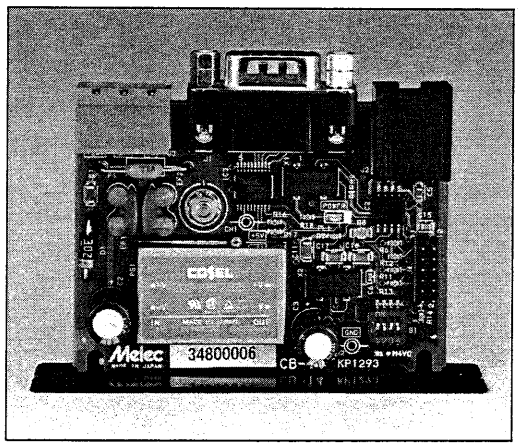
Melec



RS232Cマスタユニット

CB-14
取扱説明書
（設計者用）

USER'S MANUAL

本製品を使用する前に、この取扱説明書をよ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。

PR0746-1('02.6)

はじめに

この取扱説明書は、「AL シリーズ対応 RS232C マスターユニット CB-14」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。

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

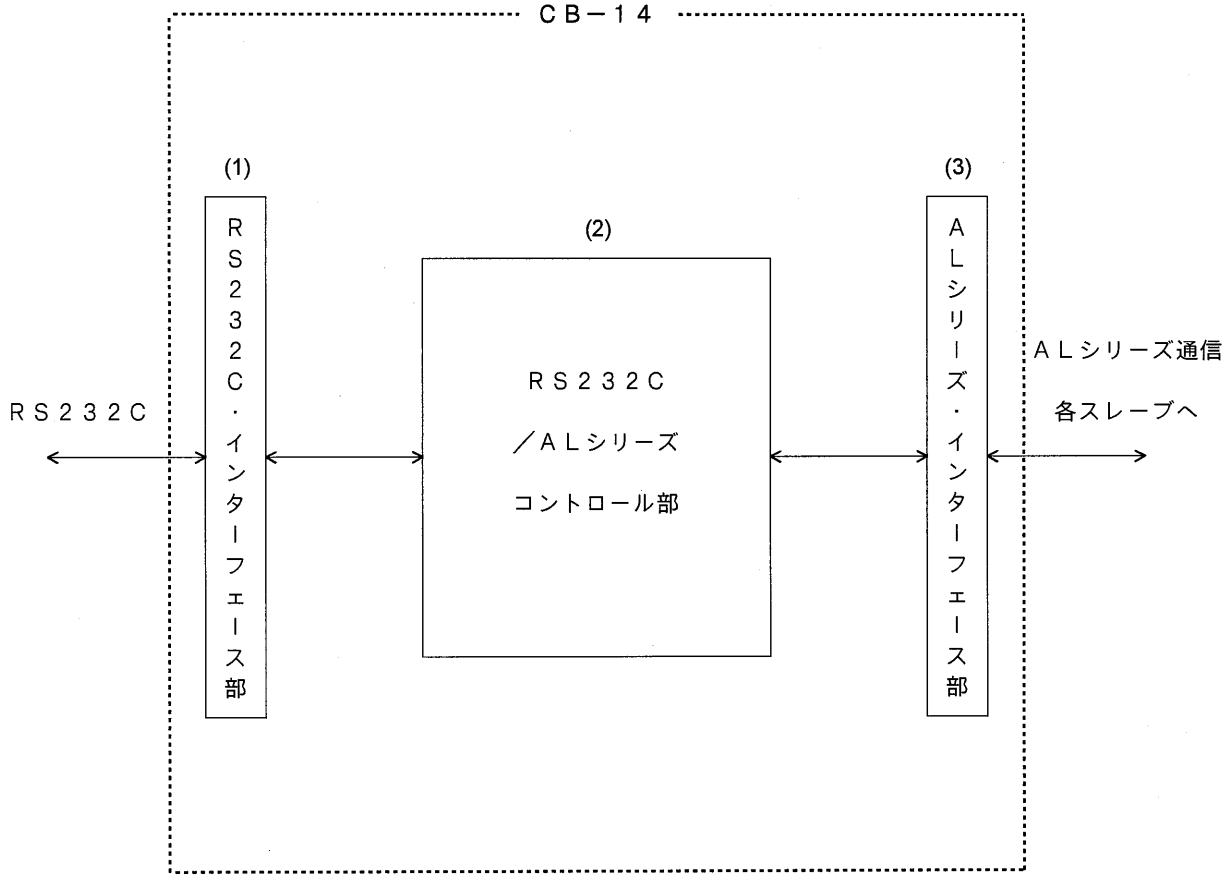
PAGE

１．概要

CB-14 は、PC の標準シリアルインターフェースである RS232C で接続可能な、 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)用のマスターユニットです。
ユーザはこのマスターユニット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。

２．基本構成

2-1.機能ブロック図



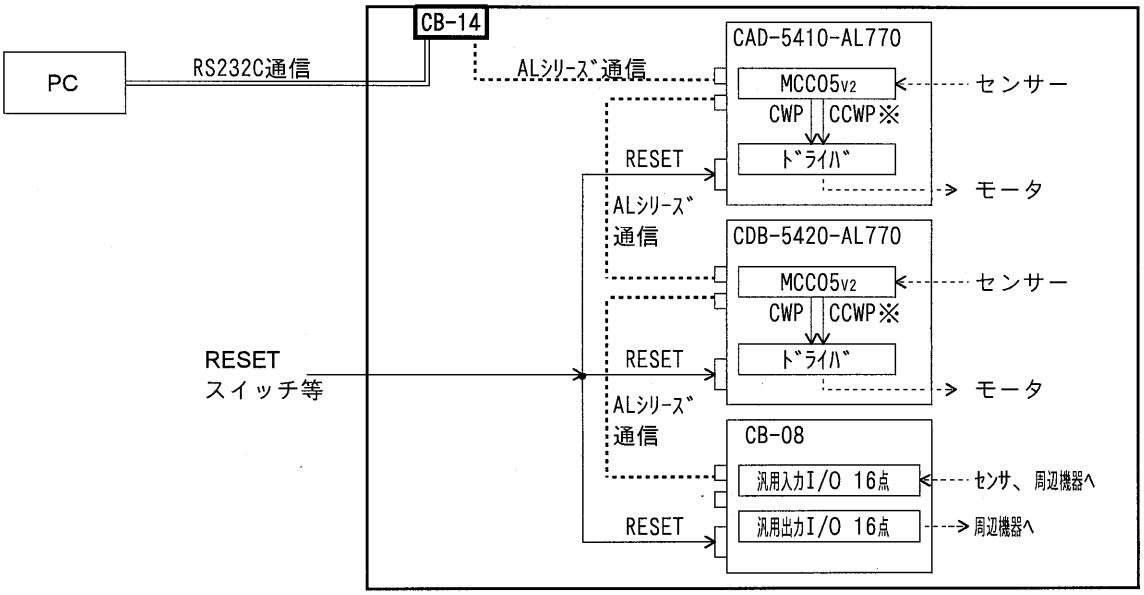
2-2.各ブロック説明

(1)RS232C・インターフェース部
RS232C 通信とのインターフェースブロックです。

(2)RS232C/AL シリーズコントロール部
RS232C 通信と AL シリーズ通信を制御するブロックです。
当コントロール部は RS232C 通信で受けたコマンドを AL シリーズ通信でスレーブに送信し、
スレーブより受けたアンサーバックを RS232C 通信で返します。

(3)AL シリーズ・インターフェース部
AL シリーズ通信とのインターフェースブロックです。

2-3.システム構成例



※ CWP は+(CW)方向正論理 PULSE 出力です(内部信号)。
CCWP は -(CCW)方向負論理 PULSE 出力です(内部信号)。

3．一般仕様

3-1.AL シリーズ仕様

(1)準拠規格	RS485(非絶縁)
(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	10m
(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.RS232C 仕様

(1)準拠規格	EIA-574
(2)通信距離	3m
(3)通信速度	9600bps/19200bps/38400bps/57600bps/115200bps
(4)データビット	7 ビット
(5)パリティビット	偶数
(6)ストップビット	1 ビット
(7)通信エラーチェック機能	パリティチェック
(8)PC との接続可能ケーブル	D-Sub9 ピン(メス) - D-Sub9 ピン(メス) クロスケーブル 例：C06-09F-09F-CROSS-910(ミスミ) C06-09FS-09FS-CROSS-SS30(ミスミ)

