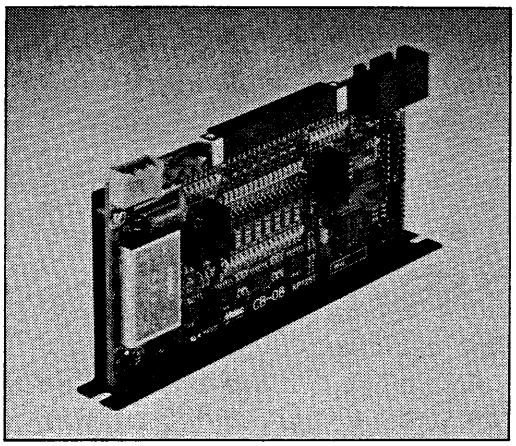
Melec



汎用入出力

CB-08
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。

PR0590-2

はじめに

この取扱説明書は、「ALシリーズ対応I/Oスレーブ CB-08」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、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。

■本製品を実動作させる前に、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．ALシリーズの設定
１１．外形寸法

■電源を供給していないスレーブを接続してシリアル通信を行った場合、通信状態が不安定になることがあります。
通信する時は、全てのスレーブへ電源を供給してください。

目　次

PAGE

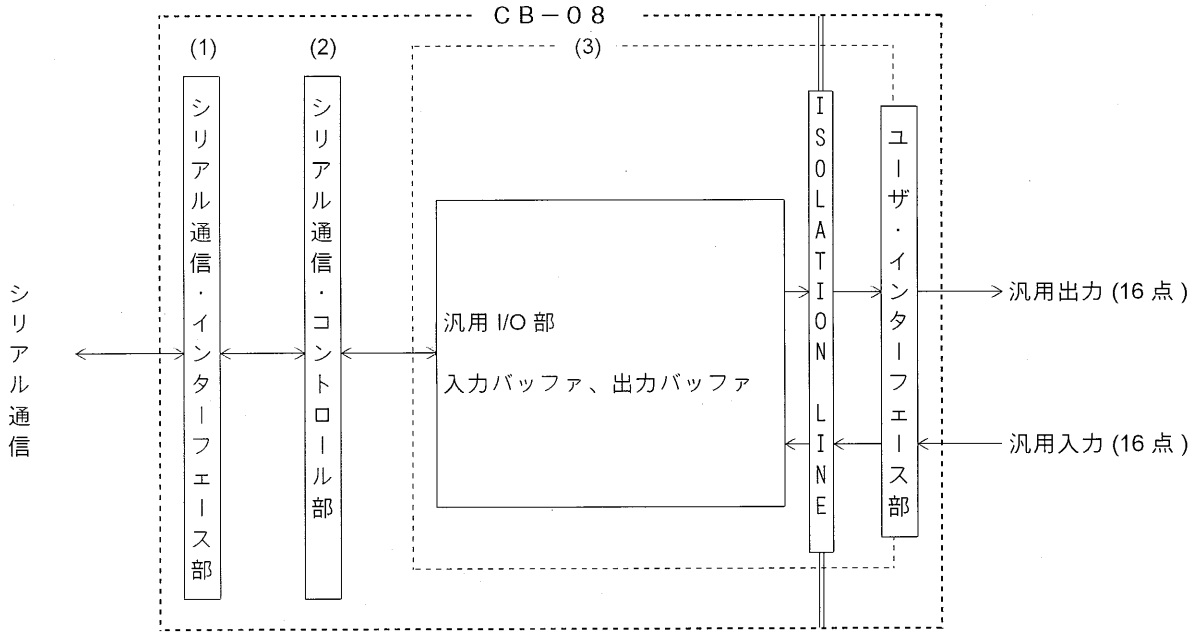
1．概要

®1

CB-08 は、 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) に接続可能な入出力各 16 点の汎用 I/O スレーブです。

2．基本構成

2-1. 機能ブロック図



2-2. 各ブロック説明

(1) シリアル通信・インターフェイス部
シリアル通信インターフェイス ブロックです。

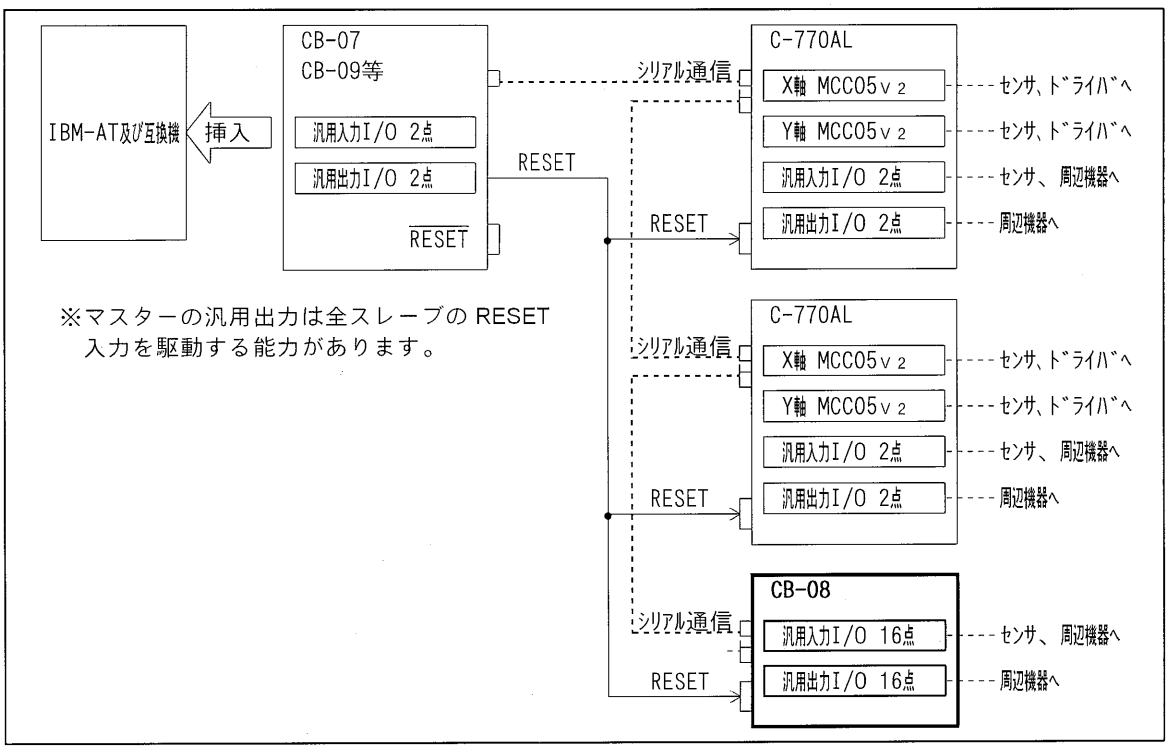
(2) シリアル通信・コントロール部
シリアル通信インターフェイスを制御します。

(3) 汎用 I/O 部
フォトカプラでアイソレーションされた入力／出力、各 16 点の I/O です。
+24V のカプラインターフェイスですので、リレー・電磁弁等の制御が可能です。

