スレーブ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

Revision	日付	変更内容
1	2006/2/2	初版
2	2012/2/15	P1 タイトルを「ファームウェア開発ガイド（スレーブ用）」から変更。 P3 「はじめに」を追加。 P5 明確化のため、ブロック図を修正。 MNM1221ブロック図を削除。 P7 XSYNC出力タイミングを追加。 P9 SH7065FをSH7216の例に変更。 P10 XINTRXの接続を削除。 P14 バス接続図を追加。 P35 占有ブロック数の設定を追加。 文言等を小変更。

はじめに

本書は、汎用スレーブのシステム設計例について説明するものです。

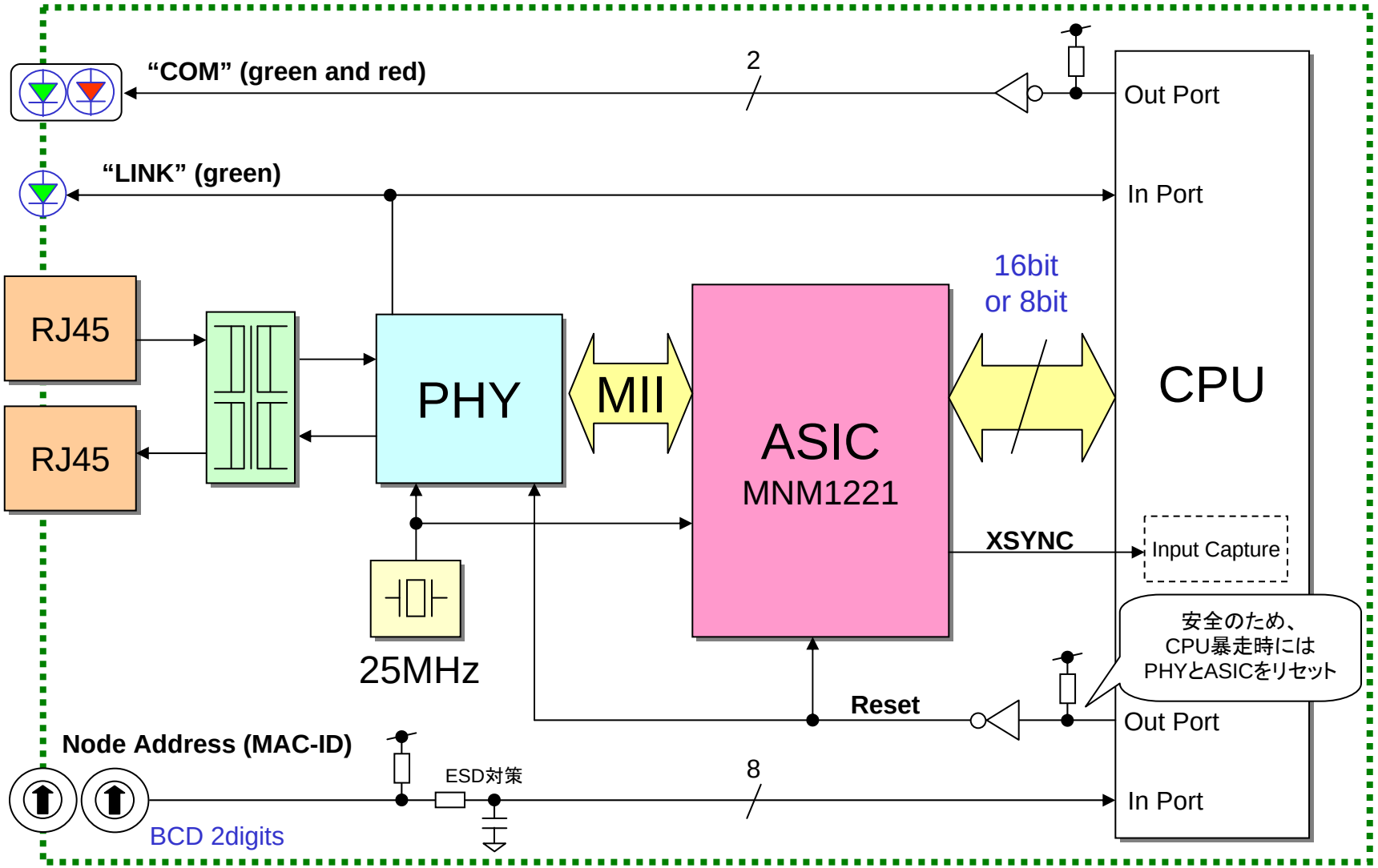
ASIC MNM1221のモード:

	マスタ	スレーブ	
		汎用	IN / OUT
CPU	使う	使う	使わない
データ端子	データバス (32 or 16bit)	データバス (16 or 8bit)	入 or 出力端子 (32)
アドレス端子	アドレスバス	アドレスバス	MAC-ID 設定入力
Pin91 の機能	XWAIT	XWAIT	XLED
送信のトリガ	内蔵タイマ or XTXTIM 入力	フレーム受信	フレーム受信
XSYNC 端子	送信タイミングで 出力	全スレーブの 受信完了後に出力 (RUNNING 状態のみ)	全スレーブの 受信完了後に出力 (RUNNING 状態のみ)
タイムアウト 発生条件	次回送信までに 受信が無い場合	一定期間 受信が無い場合 (時間はﾚｼﾞｽﾀで設定)	一定期間 受信が無い場合 (時間は 20.9ms 固定)