3-3.定格

(1)電源電圧　：+24V　50 m A

(2)周囲温度　：0 ℃～＋ 45 ℃

(3)周囲湿度　：80%RH 以下　(非結露)

(4)質量　：約 0.1 kg

(5)保存温湿度　：-10 ℃～+55 ℃、80%RH 以下(非結露)

(6)使用周囲雰囲気：直射日光の当たらない屋内に設置する事、
腐食性ガス、引火性ガス、オイルミスト、塵埃がない事

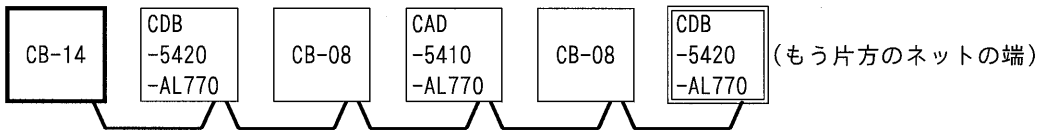
3-4.オプション

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．ユニットの設定

4-1.終端抵抗について

CB-14 の AL シリーズ・インターフェイス回路には終端抵抗が固定で付加されています。
よって CB-14 はネットの端に配置して下さい。(下図の [　　])



4-2.AL シリーズ上のアドレス設定

CB-14 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アドレス 00 Hに重複しない様に設定して下さい。

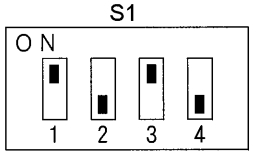
4-3.AL シリーズ通信速度設定

CB-14 の AL シリーズ通信速度(ボーレート)は、RS232C 通信による初期化リクエストで
設定します。
CB-14 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。

4-4.RS232C 通信速度設定

(1)CB-14 の RS232C 通信速度(ボーレート)を基板上のディップスイッチ(S1)1 ～ 3bit により設定します。
ディップスイッチ(S1)の設定は電源投入時に有効になりますので、設定変更後は必ず一旦、電源ＯＦＦし
電源を再投入して下さい。

例)次に示す例は RS232C 通信速度を 115200bps に設定した時のものです。



※ S1 の 4bit 目は必ず OFF にして下さい

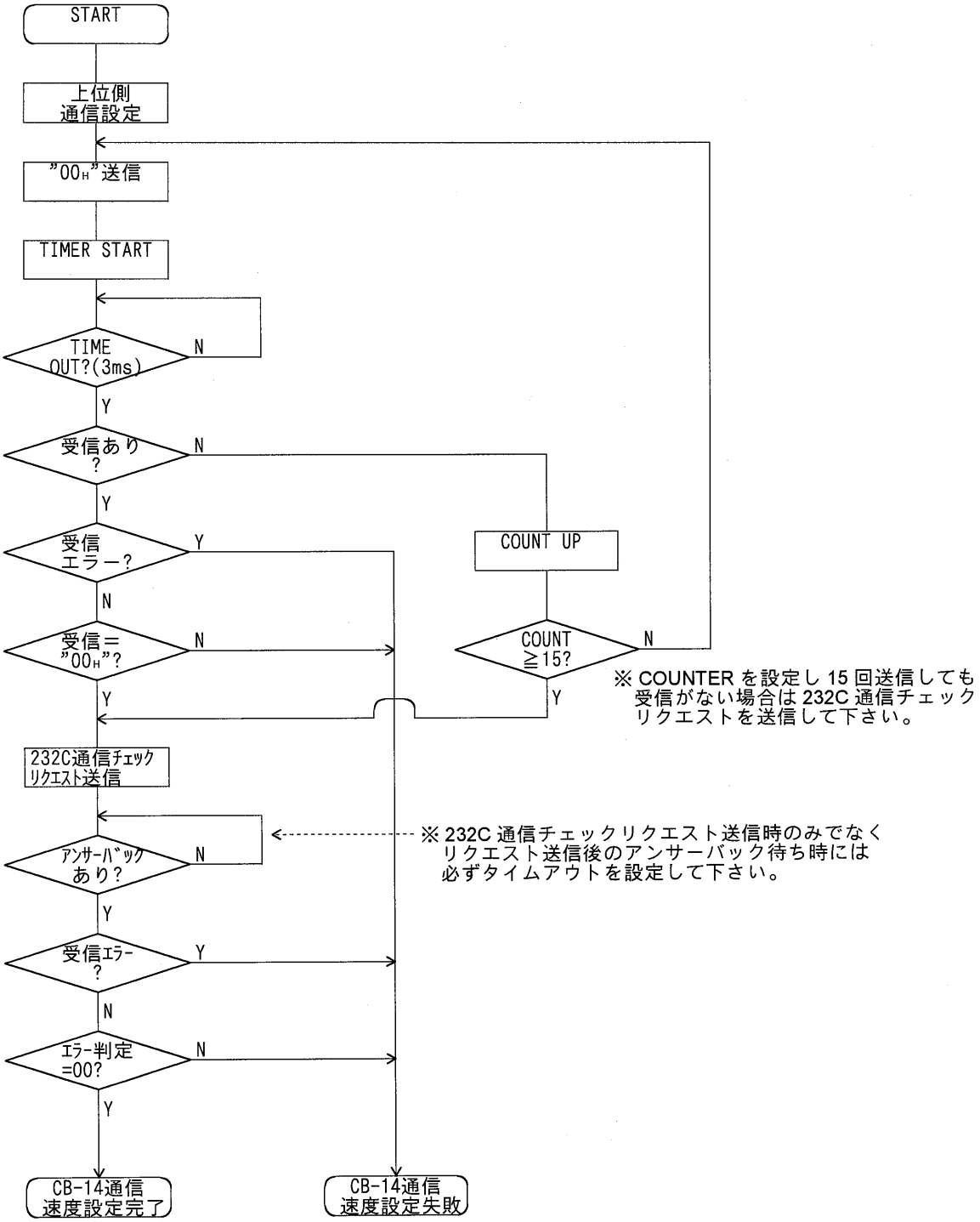
RS232C 通信速度

通信速度	1	2	3	
自動合わせ	OFF	OFF	OFF	←出荷時設定
9600bps	ON	OFF	OFF	
19200bps	OFF	ON	OFF	
38400bps	ON	ON	OFF	
57600bps	OFF	OFF	ON	
115200bps	ON	OFF	ON	
設定禁止	OFF	ON	ON	
設定禁止	ON	ON	ON	

(2)CB-14 の基板上のディップスイッチ(S1)の 1 ～ 3biT を全て OFF(自動合わせ)に設定すると、CB-14 は上位
から受信するデータの通信速度に自動合わせ込みを行います。
CB-14 の電源投入後に上位は設定したい通信速度(9600bps 、19200bps 、38400bps 、57600bps 、15200bps)
でデータ"00 H"を、CB-14 からアンサーバックデータ"00 H"を受信するまで 3ms 以上の間隔で連続送信して
下さい。("00 H"はバイナリで送信して下さい。)
上位が CB-14 からアンサーバックデータ"00 H"を受信した時点で CB-14 の RS232C 通信速度自動合わせ込み
は完了しています。
通信速度の自動合わせ込み完了後は RS232C 通信速度を変更しないで下さい。RS232C 通信速度を変更する
場合は、CB-14 を一旦電源 OFF し、電源再投入後、再び自動合わせ込みを行って下さい。

(3)RS232C 通信速度自動合わせ実行シーケンス

ユーザーアプリケーションの CB-14 RS232C 通信速度自動合わせの実行シーケンスを下記に示します。
(この実行シーケンスは CB-14 のディップスイッチ(S1)を自動合わせに設定した時のみ必要です。
ディップスイッチ(S1)を他の通信速度に設定した場合、このシーケンスは必要ありません。
また、この実行シーケンスで CB-14 通信速度設定完了後に再び、この実行シーケンスで通信速度を
変更する場合は一旦、CB-14 の電源を OFF し電源再投入後に行って下さい。)