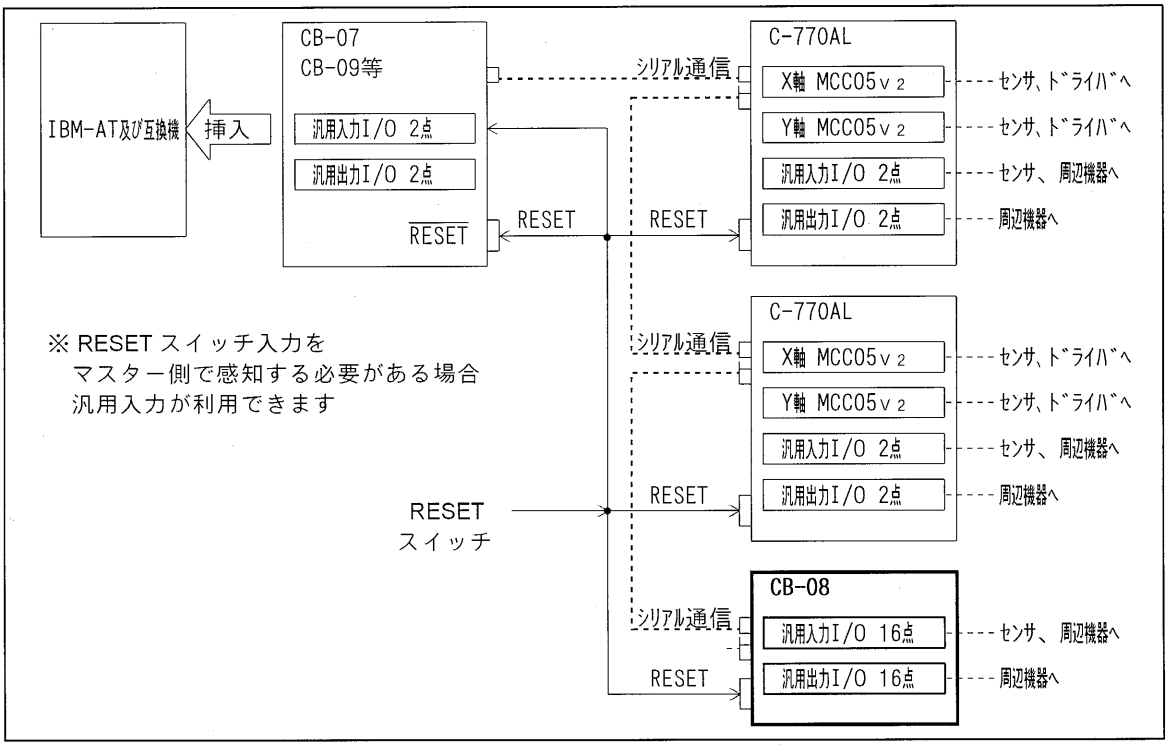
2-3. システム構成例

Ⓡ 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

3-1.AL シリーズ仕様

(1) 準拠規格	RS485(+24V 電源と絶縁)
(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 ：24V ± 2V	300mA MAX
(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． ALシリーズの設定

®1

4-1.ALシリーズ上のアドレス設定(S1)

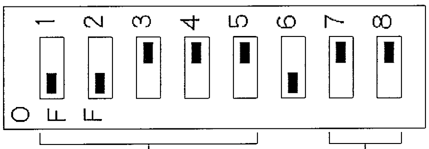
ALシリーズ上のスレーブアドレスを基板上のディップスイッチ(S1)1～5の5BITにより設定します。
マスターの専用アドレス00 H及び他のスレーブアドレスに重複しない様にスレーブ毎に設定して下さい。

4-2.ALシリーズ通信速度設定(S1)

ALシリーズの通信速度(ボーレート)を基板上のディップスイッチ(S1)7,8の2BITで設定します。
マスターに初期化コマンドで設定する通信速度と同じ通信速度をALシリーズに接続する全てのスレーブに対して設定して下さい。

例)次に示す例はスレーブアドレス=28、ボーレート=625000bpsに設定した時のものです。

**S1の設定はリセット、又は電源投入時に有効になりますので設定変更後は必ずリセットを入れて下さい。
この時、必ずマスターの初期化も同時に行って下さい。**



※S1の6ビット目は必ずOFF側にして下さい。

スレーブアドレス

(弊社出荷時設定)→ 01

ｱﾄﾞﾚｽ	1	2	3	4	5
設定禁止	OFF	OFF	OFF	OFF	OFF
01	ON	OFF	OFF	OFF	OFF
02	OFF	ON	OFF	OFF	OFF
03	ON	ON	OFF	OFF	OFF
～	～	～	～	～	～
28	**OFF**	**OFF**	**ON**	**ON**	**ON**
29	ON	OFF	ON	ON	ON
30	OFF	ON	ON	ON	ON
31	ON	ON	ON	ON	ON

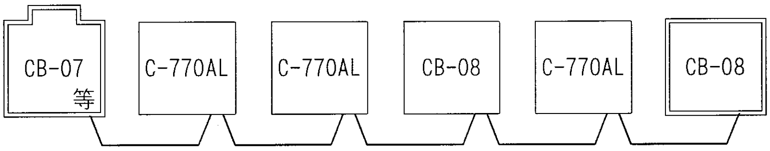
ボーレートスイッチ

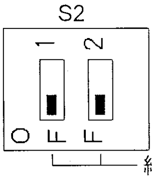
ﾎﾞｰﾚｰﾄ	7	8
9765	OFF	OFF
39062	ON	OFF
156250	OFF	ON
625000	**ON**	**ON**

(弊社出荷時設定) ← 625000

4-3.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　<u>※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。</u>
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）

5．ポート説明

5-1.I/O PORT 表

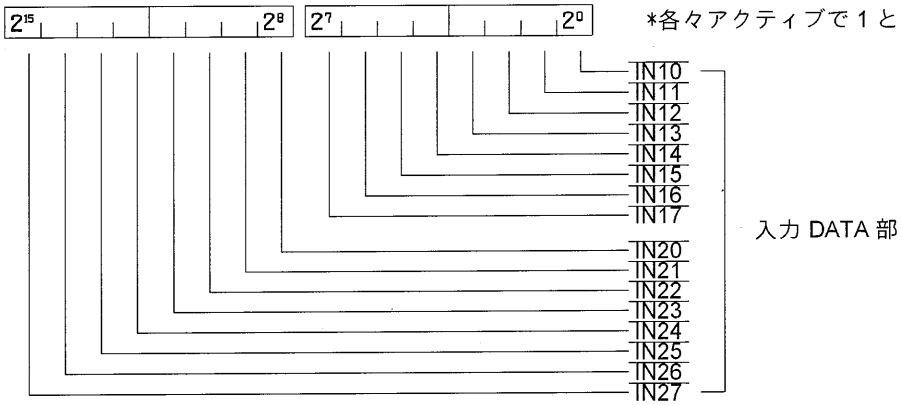
PORT 名称	PORT 機能	
汎用 I/O	出力 (16 点)	書き込み
	入力 (16 点)	読み出し