注:汎用スレーブでI/Oを構成することも可能です。

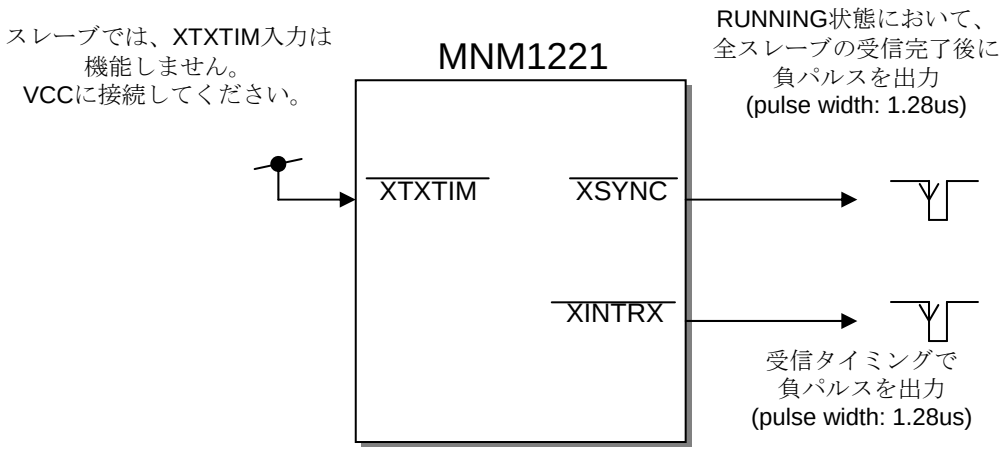
システム構成

RTEX回路ブロック図



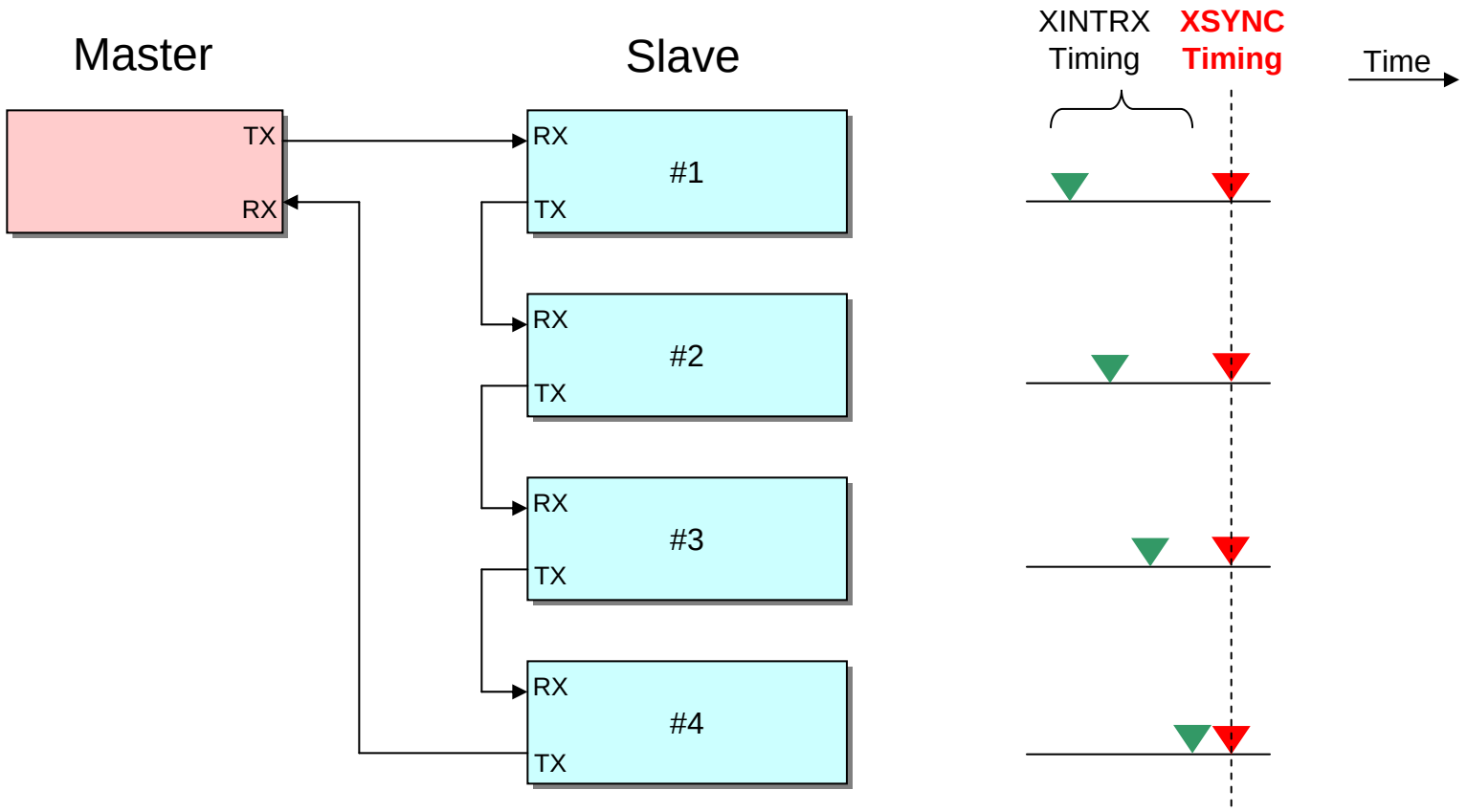
MNM1221のタイミング信号

スレーブでは、XSYNC信号は全スレーブ受信完了後の同一タイミングで出力されます。この信号に同期して制御処理を行うことで、全スレーブの同時性を実現します。



XSYNC出力タイミング

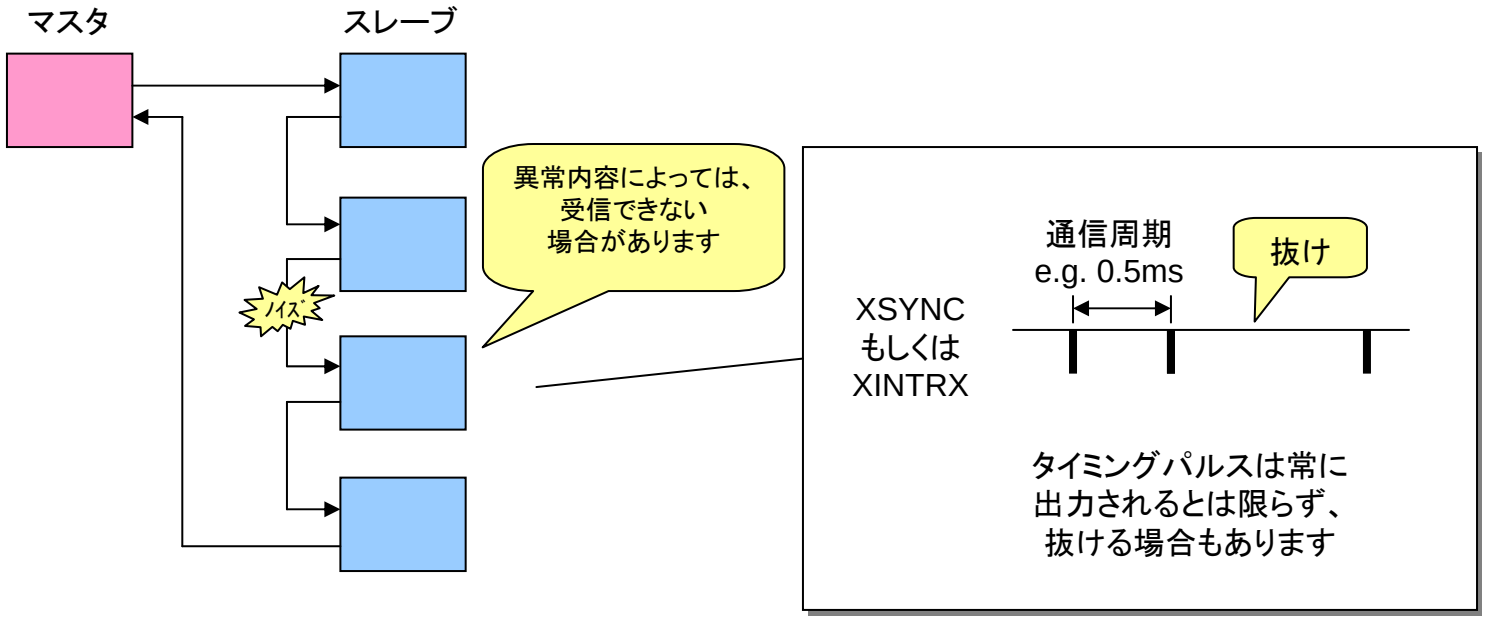
全スレーブのXSYNCは、同時に出力



通信異常への配慮

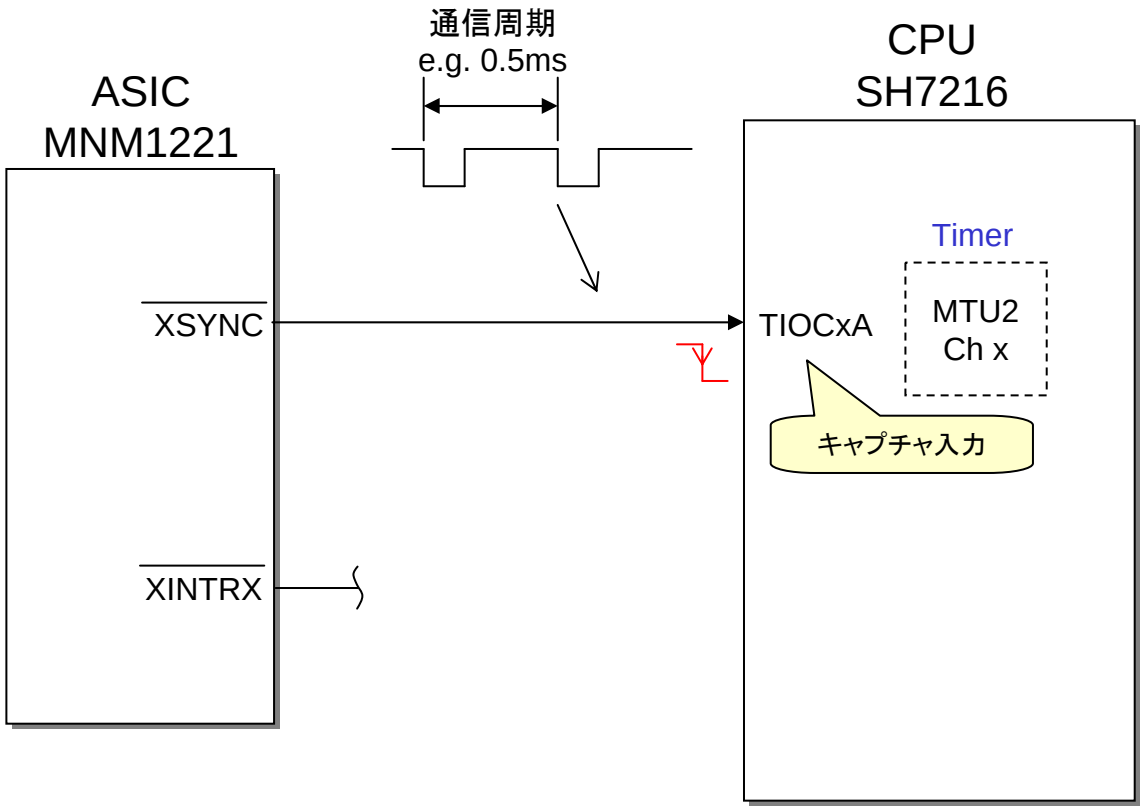
ノイズ等により通信異常が生じると、CRC異常ばかりではなく、
その通信サイクルのフレーム送信が停止する場合もあります。