※エラーが発生した場合、上位側の通信速度設定が、CB-14 の RS232C 通信仕様
と合っているか確認して下さい。
また、CB-14 のディップスイッチが自動合わせ(全て OFF)になっているか
確認して下さい。

5．初期化仕様

5-1.初期化機能

CB-14 は RS232C 通信で初期化リクエストを受信すると次の様な状態になります。

(1)CB-14 内の AL シリーズ通信コントロール部の状態が初期化されます。
（初期化リクエストパラメータで指示された AL シリーズ通信ボーレート、リトライ回数が設定されます。）

(2)CB-14 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、CB-14 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 CB-14 に初期化リクエストを送信して CB-14 の初期化を行って下さい。**

(注)既に当リクエスト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。

(3)CB-14 内の RS232C 通信コントロール部の状態が初期化されます。
（初期化リクエストパラメータで指示された以降の RS232C 通信における XID 機能の有効／無効が設定されます）

5-2.初期化方法

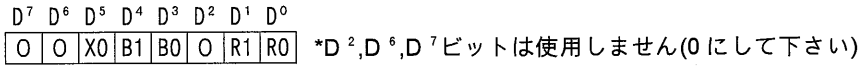
ユーザアプリケーションで初期化リクエストパラメータに AL シリーズ通信のボーレート、リトライ回数を設定し、CB-14 に初期化リクエストを送信してください。

(1)ボーレート
AL シリーズ通信のボーレートを初期化リクエストパラメータの D^4,D^3 ビットに設定します。

(2)リトライ回数
AL シリーズ通信のリトライ回数を初期化リクエストパラメータの D^1,D^0 ビットに設定します。

(3)XID 機能
RS232C 通信においての XID 機能の有効／無効を D^5 ビットに設定します。

(4)初期化リクエストパラメータ



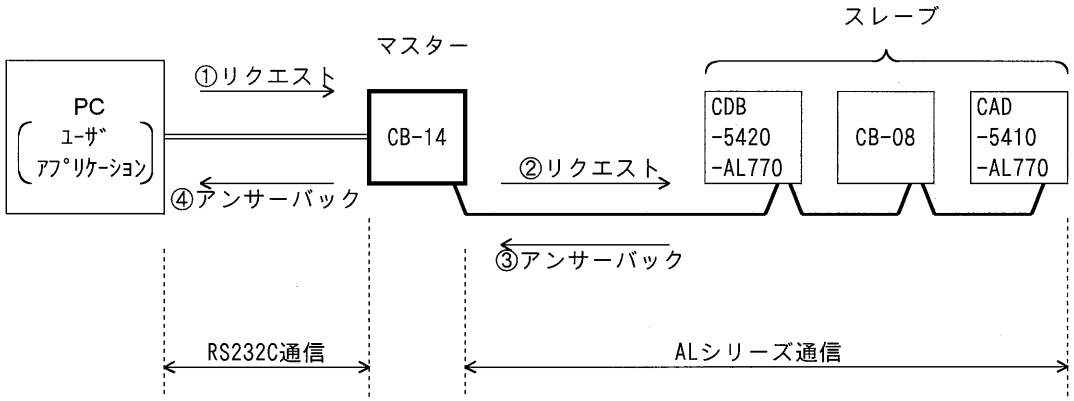
XID 機能	X0
無　効	0
有　効	1

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

6．リクエスト仕様

ユーザアプリケーションは、RS232C 通信でリクエストを CB-14 に送信する事でスレーブに指令を与えます。
CB-14 は、RS232C 通信のリクエストのエンドコードを受信すると受信したリクエストに通信エラーが無い事を確認し、スレーブに対して AL 通信のリクエスト送信を行います。
CB-14 はスレーブへのリクエスト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。
スレーブからのアンサーバックの受信が終了すると AL 通信のアンサーバックに通信エラーが無い事を確認し、ユーザアプリケーションへ RS232C 通信でアンサーバックを送信します。
ユーザアプリケーションから CB-14 へのリクエストにエラーがある場合(通信エラーは除く)、スレーブから CB-14 へのアンサーバックにエラーがある場合は CB-14 からユーザアプリケーションへの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。

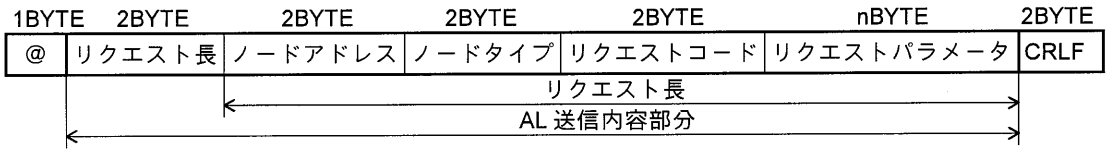


CB-14 はエラーの有無に関わらず、一旦、リクエストを受信するとアンサーバックを送信するまで次のリクエストを受け付けません。よってユーザアプリケーションはリクエスト送信後、必ずアンサーバックを受信してから次のリクエストを送信して下さい。
(リクエストに 232C 通信エラーが発生した場合のみ、CB-14 はアンサーバックを返しません。ユーザーアプリケーションではタイムアウトを設定し、タイムアウト発生後、再びリクエストを送信して下さい。)
CB-14 はハードエラー、AL 通信エラー以外のエラーが発生した場合でも、次のリクエストを正常に受信すればそのリクエストを実行します。
但し、ハードエラー、AL 通信エラーが発生した場合は、再び初期化リクエストを受信するまで他のリクエストを受付ませんので御注意下さい。
なお、CB-14 は電源 ON 後は初期化リクエストを実行するまで、他のリクエストは受付ませんので御注意下さい。

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 ASCII コードです。リクエストパラメータ、アンサーバックバラメータは、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。

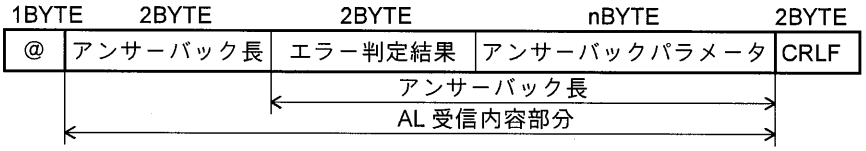
(1)リクエスト フォーマット



(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

・@(, ＊)　：スタートコードです。リクエストフォーマットの先頭に付けてください。
　XID 機能無効時にはスタートコードは@固定です。
　XID 機能有効時にはスタートコードは@、＊となり、通信毎に交互に使用します。
・リクエスト長　：AL に送信するデータの BYTE 数(バイナリ換算)を示します。
　(範囲は上図の通りです)
・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。
・ＣＲＬＦ　：エンドコードです。リクエストフォーマットの終わりに付けてください。

(2)アンサーバック フォーマット



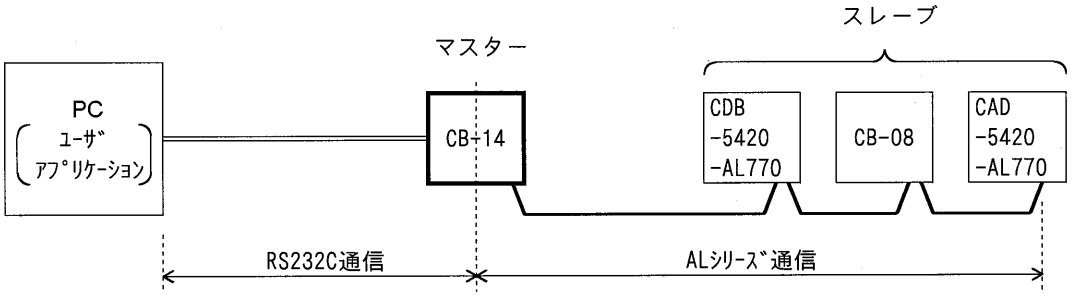
(注)アンサーバック長指定バイトはアンサーバック長に含みません

・@ ：スタートコードです。リクエストフォーマットの先頭に付きます。
・アンサーバック長 ：AL から受信したデータの BYTE 数(バイナリ換算)を示します。(範囲は上図の通りです)
・アンサーバックパラメータ ：各スレーブの取扱説明書を参照してください。
・ＣＲＬＦ ：エンドコードです。リクエストフォーマットの終わりに付きます。
・エラー判定結果 ：下表参照

