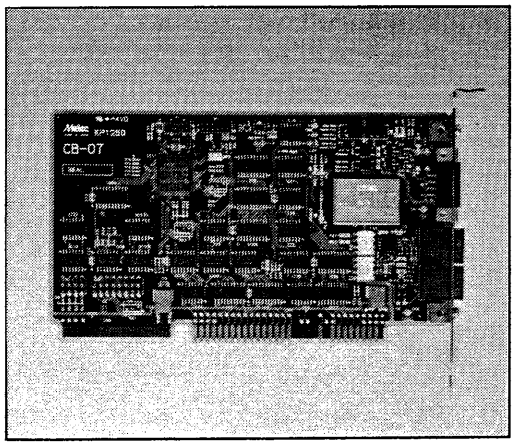
Melec



ISAバスマスタ

CB-07
取扱説明書
(設計者用)

USER'S MANUAL

本製品を使用する前に、この取扱説明書を良く読んで
十分に理解してください。
この取扱説明書は、いつでも取り出して読めるように
保管してください。

PR0589-1('00.5)

はじめに

この取扱説明書は、「ALシリーズ対応ISAバスマスターボードCB-07」を正しく安全に使用していただくために、仕様に重きをおいた取り扱い方法について、ステッピングモータ或いはサーボモータを使った制御装置の設計を担当される方を対象に説明しています。
使用する前に、この取扱説明書を良く読んで十分に理解してください。
この取扱説明書は、いつでも取り出して読めるように保管してください。

安全に関する事項の記述方法について

本製品は正しい方法で取り扱うことが大切です。
誤った方法で取り扱った場合、予期しない事故を引き起こし、人身への障害や財産の損壊等の被害を被るおそれがあります。
そのような事故の多くは、危険な状況を予め知っていれば回避することができます。
そのため、この取扱説明書では危険な状況が予想できる場合には、注意事項が記述してあります。
それらの記述は、次のようなシンボルマークとシグナルワードで示しています。

⚠ 警告	取り扱いを誤った場合に死亡、又は重傷を負うおそれのある警告事項を示します。
⚠ 注意	取り扱いを誤った場合に、軽傷を負うおそれや物的損害が発生するおそれがある注意事項を示します。

御使用の前に

■本製品は、原子力関連機器、航空宇宙関連機器、車両、船舶、人体に直接関わる医療機器、財産に大きな影響が予測される機器など、高度な信頼性が要求される装置向けには設計･製造されておりません。

■入力電源の異常や各信号線の断線、製品本体の故障時でもシステム全体が安全側に働くように、フェールセーフ対策を施してください。

■本製品は必ずこの取扱説明書に記載の指定方法および仕様の範囲内で使用してください。

■本ボードをスロットに挿入する前に、基板上の各種設定を行う必要があります。
次に示す各項を参照ください。
4．ボードの設定
11．基板形状と寸法

はじめに
安全に関する事項の記述方法について
御使用の前に

目　次

PAGE

９．コネクタ信号表

１０．入出力回路

１１．基板形状と寸法

１２．サンプル プログラム

１３．トラブルシューティング

１４．CB-07 全 COMMAND 一覧表

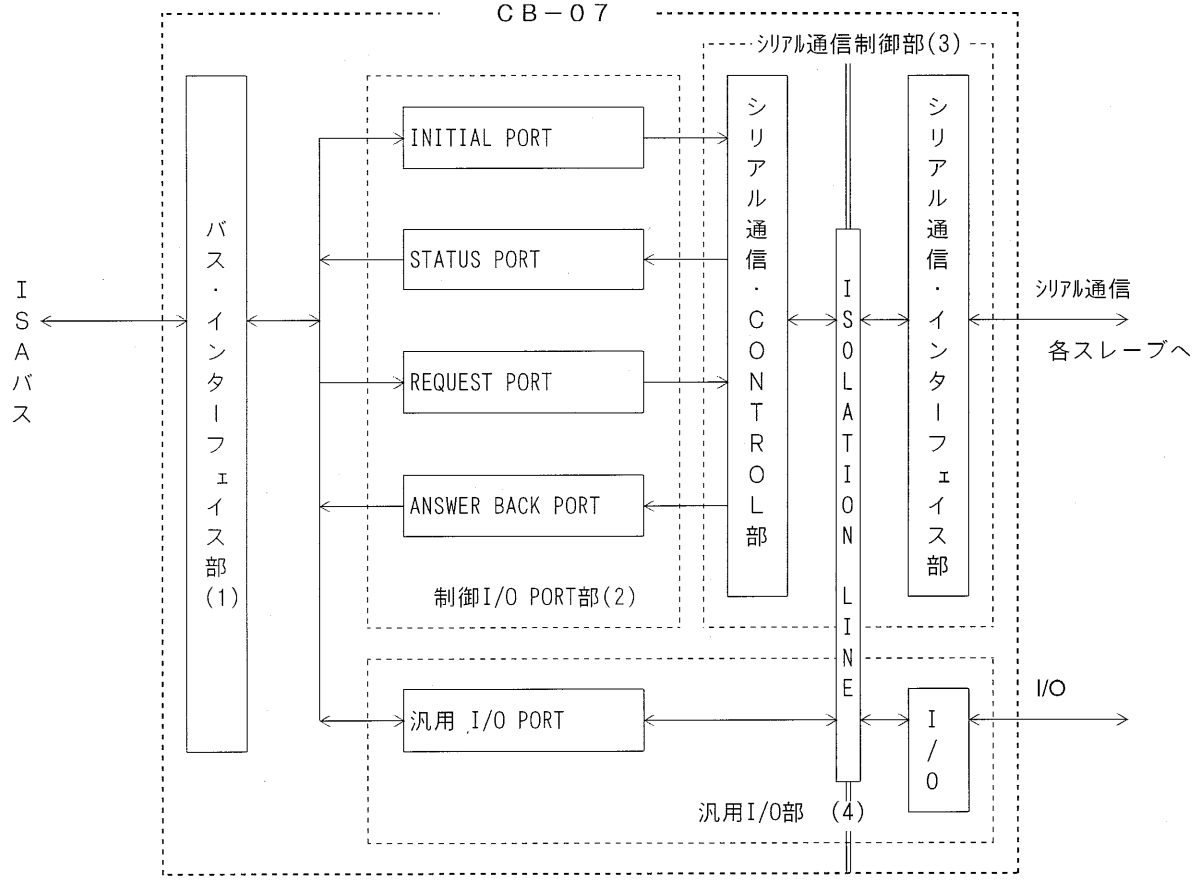
1．概要

®1

CB-07 は、IBM-PC 及びその互換機の ISA バス拡張スロットに直接挿入可能な、AL シリーズ (弊社オリジナル ステッピング／サーボ モータ コントロール システム) 用のマスターボードです。
ユーザはこのマスターボードを介してスレーブのモータコントロール及び汎用 I/O を制御します。
基板形状はコンパクトなハーフサイズ (163 × 93.6) です。

2．基本構成

2-1.機能ブロック図



2-2.各ブロック説明

(1) バス・インターフェース部
ISA バスとのインターフェースブロックです。
IBM-PC 及び互換機の I/O アドレスのデコード回路、アドレス／データバッファなどで構成されています。

(2) 制御 I/O PORT 部
ISA バスとシリアル通信間を受け渡しする制御用の専用 I/O PORT です。
・ INITIAL PORT ---------- マスター及び全スレーブを初期化する為の PORT です。
・ STATUS PORT ---------- シリアル通信 CONTROL 部の状態を読み出す PORT です。
・ REQUEST PORT --------- マスター又はスレーブに実行させる機能を要求する PORT です。
・ ANSWER BACK PORT --- マスター又はスレーブが応答したデータを読み出す PORT です。

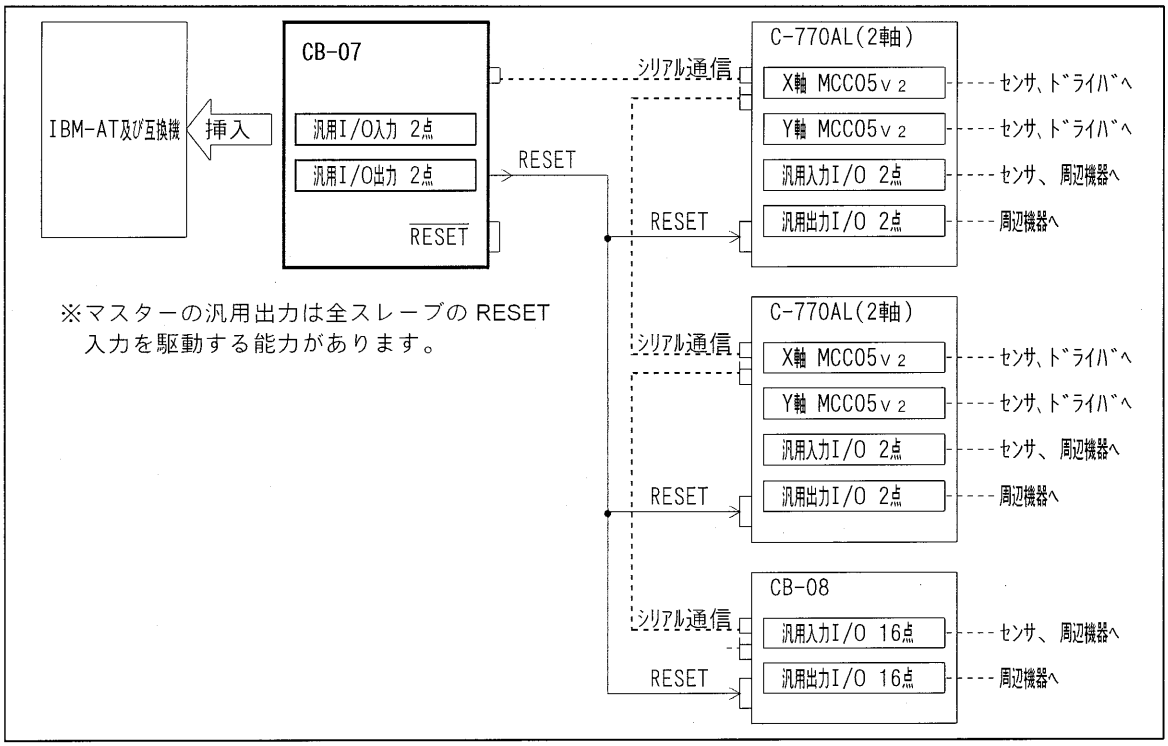
(3) シリアル通信制御部
シリアル通信を制御する回路ブロックです。
シリアル通信インターフェース部は CB-07 内部回路 (PC 電源) とアイソレーションされています。

(4) 汎用 I/O PORT 部
フォトカプラでアイソレーションされた各 2 本の入力／出力です。
+24V のカプラインターフェースですので、リレー・電磁弁及び各スレーブへの制御信号として使用可能です。
このブロックはホスト PC からの制御が他のブロックと完全に独立しています。