この場合は受信できず、MNM1221からXSYNCとXINTRXパルスは出力されません。
このような状況でも問題が生じないシステムにする必要があります。



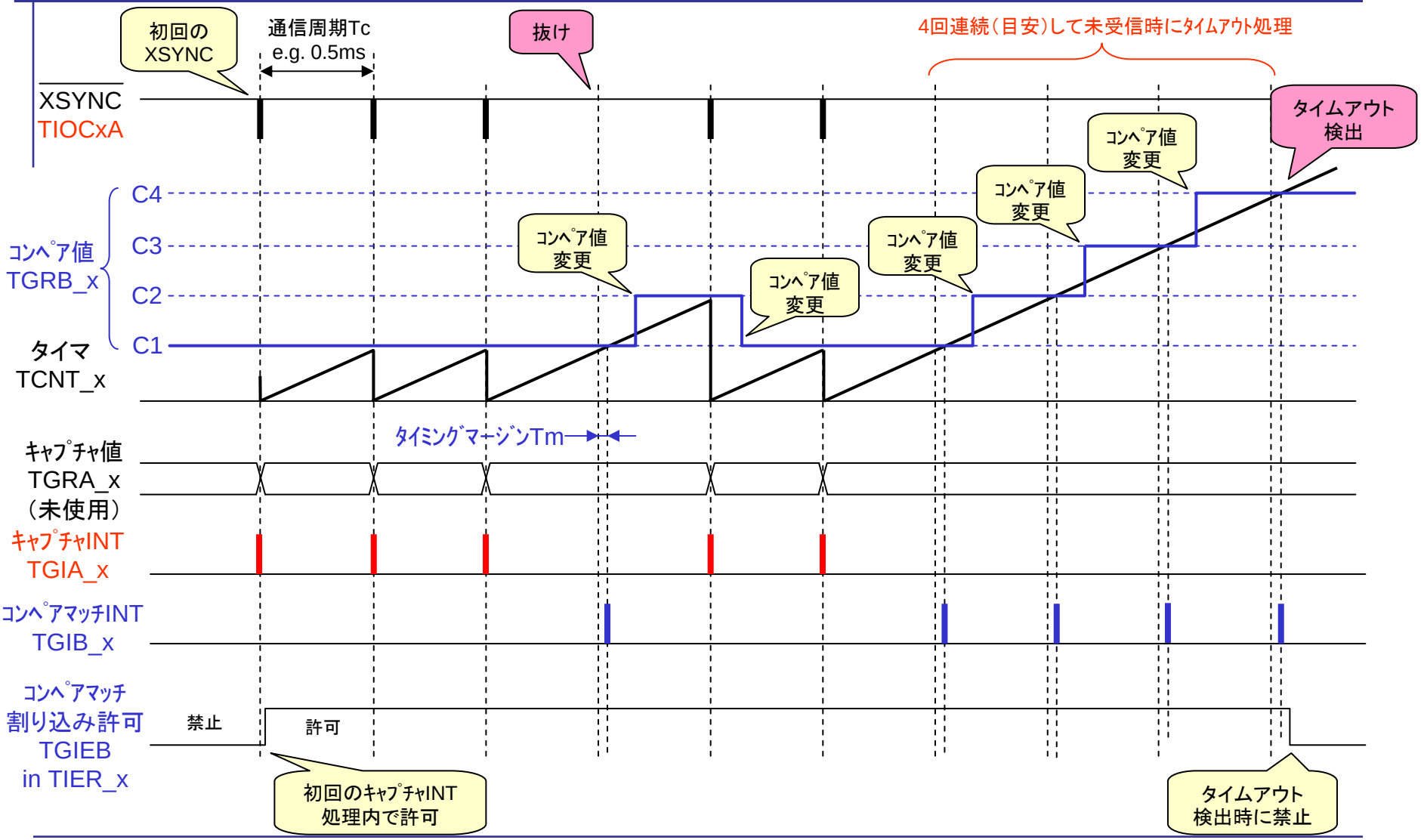
SH7216 (Renesas) との接続例

タイミング信号の接続



・XSYNCをCPU内蔵タイマのキャプチャ入力端子に接続。
・XSYNCの立ち下りエッジでタイマをクリアし、同時に割り込みを起動。

タイムチャート


初回の
XSYNC
通信周期Tc
e.g. 0.5ms
抜け
4回連続（目安）して未受信時にﾀｲﾑｱｳﾄ処理
XSYNC
TIOCxA
ｺﾝﾍﾟｱ値
TGRB_x
C4
C3
C2
C1
ｺﾝﾍﾟｱ値
変更
ｺﾝﾍﾟｱ値
変更
ｺﾝﾍﾟｱ値
変更
ｺﾝﾍﾟｱ値
変更
ｺﾝﾍﾟｱ値
変更
タイムアウト
検出
タイマ
TCNT_x
ﾀｲﾐﾝｸﾞﾏｰｼﾞﾝTm
ｷｬﾌﾟﾁｬ値
TGRA_x
（未使用）
ｷｬﾌﾟﾁｬINT
TGIA_x
ｺﾝﾍﾟｱﾏｯﾁINT
TGIB_x
ｺﾝﾍﾟｱﾏｯﾁ
割り込み許可
TGIEB
in TIER_x
禁止
許可
初回のｷｬﾌﾟﾁｬINT
処理内で許可
タイムアウト
検出時に禁止

割り込みの設定

Panasonic
ideas for life

正常受信時はインプットキャプチャ割り込みで、
また、抜け発生時はコンペアマッチ割り込みを用いることで制御処理を継続。

信号	割り込み	周期	発生要因	割り込み許可
TGIA_x	MTU2 ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬ	e.g. 0.5ms	正常受信時	常時
TGIB_x	MTU2 コンペアマッチ	e.g. 0.5ms	抜け発生時	初回XSYNC検出時に ｷｬﾌﾟﾁｬINT処理内で許可。 タイムアウト検出時に禁止。

注:

・通信抜け発生時は前回の正常受信データに基づき制御を継続。(CRC異常検出時と同様)
指令は前回の指令位置差分を用いる(速度一定であると仮定して補間)。

・インプットキャプチャは割り込み起動と同時にタイマをクリアする目的で使用。
TGRA_xのキャプチャ値は使わない。

・割り込みを起動するXSYNC信号はRUNNING状態でしか出力されないので、
リセット解除後からRUNNING状態に至るまでの期間の制御は、メインループなどで実行。

コンペア値の設定

正常受信時にキャプチャINTとコンペアマッチINTの両方が起動しないようにするため、
コンペアマッチINTの発生タイミングを制御に影響しない範囲で少し遅らせる。
このため、次にようにコンペア値を設定。