エラー判定結果	エラー名称	エ ラ ー 内 容	エラー種別	発生箇所
００H	(エラーなし)			
０１H～７ＦH	CB-14,スレーブ固有のエラー	スレーブ固有については各スレーブの取扱説明書を参照して下さい。		ﾘｸｴｽﾄのﾉｰﾄﾞによる
※1 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	RS232C 通信
※1 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	
※1 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	
※2 ０６H	232C データエラー	RS232C で受信したデータにエラーがある	書式エラー	
０７H	未初期化エラー	CB-14 が初期化されていません	ハードエラー	
(ｱﾝｻｰﾊﾞｯｸなし) ※3	232C シリアルエラー	RS232C での受信時に通信エラーが発生した	232C 通信エラー	
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	AL ｼﾘｰｽﾞ通信
８１H	シリアルエラー	スレーブからの受信時のエラー	AL 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	AL 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 03 H～ 14 Hの範囲にない	書式エラー	

※1 ０２H、０４H、０５H は対 CB-14 リクエストに対するエラーです。
※2 RS232C で受信したデータの内容が HEX に変換できないものである場合、もしくは全受信 BYTE 数が規定 BYTE 数を越えた場合に発生するエラーです。
※3 RS232C での受信時に通信エラー(パリティ、フレーミング、オーバーラン)が発生した場合、CB-14 はアンサーバックを返しません。よってユーザアプリケーションはタイムアウトを設定しエラーを検出して下さい。

エラー発生箇所：



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。
エラー発生後も次のリクエストを受け付ける。リトライは行わない。

232C 通信エラー … RS232C 通信上の異常によるもの。エラー発生後も次のリクエストを受け付ける。

AL 通信エラー … AL シリーズ通信上の異常によるもの。エラー発生後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。エラー発生後は再初期化しなければ動作しない。
リトライは行わない。

6-2.対 CB-14 リクエスト一覧表

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
E8 H	初期化リクエスト
E9 H	使用禁止
EA H	232C 通信チェックリクエスト

6-3.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

@	0	3	0	0	0	0	E	0	CRLF

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。(無効。何を入れても構いません。)
マスターアドレスを指定します。(マスターは 00 H)
リクエスト長を指定します。

(2)アンサーバック

@	0	5	0	0	0	0	8	8	0	0	3	CRLF

ノードアドレス 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

ノードアドレス 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

ノードアドレス 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

ノードアドレス 18 H～ 1F Hの接続情報です。

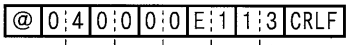
BIT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
ｽﾚｰﾌﾞｱﾄﾞﾚｽ	1F H	1E H	1D H	1C H	1B H	1A H	19 H	18 H	未接続 0
接続状態例	0	0	0	0	0	0	0	0	接続 1

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.スレーブタイプ読み出しリクエスト

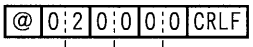
指定したアドレスのスレーブタイプを読み出します。未接続の場合には FF Hを返します。
この情報は初期化機能実行時に作成されるため、初期化機能後の接続状態変更は反映されません。
例．スレーブアドレス 13 Hのスレーブタイプが 00 Hです。(C-770AL)

(1)リクエスト



対象となるスレーブアドレスを指定します。(01 H～ 1F H)
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。(無効。何を入れても構いません。)
マスターアドレスを指定します。(マスターは 00 H)
リクエスト長を指定します。

(2)アンサーバック

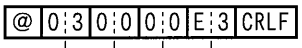


スレーブタイプです。(例：00 H… C-770AL、10 H… CB-08)
リクエストが正常に実行された事を示します。
アンサーバック長です。

6-5.エラー累計回数読み出しリクエスト

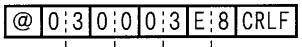
電源投入後に AL シリーズ通信上のエラーが発生した回数の累計を返します。
初期化時に設定したリトライ回数内でリトライを行ったときもエラー回数としてカウントします。
カウントは最大で 65535 回までカウントし、その後エラーが発生してもカウントはストップします。
このカウンタは RESET、電源再投入、又はエラー累計回数クリアリクエストによって 0 にクリアされます。
例．電源投入後、プロトコル異常(ノイズ等)により 1000(3E8 H)回の通信エラーが発生しました。

(1)リクエスト



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。　(無効。何を入れても構いません。)
マスターアドレスを指定します。(マスターは 00 H)
リクエスト長を指定します。

(2)アンサーバック

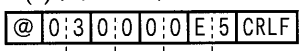


累計エラー回数(下位 8bit)
累計エラー回数(上位 8bit)
} エラー回数が 3E8 H(1000 回)であることを示します。
リクエストが正常に実行された事を示します。
アンサーバック長です。

6-6.エラー累計回数クリアリクエスト

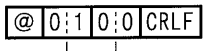
CPU 内部で記憶しているエラー累計回数を 0 にクリアします。

(1)リクエスト



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。　(無効。何を入れても構いません。)
マスターアドレスを指定します。(マスターは 00 H)
リクエスト長を指定します。

(2)アンサーバック

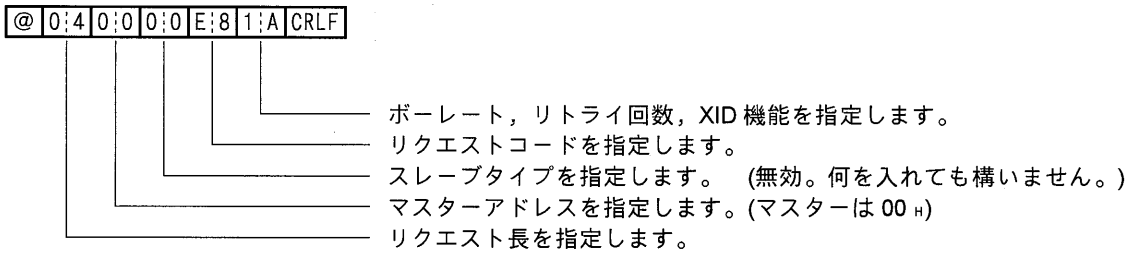


リクエストが正常に実行された事を示します。
アンサーバック長です。

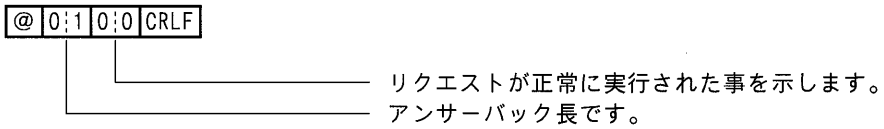
6-7.初期化リクエスト

CB-14 内の AL シリーズ通信コントロール部を初期化します。
(AL シリーズ通信ボーレート、リトライ回数が設定し、全スレーブに初期化リクエストを送信します。)
例．AL シリーズ通信コントロール部をボーレート=625000bps，リトライ回数=2 回に、
RS232C 通信コントロール部を XID 機能=無効に初期化します。

(1) リクエスト



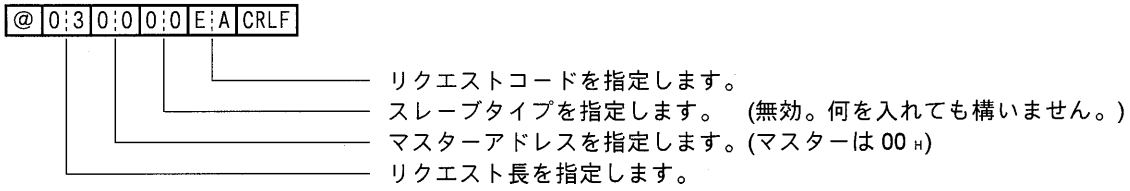
(2)アンサーバック



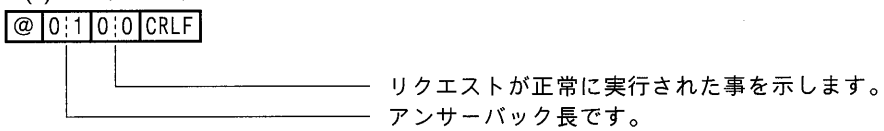
6-8.232C 通信チェックリクエスト

CB-14 との RS232C 通信が正常に行われている事を確認するために使用します。

(1)リクエスト



(2)アンサーバック



6-9.対 CB-14 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	発生箇所
００H	(エラーなし)			
※1 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	RS232C 通信
※1 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	
※1 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	
※2 ０６H	232C データエラー	RS232C で受信したデータにエラーがある	書式エラー	
０７H	未初期化エラー	CB-14 が初期化されていません	ハードエラー	
８０H～ＦＦH	全スレーブ共通エラー	6-1.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	AL ｼﾘｰｽﾞ 通信