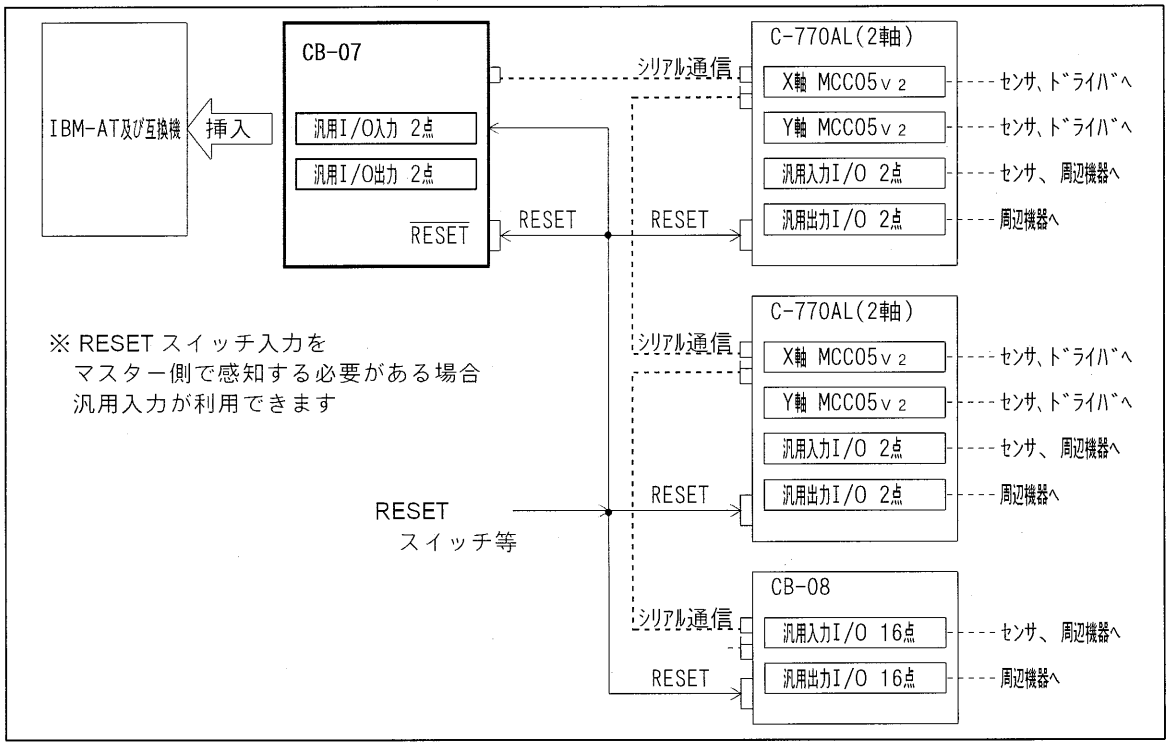
2-3. システム構成例

Ⓡ1

(1) マスターの汎用出力をスレーブへの RESET 信号にした応用例



(2) 外部 RESET 信号をマスターの汎用入力に入力した応用例



3．一般仕様

®1

3-1.AL シリーズ仕様

(1) 準拠規格	RS485(絶縁式)
(2) 通信方式	2 線式半二重
(3) 同期方式	非同期
(4) スレーブ接続局数	1 ～ 15 スレーブ (アドレス設定範囲は 01 H～ 1F H)
(5) 最大配線距離	20m
(6) ボーレート	9765bps/39062bps/156250bps/625000bps
(7) データビット	8 ビット
(8) パリティビット	偶数
(9) ストップビット	1 ビット
(10) 通信エラーチェック機能	パリティチェック、サムチェック

3-2. 定格

(1) 電源電圧：	5V ± 5%	350mA MAX
	24V ± 2V	30mA MAX(フォトカプラ・インターフェイス用)
(2) 周囲温度：	0 ℃～ 45 ℃	
(3) 周囲湿度：	20% ～ 80%	(非結露)

3-3. オプション

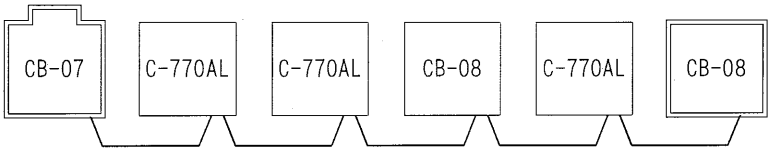
AL シリーズにはオプションが用意されています。
オプションについては別途お問い合わせ下さい。

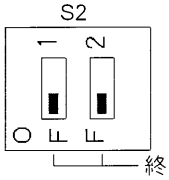
4．ボードの設定

®1

4-1. 終端抵抗の設定 (S2)

基板上のディップスイッチ S2 により AL シリーズネットワーク上の終端抵抗の有無を設定します。
終端抵抗はネットの両端に位置するスレーブ又はマスターのみ有りにして下さい。(下図の □)



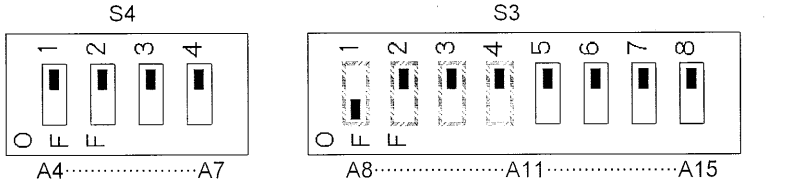


終端抵抗スイッチ　　※S2 の一方の BIT だけを ON にした状態で電源を入れないで下さい。
1,2 とも ON : 抵抗有り
1,2 とも OFF: 抵抗無し (弊社出荷時設定)

4-2. ボードアドレスの設定 (S3,S4)

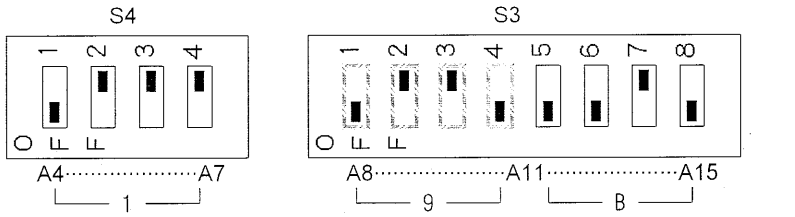
CB-07 の ISA バス上のボートアドレスは、基板上のディップスイッチ S3,S4 により上位 12BIT で設定します。
複数のインターフェイスボードを使用する場合、アドレスが重複しない様に設定して下さい。
尚、 S3 の A8,A9,A10,A11(□ 部) を全て ON(ｱﾄﾞﾚｽ =x0xx H) に設定することは禁止します。
この BIT を 4 つとも ON にすると、 CB-07 はアクセスされません。

上位 12BIT 設定(弊社出荷時設定は下記 010x H です。)



"ON" → 0
"OFF" → 1

(例) 上位 12BIT アドレス =B91 H に設定する場合。



この場合、各制御 I/O PORT のアドレスは次の様になります。

I/O PORT 名称	PORT アドレス
REQUEST PORT	B910 H
INITIAL PORT	B911 H
ANSWER BACK PORT	B912 H
STATUS PORT	B913 H

4-3.AL シリーズ上のアドレス設定

マスター CB-07 は、 AL シリーズ上のアドレスは 00 H 固定で設定は不要です。
AL シリーズに接続するスレーブ側をマスターアドレス 00 H に重複しない様に設定して下さい。

4-4.AL シリーズ通信速度設定

マスター CB-07 の通信速度(ボーレート)は、 INITIAL PORT に書き込む初期化コマンドで設定します。
マスターに設定したボーレートに合わせて AL シリーズに接続される全スレーブのボーレートを設定して下さい。

5．I/O ポートの説明

5-1.I/O PORT 表

アドレス	PORT 名称	
×××0H	REQUEST PORT	書き込み
×××1H	INITIAL PORT	書き込み
×××2H	ANSWER BACK PORT	読み出し
×××3H	STATUS PORT	読み出し
×××4H	汎用 I/O PORT	読出/書込

5-2.REQUEST PORT

リクエストを書き込む PORT です。

D^7	D^6	D^5	D^4	D^3	D^2	D^1	D^0

5-3.INITIAL PORT

初期設定機能を指令する PORT です。

D^7	D^6	D^5	D^4	D^3	D^2	D^1	D^0

DATA部
CODE部

5-4.ANSWER BACK PORT

アンサーバックを読み出す PORT です。

D^7	D^6	D^5	D^4	D^3	D^2	D^1	D^0

5-5.STATUS PORT

シリアル通信 CONTROL 部の状態を読み出す PORT です。

					D^2	D^1	D^0

REQRDY BIT　アクティブで0
INIRDY BIT　アクティブで0
ANSRDY BIT　アクティブで0

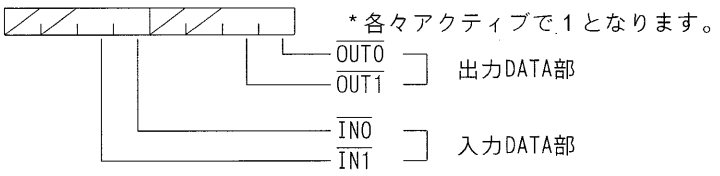
REQRDY BIT ： リクエストが受付可能であることを示します。
INIRDY BIT ： 初期設定機能が指令可能であることを示します。
ANSRDY BIT ： アンサーバックの受取りが可能であることを示します。

5-6. 汎用 I/O PORT

CB-07 は、入力 2 点 [$\overline{IN0}$ ～ $\overline{IN1}$ 入力信号]、出力 2 点 [$\overline{OUT0}$ ～ $\overline{OUT1}$ 出力信号] の汎用 I/O を備えており
ユーザはこれを自由に使用する事が出来ます。
これらの信号は ACTIVE LOW となっており、各々は ACTIVE 時、基板上の LED が点灯する様になっています。

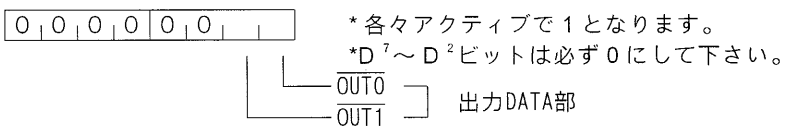
(1) 入力 PORT

入力 PORT は下記に示す通り、入力 DATA 部と出力 DATA 部により構成されています。
外部入力 [$\overline{IN0}$ ～ $\overline{IN1}$] の状態は、入力 DATA 部に取り込まれます。
出力 DATA 部には、現在の出力 PORT の状態 (前回、出力 PORT へ出力した DATA) を取り込みます。



(2) 出力 PORT

出力 PORT は下記に示す構成となっており、下位 2BIT の内容を外部 [$\overline{OUT0}$ ～ $\overline{OUT1}$] へ出力します。



出力 PORT は、 POWER ON/RESET 時、 OFF 出力 (NOT ACTIVE) となります。

6．初期化仕様

®1

6-1. 初期化機能

CB-07 の INITIAL PORT に初期化コマンドが書き込まれると次の様な状態になります。

(1)CB-07 内のシリアル通信 CONTROL 部(通信制御)内部の状態が初期化されます。

(2)CB-07 がリクエストの書き込み処理中又は、アンサーバックの読み出し処理中であっても、処理を中止してリクエスト長入力待ち状態になります。
よって、ユーザアプリケーション先頭で初期化コマンドを実行することにより、ユーザアプリケーションをリスタートしても正常に動作することが可能です。

(3)CB-07 は自動的に全スレーブに対して初期化リクエストを送信します。
スレーブ側は RESET 入力又は電源投入で初期化された場合、マスターからの初期化リクエストを受信するまでは、他のリクエストが送信されるとイニシャルエラーを返します。
よって、**スレーブ側で RESET 入力又は電源投入を行った時には必ず初期化コマンドを実行してマスターの初期化を行って下さい。**

(注) 既に当コマンドによってスレーブが通常状態になっている場合に、再び当機能を実行することは可能です。この場合、スレーブは何も変化しません。

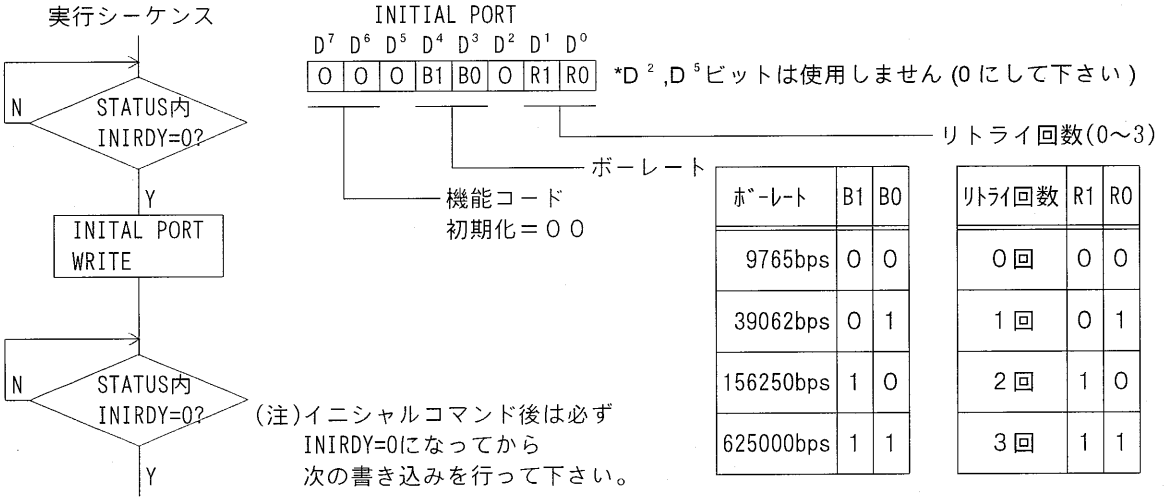
6-2. 初期化方法

機能コード		機能名
D^7	D^6	
0	0	初期化
0	1	使用禁止
1	0	使用禁止
1	1	使用禁止

コマンドの各部分は次のようになっています。

(1) ボーレート　　AL シリーズのボーレートを設定します。

(2) リトライ回数　通信リトライ回数を設定します。



ﾎﾞｰﾚｰﾄ	B1	B0
9765bps	0	0
39062bps	0	1
156250bps	1	0
625000bps	1	1

ﾘﾄﾗｲ回数	R1	R0
0回	0	0
1回	0	1
2回	1	0
3回	1	1

7．リクエスト仕様 ®1

ユーザアプリケーションは、リクエストを REQUEST PORT に書き込んで指令を与えます。
CB-07 は、リクエストの最終バイトが書き込まれるとリクエストのサムチェックコードをリクエストの最終バイトに付加した後、スレーブに対して送信を開始します。
送信が終了するとスレーブからのアンサーバックを待ちます。
アンサーバックの受信が終了するとサムチェックを行い、通信エラーがある場合はアンサーバックのエラー判定結果を **0 以外**にしてユーザへ通知します。
エラーの有無に関わらず、アンサーバックが読み出されるまで次のリクエストを受け付けませんので、ユーザアプリケーションはリクエスト書き込み後、必ず ANSRDY を確認してアンサーバックを読み出さなければなりません。
また、エラーがあった場合、再び初期化を行うまで次のリクエストの書き込みは行えませんので御注意下さい。