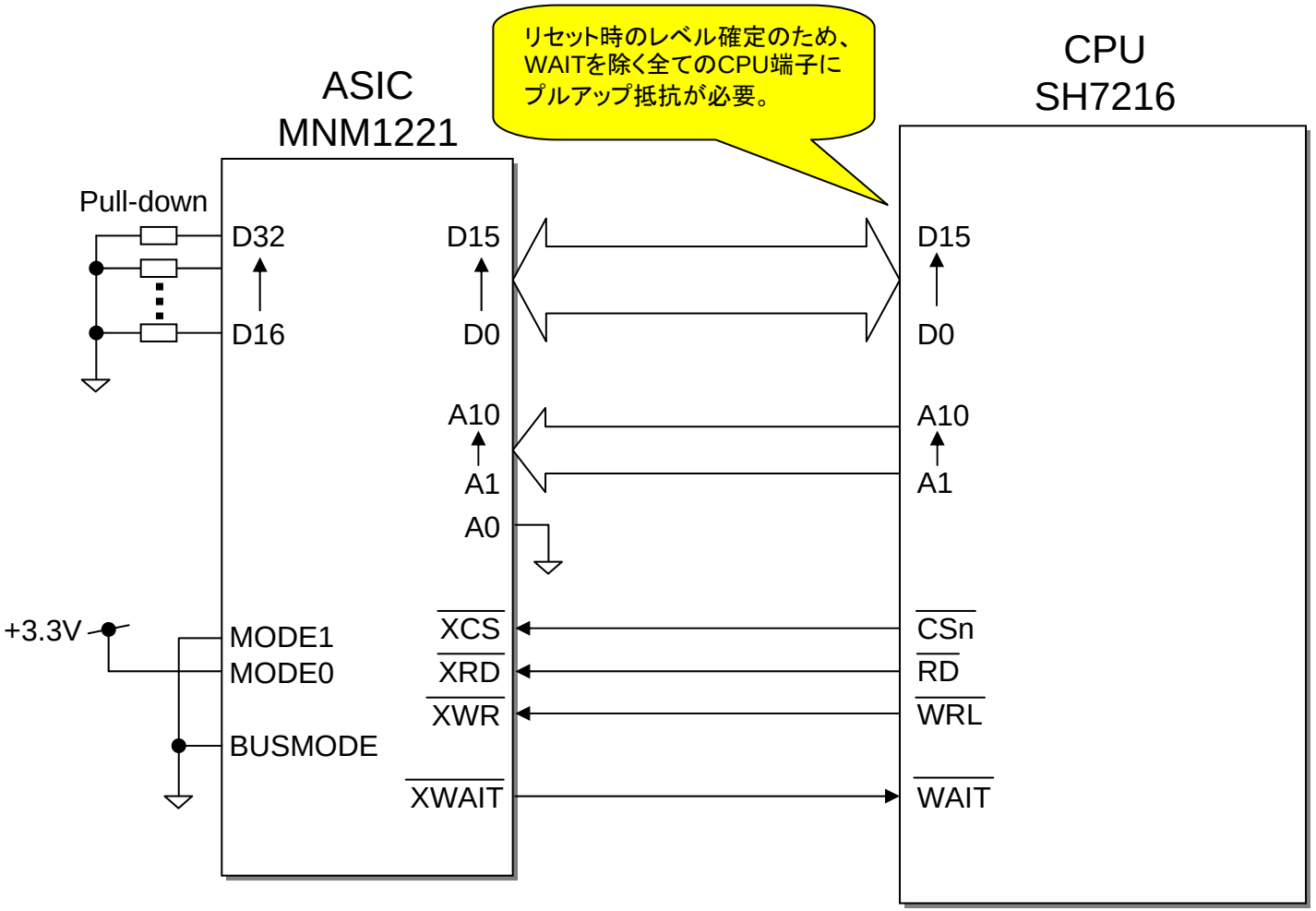
	Setting
C1	Tc + Tm
C2	2Tc + Tm
C3	3Tc + Tm
C4	4Tc + Tm

Tc: 通信周期
Tm: タイミングマージン
（ XSYNCのジッタよりも大きく設定）

受信状況に応じて、TGRB_xに次のコンペア値を設定。

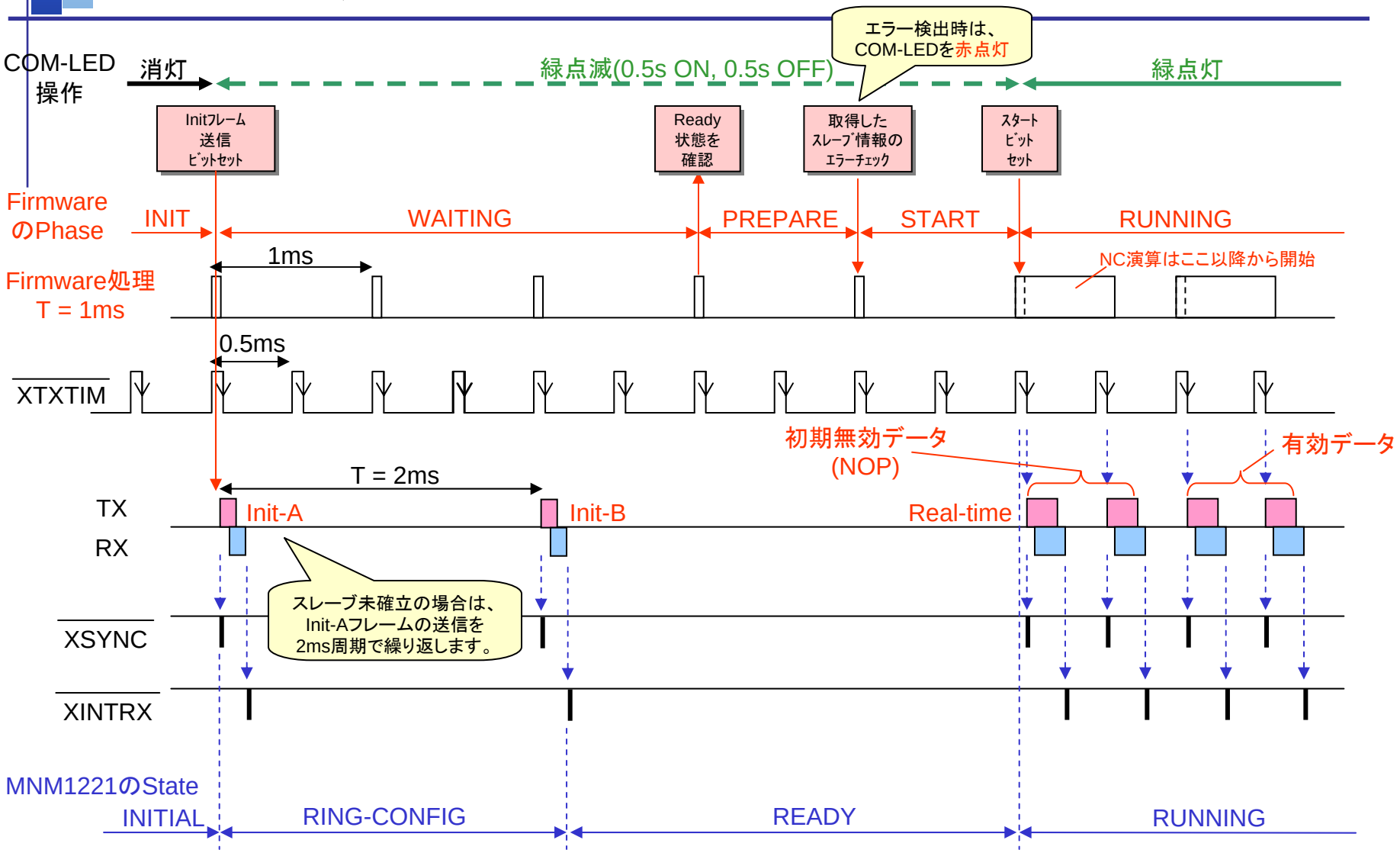
キャプチャ割り込み処理	コンペアマッチ割り込み処理
C1を設定 （初期化）	割り込み発生毎に、現在の コンペア値をインクリメント （現在がC1ならC2に変更）