※1 ０２H、０４H、０５H は対 CB-14 リクエストに対するエラーです。
※2 RS232C で受信したデータの内容が HEX に変換できないものである場合、もしくは全受信 BYTE 数が規定 BYTE 数を越えた場合に発生するエラーです。

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。
エラー発生後も次のリクエストを受け付ける。リトライは行わない。
ハードエラー … ハードの異常によるもの。エラー発生後は再初期化しなければ動作しない。
リトライは行わない。

6-10.エラー処理例

(1)エラー判定結果≠ 0 のアンサーバックを受信します。
(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。
(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-14 はエラー判定結果≠ 0 のアンサーバックを送信した後初期化リクエスト以外のリクエストは受け付けません。

6-11.イニシャルエラー

CB-14 に初期化リクエストを送信することにより、CB-14 は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 RESET 後(電源 ON 後)、このリクエストを受信するまでは他のリクエストを受け付けません。(他のリクエストを受信した場合、イニシャルエラーを返します。)
よって、スレーブの RESET 後(電源 ON 後)には必ず CB-14 に初期化リクエストを実行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。

6-12.リトライ機能(AL シリーズ通信)

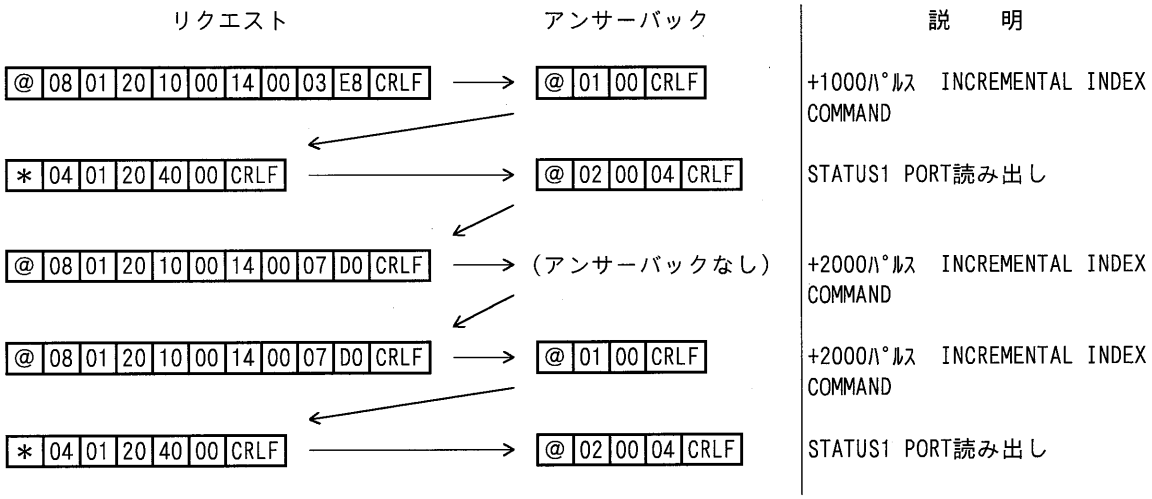
ノイズの影響により AL シリーズ通信上で通信エラーが発生し、場合によってはシステム全体がロック状態になる事も考えられます。CB-14 にはこの事を考慮して、リトライ機能があります。
AL シリーズ通信上の通信エラーにより、ロック状態になった時(時間計測によって検出)リトライします。
尚、リトライ回数は、初期化リクエストで設定します。
リトライ回数に達しても、尚かつ通信エラーが発生した場合、アンサーバック内のエラー判定結果≠ 0 にしてユーザアプリケーションに通知します。この時のエラー判定結果は最後に発生したエラーです。
リトライ機能の対象となるエラーは AL シリーズ通信上のエラー(シリアルエラー、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。

6-13.XID 機能(RS232C 通信)

(1)ノイズの影響により RS232C 通信上で通信エラーが発生し、ユーザアプリケーション側で CB-14 からのアンサーバックを受信できない事があります。
この場合、再度 CB-14 へ前回と同じリクエストを送信すると、CB-14 は前回のリクエストを実行している可能性がある為、CB-14 は結果的に同じリクエストを 2 回実行する事になります。
この様にユーザーアプリケーションでのアンサーバック受信エラー発生時のエラー処理に対して、CB-14 リクエストの二重実行を防止する為に XID 機能を用意しています。

(2)XID 機能とは CB-14 が受け付けるリクエストのスタートコードがリクエスト毎に@、＊と切り替わるものです。XID 機能を有効にした場合、ユーザアプリケーションはリクエスト毎にリクエストのスタートコードを@、＊に切り替える必要があります。
但し、アンサーバック受信エラー発生時に再度 CB-14 へ前回と同じリクエストを送信する場合のみ、前回と同じスタートコードでリクエストを送信して下さい。
これにより CB-14 は前回リクエストを実行していれば動作は行わず、前回返したアンサーバックを再び返します。前回リクエストが実行されていない場合は動作を行い、そのアンサーバックを返します。

例) XID 機能を使用した場合のリクエスト／アンサーバック
(CDB-5420-AL770(スレーブアドレス 01 H)に対するリクエストを例とします)



(3)XID 機能を有効とした場合、基本的に全てのリクエストにおいて、リクエスト毎に受け付けるスタートコードが@、＊に切り替わりますが、例外として以下のリクエストのみは XID を無視したスタートコードでも受け付けます。
これらのリクエストを受け付けた場合、そのリクエストでのスタートコードに XID がリフレッシュされますので次のリクエストはその反対のスタートコードで送信して下さい。