7-1. リクエスト、アンサーバック フォーマット

データはすべてバイナリーです。リクエストパラメータ、アンサーバックパラメータは、各リクエスト、アンサーバックにより長さが異なります。
AL シリーズに接続される固有のアドレスを有するものをノードと言い、そのノードにはマスター(00 H)とスレーブ (01 H～ 1F H) があります。

(1) リクエスト フォーマット

1BYTE	1BYTE	1BYTE	1BYTE	
リクエスト長	ノードアドレス	ノードタイプ	リクエストコード	リクエストパラメータ
	←リクエスト長→			

(注)リクエスト長指定バイトはリクエスト長に含みません

・ノードアドレス　：00 H～ 1F H （但しマスターのアドレスは 00 H固定）
・ノードタイプ　：各ノード（スレーブ）の取扱説明書を参照してください。
・リクエストコード　：各ノード（スレーブ）の取扱説明書を参照してください。
・リクエストパラメータ　：各ノード（スレーブ）の取扱説明書を参照してください。

(2) アンサーバック フォーマット

1BYTE	1BYTE	
アンサーバック長	エラー判定結果	アンサーバックパラメータ
	←アンサーバック長→	

(注)アンサーバック長指定バイトはアンサーバック長に含みません

エラー判定結果	エラー名称	エラー内容	エラー種別
００H	(エラーなし)		
０１H～７ＦH	スレーブ固有のエラー	各スレーブの取扱説明書を参照してください	書式エラー
０２H	未定義ﾘｸｴｽﾄｴﾗｰ	未定義のリクエストコードを受信しました	書式エラー
０４H	リクエスト長エラー	リクエスト長がリクエストとあっていません	書式エラー
０５H	フォーマットエラー	パラメータが範囲外です	書式エラー
８０H	イニシャルエラー	スレーブが不正に初期化されている	ハードエラー
８１H	シリアルエラー	スレーブからの受信時のエラー	通信エラー
８２H	タイムアウトエラー	スレーブへの送信時の全エラー	通信エラー
８４H	ﾘｸｴｽﾄ長ﾌｫｰﾏｯﾄｴﾗｰ	リクエスト長が03H～14Hの範囲にない	書式エラー

※０２H、０４H、０５Hは対マスターリクエストに対するエラーです。

エラー種別の意味：

書式エラー … プログラムの書式フォーマットの間違いによるもの。
エラー発生後も REQRDY=L となり、プログラムの続行が可能。リトライは行わない。

通信エラー … シリアル回線上の異常によるもの。エラー発生時には再初期化しなければ動作しない。
リトライ設定時にはリトライを行う。

ハードエラー…ハードの異常によるもの。エラー発生時には再初期化しなければ動作しない。
リトライは行わない。

・アンサーバックパラメータ　：各ノード（スレーブ）の取扱説明書を参照してください。

7-2. 対マスターリクエスト一覧表

®1

ｺｰﾄﾞ	リクエスト名
E0 H	有効アドレスチェックリクエスト
E1 H	スレーブタイプ読み出しリクエスト
E2 H	使用禁止
E3 H	エラー累計回数読み出しリクエスト
E4 H	使用禁止
E5 H	エラー累計回数クリアリクエスト
E6 H	使用禁止
E7 H	使用禁止

7-3. 有効アドレスチェックリクエスト

現在接続を認識しているノードの一覧を読み出します。
接続している場合は対応ビットが 1、未接続の場合は対応ビットが 0 になります。
この情報は初期化機能実行時に作成されるため、初期化機能後の接続状態変更は反映されません。
例．スレーブアドレス 01 H、 0F H、 13 Hが接続（ 00 Hはマスターアドレスなので常に 1 ）

(1)リクエスト

03H	00H	00H	E0H

- E0H：リクエストコードを指定します。
- 00H：スレーブタイプを指定します。(無効。何を入れても構いません。)
- 00H：マスターアドレスを指定します。(マスターは 00 H)
- 03H：リクエスト長を指定します。

(2)アンサーバック

05H	00H	00H	08H	80H	03H

- 03H：ノード(マスターとスレーブを含む)アドレス 00 H～ 07 Hの接続情報です。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
ﾉｰﾄﾞｱﾄﾞﾚｽ	07 H	06 H	05 H	04 H	03 H	02 H	01 H	00 H	未接続 0
接続状態例	0	0	0	0	0	0	1	1	接続 1

- 80H：スレーブアドレス 08 H～ 0F Hの接続情報です。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
ｽﾚｰﾌﾞｱﾄﾞﾚｽ	0F H	0E H	0D H	0C H	0B H	0A H	09 H	08 H	未接続 0
接続状態例	1	0	0	0	0	0	0	0	接続 1

- 08H：スレーブアドレス 10 H～ 17 Hの接続情報です。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
ｽﾚｰﾌﾞｱﾄﾞﾚｽ	17 H	16 H	15 H	14 H	13 H	12 H	11 H	10 H	未接続 0
接続状態例	0	0	0	0	1	0	0	0	接続 1

- 00H：スレーブアドレス 18 H～ 1F Hの接続情報です。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
ｽﾚｰﾌﾞｱﾄﾞﾚｽ	1F H	1E H	1D H	1C H	1B H	1A H	19 H	18 H	未接続 0
接続状態例	0	0	0	0	0	0	0	0	接続 1

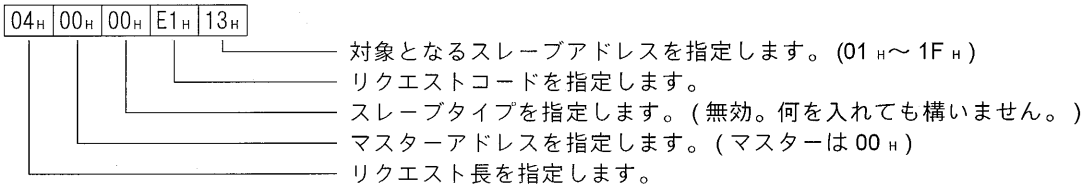
- 00H：リクエストが正常に実行された事を示します。
- 03H：アンサーバック長です。

7-4. スレーブタイプ読み出しリクエスト

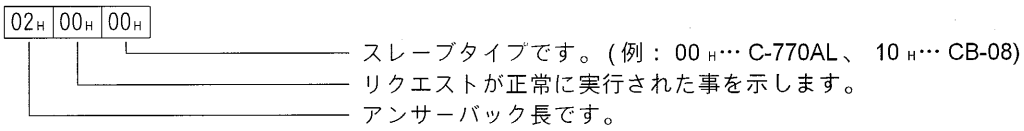
®1

指定したアドレスのスレーブタイプを読み出します。未接続の場合には FF Hを返します。
この情報は初期化機能実行時に作成されるため、初期化機能後の接続状態変更は反映されません。
例．スレーブアドレス 13 Hのスレーブタイプが 00 Hです。(C-770AL)

(1)リクエスト



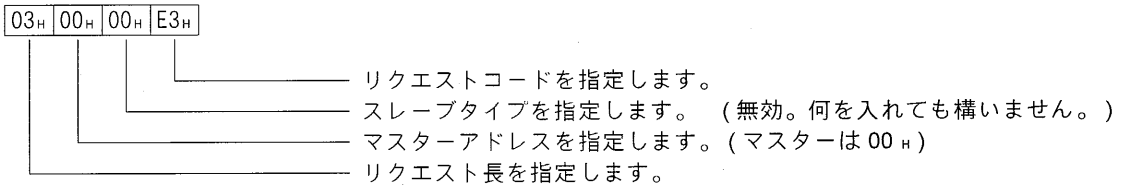
(2)アンサーバック



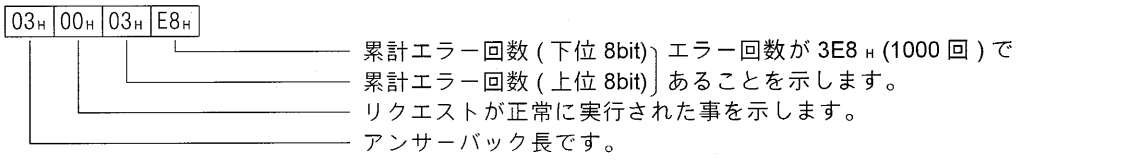
7-5. エラー累計回数読み出しリクエスト

電源投入後に AL シリーズ通信上のエラーが発生した回数の累計を返します。
初期化時に設定したリトライ回数内でリトライを行ったときもエラー回数としてカウントします。
カウントは最大で 65535 回までカウントし、その後エラーが発生してもカウントはストップします。
このカウンタは RESET、電源再投入、又はエラー累計回数クリアリクエストによって 0 にクリアされます。
例．電源投入後、プロトコル異常(ノイズ等)により 1000(3E8 H)回の通信エラーが発生しました。

(1) リクエスト



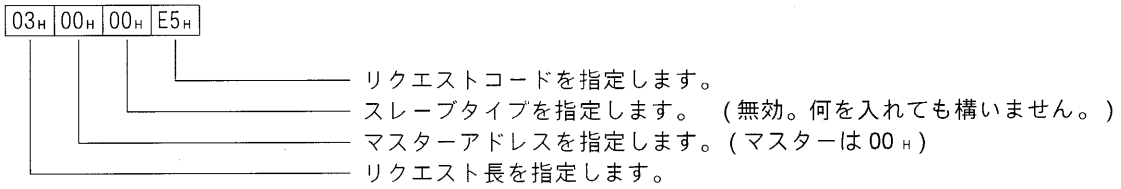
(2) アンサーバック



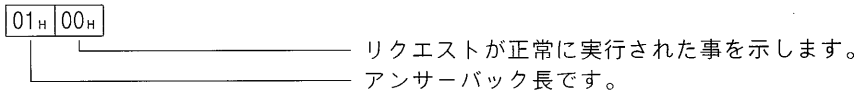
7-6. エラー累計回数クリアリクエスト

CPU 内部で記憶しているエラー累計回数を 0 にクリアします。

(1) リクエスト



(2) アンサーバック



7-7. 対マスターリクエストのエラー判定結果

エラー判定結果	エラー名称	エラー内容	エラー種別
00 H	(エラーなし)		
02 H	未定義ﾘｸｴｽﾄエラー	未定義のリクエストコードを受信しました	書式エラー
04 H	リクエスト長エラー	リクエスト長がリクエストとあっていません	書式エラー
05 H	フォーマットエラー	パラメータが範囲外です	書式エラー
80 H～ FF H	全スレーブ共通エラー	7-1. リクエスト、アンサーバック フォーマット参照	エラーによる

エラー種別の意味：

書式エラー … プログラムの書式フォーマットの間違いによるもの。
エラー発生後も REQRDY=L となり、プログラムの続行が可能。リトライは行わない。

7-8. エラー処理例

(1) エラー判定結果≠ 0 のアンサーバックを読み出します。
(2) ユーザアプリケーションを終了させます。
(3) スレーブに RESET をかけるか、もしくは電源を切ります。
RESET にはマスターの汎用 I/O が使用できます。
(4) 配線、通信設定等原因と思われる箇所をチェックします。
(5) スレーブの電源を入れます。 ((3) で電源を切った場合)
(6) プログラムを再起動します。
(注) 通信、ハードエラーがあった場合、 CB-07 はエラー判定結果≠ 0 のアンサーバックを読み出した後は
初期化機能以外の書き込みを受け付けません。