バス接続

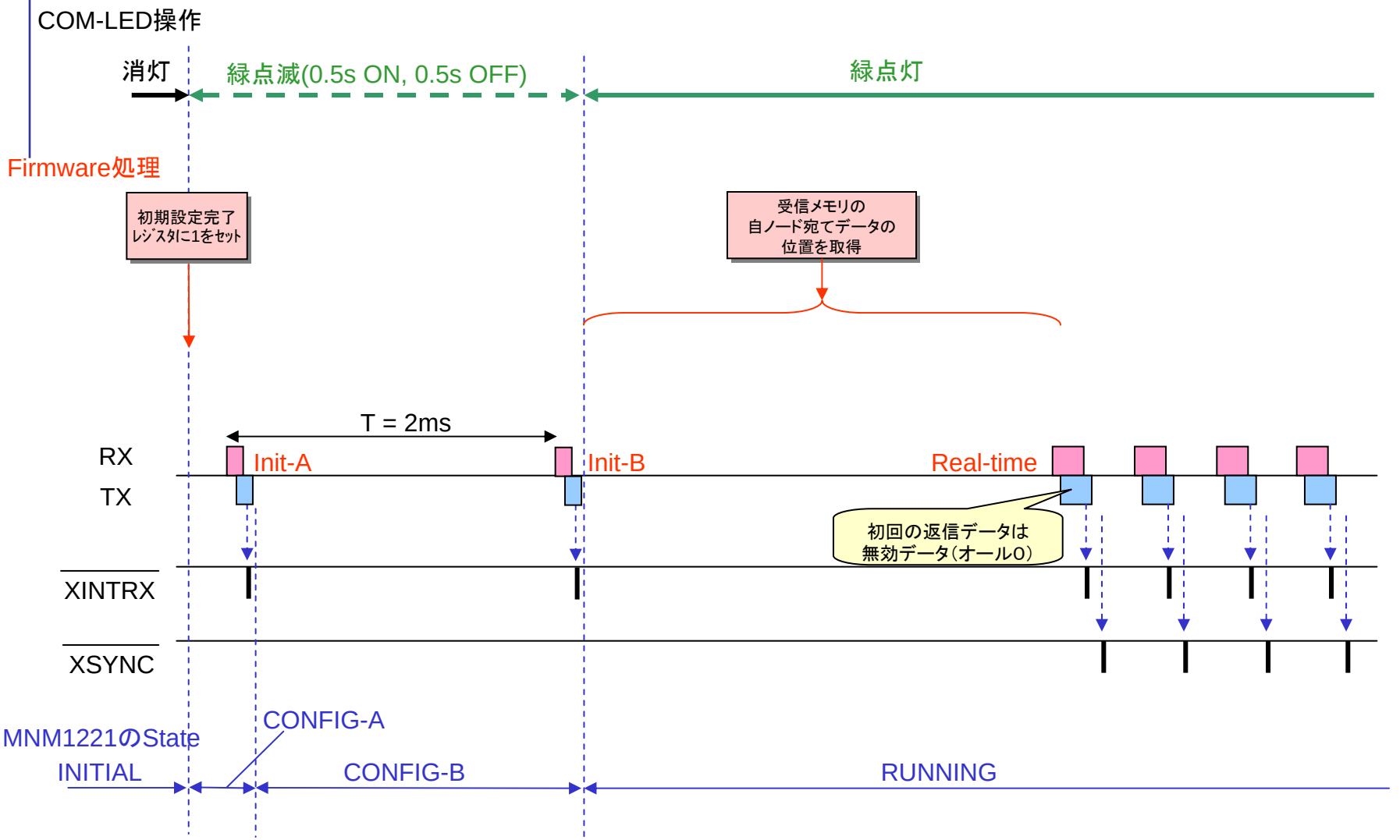

リセット時のレベル確定のため、
WAITを除く全てのCPU端子に
プルアップ抵抗が必要。
ASIC
MNM1221
CPU
SH7216
Pull-down
D32
D16
D15
D0
A10
A1
A0
+3.3V
MODE1
MODE0
BUSMODE
XCS
XRD
XWR
XWAIT
D15
D0
A10
A1
CSn
RD
WRL
WAIT

起動時タイムチャート

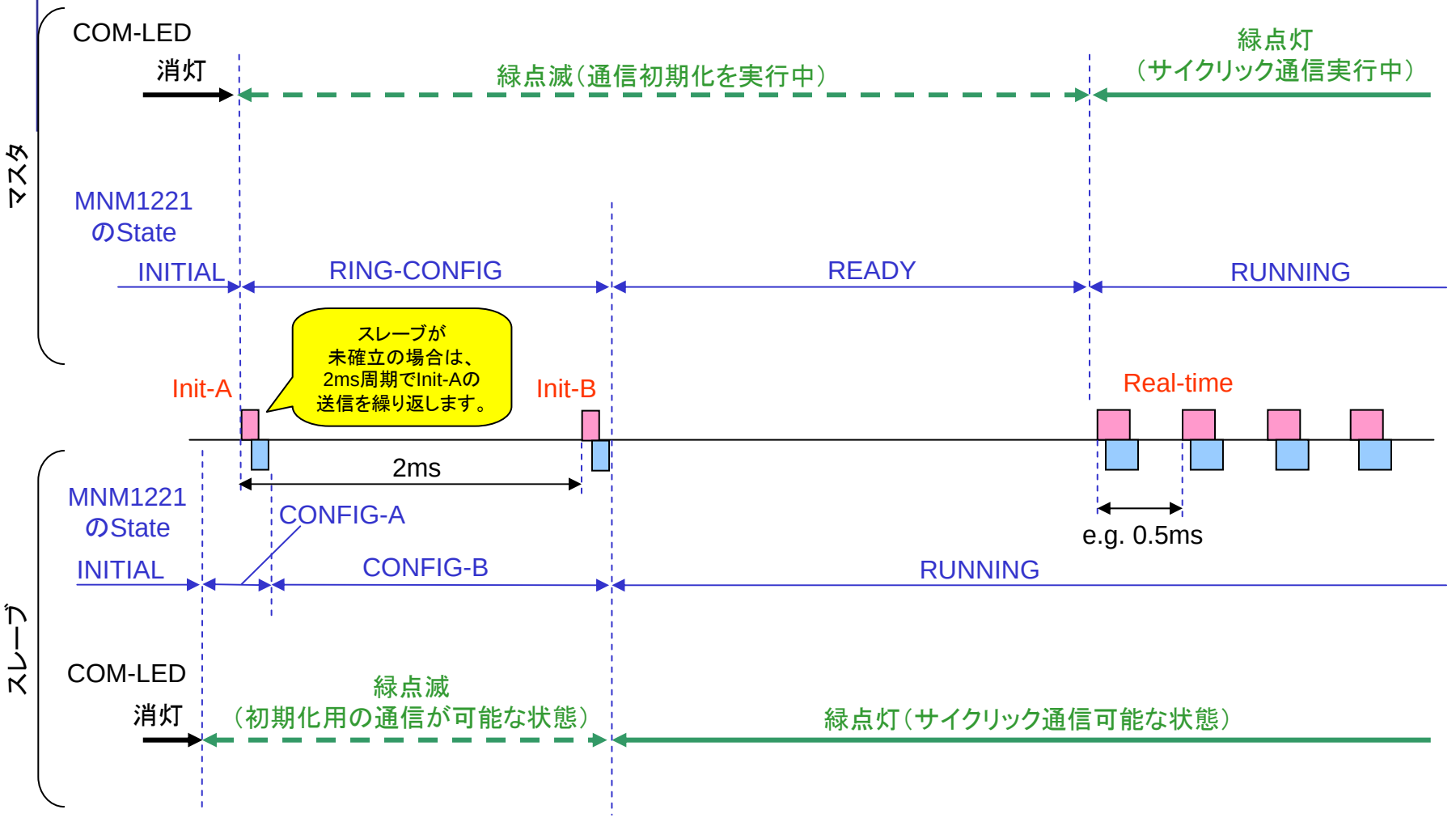
マスタ起動時タイミング



スレーブ起動時タイミング



起動時の状態遷移


マスタ
COM-LED
消灯
緑点滅（通信初期化を実行中）
緑点灯
（サイクリック通信実行中）
MNM1221
のState
INITIAL
RING-CONFIG
READY
RUNNING
スレーブが
未確立の場合は、
2ms周期でInit-Aの
送信を繰り返します。
Init-A
Init-B
Real-time
2ms
e.g. 0.5ms
MNM1221
のState
CONFIG-A
INITIAL
CONFIG-B
RUNNING
スレーブ
COM-LED
消灯
緑点滅
（初期化用の通信が可能な状態）
緑点灯（サイクリック通信可能な状態）