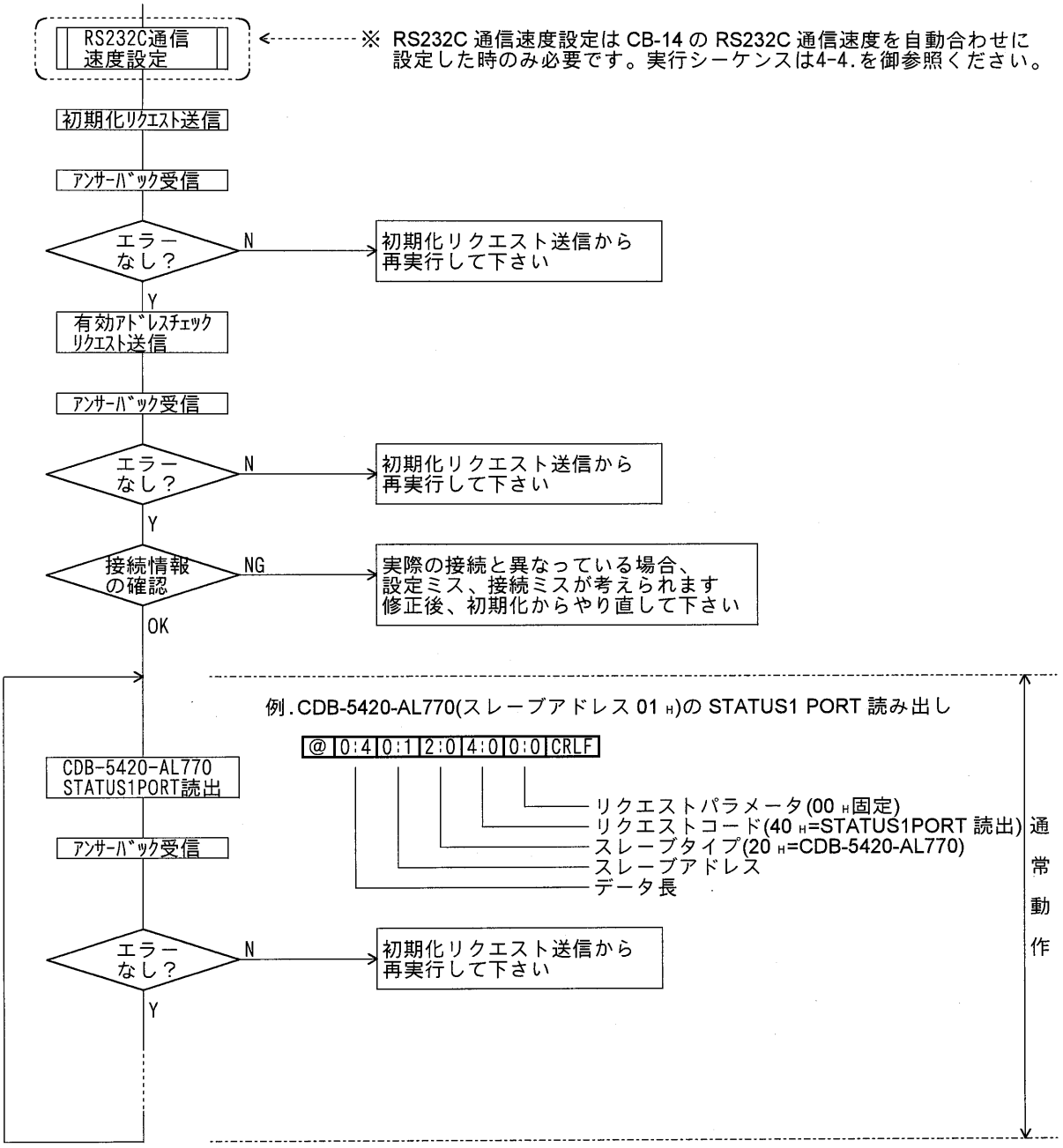
XID 無効リクエスト

対象機種	CODE	リクエスト名
CB-14	E8 H	初期化リクエスト
	EA H	232C 通信チェックリクエスト

(4)XID 機能有効／無効の選択は初期化リクエストで行います。詳細については 5.初期化仕様を参照して下さい。

6-14.実行シーケンス

(1)全体の流れ



7．タイミング

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

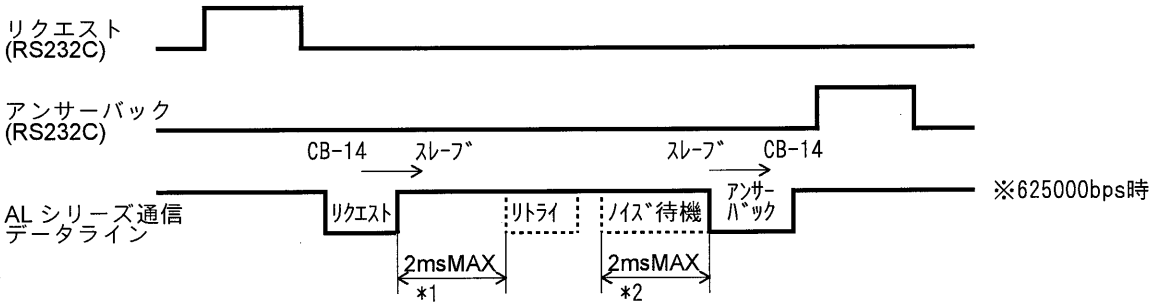
(2)リトライ回数
リトライ回数を 1 回以上に設定した場合、リトライ 1 回あたりの回数に応じて最大で下記の時間が遅れます。
これはリトライ動作による遅れなので、リトライを有効にしていてもノイズがのらない環境であれば実行時間に変化はありません。

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。

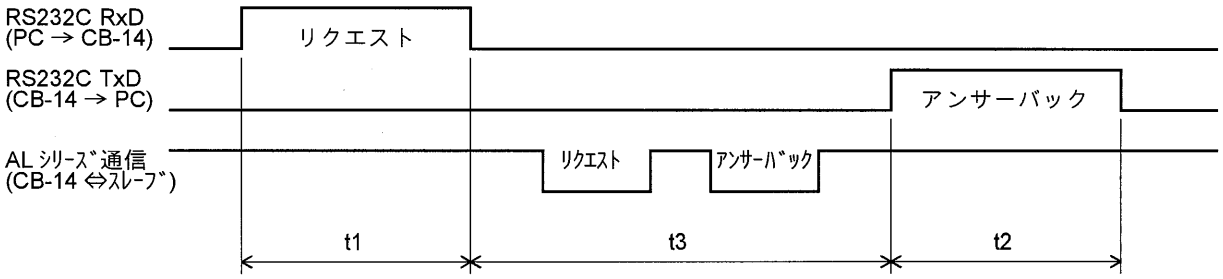
通信速度(bps)	9765	39062	156250	625000
ｱﾝｻｰﾊﾞｯｸ遅れ (ms)	128.00	32.00	8.00	2.00



*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、CB-14 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 AL シリーズ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～アンサーバック TIMING



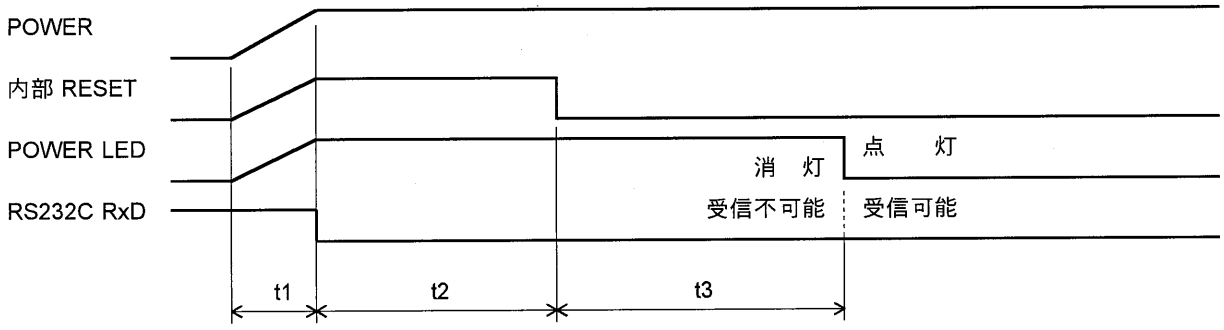
通信速度(bps)	RS232C 通信				
	9600	19200	38400	57600	115200
t1	1.2ms×nBYTE	0.6ms×nBYTE	0.3ms×nBYTE	0.2ms×nBYTE	0.1ms×nBYTE
t2	1.2ms×nBYTE	0.6ms×nBYTE	0.3ms×nBYTE	0.2ms×nBYTE	0.1ms×nBYTE

(注 1) nBYTE は ASCII コードでの送受信 BYTE 数になります。

通信速度(bps)		AL シリーズ通信			
		9765	39062	156250	625000
t3 (注 2)	対 CB-14	<0.25ms			
	対 CB-14(初期化ﾘｸｴｽﾄ)	<4000ms	<1000ms	<250ms	<63ms
	対 C-770AL 例	<28.90ms	<7.30ms	<1.90ms	<0.55ms
	対 CB-08 例	<21.22ms	<5.38ms	<1.42ms	<0.43ms

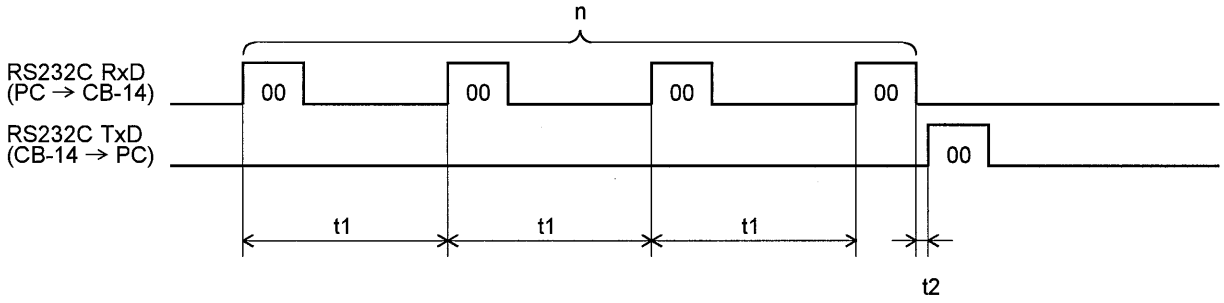
(注 2)ノードのタイプ、各ノードのリクエストによって詳細の時間は異なります。
各ノード(スレーブ)の取扱説明書を参照して下さい。
尚、対 CB-14 リクエストの応答時間は AL シリーズ通信時間がない為、通信速度設定に依存しません。
但し、対 CB-14 リクエストの内、初期化リクエストについては全てのスレーブに
イニシャルリクエストを送信しているため、他の対 CB-14 リクエストとは実行時間が異なります。