7-9. イニシャルエラー

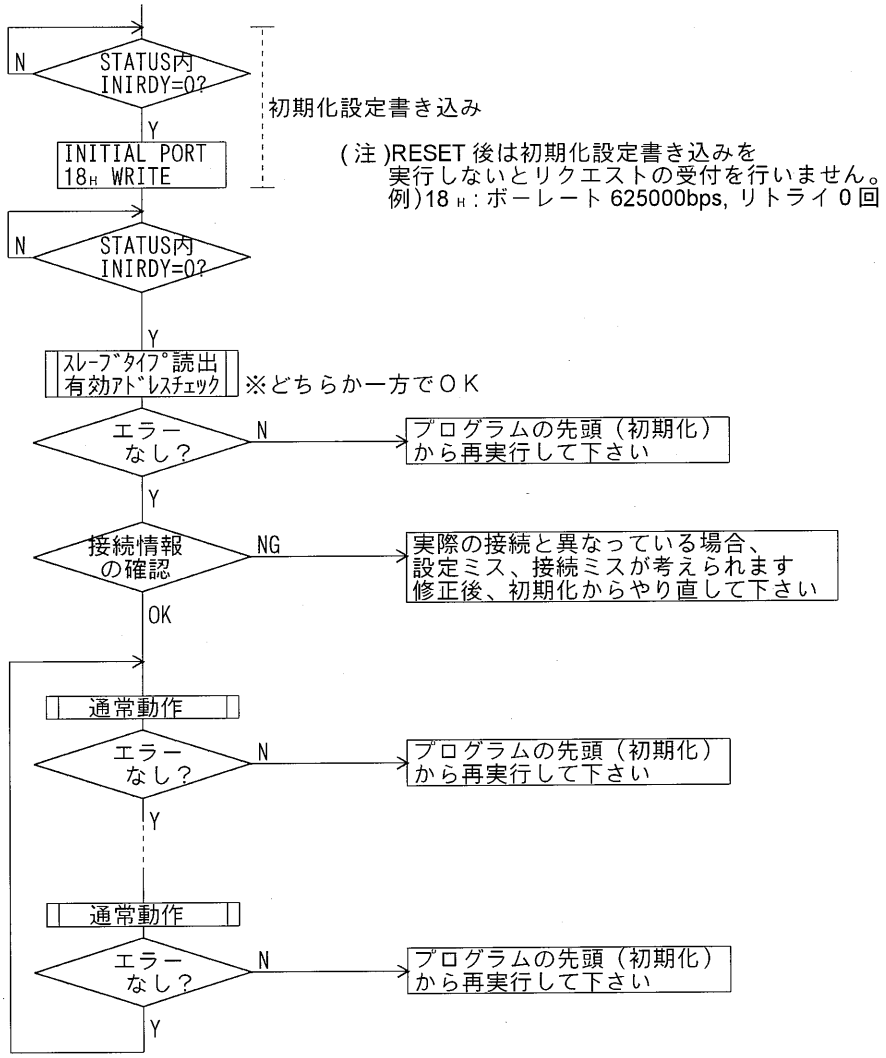
マスターが初期化を行うことにより、自動的に全スレーブに対しイニシャルリクエストを送信します。
スレーブ側はこのリクエストを受信するまではイニシャルエラーを返します。
よって、スレーブの初期化後には必ずマスターの初期化をする必要があります。
この機能により、スレーブ側に瞬時停電等の不意の理由により RESET が入った場合、不正なデータでの動作続行を防止する事が出来ます。

7-10. リトライ機能

ノイズの影響により通信エラーが発生し、場合によってはシステム全体がロック状態になる事も考えられます。
CB-07 にはこの事を考慮して、リトライ機能があります。通信エラーにより、ロック状態になった時 (時間計測によって検出) リトライします。尚、リトライ回数は、初期化機能で設定する事が出来ます。
リトライ回数に達しても、尚かつ通信エラーが発生した場合、アンサーバック内のエラー判定結果≠ 0 にしてユーザアプリケーションに通知します。この時のエラー判定結果は最後に発生したエラーです。
リトライ機能の対象となるエラーは通信上のエラー (シリアルエラー、タイムアウトエラー) のみです。
その他のエラーが発生した場合にはリトライを行わずにエラーステータスを返します。

7-11. 実行シーケンス

(1) 全体の流れ

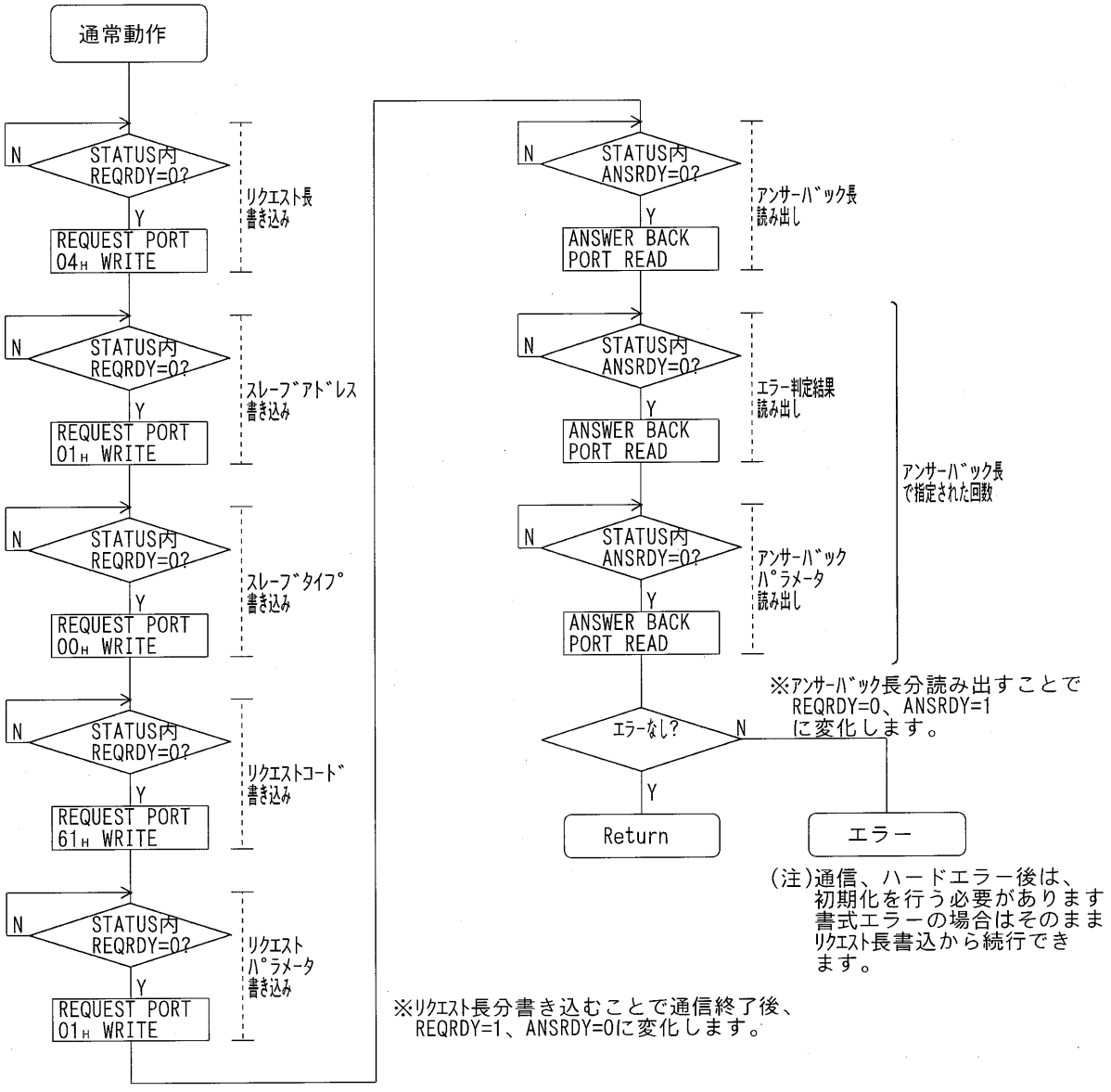


(2) リクエスト書き込み、アンサーバック読み出しのサブルーチン

例 . 次に示すリクエストを書き込み後、アンサーバックを読み出します。

リクエスト長 =04 H 、 スレーブアドレス =01 H 、 スレーブタイプ゜ =00 H(C-770AL)

リクエストコード =61 H(汎用 I/O 指定 BIT 読み出し)、リクエストパラメータ =01 H(OUT0=1,OUT1=0)



(注)通信、ハードエラー後は、初期化を行う必要があります 書式エラーの場合はそのままリクエスト長書込から続行できます。

8．タイミング

®1

8-1.AL シリーズシリアル通信時間

当製品はシリアル通信を行うため、幾つかの要因により、実行時間に差が生じます。
設計時には次の点に留意して実行時間を確認して下さい。
尚、通信速度 =625000bps で且つノイズ等がない通常動作をしている限り、このタイミングは関係ありません。

(1) 通信速度(ボーレート)

通信速度の設定によって下記の様に時間が加算されます。

通信速度 (bps)	9765	39062	156250	625000
書き込み時時間差 (ms)	12.60	3.00	0.60	0
読み出し時時間差 (ms)	25.20	6.00	1.20	0

(2) リトライ回数

リトライ回数を１回以上に設定した場合、リトライ１回あたりの回数に応じて最大で下記の時間が遅れます。
これはリトライ動作による遅れなので、リトライを有効にしていてもノイズがのらない環境であれば実行時間に変化はありません。

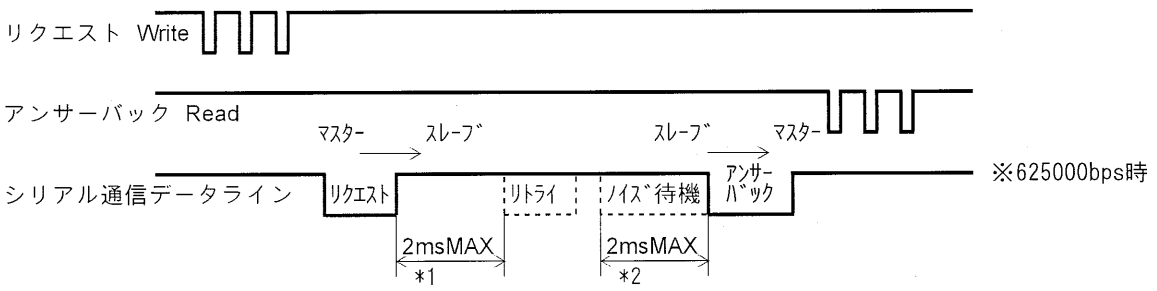
通信速度 (bps)	9765	39062	156250	625000
書き込み時遅れ (ms)	128.00	32.00	8.00	2.00
読み出し時遅れ (ms)	256.00	64.00	16.00	4.00

※リトライ1回あたりの時間

(3) アンサーバック

スレーブがアンサーバックを返すときにシリアル通信データラインにノイズが入っていると、スレーブはノイズがなくなるまで待機します。この時の最大待ち時間は下記の通りです。

通信速度 (bps)	9765	39062	156250	625000
ｱﾝｻｰﾊﾞｯｸ遅れ (ms)	128.00	32.00	8.00	2.00



*1

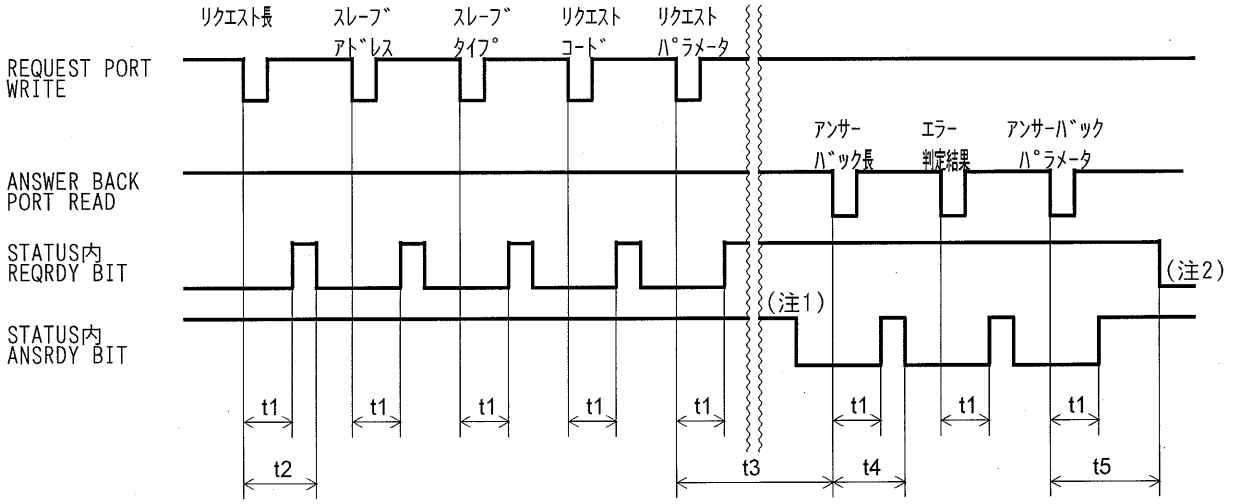
この間にスレーブからアンサーバックが返らないとリトライ設定に応じてリトライを実行します。
時間待ち又は指定回数リトライを実行してもアンサーバックが返らない場合は、マスターはユーザへのアンサーバックの中にエラー判定結果としてタイムアウトエラーを通知します。