サンプルコード

サンプルコードの内容

Panasonic
ideas for life

サンプルコードにはMNM1221をスレーブとして動作させるための基本的な処理を収めています。
受信データをRAM上の受信バッファに展開し、また、送信バッファのデータを送信する処理が
行われるので、この送受信バッファを用いたアプリケーションを組み込んでください。

ファイル:

ファイル名	内容
mnm1221_s.h	レジスタ定義ヘッダファイル
mnm1221_s.c	ソースファイル

関数:

関数名	配置	内容
init_mnm1221_s()	リセット解除後	初期化
get_state_mnm1221_s()	メインループ	MNM1221の状態読み出し
int_sync_mnm1221_s()	XSYNCで起動する割り込み	通信データ交換
timeout_mnm1221_s()	タイムアウト検出時	タイムアウト時の処理

変数:

変数名	配置	内容
my_rx_buf[]	CPU内蔵RAM	受信バッファ
my_tx_buf[]	CPU内蔵RAM	送信バッファ

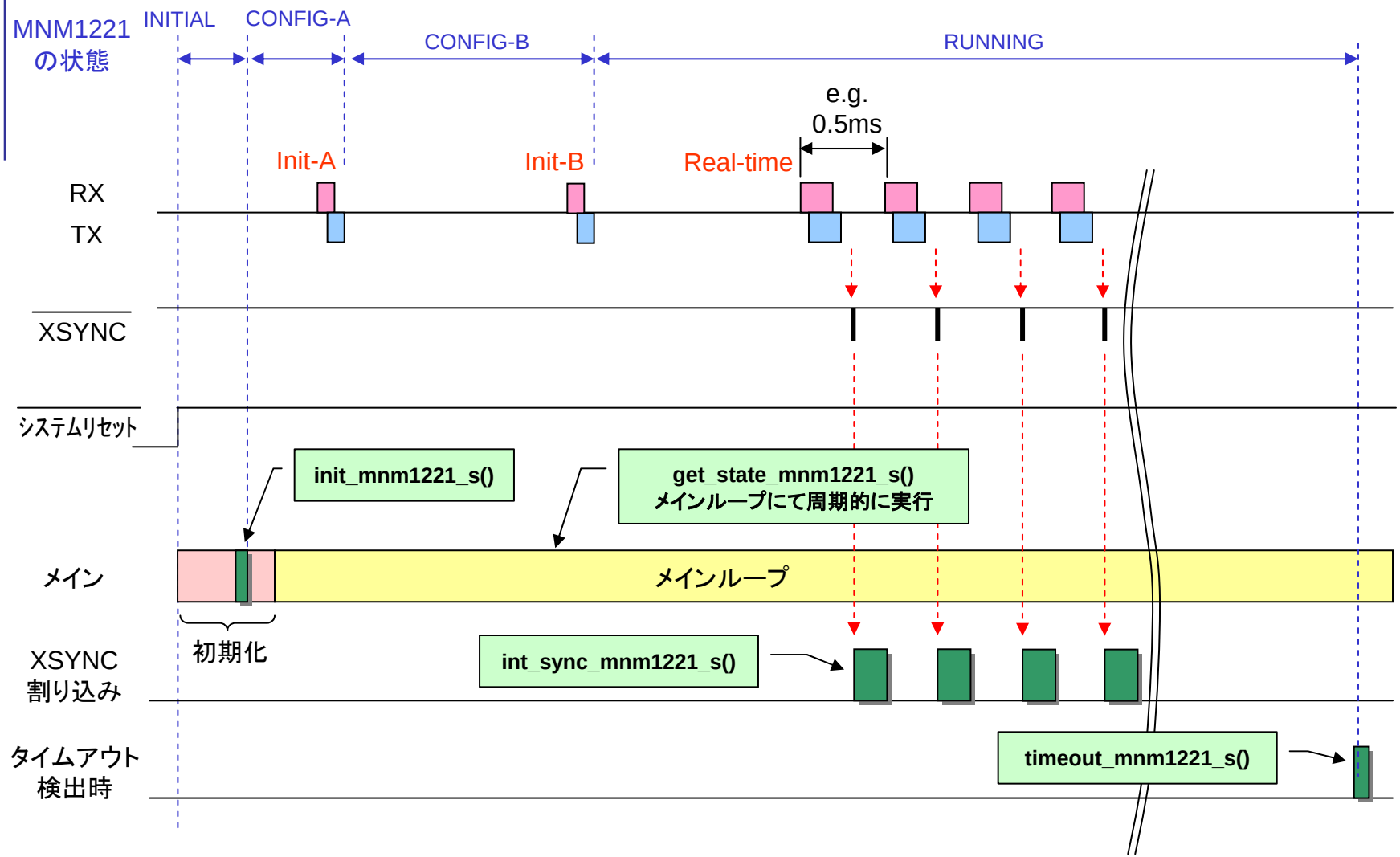
サンプルコードに含まれない処理

次の処理はサンプルコードに含まれていないので、別途作成してください。

項目	内容	備考
アプリケーション	各機器に応じた処理	コマンドエラー検出処理 を含む
MAC-ID読み込み	ロータリSWの値を 電源投入時に一度だけ読み込み	BCD→BIN変換、 設定範囲外の異常検出、等
“COM” LED制御	MNM1221の状態とエラー内容 に応じて点灯制御	赤、緑の2色発光LED
タイムアウト検出	RUNNING状態において、 所定の回数以上、連続して 受信が無い場合	断線等で発生
連続CRC異常検出	RUNNING状態において、 所定の回数以上、連続して CRC異常を検出した場合	安全性等を考慮し、 必要に応じて実装 （モーション制御では必須）

注：タイムアウトの検出時間や連続CRC異常の検出回数を大きく設定すると耐ノイズ性が高まりますが、異常と判断するまでの検出遅れが長くなり安全性の問題が生じるというトレードオフがあります。このことを十分に考慮し、用途に応じた設定をしてください。
また、異常検出時には、動作を停止するなど安全側に制御してください。

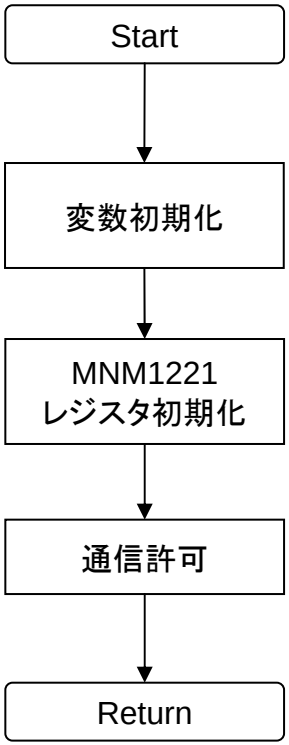
サンプルコードの配置


MNM1221
の状態
INITIAL
CONFIG-A
CONFIG-B
RUNNING
e.g.
0.5ms
Init-A
Init-B
Real-time
RX
TX
XSYNC
システムリセット
init_mnm1221_s()
get_state_mnm1221_s()
メインループにて周期的に実行
メイン
メインループ
初期化
XSYNC
割り込み
int_sync_mnm1221_s()
タイムアウト
検出時
timeout_mnm1221_s()