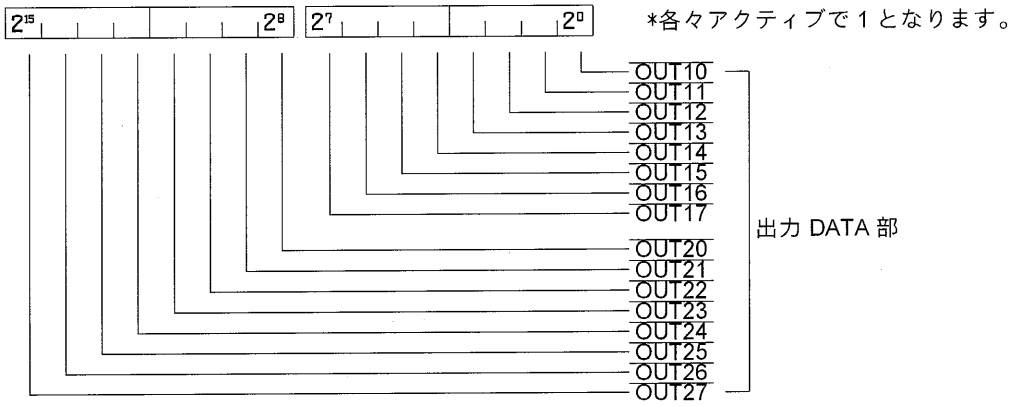
5-2. 汎用 I/O PORT

本製品は、入力 16 点 [$\overline{IN10}$ ～ $\overline{IN17}$、 $\overline{IN20}$ ～ $\overline{IN27}$ 入力信号]、出力 16 点 [$\overline{OUT10}$ ～ $\overline{OUT17}$、$\overline{OUT20}$ ～ $\overline{OUT27}$ 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 時に基板上の LED が点灯する様になっています。

(1) 入力 PORT

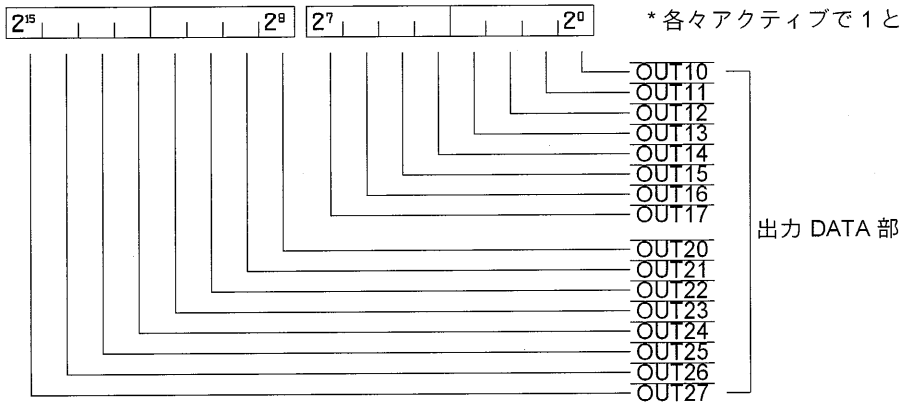
入力 PORT は下記に示す通り、入力 DATA 部と出力 DATA 部により構成されています。
外部入力 [[$\overline{IN10}$ ～ $\overline{IN17}$、 $\overline{IN20}$ ～ $\overline{IN27}$] の状態は、入力 DATA 部に取り込まれます。
出力 DATA 部には、現在の出力 PORT の状態 (前回、出力 PORT へ出力した DATA) を取り込みます。





(2) 出力 PORT

出力 PORT は下記に示す構成となっており書き込まれた内容を外部 [$\overline{OUT10}$ ～ $\overline{OUT17}$、 $\overline{OUT20}$ ～ $\overline{OUT27}$] 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．リクエスト仕様 ®1

マスター(ユーザアプリケーション)から次に示すリクエストを送信して、CB-08 に実装されている汎用 I/O の各ポートにアクセス(書き込み、読み出し)します。
マスターからリクエストを送信すると、CB-08 はアンサーバックを返信します。
よってマスターからリクエストを送信した後は、マスターは必ずアンサーバックを受け取らなければなりません。

6-1. 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。

(1) リクエスト フォーマット

1BYTE	1BYTE	1BYTE	1BYTE	
リクエスト長	スレーブアドレス	スレーブタイプ	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

・スレーブアドレス　　：01 H～1F H
・スレーブタイプ　　：CB-08 のスレーブタイプは、10 Hです。
・リクエストコード　　：リクエスト一覧表を参照してください。
・リクエストパラメータ：詳細については、各リクエストを参照してください。

(2) アンサーバック フォーマット

アンサーバックにはリクエストにエラーがないことを示すアンサーバックとエラーがあることを示すアンサーバックの 2 通りがあります。

◎リクエストに論理上のエラーがないことを示すアンサーバック
リクエストが実行された後に返信されます。(リクエストが正常に実行された事を示します。)

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

エラー判定結果は、00 H(エラーなし)になります。
詳細については、各リクエストを参照してください。

◎リクエストに論理上のエラーがある場合

1BYTE	1BYTE
アンサーバック長	エラー判定結果
	←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
00 H	(エラーなし)		
01 H	スレーブタイプエラー	スレーブタイプが CB-08 を指定していません	書式エラー
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	マスターの取扱説明書を参照してください	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。

・アンサーバックパラメータ　：　各リクエストを参照してください。
・エラーの処理例　　：　マスターの取扱説明書を参照して下さい。

6-2. 対 CB-08 リクエスト一覧表

(1) 汎用 I/O 書き込みリクエスト

ｺｰﾄﾞ	リクエスト名
50 H	汎用 I/O 書き込み
51 H	汎用 I/O 指定ビット書き込み
52 H	汎用 I/O AND 書き込み
53 H	汎用 I/O OR 書き込み

(2) 汎用 I/O 読み出しリクエスト

ｺｰﾄﾞ	リクエスト名
60 H	汎用 I/O 読み出し
61 H	汎用 I/O 指定ビット読み出し

6-3. 汎用 I/O PORT 書き込みリクエスト

汎用 I/O PORT にデータを書き込みます。

例 . 汎用 I/O PORT の OUT27 ～ OUT20 に AB Hを、 OUT17 ～ OUT10 に 37 Hを書き込みます。

(1) リクエスト

05H	01H	10H	50H	ABH	37H

OUT17 ～ OUT10 に書き込むデータを指定します。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
OUT PORT	O_17	O_16	O_15	O_14	O_13	O_12	O_11	O_10
書込ﾃﾞｰﾀ例	0	0	1	1	0	1	1	1

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

OUT27 ～ OUT20 に書き込むデータを指定します。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
OUT PORT	O_27	O_26	O_25	O_24	O_23	O_22	O_21	O_20
書込ﾃﾞｰﾀ例	1	0	1	0	1	0	1	1

