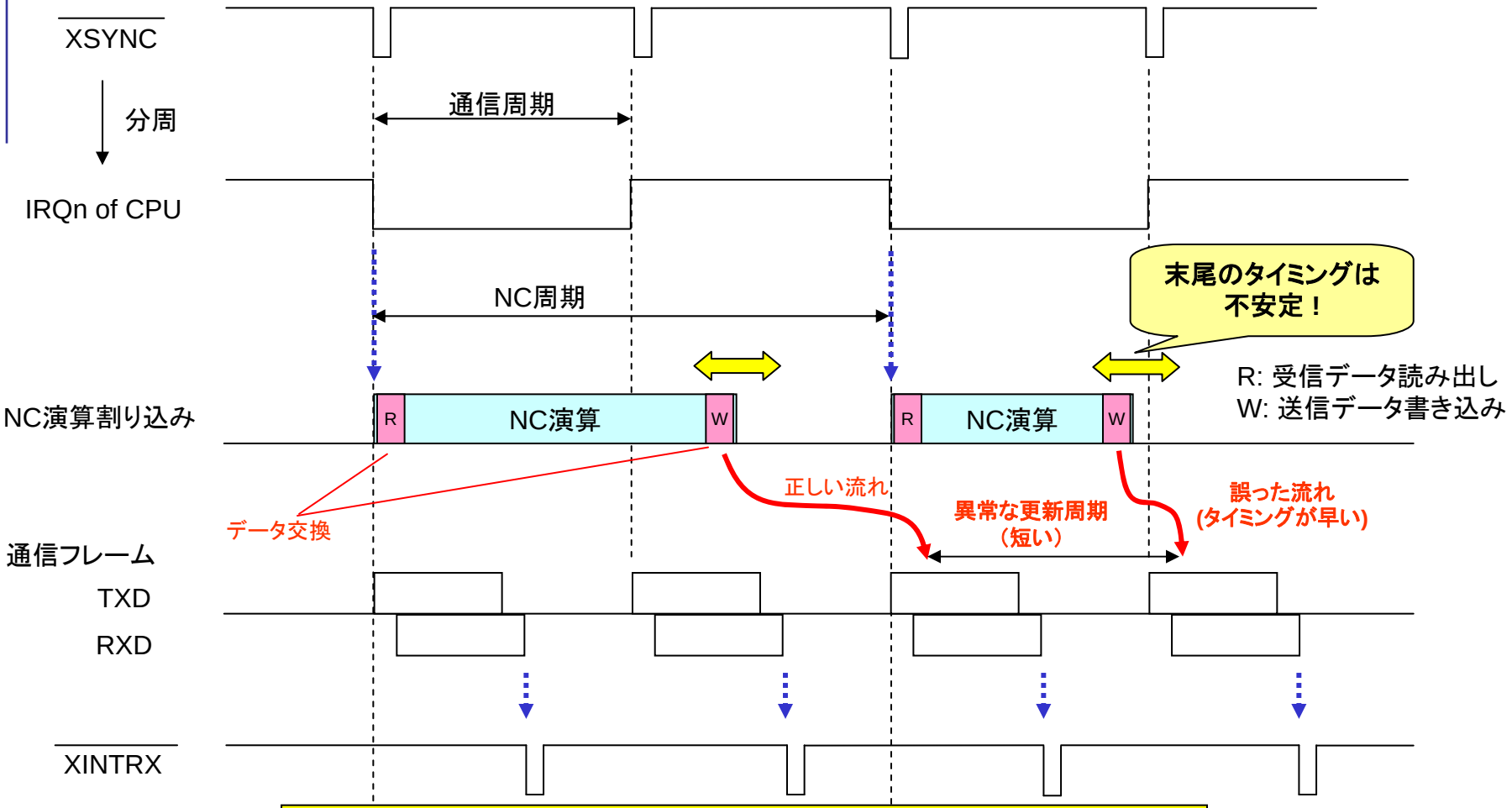
送信データの伝送遅れが長いため、本システムは推奨できません。

タイムチャート



注:
XSYNCの立ち下りエッジ時点ですでにフレーム送信を開始しているため、そこで送信メモリのバンク切り替えを指示しても
実際の切り替えは送信が完了するまで保留されます。このため、送信タイミングは 1通信周期分、遅れます。

悪い例



送信データ書き込みをNC演算の末尾で行えば送信遅延を短くできるが、
末尾タイミングは不安定なので、更新周期を一定に保つことができない。