フローチャート

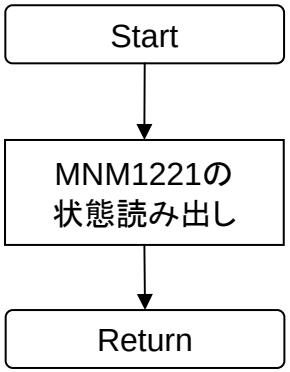
初期化処理

関数名	内容
init_mnm1221_s()	リセット解除後の初期化



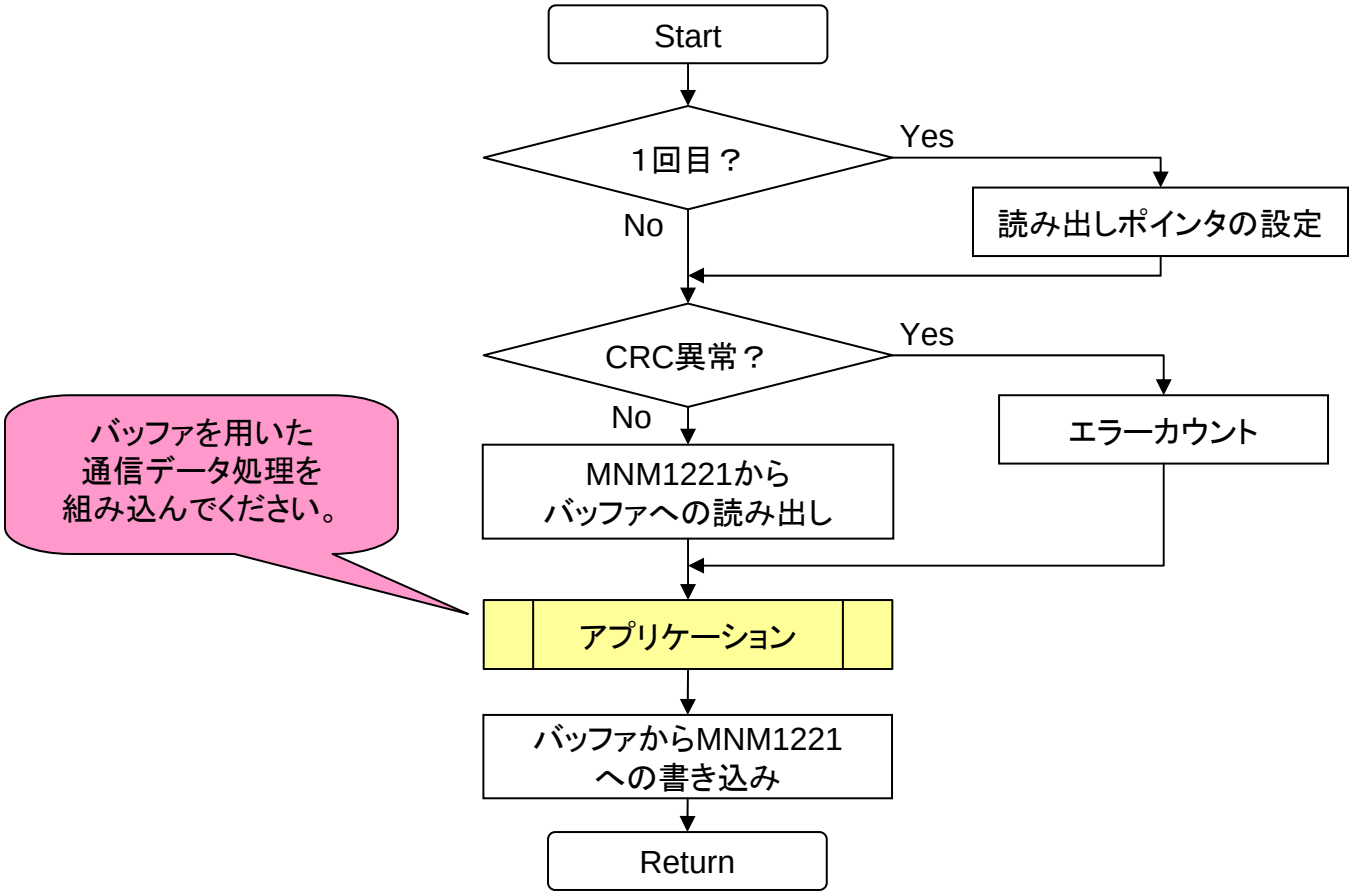
メインループ処理

関数名	内容
get_state_mnm1221_s()	メインループで実行する状態読み出し



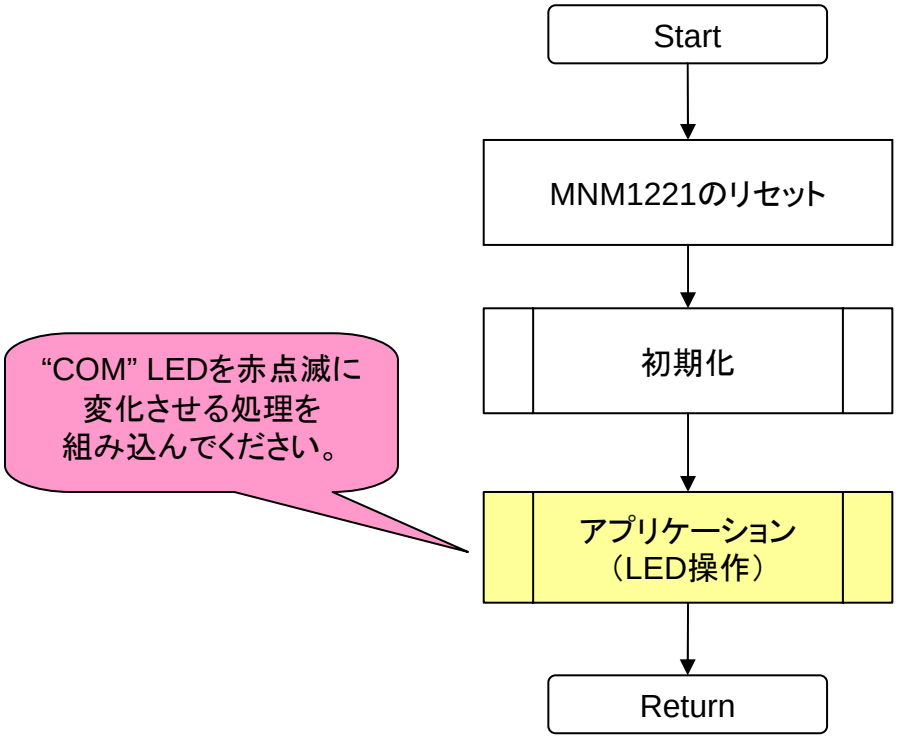
SYNC割り込み処理

関数名	内容
int_sync_mnm1221_s()	MNM1221のXSYNCで起動する割り込み



タイムアウト検出時処理

関数名	内容
timeout_mnm1221_s()	タイムアウト検出時の初期化



送受信バッファの構造

送受信バッファの構造(16bit)

下図は16ビットアクセス時における my_rx_buf[] 配列を示します。my_tx_buf[]も同様です。

	Bit 15 … Bit 8	Bit 7 … Bit 0
my_rx_buf[0]	byte1	byte0
my_rx_buf[1]	byte3	byte2
my_rx_buf[2]	byte5	byte4
my_rx_buf[3]	byte7	byte6
my_rx_buf[4]	byte9	byte8
my_rx_buf[5]	byte11	byte10
my_rx_buf[6]	byte13	byte12
my_rx_buf[7]	byte15	byte14

「byte0 ~ 15」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。

送受信バッファの構造(8bit)

Panasonic
ideas for life

下図は8ビットアクセス時における my_rx_buf[] 配列を示します。my_tx_buf[]も同様です。

	Bit 7 … Bit 0
my_rx_buf[0]	byte0
my_rx_buf[1]	byte1
my_rx_buf[2]	byte2
my_rx_buf[3]	byte3
my_rx_buf[4]	byte4
my_rx_buf[5]	byte5
my_rx_buf[6]	byte6
my_rx_buf[7]	byte7

	Bit 7 … Bit 0
my_rx_buf[8]	byte8
my_rx_buf[9]	byte9
my_rx_buf[10]	byte10
my_rx_buf[11]	byte11
my_rx_buf[12]	byte12
my_rx_buf[13]	byte13
my_rx_buf[14]	byte14
my_rx_buf[15]	byte15

「byte0 ~ 15」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。

LED操作

“COM” LED 操作

Panasonic
ideas for life

“COM” LED（赤、緑の2色発光）は下表に示すように点灯制御（出力ポート操作）してください。

通常時

get_state_mnm1221_s() の戻り値	“COM” LED
0x0008 (INITIAL)	消灯
0x0004 (CONFIG-A)	緑点滅 (0.5s ON, 0.5s OFF)
0x0002 (CONFIG-B)	
0x0001 (RUNNING)	緑点灯