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

リクエストコードを指定します。
スレーブタイプを指定します。
スレーブアドレスを指定します。
リクエスト長を指定します。

(2) アンサーバック

01H	00H

リクエストが正常に実行された事を示します。
アンサーバック長です。

6-4. 汎用 I/O PORT 指定ビット書き込みリクエスト

指定した汎用 I/O PORT の 1BIT にデータを書き込みます。指定外の汎用 I/O は変化しません。

例 . 汎用 I/O PORT OUT13 のみ NOT ACTIVE(0) から ACTIVE(1) にします。

(元の状態は OUT17 H～ 10 H =37 Hと仮定)

(1) リクエスト

05H	01H	10H	51H	13H	01H

指定した汎用 I/O に書き込むデータを指定します。
01 H =ACTIVE(Low)　　00 H =NOT ACTIVE(High)

書き込むビットを指定します。(10 H～ 17 H、 20 H～ 27 Hの何れかを指定)

OUT PORT	O_17	O_16	O_15	O_14	O_13	O_12	O_11	O_10
書込前ﾃﾞｰﾀ	0	0	1	1	0	1	1	1
書込後ﾃﾞｰﾀ	0	0	1	1	1	1	1	1

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

リクエストコードを指定します。
スレーブタイプを指定します。
スレーブアドレスを指定します。
リクエスト長を指定します。

(2) アンサーバック

01H	00H

リクエストが正常に実行された事を示します。
アンサーバック長です。

6-5. 汎用 I/O PORT AND 書き込みリクエスト

AND 書き込みで 0 にしたビットに対して汎用 I/O PORT の出力を NOT ACTIVE(0) にします。
AND 書き込みで 1 にしたビットに対しては元のデータのまま汎用 I/O PORT の出力は変化しません。
例.汎用 I/O PORT(OUT17 ～ OUT10) の元のデータ 37 Hに 3C Hで AND をとります。
汎用 I/O PORT(OUT27 ～ OUT20) の元のデータ AB Hに 5A Hで AND をとります。

(1) リクエスト

05H	01H	10H	52H	5AH	3CH

OUT17 ～ OUT10 に AND データを指定します。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
OUT PORT	O_17	O_16	O_15	O_14	O_13	O_12	O_11	O_10
書込前ﾃﾞｰﾀ	0	0	1	1	0	1	1	1
ANDﾃﾞｰﾀ3CH	0	0	1	1	1	1	0	0
書込後ﾃﾞｰﾀ	0	0	1	1	0	1	0	0

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

OUT27 ～ OUT20 に AND データを指定します。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
OUT PORT	O_27	O_26	O_25	O_24	O_23	O_22	O_21	O_20
書込前ﾃﾞｰﾀ	1	0	1	0	1	0	1	1
ANDﾃﾞｰﾀ5AH	0	1	0	1	1	0	1	0
書込後ﾃﾞｰﾀ	0	0	0	0	1	0	1	0

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

リクエストコードを指定します。
スレーブタイプを指定します。
スレーブアドレスを指定します。
リクエスト長を指定します。

(2) アンサーバック

01H	00H

リクエストが正常に実行された事を示します。
アンサーバック長です。

6-6. 汎用 I/O PORT OR 書き込みリクエスト

OR 書き込みで 1 にしたビットに対して汎用 I/O PORT の出力を ACTIVE(1) にします。
OR 書き込みで 0 にしたビットに対しては元のデータのまま汎用 I/O PORT の出力は変化しません。
例.汎用 I/O PORT(OUT17 ～ OUT10) の元のデータ 37 Hに 3C Hで OR をとります。
汎用 I/O PORT(OUT27 ～ OUT20) の元のデータ AB Hに 5A Hで OR をとります。

(1) リクエスト

05H	01H	10H	53H	5AH	3CH

OUT17 ～ OUT10 に OR データを指定します。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
OUT PORT	O_17	O_16	O_15	O_14	O_13	O_12	O_11	O_10
書込前ﾃﾞｰﾀ	0	0	1	1	0	1	1	1
ORﾃﾞｰﾀ3CH	0	0	1	1	1	1	0	0
書込後ﾃﾞｰﾀ	0	0	1	1	1	1	1	1

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

OUT27 ～ OUT20 に OR データを指定します。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
OUT PORT	O_27	O_26	O_25	O_24	O_23	O_22	O_21	O_20
書込前ﾃﾞｰﾀ	1	0	1	0	1	0	1	1
ORﾃﾞｰﾀ5AH	0	1	0	1	1	0	1	0
書込後ﾃﾞｰﾀ	1	1	1	1	1	0	1	1

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

リクエストコードを指定します。
スレーブタイプを指定します。
スレーブアドレスを指定します。
リクエスト長を指定します。

(2) アンサーバック

01H	00H

リクエストが正常に実行された事を示します。
アンサーバック長です。

6-7. 汎用 I/O PORT 読み出しリクエスト

汎用 I/O PORT を読み出します。

例 . 読み出した内容が IN27 ～ IN20 = AC H、 IN17 ～ IN10 = 56 Hであることを示しています。

又、現在出力している状態は OUT17 ～ OUT10=3F H、OUT27 ～ OUT20=FB Hであることを示します。

(1) リクエスト

03H	01H	10H	60H

60H — リクエストコードを指定します。
10H — スレーブタイプを指定します。
01H — スレーブアドレスを指定します。
03H — リクエスト長を指定します。

(2) アンサーバック

05H	00H	ACH	56H	FBH	3FH

3FH — 現在の出力 PORT の状態 (OUT17 ～ OUT10) を取り込みます。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
OUT PORT	O_17	O_16	O_15	O_14	O_13	O_12	O_11	O_10
読出ﾃﾞｰﾀ例	0	0	1	1	1	1	1	1

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

FBH — 現在の出力 PORT の状態 (OUT27 ～ OUT20) を取り込みます。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
OUT PORT	O_27	O_26	O_25	O_24	O_23	O_22	O_21	O_20
読出ﾃﾞｰﾀ例	1	1	1	1	1	0	1	1

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

56H — 読み出した IN17 ～ IN10 の内容です。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
IN PORT	IN17	IN16	IN15	IN14	IN13	IN12	IN11	IN10
読出ﾃﾞｰﾀ例	0	1	0	1	0	1	1	0

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

ACH — 読み出した IN27 ～ IN20 の内容です。

BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
IN PORT	IN27	IN26	IN25	IN24	IN23	IN22	IN21	IN20
読出ﾃﾞｰﾀ例	1	0	1	0	1	1	0	0

※ 1=ACTIVE(Low)　0=NOT ACTIVE(High)

00H — リクエストが正常に実行された事を示します。
05H — アンサーバック長です。

6-8. 汎用 I/O PORT 指定ビット読み出しリクエスト

指定した汎用 I/O PORT にデータを読み出します。

例 . 汎用 I/O PORT IN13 を読み出し

(1) リクエスト

04H	01H	10H	61H	13H

13H — 読み出すビットを指定します。(10 H～ 17 H、 20 H～ 27 H)
61H — リクエストコードを指定します。
10H — スレーブタイプを指定します。
01H — スレーブアドレスを指定します。
04H — リクエスト長を指定します。