7-3.POWER ON(RESET) TIMING



t1	≦ 200ms
t2	≦ 400ms
t3	≦100ms

7-4.RS232C 通信速度自動合わせ TIMING

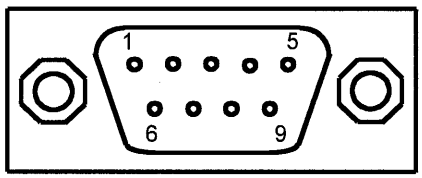


t1	≧ 3ms	
t2	≦ 0.1ms	
n	115200bps	1
	57600bps	2
	38400bps	3
	19200bps	4
	9600bps	5

8．コネクタ信号表

8-1.RS232C 通信コネクタ(J1)

(1)コネクタ型名　DELC-J9PAF-23L9(JAE)



(2)信号表

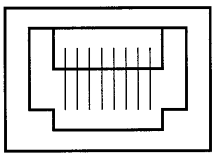
ピン	方 向	信 号 名	説　明
1	－	N.C	使用禁止
2	入	RxD	RS232C 受信データ信号
3	出	TxD	RS232C 送信データ信号
4	－	N.C	使用禁止
5	－	GND	グランド
6	－	N.C	使用禁止
7	－	N.C	使用禁止
8	－	N.C	使用禁止
9	－	N.C	使用禁止

8-2.AL シリーズ通信コネクタ(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

J2



12345678

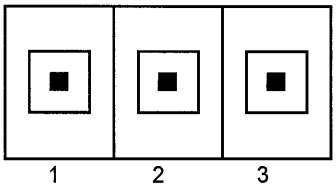
(2)信号表

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	AL シリーズ通信データ入出力信号 (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	AL シリーズ通信データ入出力信号 (ラインドライバ負論理)
7	－	N.C	使用禁止
8	－	N.C	使用禁止

8-3.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) :ﾃﾞｪｼﾞﾁｪｰﾝ時

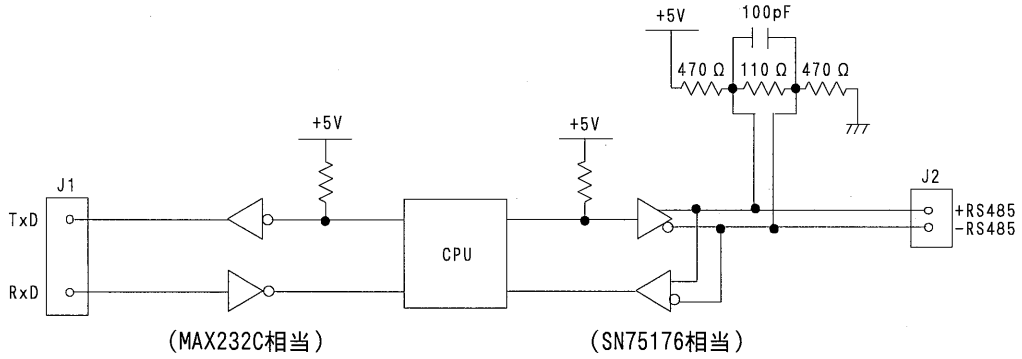
(基板外側から見た図)

(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

９．入出力回路

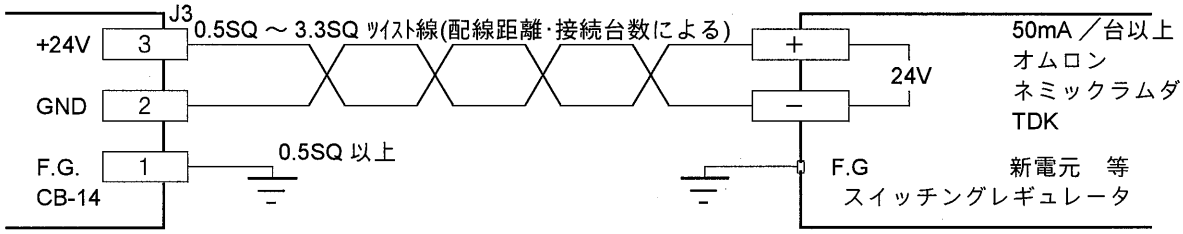
9-1.RS232C 通信コネクタ～AL シリーズ通信コネクタ間等価回路(J1 ～ J2)



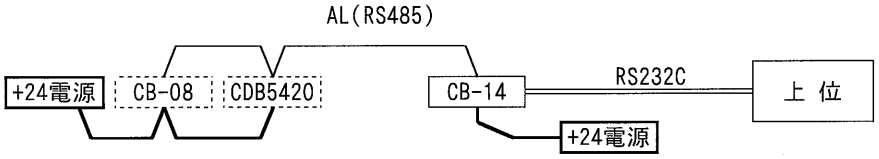
１０．接続

10-1.電源との接続例

CB-14 の電源は、DC+24V です。この電源は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　50mA ／台以上の安定化電源を御使用下さい。
CB-14 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。



10-2.システム電源シーケンス



(1)上位の電源が OFF の場合
CB-14 は上位からのリクエストを受信するまで動作しません。

(2)CB-14 の電源が OFF の場合
CB-14 は動作を行えません。上位アプリケーションがリクエストを送信した場合、アンサーバックを返せませんのでアプリケーション側でタイムアウトを設ける必要があります。

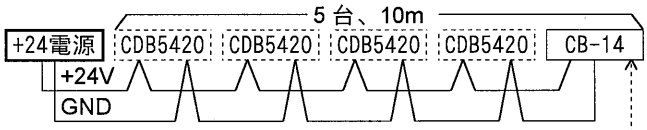
(3)AL スレーブ機器の電源が OFF の場合
CB-14 は上位アプリケーションからリクエストを受信した場合、AL 通信を行いますが
タイムアウトエラーが発生します。CB-14 は上位アプリケーションにタイムアウトエラー発生を
通知します。

※上記(3)は CB-14 と AL スレーブ機器の電源が別電源である場合を想定しています。
CB-14 と AL スレーブ機器に同一電源を使用しても構いません。

10-3.CB-14、スレーブへの電源供給例

(1)CB-14、スレーブ電源をネットワーク端に 1 台配置した場合

CB-14、スレーブ(CDB-5420-AL770)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 1.5mm ^{2} の電源線を使用し、総配線距離 10m で 5 台(CB-14:1 台,CDB-5420-AL770:4 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 10m で CB-14,CDB-5420-AL770 電源線を 1.5mm ^{2} で 5 台デェジチェーン接続した状態

入力電圧 22.3V

以下にその確認方法を説明します。

①電源を供給する CB-14,CDB-5420-AL770 の消費電流の合計を求めます。
例：CB-14,CDB-5420-AL770 の電源消費電流は 50mA／台、2A／台より 0.05 ＋ 2.00 × 4 台＝ 8.05A

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=10m
但し、+24V 線と GND 線の両方あり、直線路で考えると× 2 倍の 20m 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。
AWG15 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)
20m ×(10 Ω ÷ 1000m)× 8.05A ≒ 1.7V

④ CB-14,CDB-5420-AL770 の入力電源仕様範囲と照合します。
24V の CB-14,CDB-5420-AL770 仕様に対し 1.7V の電圧降下が見積もれることより、24V-1.7V ＝ 22.3V の電源電圧が最終端の CB-14 に給電できる結果となりました。

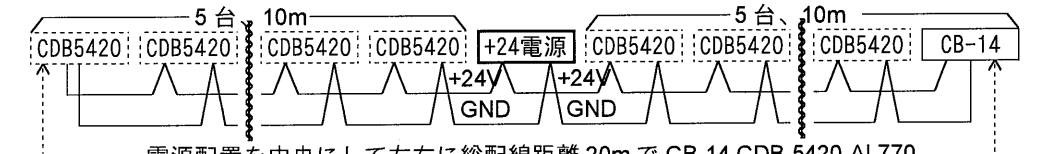
【CB-14 本体電源仕様にて制約される事項】

No.	項 目	仕 様	備 考
1	入力電圧	DC+24V	許容範囲± 2V
2	消費電流	50mA(max)	
3	本体電源コネクタの適合最大電線径	AWG12(約 3.3mm ^{2})	1 本接続時
		AWG15(約 1.5mm ^{2})× 2	2 本接続(デェジチェーン)時
4	本体電源コネクタの最大電流	10.0A(max)	