エラー発生時

エラーの内容	“COM” LED
タイムアウト	赤点滅 (0.5s ON, 0.5s OFF)
連続CRC異常	
サイクリックデータ受信不能 （受信データのMAC-ID不一致など）	
スレーブ側でのMAC-ID設定範囲外	赤点灯

注：検出したエラーはラッチし、異常要因が無くなってもクリア処理が行われるまではエラー状態を保持してください。
赤点灯は、システムリセットしないとクリアができないことを示します。
「タイムアウト」と「連続CRC異常」の検出は4通信周期を目安としますが、要求される安全性と耐ノイズ性等に応じて適切に設定してください。

サンプルコードの変更

バスアクセスの定義

Panasonic
ideas for life

mnm1221_s.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 SLAVE_16BIT_ACCESS
/* If NOT 16bits BUT 8bits, change this definition to comments or delete it. */
/*-------------------------------------------------------------------------*/
```

MNM1221が配置されているアドレス
（バイト単位アドレス）に変更

データバス幅が8bitの場合は、削除。
16bitの場合は、このまま残す。

占有ブロック数の設定

Panasonic
ideas for life

mnm1221_s.c

複数のブロックを占有する場合は、この値を変更
(1 block = 16 bytes)

```
/*-----------------------------------------------------------------------*/
/* Definition of constants                                               */
/*-----------------------------------------------------------------------*/

#define N_MY_BLOCK      1       /* the number of my data blocks */
                                /* setting range: 1 to 32, normally 1 */
#if N_MY_BLOCK >= 32
    #define MSK_DATA_CRCE   0xffffffff
#else
    #define MSK_DATA_CRCE   (((unsigned long)1 << N_MY_BLOCK) - 1)
#endif

#define N_MY_DATA       (SIZE_OF_BLOCK * N_MY_BLOCK)
```

アプリケーションの実装

Panasonic
ideas for life

mnm1221_s.c

Int_sync_mnm1221_s()

```
if (!done_1st_cycle) {
    done_1st_cycle = 1;

    order = MNM1221_S_BLK_ORDER & 0x001f;
    p_my_rx_bgn = P_MNM1221_RX_MEM_BGN + (SIZE_OF_BLOCK * order);
    my_dcrc_mask = MSK_DATA_CRCE << order;
}

/* CRC error checking */
crc_err = chk_crc_err(my_dcrc_mask);

/* get command from MNM1221 to my_rx_buf[] */
if (!crc_err) {
    get_cmd_mnm1221();
}

/*--- Here, put your application using my_rx_buf[] and my_tx_buf[] -----*/

/* This function is just for example */
set_resp_example(my_mac_id, crc_err);

/*------------------------------------------ end of your application ---*/

/* set response from my_tx_buf[] to MNM1221 */
set_rsp_mnm1221();
MNM1221_S_TXMEM_SW = 1;                    /* switching TX bank */

return crc_err;
```

このテスト関数を
アプリケーション処理
関数に置換。
この関数の処理時間は
通信周期よりも十分に
短くなるようにしてください。

タイムアウト時のLED制御

Panasonic
ideas for life

mnm1221_s.c

“COM” LEDを赤点滅に変化させる処理を追加

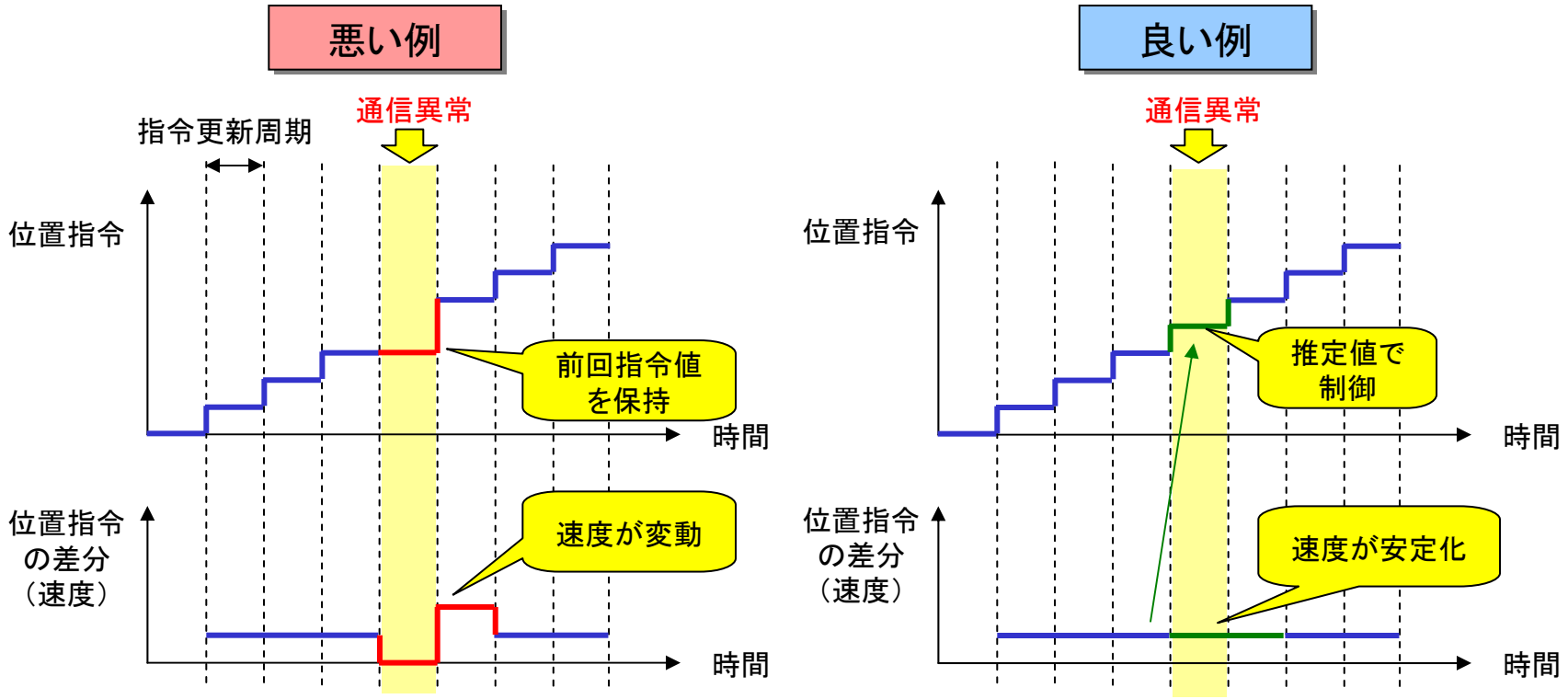
```
void timeout_mnm1221_s(short mac_id)
{
    MNM1221_S_RESET = 1;          /* reset MNM1221 */
                                  /* Another way is to control an output port */
                                  /* connected with XRST input of MNM1221. */
    init_mnm1221_s(mac_id);

    /*-------------------------------------------------------------------------*/
    /* Here, put the operation to blink the "COM" LED (ON:0.5s, OFF:0.5s) with red. */
    /*-------------------------------------------------------------------------*/
}
```

サイクリック位置I/F時の注意事項

通信異常時の動作継続

位置指令をインターフェースとする機器の場合は、通信異常（CRC異常や抜け）が発生した時に、位置指令を前回値のまま保持すると一時的に速度ゼロを指令されたのと同様になります。
このため動作が不安定になるので、前回値を保持するのではなく、速度が前回と同じであるとの仮定に基づき推定した位置指令を用いて制御してください。



注：本説明は一時的に生じた通信異常に関するものです。異常が一定回数以上、連続した場合には、安全確保のため動作を停止させなければなりません。

データ更新タイミング

Panasonic
ideas for life

通信周期よりもデータ更新周期が長い場合には、
「2回連続して正常受信し、なおかつ、この時にUpdate Counterが変化した場合」に、
更新タイミングであると判定してください。

前回受信	今回受信	Update Counter の変化	更新タイミング かどうか
正常	正常	変化	Yes
正常	正常	変化なし	No
少なくともいずれかが異常		-	不明

注:
- 正常受信とは、CRC異常と抜けの両方の異常が無いことを意味します。
- Update Counterはコマンドブロックのbyte0のbit5,6に配置されるデータです。