(2) アンサーバック

02H	00H	01H

01H — 指定した汎用 I/O のデータです。
01 H =ACTIVE(Low)　　00 H =NOT ACTIVE(High)
00H — リクエストが正常に実行された事を示します。
02H — アンサーバック長です。

6-9. 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 (コード 80 H) 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．タイミング

®1

7-1.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。

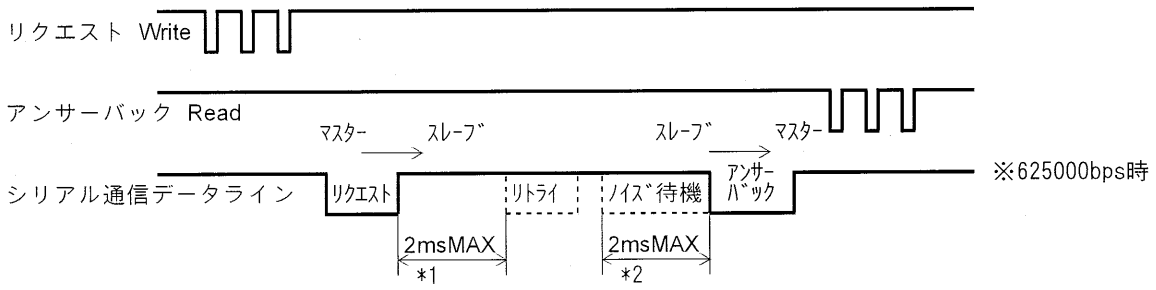
通信速度 (bps)	9765	39062	156250	625000
書き込み時遅れ (ms)	128.00	32.00	8.00	2.00
読み出し時遅れ (ms)	256.00	64.00	16.00	4.00

※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



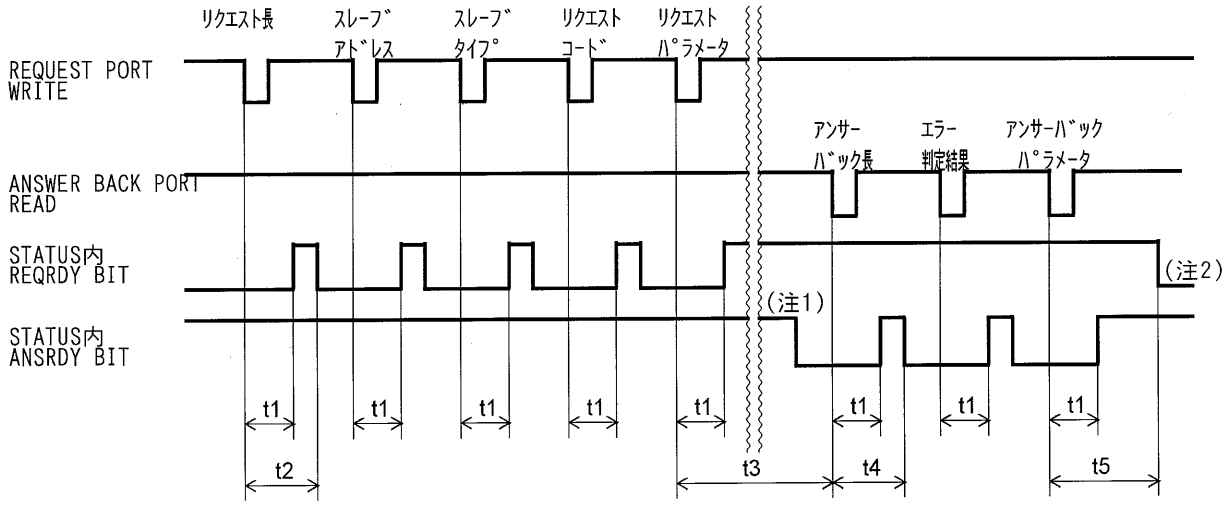
*1

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。

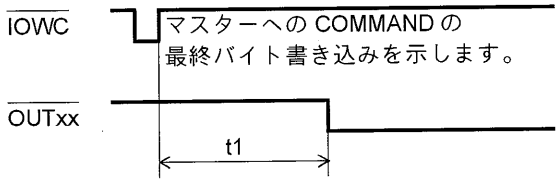
7-2. リクエスト書き込み、アンサーバック読み出し TIMING(マスターの TIMING)



通信速度 (bps)		9765	39062	156250	625000
t1		< 200n s			
t2		REQRDY BIT=0 まで (< 30 μ s)			
t3	汎用I/O読出し	<21.22ms	<5.38ms	<1.42ms	<0.43ms
	指定BIT読出し	<19.30ms	<4.90ms	<1.30ms	<0.40ms
t4		ANSRDY BIT= 0 まで (< 30 μ s)			
t5		REQRDY BIT=0 まで (< 30 μ s)			

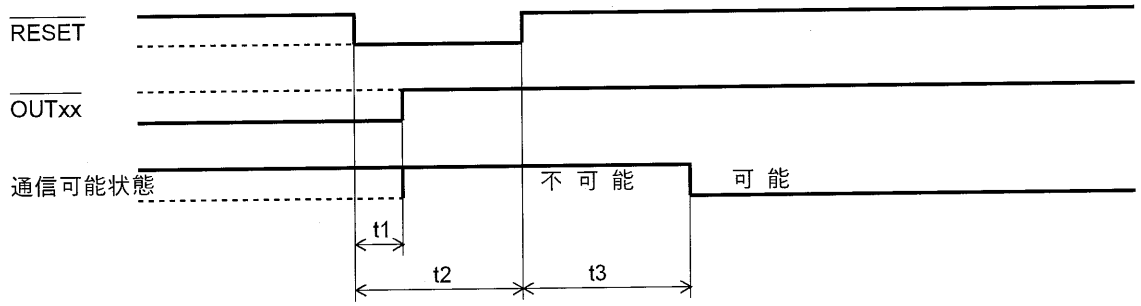
(注 1) スレーブからアンサーバックを受信すると ANSRDY BIT=0 になります。
(注 2) アンサーバックの最終バイトが読み出されると REQRDY BIT=0 になります。

7-3. 汎用出力 TIMING



通信速度 (bps)	9765	39062	156250	625000
t1	<11.57ms	<2.93ms	<0.77ms	<0.23ms

7-4.RESET TIMING



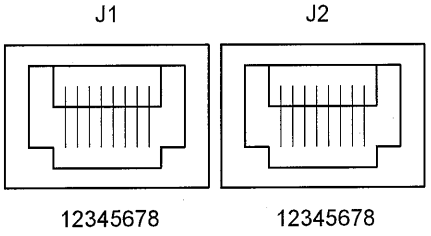
t1	< 15ms
t2	≧ 20ms
t3	< 22ms

8．コネクタ信号表

8-1. 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。

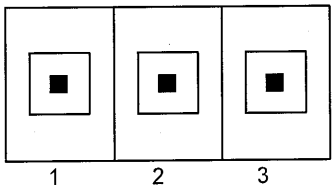
・ マルチドロップ接続する時に J1 又は J2 コネクタを介して他の機器に分岐接続します。

8-2. 電源コネクタ(J3)

(1) コネクタ型名　MSTBA2.5/3-G-5.08(ﾌｪﾆｯｸｽ)