*2

スレーブはアンサーバックを返す時点でシリアル通信データラインの状態を確認し、ノイズ等がないクリヤーな状態になるまで待機します。この待機時間の最大がアンサーバック遅れです。
この待機時間を経過してもシリアル通信データラインがクリヤーにならなかった場合は*1の処理になります。

8-2. リクエスト書き込み、アンサーバック読み出し TIMING

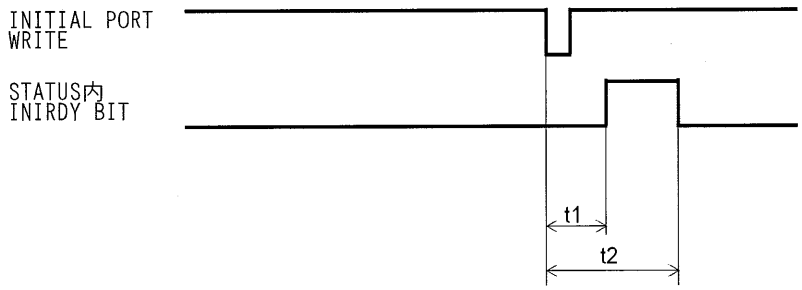
®1



通信速度 (bps)		9765	39062	156250	625000
t1		< 200ns			
t2		REQRDY BIT=0 まで (< 30 μs)			
t3	対マスター	<0.25ms			
	対 C-770AL 例	<28.90ms	<7.30ms	<1.90ms	<0.55ms
	対 CB-08 例	<21.22ms	<5.38ms	<1.42ms	<0.43ms
t4		ANSRDY BIT= 0 まで (< 30 μs)			
t5		REQRDY BIT=0 まで (< 30 μs)			

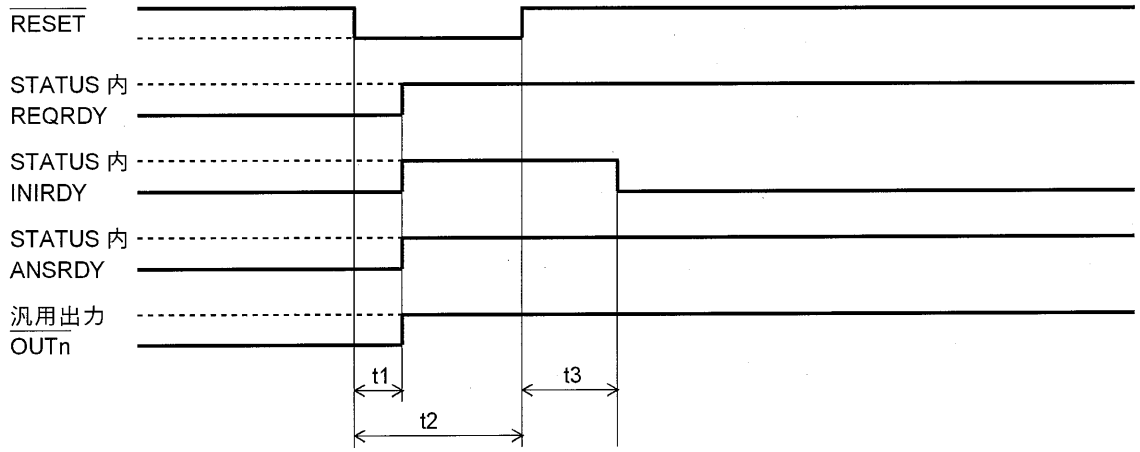
(注 1) スレーブからアンサーバックを受信すると ANSRDY BIT=0 になります。
(注 2) アンサーバックの最終バイトが読み出されると REQRDY BIT=0 になります。
(注 3) ノードのタイプ、各ノードのリクエストによって詳細の時間は異なります。
各ノード（スレーブ）の取扱説明書を参照して下さい。
尚、対マスター (CB-07) の応答時間はシリアル通信時間がない為、通信速度設定に依存しません。

8-3. 初期化 TIMING



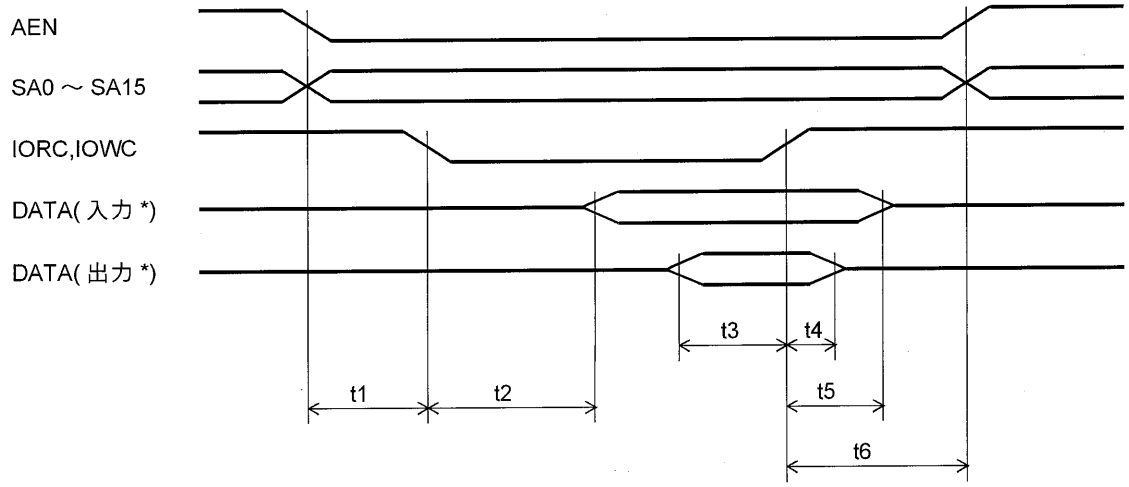
通信速度 (bps)	9765	39062	156250	625000
t1	< 200ns			
t2	<4000ms	<1000ms	<250ms	<63ms

8-4.RESET TIMING



タイミング	I/O $\overline{\text{RESET}}$	システム(PC)RESET
t1	＜ 15ms	＜ 200 μ s
t2	≧ 20ms	≧ 200 μ s
t3	＜ 100ms	

8-5.BUS TIMING



* 入出力はホスト PC 側から見た状態です。

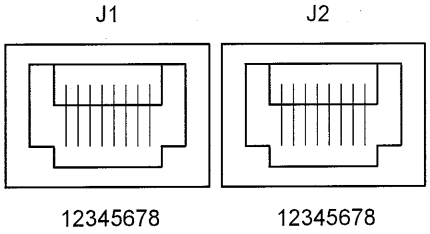
t1	＞ 50ns
t2	＜ 70ns
t3	＞ 10ns
t4	＞ 22ns
t5	5ns ＜ t5 ＜ 25ns
t6	＞ 20ns

9．コネクタ信号表

9-1. シリアル通信コネクタ(J1,J2)

(1) コネクタ型名　TM5RL-88(ヒロセ)

適合ソケット (付属品ではありません)
TM8P-88P,TM11AP1-88(ヒロセ) 及び同等品



(2) 信号表

ピン	方 向	信 号 名	説　明
1	−	N.C	使用禁止
2	−	N.C	使用禁止
3	入/出	+RS485	シリアルデータの入出力信号 (ラインドライバ正論理)
4	−	N.C	使用禁止
5	−	N.C	使用禁止
6	入/出	-RS485	シリアルデータの入出力信号 (ラインドライバ負論理)
7	−	N.C	使用禁止
8	−	N.C	使用禁止

・ J1 と J2 は同じ端子配列です。

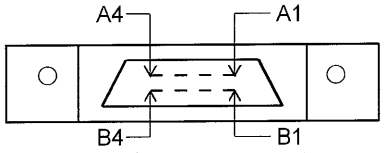
・ J1,J2 のどちらに接続しても構いません。

・マルチドロップ接続する時に J1 又は J2 コネクタを介して他の機器に分岐接続します。

9-2. ユーザ I/O コネクタ(J4)

(1) コネクタ型名　FCN-365P008-AU（富士通）

適合ソケット (付属品)
FCN-361J008-AU（富士通）



(2) 信号表

ピン	方向	信号名	説　明	ピン	方向	信号名	説　明
A1	入	+24V	+24V	B1	−	+24VGND	+24VGND
A2	入	$\overline{\text{IN0}}$	汎用入力ポート 0	B2	出	$\overline{\text{OUT0}}$	汎用出力ポート 0
A3	入	$\overline{\text{IN1}}$	汎用入力ポート 1	B3	出	$\overline{\text{OUT1}}$	汎用出力ポート 1
A4	入	$\overline{\text{RESET}}$	リセット入力	B4	−	N.C	使用禁止

9-3. 基板エッジコネクタ

(1)CN1

端子番号	信号名	方 向	機　　能	端子番号	信号名	方 向	機　　能
A1	―――――			B1	GND		
A2	D7	I/O	ﾃﾞｰﾀﾊﾞｽ	B2	RESDRV	I	RESET
A3	D6	I/O	ﾃﾞｰﾀﾊﾞｽ	B3	+5V		
A4	D5	I/O	ﾃﾞｰﾀﾊﾞｽ	B4	―――――		
A5	D4	I/O	ﾃﾞｰﾀﾊﾞｽ	B5	―――――		
A6	D3	I/O	ﾃﾞｰﾀﾊﾞｽ	B6	―――――		
A7	D2	I/O	ﾃﾞｰﾀﾊﾞｽ	B7	―――――		
A8	D1	I/O	ﾃﾞｰﾀﾊﾞｽ	B8	―――――		
A9	D0	I/O	ﾃﾞｰﾀﾊﾞｽ	B9	―――――		
A10	―――――			B10	GND		
A11	AEN	I	ｱﾄﾞﾚｽｲﾈｰﾌﾞﾙ	B11	―――――		
A12	―――――			B12	―――――		
A13	―――――			B13	IOWC	I	ｺﾏﾝﾄﾞ
A14	―――――			B14	IORC	I	ｺﾏﾝﾄﾞ
A15	―――――			B15	―――――		
A16	SA15	I	ｱﾄﾞﾚｽﾊﾞｽ	B16	―――――		
A17	SA14	I	ｱﾄﾞﾚｽﾊﾞｽ	B17	―――――		
A18	SA13	I	ｱﾄﾞﾚｽﾊﾞｽ	B18	―――――		
A19	SA12	I	ｱﾄﾞﾚｽﾊﾞｽ	B19	―――――		
A20	SA11	I	ｱﾄﾞﾚｽﾊﾞｽ	B20	―――――		
A21	SA10	I	ｱﾄﾞﾚｽﾊﾞｽ	B21	―――――		
A22	SA9	I	ｱﾄﾞﾚｽﾊﾞｽ	B22	―――――		
A23	SA8	I	ｱﾄﾞﾚｽﾊﾞｽ	B23	―――――		
A24	SA7	I	ｱﾄﾞﾚｽﾊﾞｽ	B24	―――――		
A25	SA6	I	ｱﾄﾞﾚｽﾊﾞｽ	B25	―――――		
A26	SA5	I	ｱﾄﾞﾚｽﾊﾞｽ	B26	―――――		
A27	SA4	I	ｱﾄﾞﾚｽﾊﾞｽ	B27	―――――		
A28	SA3	I	ｱﾄﾞﾚｽﾊﾞｽ	B28	―――――		
A29	SA2	I	ｱﾄﾞﾚｽﾊﾞｽ	B29	+5V		
A30	SA1	I	ｱﾄﾞﾚｽﾊﾞｽ	B30	―――――		
A31	SA0	I	ｱﾄﾞﾚｽﾊﾞｽ	B31	GND		

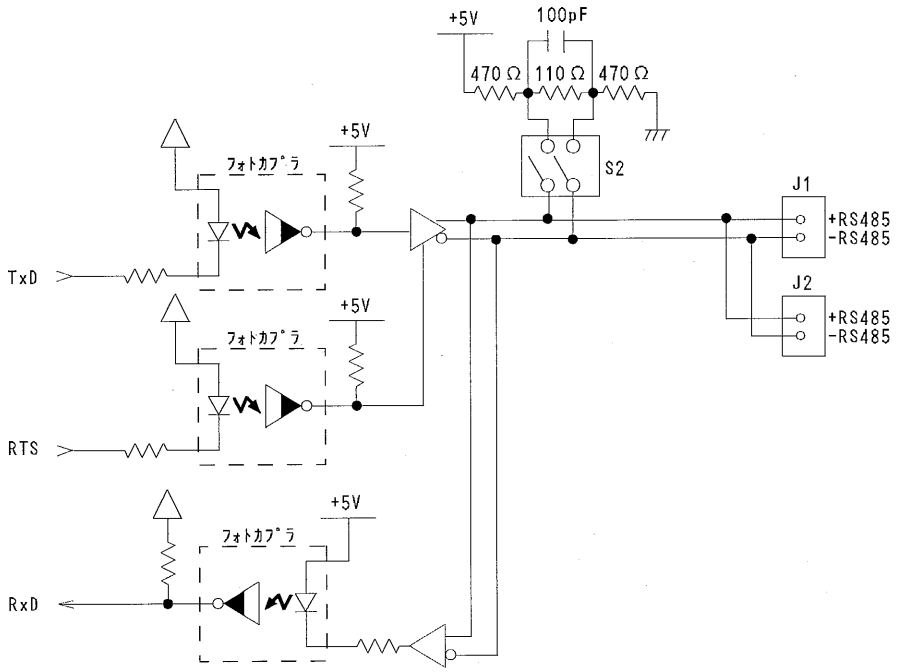
(2)CN2

端子番号	信号名	方 向	機　　能	端子番号	信号名	方 向	機　　能
A1	―――――			B1	―――――		
A2	―――――			B2	―――――		
A3	―――――			B3	―――――		
A4	―――――			B4	―――――		
A5	―――――			B5	―――――		
A6	―――――			B6	―――――		
A7	―――――			B7	―――――		
A8	―――――			B8	―――――		
A9	―――――			B9	―――――		
A10	―――――			B10	―――――		
A11	―――――			B11	―――――		
A12	―――――			B12	―――――		
A13	―――――			B13	―――――		
A14	―――――			B14	―――――		
A15	―――――			B15	―――――		
A16	―――――			B16	+5V		
A17	―――――			B17	―――――		
A18	―――――			B18	GND		

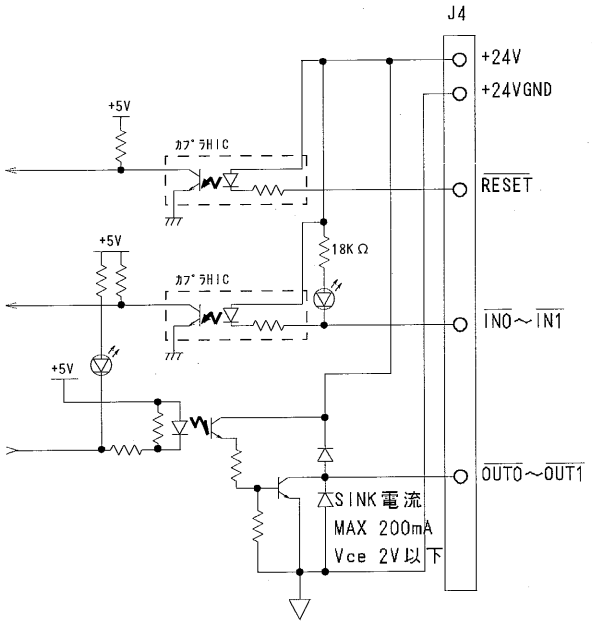
――――― 部は、本ボードでは使用しておりません。

１０．入出力回路

10-1. シリアル通信コネクタ等価回路 (J1,J2)



10-2. ユーザ I/O コネクタ入出力回路 (J4)

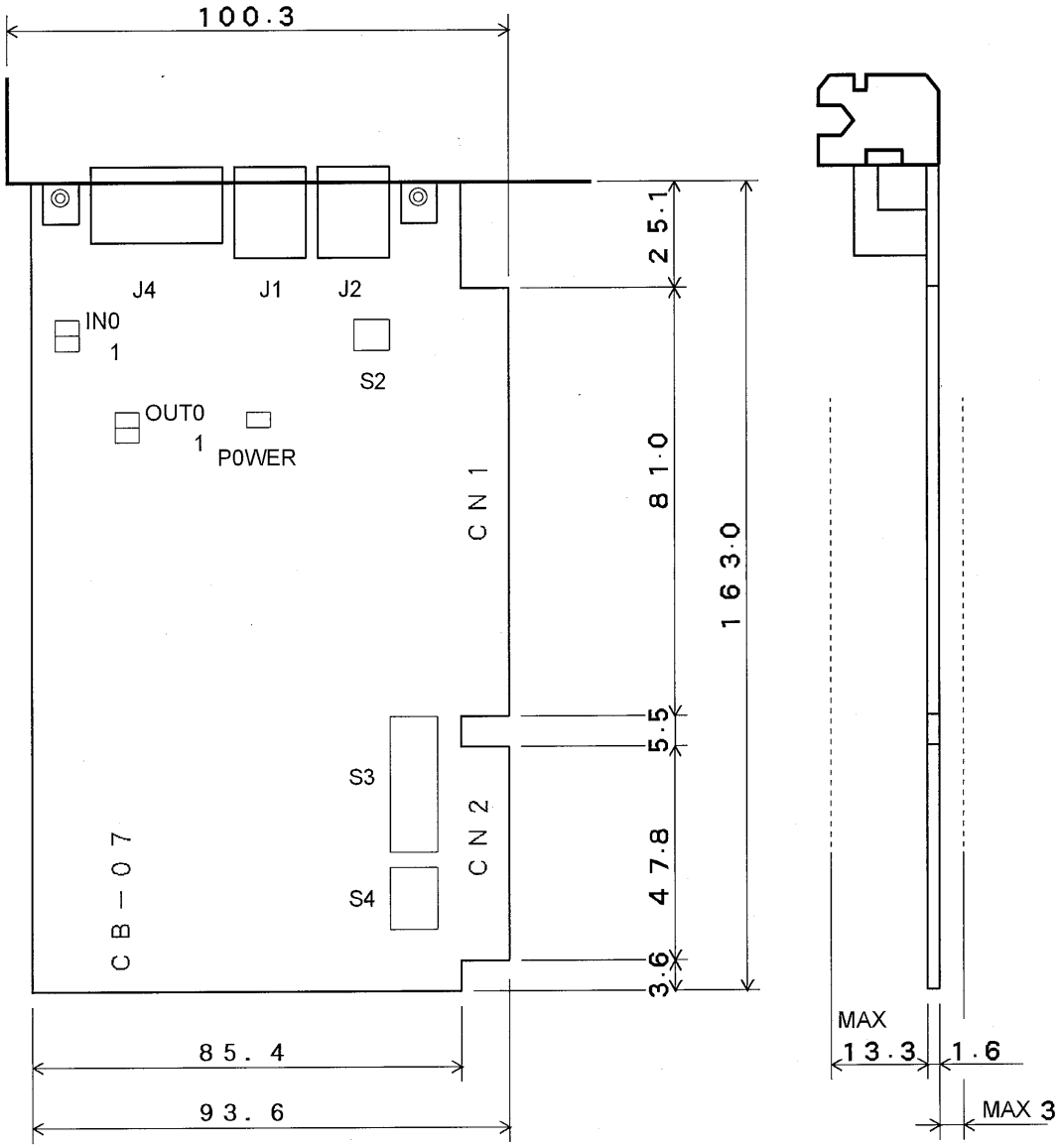


HIGHレベル 1.0mA以下
LOWレベル 3.5mA以上
20V以上 HIGH
4V以下 LOW
電源＝24V時

HIGHレベル 1.3mA以下
LOWレベル 7.0mA以上
20V以上 HIGH
3V以下 LOW
電源＝24V時

HIGHレベル ハイインピーダンス
(リーク電流100μA以下)
LOWレベル 0.6V以下
(SINK180mA時)

１１．基板形状と寸法



単位： mm

J1,J2 ---------- シリアル通信コネクタ(マルチドロップ可能)
J4 ---------- ユーザ I/O 入力コネクタ
CN1,CN2 ---------- キバンエッジコネクタ
POWER ---------- 電源 ON 時に点灯する LED
IN (0,1) ---------- 各汎用入力がアクティブの時点灯する LED
OUT(0,1) ---------- 各汎用出力がアクティブの時点灯する LED
S2 ---------- 終端抵抗設定スイッチ
S3,S4 ---------- I/O ADDRESS 設定スイッチ

１２．サンプル プログラム

本章では CB-07(ISA マスター) と C-770AL(MCC スレーブ) と CB08(I/O スレーブ) を各 1 台接続し、2 軸と I/O を制御するユーザプログラム例を示します。(ANSI 規格 C 言語)

・ CB-07 のベース ADDRESS を B910 H (4-2.のディップスイッチの設定例) とします。
・ C-770AL のスレーブアドレスは、1 を使用します。
・ CB-08 のスレーブアドレスは、3 を使用します。

12-1.AL シリーズ システム設定例

```
/*************************************************************************/
/*                          DEFINITION                                   */
/*************************************************************************/
#define UC      unsigned char
#define US      unsigned short
#define UL      unsigned long

#define ADR     0xb910
#define REQP    ADR + 0x00
#define INIP    ADR + 0x01
#define ANSP    ADR + 0x02
#define STSP    ADR + 0x03
#define GPIP    ADR + 0x04

#define DUMY        0x00    /* DUMMY DATA           */


#define ADR1        0x01    /* C770AL ADDRESS       */
#define ADR2        0x03    /* CB08 ADDRESS         */
#define AXIS_X      0x00    /* C770AL AXIS=X        */
#define AXIS_Y      0x01    /* C770AL AXIS=Y        */
#define TYPE_MASTER 0x00    /* SLAVE_TYPE(MASTER)(INVALID)  */
#define TYPE_C770AL 0x00    /* SLAVE_TYPE(C770AL)   */
#define TYPE_CB08   0x10    /* SLAVE_TYPE(CB08)     */
#define IN10        0x10000 /* SENSOR1(IN10)        */
#define IN11        0x20000 /* SENSOR2(IN11)        */

#define ADDRESS_MAP 0x0000000b  /* CONNECT CHECK COMPARE DATA */
                                /* VALID SLAVE ADDRESS=01,03,MASTAR */

UC trs_ptr[20];     /* TRANS BUFFER     */
UC rev_ptr[20];     /* RECIEVE BUFFER   */

void    c770al_main(void);
void    cb08_main(void);
void    xmcc05inz(void);
void    xjog(void);
void    xscan(void);
void    xabsindex(void);
void    xorg(void);
UC      xsts1read(void);
long    xcntred(void);
void    xdall( UC, UC, UC, UC );
void    xdcom( UC );
void    xcall( UC, UC, UC, UC );
void    request(void);
UL      adr_map(void);
void    cb08iowrite(US);
UL      cb08ioread(void);
void    error_op(void);
```

頻繁に現れるマスターのステータスポート確認、及び C-770AL の MCC05v2 の RDY 確認をマクロ化し、PROGRAM の簡素化を図ります。 ®1

```
#define reqprdy()   while( inp( STSP ) & 0x01 ) /* REQUEST PORT READY WAIT */
#define iniprdy()   while( inp( STSP ) & 0x02 ) /* INITIAL PORT READY WAIT */
#define ansprdy()   while( inp( STSP ) & 0x04 ) /* ANSWER BACK PORT READY WAIT */
#define xmccrdy()   while( xsts1read() & 0x01 ) /* X-AXIS MCC05 READY WAIT  */
```

以下、X 軸についてのみ例を示しますが、Y 軸、及び C-770AL を複数使用した場合も同様の手順です。
当 PROGRAM 例で使用する RAM エリアを下記のように定義します。

```
/*************************************************************************/
/*                          RAM AREA                                     */
/*************************************************************************/
UC      urate;      /* UP RATE No.          */
UC      drate;      /* DOWN RATE No.        */
UL      lspd;       /* LOW SPEED DATA       */
UL      hspd;       /* HIGH SPEED DATA      */
UL      cspd;       /* CONSTANT SPEED DATA  */
long    absdt;      /* OBJECT ADDRESS DATA FOR INDEX DRIVE  */
UC      orgno;      /* ORG TYPE No.         */
UC      offset;     /* OFFSET PULSE DATA    */
UC      ldelay;     /* LIMIT DELAY TIME     */
UC      sdelay;     /* SCAN DELAY TIME      */
UC      jdelay;     /* JOG DELAY TIME       */
```