適合ソケット (付属品)
MSTB2.5/3-ST-5.08(ﾌｪﾆｯｸｽ)

適合線材
AWG20(0.5SQ) ～ AWG12(3.3SQ)
AWG20(0.5SQ) ～ AWG15(1.5SQ) : ﾃﾞｴｼﾞﾁｪｰﾝ時



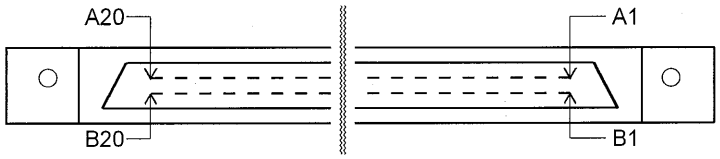
(基板外側から見た図)

(2) 信号表

ピン	方 向	信 号 名	説　明
1	−	F.G.	F.G.
2	−	GND	電源 GND
3	入	+24V	DC +24V 電源

8-3. ユーザ I/O コネクタ (J4)

(1) コネクタ型名　FCN-365P040-AU(富士通)　　適合ソケット (付属品)
FCN-361J040-AU(富士通)



(2) 信号表

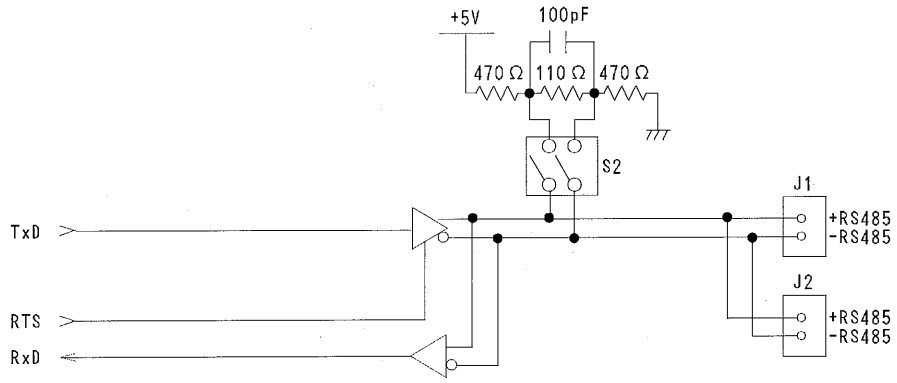
※信号はカプラ絶縁されています。

ピン	方向	信号名	説　明	ピン	方向	信号名	説　明
A1	−	GND	リターン GND　(注 1)	B1	−	GND	リターン GND　(注 1)
A2	入	$\overline{IN10}$	汎用入力ポート 1-2 ^{0}	B2	出	$\overline{OUT10}$	汎用出力ポート 1-2 ^{0}
A3	入	$\overline{IN11}$	汎用入力ポート 1-2 ^{1}	B3	出	$\overline{OUT11}$	汎用出力ポート 1-2 ^{1}
A4	入	$\overline{IN12}$	汎用入力ポート 1-2 ^{2}	B4	出	$\overline{OUT12}$	汎用出力ポート 1-2 ^{2}
A5	入	$\overline{IN13}$	汎用入力ポート 1-2 ^{3}	B5	出	$\overline{OUT13}$	汎用出力ポート 1-2 ^{3}
A6	入	$\overline{IN14}$	汎用入力ポート 1-2 ^{4}	B6	出	$\overline{OUT14}$	汎用出力ポート 1-2 ^{4}
A7	入	$\overline{IN15}$	汎用入力ポート 1-2 ^{5}	B7	出	$\overline{OUT15}$	汎用出力ポート 1-2 ^{5}
A8	入	$\overline{IN16}$	汎用入力ポート 1-2 ^{6}	B8	出	$\overline{OUT16}$	汎用出力ポート 1-2 ^{6}
A9	入	$\overline{IN17}$	汎用入力ポート 1-2 ^{7}	B9	出	$\overline{OUT17}$	汎用出力ポート 1-2 ^{7}
A10	−	N.C	使用禁止	B10	−	N.C	使用禁止
A11	−	GND	リターン GND　(注 1)	B11	−	GND	リターン GND　(注 1)
A12	入	$\overline{IN20}$	汎用入力ポート 2-2 ^{0}	B12	出	$\overline{OUT20}$	汎用出力ポート 2-2 ^{0}
A13	入	$\overline{IN21}$	汎用入力ポート 2-2 ^{1}	B13	出	$\overline{OUT21}$	汎用出力ポート 2-2 ^{1}
A14	入	$\overline{IN22}$	汎用入力ポート 2-2 ^{2}	B14	出	$\overline{OUT22}$	汎用出力ポート 2-2 ^{2}
A15	入	$\overline{IN23}$	汎用入力ポート 2-2 ^{3}	B15	出	$\overline{OUT23}$	汎用出力ポート 2-2 ^{3}
A16	入	$\overline{IN24}$	汎用入力ポート 2-2 ^{4}	B16	出	$\overline{OUT24}$	汎用出力ポート 2-2 ^{4}
A17	入	$\overline{IN25}$	汎用入力ポート 2-2 ^{5}	B17	出	$\overline{OUT25}$	汎用出力ポート 2-2 ^{5}
A18	入	$\overline{IN26}$	汎用入力ポート 2-2 ^{6}	B18	出	$\overline{OUT26}$	汎用出力ポート 2-2 ^{6}
A19	入	$\overline{IN27}$	汎用入力ポート 2-2 ^{7}	B19	出	$\overline{OUT27}$	汎用出力ポート 2-2 ^{7}
A20	入	$\overline{RESET}$	リセット入力	B20	−	N.C	使用禁止

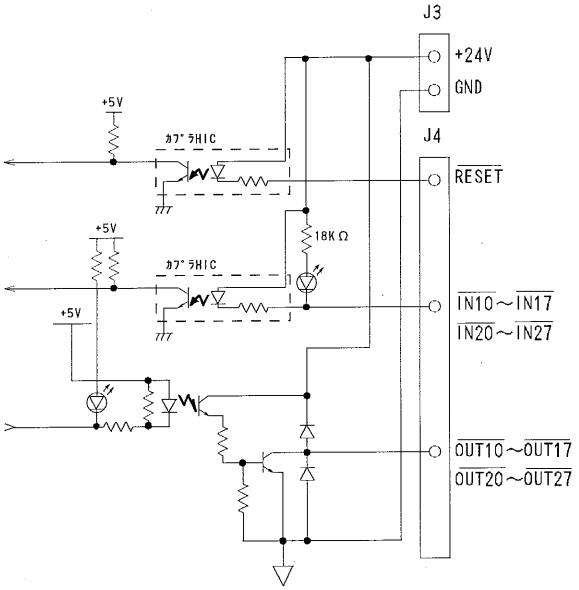
(注 1)+24V 電源の GND と内部で接続されています。

9．入出力回路

9-1. シリアル通信コネクタ等価回路 (J1,J2)



9-2. ユーザ I/O コネクタ入出力回路 (J3,J4)