尚、本章に示す PROGRAM 例はあくまでも参考例であり、必ずしもこれに従う必要はありません。

12-2.AL シリーズ REQUEST 関数例

リクエスト書き込み→アンサーバック読み出しまでを行う関数例です。
ここでは、あらかじめ送信バッファ (trs_ptr) に書き込んだリクエストを送信し、受信したアンサーバックを受信バッファ (rev_ptr) に書き込む仕様とします。

```
/*-----------------------------------------------------------------------*/
/*      REQUEST ROUTINE                                                  */
/*-----------------------------------------------------------------------*/
void    request( void ){
    UC i, cnt;

    cnt = *trs_ptr;     /* REQUEST LENGTH SET   */
    for( i = 0; i <= cnt; i ++ ){
        reqprdy();      /* REQUEST READY WAIT   */
        outp( REQP, *(trs_ptr+i) ); /* REQUEST PORT WRITE   */
    }

    ansprdy();
    cnt = inp(ANSP);    /* ANSWER BACK PORT READ    */
    *rev_ptr = cnt;     /* ANSWER BACK LENGTH SET   */
    for( i = 1; i <= cnt; i ++ ){
        ansprdy();      /* ANSWER BACK READY WAIT   */
        *(rev_ptr+i) = inp(ANSP);   /* ANSWER BACK PORT READ    */
    }

    if( *(rev_ptr+1) ){ /* ERROR COMPERA ≠ 0        */
        error_op();     /* ERROR OPERATION          */
    }
}
```

12-3.AL シリーズ ADDRESS CHECK 関数例 ®1

現在接続を確認しているスレーブアドレスを読み出す関数例です。
返値は 4byte データで最上位 bit がスレーブアドレス =31 Hとなる仕様とします。

```
/*--------------------------------------------------------------------*/
/*      AL SERIES ADDRESS CHECK                                       */
/*--------------------------------------------------------------------*/
UL  adr_map( void )
{
    UL  a;

    *trs_ptr     = 0x03;        /* REQUEST LENGTH              */
    *(trs_ptr+1) = 0x00;        /* MASTER ADDRESS              */
    *(trs_ptr+2) = TYPE_MASTER; /* SLAVE TYPE (INVALID)        */
    *(trs_ptr+3) = 0xe0;        /* ADDRESS CHECK READ REQUEST  */
    request();

    *( (UC *)&a + 3 ) = *(rev_ptr+2);
    *( (UC *)&a + 2 ) = *(rev_ptr+3);
    *( (UC *)&a + 1 ) = *(rev_ptr+4);
    *( (UC *)&a     ) = *(rev_ptr+5);

    return( a );
}
```

12-4.AL シリーズ INITIALIZE PROGRAM 例

プログラム実行時に最初に実行して下さい。
この例は以下の仕様に基づいています。

(1)AL シリーズ設定
通信速度　　… 625000bps
リトライ回数 … 0 回

```
/*--------------------------------------------------------------------*/
/*      AL SERIES INITIALIZE                                          */
/*--------------------------------------------------------------------*/
void    main( void )
{
    iniprdy();                      /* INITIAL PORT READY WAIT */
    outp( INIP, 0x18 + 0x00 );      /* master initial( rate=625000bps, retry=0 ) */
    iniprdy();                      /* INITIAL PORT READY WAIT */

    if( adr_map() != ADDRESS_MAP )  /* CONNECT CHECK    */
    {
        error_op();                 /* ERROR OPERATION  */
    }
}
```

12-5.C-770AL アクセス関数例

(1)C-770AL のドライブポートに一度にデータを書き込む関数例です。

```
/*-------------------------------------------------------------------*/
/*      X DRIVE COMMAND ALL WRITE                                    */
/*-------------------------------------------------------------------*/
void xdall( UC com, UC dt1, UC dt2, UC dt3 )
{
    *trs_ptr     = 0x08;        /* REQUEST LENGTH SET   */
    *(trs_ptr+1) = ADR1;        /* SLAVE ADDRESS SET    */
    *(trs_ptr+2) = TYPE_C770AL; /* SLAVE TYPE SET       */
    *(trs_ptr+3) = 0x10;        /* DRIVE COMMAND ALL WRITE REQUEST SET  */
    *(trs_ptr+4) = AXIS_X;      /* X AXIS SET           */
    *(trs_ptr+5) = com;         /* MCC DRIVE COMMAND SET*/
    *(trs_ptr+6) = dt1;         /* MCC DRIVE DATA1 SET  */
    *(trs_ptr+7) = dt2;         /* MCC DRIVE DATA2 SET  */
    *(trs_ptr+8) = dt3;         /* MCC DRIVE DATA3 SET  */
    request();                  /* REQUEST START        */
}
```

(2)C-770AL のドライブコマンドポートだけにデータを書き込む関数例です。

```
/*-------------------------------------------------------------------*/
/*      X DRIVE COMMAND PORT WRITE                                   */
/*-------------------------------------------------------------------*/
void xdcom( UC com )
{
    *trs_ptr     = 0x05;        /* REQUEST LENGTH SET   */
    *(trs_ptr+1) = ADR1;        /* SLAVE ADDRESS SET    */
    *(trs_ptr+2) = TYPE_C770AL; /* SLAVE TYPE SET       */
    *(trs_ptr+3) = 0x11;        /* DRIVE COMMAND PORT WRITE REQUEST SET */
    *(trs_ptr+4) = AXIS_X;      /* X AXIS SET           */
    *(trs_ptr+5) = com;         /* MCC DRIVE COMMAND SET*/
    request();                  /* REQUEST START        */
}
```

(3)C-770AL のカウンターポートに一度にデータを書き込む関数例です。

```
/*-------------------------------------------------------------------*/
/*      X COUNTER COMMAND ALL WRITE                                  */
/*-------------------------------------------------------------------*/
void xcall( UC com, UC dt1, UC dt2, UC dt3 )
{
    *trs_ptr     = 0x08;        /* REQUEST LENGTH SET       */
    *(trs_ptr+1) = ADR1;        /* SLAVE ADDRESS SET        */
    *(trs_ptr+2) = TYPE_C770AL; /* SLAVE TYPE SET           */
    *(trs_ptr+3) = 0x20;        /* COUNTER COMMAND ALL WRITE REQUEST SET    */
    *(trs_ptr+4) = AXIS_X;      /* X AXIS SET               */
    *(trs_ptr+5) = com;         /* MCC COUNTER COMMAND SET  */
    *(trs_ptr+6) = dt1;         /* MCC COUNTER DATA1 SET    */
    *(trs_ptr+7) = dt2;         /* MCC COUNTER DATA2 SET    */
    *(trs_ptr+8) = dt3;         /* MCC COUNTER DATA3 SET    */
    request();                  /* REQUEST START            */
}
```

(4)C-770AL のステータス 1 ポートの内容を読み出す関数例です。

```
/*-------------------------------------------------------------------*/
/*      X STATUS1 PORT READ                                          */
/*-------------------------------------------------------------------*/
UC xsts1read(){
    *trs_ptr     = 0x04;        /* REQUEST LENGTH SET   */
    *(trs_ptr+1) = ADR1;        /* SLAVE ADDRESS SET    */
    *(trs_ptr+2) = TYPE_C770AL; /* SLAVE TYPE SET       */
    *(trs_ptr+3) = 0x40;        /* STATUS1 PORT READ REQUEST SET    */
    *(trs_ptr+4) = AXIS_X;      /* X AXIS SET           */
    request();                  /* REQUEST START        */
    return( *(rev_ptr+2) );
}
```

(5)PULSE COUNTER DATA READ PROGRAM 例

ここでは読み出した PULSE COUNTER の COUNT 値を RETURN 値とする関数例を示します。

```
/*-------------------------------------------------------------------*/
/*      X-AXIS COUNTER READ                                          */
/*-------------------------------------------------------------------*/
long xcntred( void )
{
    long    a;

    xdcom( 0xfc );              /* PULSE COUNTER PORT SELECT COMMAND OUT*/

    *trs_ptr     = 0x04;        /* REQUEST LENGTH SET   */
    *(trs_ptr+1) = ADR1;        /* SLAVE ADDRESS SET    */
    *(trs_ptr+2) = TYPE_C770AL; /* SLAVE TYPE SET       */
    *(trs_ptr+3) = 0x30;        /* DRIVE DATA PORT ALL READ REQUEST SET */
    *(trs_ptr+4) = AXIS_X;      /* X AXIS SET           */
    request();                  /* REQUEST START        */

    *( (UC *)&a + 2 ) = *(rev_ptr+2);       /* COUNTER MSB IN   */
    *( (UC *)&a + 1 ) = *(rev_ptr+3);
    *( (UC *)&a     ) = *(rev_ptr+4);       /* COUNTER LSB IN   */

    if( (*( (UC *)&a + 2 ) & 0x80 ) != 0 )  /* SIGN BIT ON?     */
    {
        *( (UC *)&a + 3 ) = 0xff;
    }else{
        *( (UC *)&a + 3 ) = 0x00;
    }
    return( a );
}
```

12-6.CB-08 アクセス関数例

(1)CB-08 の I/O データを読み出す関数例です。

```
/*-------------------------------------------------------------------*/
/*      CB08 I/O READ                                                */
/*-------------------------------------------------------------------*/
UL cb08ioread( void )
{
    UL  a;

    *trs_ptr     = 0x03;        /* REQUEST LENGTH SET   */
    *(trs_ptr+1) = ADR2;        /* SLAVE ADDRESS SET    */
    *(trs_ptr+2) = TYPE_CB08;   /* SLAVE TYPE SET       */
    *(trs_ptr+3) = 0x60;        /* CB08 I/O READ REQUEST SET    */
    request();                  /* REQUEST START        */

    *( (UC *)&a + 3 ) = *(rev_ptr+2);
    *( (UC *)&a + 2 ) = *(rev_ptr+3);
    *( (UC *)&a + 1 ) = *(rev_ptr+4);
    *( (UC *)&a     ) = *(rev_ptr+5);

    return( a );
}
```

(2)CB-08 の I/O データを書き込む関数例です。

```
/*-------------------------------------------------------------------*/
/*      CB08 I/O WRITE                                               */
/*-------------------------------------------------------------------*/
void cb08iowrite( US data )
{
    *trs_ptr     = 0x05;        /* REQUEST LENGTH SET   */
    *(trs_ptr+1) = ADR2;        /* SLAVE ADDRESS SET    */
    *(trs_ptr+2) = TYPE_CB08;   /* SLAVE TYPE SET       */
    *(trs_ptr+3) = 0x50;        /* CB08 I/O WRITE REQUEST SET   */
    *(trs_ptr+4) = *( (UC *)&data   );  /* OUT20-27 SET */
    *(trs_ptr+5) = *( (UC *)&data+1 );  /* OUT10-17 SET */
    request();                  /* REQUEST START        */
}
```

12-7. エラー時処理ルーチン

エラーが発生したときの処理を記述する関数です。
ユーザー各自の処理を記述した後は、プログラムを再起動して下さい。

```
/*-------------------------------------------------------------------*/
/*      ERROR OPERATION                                              */
/*-------------------------------------------------------------------*/
void error_op( void )
{
    /*エラー処理ルーチン*/
}
```

12-8.C-770AL(MCC05$_{V2}$) INITIALIZE PROGRAM 例 ®1

C-770AL の RESET 時に必要に応じて実行して下さい。
この例は以下の仕様に基づいています。

(1)DRIVE 仕様

DRIVE TYPE=L、 LIMIT STOP TYPE= 即時停止、 MOTOR TYPE=STEPPING、 RDYINT TYPE= いかなる場合も出力せず を指定します。

(2)PULSE COUNTER、 COMPARATOR 仕様

PULSE COUNTER は MCC05 DRIVE PULSE で動作させるものとし、 COMPARE REGISTER1 の一致出力を CNTINT に出力する仕様とします。 COMPARE REGISTER1 の検出値は、 10000(2710 H) 番地とし、 COMP STOP TYPE は、減速停止とします。

(3)ADDRESS 仕様

MOTOR の現在 ADDRESS を 1000(3E8 H) 番地として定義し、 PULSE COUNTER にも 1000(3E8 H) を PRESET します。

```
/*-------------------------------------------------------------------*/
/*      X-AXIS MCC05 INITIALIZE                                      */
/*-------------------------------------------------------------------*/
void    xmcc05inz( void )
{
    /** SPEC INITIALIZE1 COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT        */
    xdall( 0x01, 0x28, DUMY, DUMY );    /* SPEC INITIALIZE1 COMMAND OUT */

    /** PULSE COUNTER INITIALIZE COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT        */
    xdall( 0x02, 0x01, 0x20, 0x00 );    /* PULSE COUNTER INITIALIZE COMMAND OUT */

    /** ADDRESS INITIALIZE COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT        */
    xdall( 0x03, 0x00, 0x03, 0xe8 );    /* ADDRESS INITIALIZE COMMAND OUT */

    /** COUNTER PRESET COMMAND **/
    xcall( 0x00, 0x00, 0x03, 0xe8 );    /* COUNTER PRESET COMMAND OUT */

    /** COUNTER REGISTER1 SET COMMAND **/
    xcall( 0x01, 0x00, 0x27, 0x10 );    /* COUNTER REGISTER1 SET COMMAND OUT */
}
```

(注) 前述の設定内容は全て C-770AL の RESET 時、特定の仕様に INITIALIZE されています。従って初期仕様に対して変更が必要な場合のみ上述の処理を行って下さい。初期仕様についての詳細は C-770AL の取扱説明書を参照下さい。

12-9.C-770AL(MCC05$_{V2}$) 実動作プログラム例

(1)JOG DRIVE PROGRAM 例

JOG DRIVE に必要な DATA はありません。従って JOG COMMAND で直接起動することが出来ます。

```
/*-------------------------------------------------------------------*/
/*      X-AXIS +JOG DRIVE                                            */
/*-------------------------------------------------------------------*/
void xjog( void )
{
    xmccrdy();                  /* X-AXIS MCC05 RDY WAIT    */
    xdcom( 0x10 );              /* JOG DRIVE COMMAND OUT    */
}
```

(2)SCAN DRIVE PROGRAM 例 ®1

SCAN DRIVE には URATE,DRATE,LSPD,HSPD の各 DATA が必要となる為、これらの DATA を DRIVE 開始前に予め設定しておく必要があります。尚、これらの RATE,SPEED DATA は一度設定が行われていれば変更が必要な場合を除き再設定は不要です。

```
/*-------------------------------------------------------------------*/
/*      X-AXIS +SCAN DRIVE                                           */
/*-------------------------------------------------------------------*/
void xscan( void )
{
    /** RATE SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x06, DUMY, urate, drate );  /* RATE SET COMMAND OUT     */

    /** LSPD SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x07, *((UC *)&lspd+1), *((UC *)&lspd+2), *((UC *)&lspd+3) );    /* LSPD SET COMMAND OUT */

    /** HSPD SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x08, *((UC *)&hspd+1), *((UC *)&hspd+2), *((UC *)&hspd+3) );    /* HSPD SET COMMAND OUT */

    /** SCAN DRIVE COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdcom( 0x12 );                      /* +SCAN DRIVE COMMAND OUT  */
}
```

(注)RAM エリア urate,drate には RATE DATA TABLE の No. が、又 lspd,hspd には PPS 単位で SPEED DATA が格納されているものとします。

(3) 絶対指定の INDEX DRIVE PROGRAM 例

絶対指定の INDEX DRIVE には URATE,DRATE,LSPD,HSPD の各 DATA が必要となる為、これらの DATA を DRIVE 開始前に予め設定しておく必要があります。尚、これらの RATE,SPEED DATA は一度設定が行われていれば変更が必要な場合を除き再設定は不要です。

又、 DRIVE の目的 ADDRESS は INDEX DRIVE 起動時に設定を行います。この DATA は DRIVE ごとに必ず設定する必要があります。

```
/*-------------------------------------------------------------------*/
/*      X-AXIS ABSOLUTE INDEX DRIVE                                  */
/*-------------------------------------------------------------------*/
void xabsindex( void )
{
    /** RATE SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x06, DUMY, urate, drate );  /* RATE SET COMMAND OUT     */

    /** LSPD SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x07, *((UC *)&lspd+1), *((UC *)&lspd+2), *((UC *)&lspd+3) );    /* LSPD SET COMMAND OUT */

    /** HSPD SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x08, *((UC *)&hspd+1), *((UC *)&hspd+2), *((UC *)&hspd+3) );    /* HSPD SET COMMAND OUT */

    /** ABSOLUTE INDEX DRIVE COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x15, *((UC *)&absdt+1), *((UC *)&absdt+2), *((UC *)&absdt+3) );
                                                    /*ABSOLUTE INDEX DRIVE COMMAND OUT*/
}
```

(注)RAM AREA urate,drate には RATE DATA TABLE の No. が、 lspd,hspd には PPS 単位で SPEED DATA が格納されているものとします。又、 absdt には目的 ADDRESS が格納されているものとします。

(4)ORIGIN DRIVE PROGRAM 例 ®1

ORIGIN DRIVE には URATE,DRATE,LSPD,HSPD,CSPD,OFFSET PULSE,LDELAY,SDELAY,JDELAY の各 DATA が必要となる為、これらの DATA を DRIVE 開始前に予め設定しておく必要があります。尚、これらの DATA は一度設定が行われていれば変更が必要な場合を除き再設定は不要です。

又、ORIGIN DRIVE 時の機械原点検出型式は DRIVE 起動時に設定を行います。この DATA は DRIVE ごとに必ず設定する必要があります。

```
/*-------------------------------------------------------------------*/
/*      X-AXIS ORIGIN DRIVE                                          */
/*-------------------------------------------------------------------*/
void xorg( void )
{
    /** RATE SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x06, DUMY, urate, drate );  /* RATE SET COMMAND OUT     */

    /** LSPD SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x07, *((UC *)&lspd+1), *((UC *)&lspd+2), *((UC *)&lspd+3) );    /* LSPD SET COMMAND OUT */

    /** HSPD SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x08, *((UC *)&hspd+1), *((UC *)&hspd+2), *((UC *)&hspd+3) );    /* HSPD SET COMMAND OUT */

    /** CSPD SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x1a, *((UC *)&cspd+1), *((UC *)&cspd+2), *((UC *)&cspd+3) );    /* CSPD SET COMMAND OUT */

    /** OFFSET PULSE SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x1b, DUMY, DUMY, offset );  /* OFFSET PULSE SET COMMAND OUT */

    /** ORG DELAY SET COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x1c, ldelay, sdelay, jdelay );  /* OFFSET PULSE SET COMMAND OUT */

    /** ORIGIN DRIVE COMMAND **/
    xmccrdy();                          /* X-AXIS MCC05 RDY WAIT    */
    xdall( 0x1e, orgno, DUMY, DUMY );   /* ORIGIN DRIVE COMMAND OUT */
}
```

(注)RAM エリア urate,drate には RATE DATA TABLE の No. が、lspd,hspd,cspd には PPS 単位で SPEED DATA が、offset には OFFSET PULSE 数が、更に ldelay,sdelay,jdelay には各々の DELAY TIME DATA が格納されているものとします。

又、orgno には機械原点検出型式が格納されているものとします。

12-10.CB-08 実動作プログラム例

この例では、IN10 が ON になったときの IN11 の状態を全 OUT に反映する仕様とします。

(1)CB-08 PROGRAM 例

```
/*-------------------------------------------------------------------*/
/*      CB08 MAIN                                                    */
/*-------------------------------------------------------------------*/
void    cb08_main( void )
{
    UL  a;

    a = cb08ioread();
    if( (a & IN10) != 0x00 ){   /* SENSOR1 ON?  */
        if( (a & IN11) != 0x00){    /* SENSOR2 ON?  */
            cb08iowrite( 0xffff );  /* ALL I/O ON   */
        }else{
            cb08iowrite( 0x0000 );  /* ALL I/O OFF  */
        }
    }
}
```

１３．トラブルシューティング

ここでは、CB-07 を使用する上で考えられるトラブル及びその時のチェックポイントを示します。

	現　　象	チェックポイント
1	*INIRDY がいつまで待っても 0 に ならない(電源投入直後)	*$\overline{\text{RESET}}$ 信号に LOW LEVEL が入力されていませんか？ *I/O ADDRESS 設定が合っていますか？ 又、他のボードと ADDRESS が重複していませんか？
2	*REQRDY がいつまで待っても 0 に ならない(電源投入直後)	*INITIAL PORT への書き込みで正しく初期化を行いましたか？ 初期化を実行しない限り、CB-07 はリクエストを受け付けません。
	*REQRDY がいつまで待っても 0 に ならない (アンサーバック読み出し後)	* アンサーバック長で指定されたバイト分読み出しを行いましたか？ 読み出しデータが残っている場合、REQRDY は 0 になりません。 * エラー判定結果が 0 以外ではありませんか？ エラーがあった場合、CB-07 は初期化以外の書き込みを受け付けません。
	*REQRDY がいつまで待っても 0 に ならない (リクエスト書き込み中)	* リクエスト長を正しく指定しましたか？ CB-07 は指定されたバイト分書き込まれると REQRDY=1 として送信を開始します。
3	*ANSRDY がいつまで待っても 0 に ならない (リクエスト書き込み後)	* リクエスト長で指定したバイト分書き込みを行いましたか? 書き込みデータ数が指定より少ない場合、ANSRDY は 0 になりません。
	*ANSRDY がいつまで待っても0に ならない (アンサーバック読み出し中)	* アンサーバック長以上の読み出しをしようとしていませんか ?CB-07 は指定されたバイト分読み出されると ANSRDY=1 として次のリクエストを待ちます。
4	* アンサーバックのエラー判定結果が イニシャルエラー (80 H) である。	* スレーブの初期化後に INITIAL PORT への書き込みで正しく初期化を行いましたか？ 初期化を実行しない限り、スレーブはリクエストを受け付けません。 初期化はスレーブの電源 ON →マスターの初期化書き込みの順序で行って下さい。 * プログラムの実行中にこのエラーが発生した場合、スレーブに予期せぬ RESET が入ったことが考えられます。

１４． CB-07全COMMAND一覧表

®1

14-1. 初期化機能一覧表

初期化時にイニシャルポートに書き込むコードです。

$D^7D^6D^5D^4D^3D^2D^1D^0$	HEX CODE	ボーレート	リトライ回数	参照ページ	備考
0 0 0 0 0 0 0 0	００	9765bps	０回	１０	
0 0 0 0 0 0 0 1	０１	9765bps	１回	１０	
0 0 0 0 0 0 1 0	０２	9765bps	２回	１０	
0 0 0 0 0 0 1 1	０３	9765bps	３回	１０	
0 0 0 0 1 0 0 0	０８	39062bps	０回	１０	
0 0 0 0 1 0 0 1	０９	39062bps	１回	１０	
0 0 0 0 1 0 1 0	０Ａ	39062bps	２回	１０	
0 0 0 0 1 0 1 1	０Ｂ	39062bps	３回	１０	
0 0 0 1 0 0 0 0	１０	156250bps	０回	１０	
0 0 0 1 0 0 0 1	１１	156250bps	１回	１０	
0 0 0 1 0 0 1 0	１２	156250bps	２回	１０	
0 0 0 1 0 0 1 1	１３	156250bps	３回	１０	
0 0 0 1 1 0 0 0	１８	625000bps	０回	１０	
0 0 0 1 1 0 0 1	１９	625000bps	１回	１０	
0 0 0 1 1 0 1 0	１Ａ	625000bps	２回	１０	
0 0 0 1 1 0 1 1	１Ｂ	625000bps	３回	１０	

※上記以外は設定禁止です。

14-2. 対マスターリクエスト一覧表

リクエストポートに書き込むコードです。

$D^7D^6D^5D^4D^3D^2D^1D^0$	HEX CODE	REQUEST　NAME	参照ページ	備考
1 1 1 0 0 0 0 0	Ｅ０	有効アドレスチェックリクエスト	１２	
1 1 1 0 0 0 0 1	Ｅ１	スレーブタイプ読み出しリクエスト	１３	
1 1 1 0 0 0 1 0	Ｅ２	使用禁止	--------	
1 1 1 0 0 0 1 1	Ｅ３	エラー累計回数読み出しリクエスト	１３	
1 1 1 0 0 1 0 0	Ｅ４	使用禁止	--------	
1 1 1 0 0 1 0 1	Ｅ５	エラー累計回数クリアリクエスト	１３	
1 1 1 0 0 1 1 0	Ｅ６	使用禁止	--------	
1 1 1 0 0 1 1 1	Ｅ７	使用禁止	--------	

お問い合わせ先

株式会社メレック　制御機器部　〒193-0834 東京都八王子市東浅川町516-10

技　術　相　談／TEL.(0426)64-5382 FAX.(0426)66-5664

八王子営業所／TEL.(0426)64-5382 FAX.(0426)66-5664
東京営業所／TEL.(042)300-3320 FAX.(042)300-3323
大阪営業所／TEL.(06)6386-5135 FAX.(06)6386-5375