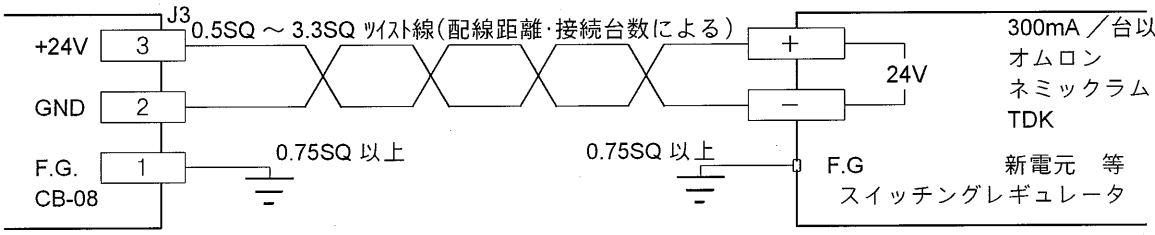


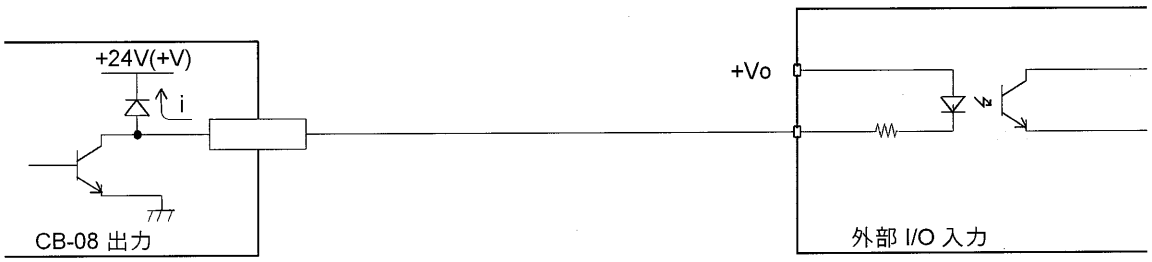
HIGHレベル 1.0mA以下
LOWレベル 3.5mA以上
[20V以上 HIGH
4V以下 LOW
電源＝24V時]
HIGHレベル 1.3mA以下
LOWレベル 7.0mA以上
[20V以上 HIGH
3V以下 LOW
電源＝24V時]
HIGHレベル ハイインピーダンス
(リーク電流100μA以下)
LOWレベル 0.6V以下
(最大SINK100mA時)

１０．接続

10-1. 電源との接続例

CB-08 の電源は、 DC+24V です。この電源は、カプラインターフェイス用と内部コントロール用 (24V とは絶縁されています。) として使用されています。
電源としては、電圧 +24 ± 2V、出力容量　300mA ／台以上の安定化電源を御使用下さい。
CB-08 の電源の配線は、 0.5SQ 以上の線材を使用し、より線にして下さい。
又、他機器の主回路、動力線とは別束し、 50mm 以上離して下さい。



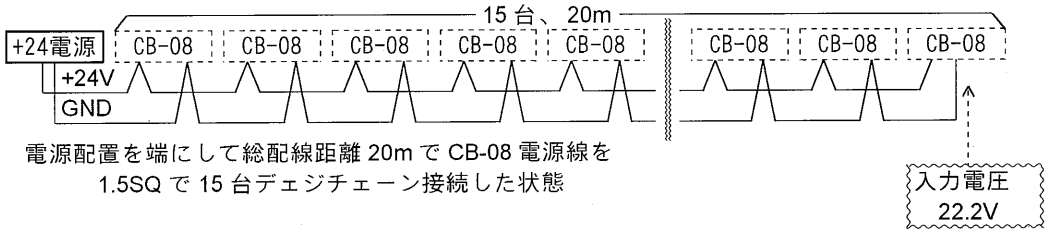


※ 外部 I/O への供給電源 (+V。) ＞ CB-08 への供給電源 (+V) となると出力回路の保護ダイオードを通してリーク電流 i が流れる為、外部 I/O 入力が常に ON 状態となります。
外部 I/O への供給電源 (+V。)=CB-08 への供給電源 (+V)、となるように外部 I/O 及び CB-08 に接続して下さい。
他機器との電源供給に注意し、 POWER ON/OFF は CB-08 が先又は、同時に ON/OFF する様にして御使用下さい。

10-2. スレーブへの電源供給例

(1) スレーブ電源をネットワーク端に 1 台配置した場合

スレーブ電源をネットワーク端に 1 台配置した構成例では、下図の様に 1.5SQ の電源線を使用し、総配線距離 20m で 15 台デェジチェーン接続した条件まで給電できます。



電源配置を端にして総配線距離 20m で CB-08 電源線を
1.5SQ で 15 台デェジチェーン接続した状態

以下にその確認方法を説明します。

①電源を供給するスレーブの消費電流の合計を求めます。
例： CB-08 本体の電源消費電流は 300mA ／台より　0.30 × 15 台＝ 4.5A

②電源供給で使用する電線の総延長を求めます。
総配線距離 =20m
但し、 +24V 線と GND 線の両方あり、直線路で考えると× 2 倍の 40m となります。

③電源ケーブルの配線抵抗を電線カタログで調べ、流す電流から電圧降下の値を調べます。
AWG(15) の導体抵抗で線材の電圧降下を計算します。 (カタログでは約 10 Ω /1000m)
40m × (10 Ω ÷ 1000m) × 4.5A ≒ 1.8V

④スレーブの入力電源仕様範囲と照合します。
24V ± 2V の CB-08 仕様に対し 1.8V の電圧降下が見積もれることより、 24V-1.8V ＝ 22.2V の電源電圧が最終端の CB-08 に給電できる結果となりました。

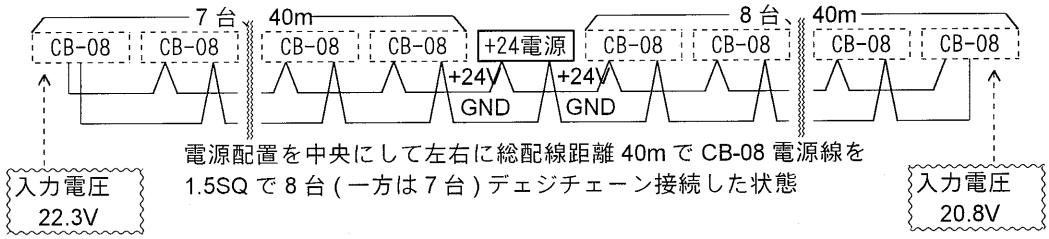
【 CB-08 本体電源仕様にて制約される事項】

No.	項 目	仕 様	備 考
1	入力電圧	DC+24V ± 2V	
2	消費電流	300mA(max)	
3	本体電源コネクタ	AWG12　　(約 3.3SQ)	1 本接続時
	の適合最大電線径	AWG(15) × 2(約 1.5SQ) × 2	2 本接続(ﾃﾞｪｼﾞﾁｪｰﾝ)時

(2) スレーブの入力電圧仕様を満足できない電圧降下が計算された場合

例：シリアルライン配線距離よりも電源配線距離の方が長い場合

①ネットワーク中 付近に電源を配置することで全てのスレーブに給 電できるか確認してください。

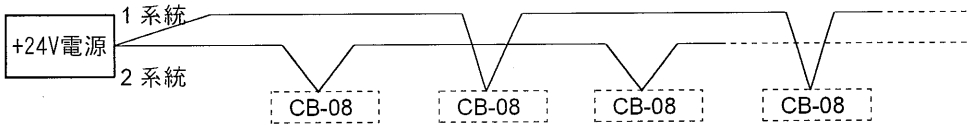


電源配置を中央にして左右に総配線距離 40m で CB-08 電源線を
1.5SQ で 8 台 (一方は 7 台) デェジチェーン接続した状態

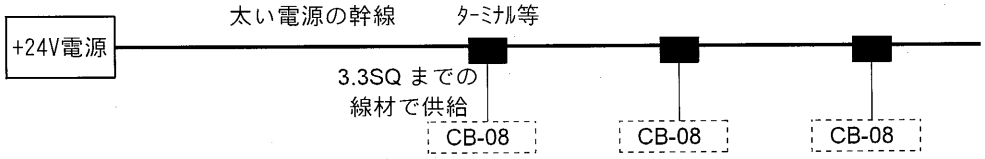
②上記①で確認した方法でも給電できない場合は以下を検討してください。

・電源装置の台数を増やして1系統当たりの電源総配線距離を短くする。

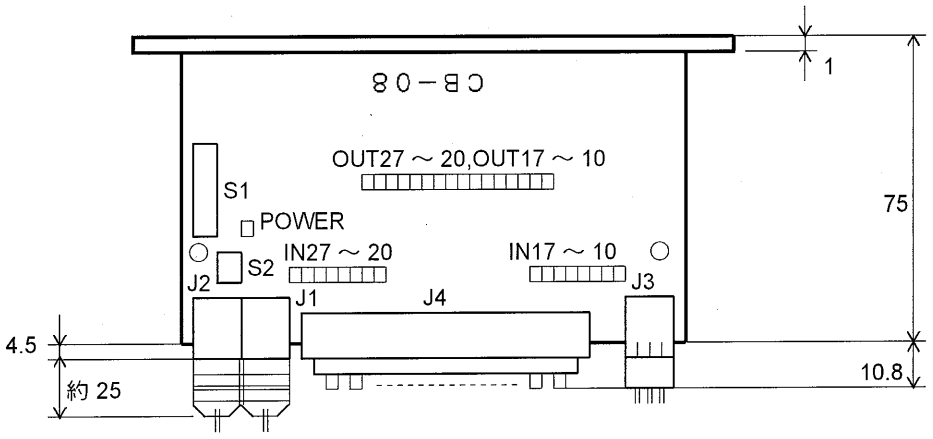
・電源の出力から直接スレーブへ配線する本数を増やしてデェジチェーン数を減らす。

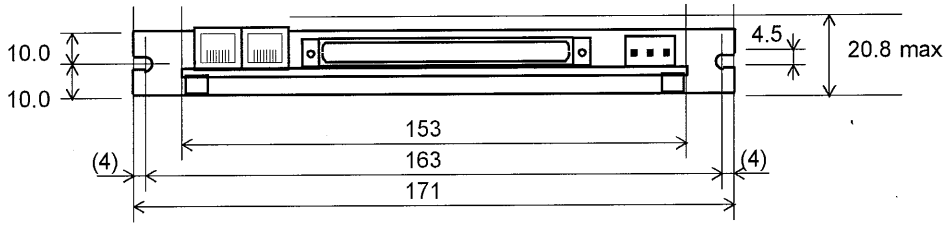


・電圧降下の低い太い電源線を幹線として、各スレーブへ幹線ターミナルから直接給電する。



１１．外形寸法





J1,J2 ---------- シリアル通信コネクタ
J3 ---------- 電源コネクタ
J4 ---------- ユーザ I/O コネクタ
POWER ---------- 電源 ON 時に点灯する LED
IN (10 ～ 17,20 ～ 27)---------- 各汎用入力がアクティブの時点灯する LED
OUT(10 ～ 17,20 ～ 27) ---------- 各汎用出力がアクティブの時点灯する LED
S1 ---------- レーブアドレス、通信速度設定スイッチ
S2 ---------- 終端抵抗設定スイッチ

１２．サンプル プログラム

®1

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 (マスター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(MCC05v2) 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(MCC05v2) 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-08 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。
解決しない場合には、マスターボードのトラブルシューティングも併せて参照して下さい。

	現　　象	チェックポイント
1	* 通信が正常に出来ない	*$\overline{\text{RESET}}$ に LOW LEVEL が入力されていませんか？ * ケーブルは正しく接続されていますか？ * 本体電源が供給されていないスレーブがネットワークに接続されていませんか？ * 終端抵抗は正しく設定しましたか？ ネットワーク終端のノードだけ ON にして下さい。 * 通信速度の設定はマスターと一致していますか？ ディップスイッチの設定は RESET 時に更新されます。 * スレーブのアドレス設定は正しいですか？ マスターアドレス及び他のスレーブアドレスと重複していないか確認して下さい。 * リクエストのフォーマットは間違っていませんか？ * リクエスト長と送信するバイト数は一致していますか？　リクエスト長を示すバイトはリクエスト長に含みません。
2	* エラー判定結果が返ってくる (1) ｺｰﾄﾞ =01 H (2) ｺｰﾄﾞ =02 H (3) ｺｰﾄﾞ =04 H (4) ｺｰﾄﾞ =05 H (5) その他のｺｰﾄﾞ	* スレーブタイプはあっていますか？ CB-08 のスレーブタイプは 10 H です。 * リクエストコードはあっていますか？ 6-2. リクエスト一覧表で確認して下さい。 * リクエスト長はリクエストコードに対応したものですか？ 各リクエストのフォーマットを確認して下さい。 * データがリクエストで要求されている範囲にありません。 各リクエストのフォーマットを確認して下さい。 * マスターのエラーです。 マスターのマニュアルを参照して下さい。

１４．CB-08 全 COMMAND 一覧表

$D^7D^6D^5D^4D^3D^2D^1D^0$	HEX CODE	REQUEST　NAME	参照ページ	備考
0 1 0 1 0 0 0 0	5 0	汎用 I/O 書き込み	1 1	
0 1 0 1 0 0 0 1	5 1	汎用 I/O 指定ビット書き込み	1 1	
0 1 0 1 0 0 1 0	5 2	汎用 I/O AND 書き込み	1 2	
0 1 0 1 0 0 1 1	5 3	汎用 I/O OR 書き込み	1 2	
0 1 1 0 0 0 0 0	6 0	汎用 I/O 書き込み	1 3	
0 1 1 0 0 0 0 1	6 1	汎用 I/O 指定ビット書き込み	1 3	

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. (042)664-5382　FAX.(042)666-2031

標準営業グループ／TEL.(042)664-5384　FAX.(042)666-2031

大 阪 営 業 所／TEL. (06) 6386-5135 FAX. (06) 6386-5375