(2)CB-14 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：ＡＬシリーズ通信の配線距離よりも電源配線距離の方が長い場合

①ネットワーク中央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 20m で CB-14,CDB-5420-AL770 電源線を 1.5mm ^{2} で 5 台デェジチェーン接続した状態

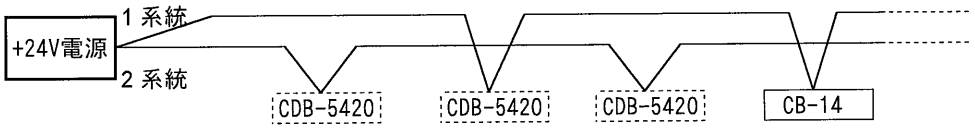
入力電圧 22.0V

入力電圧 22.3V

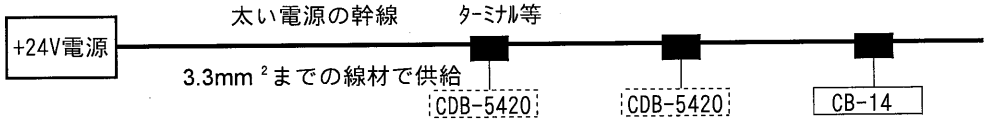
②上記①で確認した方法でも給電できない場合は以下を検討してください。

・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。

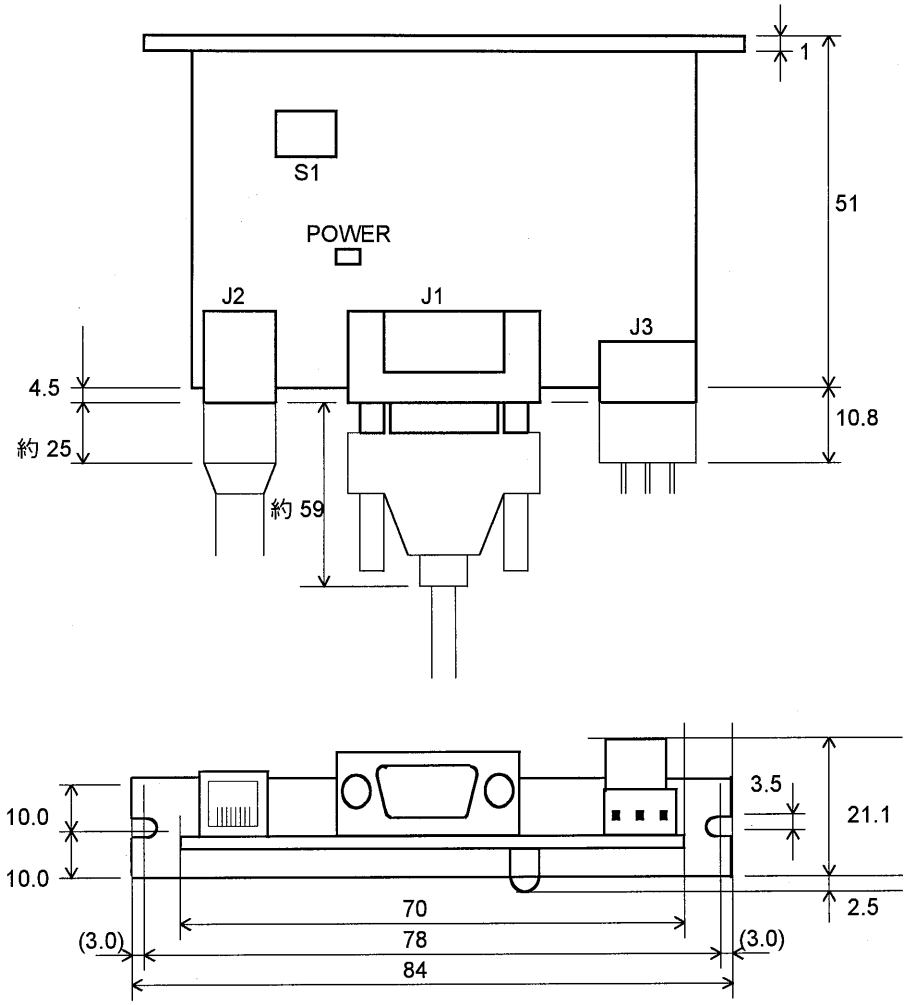
・電源の出力から直接スレーブへ配線する本数を増やしてデェジチェーン数を減らす。



・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	----------	RS232C 通信コネクタ
J2	----------	AL シリーズ通信(RS485)コネクタ
J3	----------	電源コネクタ
POWER	----------	電源 ON 時に点灯する LED
S1	----------	RS232C 通信速度設定スイッチ

１２．サンプル プログラム

®1

本章では CB-14 と C-770AL(MCC スレーブ)と CB-08(I/O スレーブ)を各 1 台接続し、1 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)

・この例では、PC の COM1 により RS232C で CB-14 と接続しているものとします。
　また、使用する PC は RS232C のデバイスが PC16550 相当であるものとします。
・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.RS232C 通信初期化関数例

RS232C 通信の初期化を行います。プログラムを実行する際には初めに必ず実行させてください。
ここでは通信速度を 115200bps、データビットを 7 ビット、パリティを偶数、ストップビットを 1 ビットとします。

```
/*-------------------------------------------------------------------*/
/*              RS232C INITIALIZE                                    */
/*-------------------------------------------------------------------*/

  void Rs232c_Initialize( void )
  {
        US  bps;
        UC  bpsA, bpsB;

    bps = 115385 / 115200;
    bpsA = bps >> 8;
    bpsB = bps;
    _outp( LCR, 0x80 );          /* DEVISOR LATCH ACCESS BIT ON */
    _outp( DVLA_B, bpsB );       /* BAUD RATE UPPER SET */
    _outp( DVLA_A, bpsA );       /* BAUD RATE LOWER SET */
    _outp( LCR, 0x1a );          /* DEVISOR LATCH ACCESS BIT OFF, PARAMATER SET*/
  }
```

12-2.DATA 変換関数例

(1)ASCII コード文字列を BINARY に変換します。

```
/*-------------------------------------------------------------------*/
/*              ASCII  →  BINARY                                     */
/*-------------------------------------------------------------------*/

   UC  ASCII_to_Binary( UC *Str )
   {
        UC     dt[2];
        int i;

    for( i = 0; i < 2 ; i++ ){
        if( ( Str[i] >= 'A' ) && ( Str[i] <= 'F' ) )
            dt[i] = Str[i] - '7';
        if( ( Str[i] >= '0' ) && ( Str[i] <= '9' ) )
            dt[i] = Str[i] - '0';
        }
    return dt[1] << 4 | dt[2];
    }
```

(2)BINARY コード文字列を ASCII に変換します。 ®1

BINARY コードの上位 4 ビットは引数で指定されたアドレスに、下位 4 ビットは 1 インクリメントされたアドレスに格納されます。

例

Binary_to_ASCII(0x10, &data[5]); → data[5] = '1' data[6]= '0' となります。

```
/*-------------------------------------------------------------------*/
/*              BINARY→  ASCII                                       */
/*-------------------------------------------------------------------*/

  void Binary_to_ASCII( UC dt, UC *str )
  {
    UC udt, ldt;

        udt = dt >> 4;
        ldt = dt & 0x0F;

        if( udt <10 )
              udt += '0';
        else
              udt += '7';

        if( ldt <10 )
             ldt += '0';
        else
             ldt += '7';

       *str          = udt;
       *( str + 1 )  = ldt;

  }
```

12-3.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  RBR            0x03f8      /* RECEIVE BUFFER REGISTER         */
#define  THR            RBR         /* TRANSMISSION HOLDING REGISTER   */
#define  DVLA_B         RBR         /* DEVISOR LATCH LOWER             */
#define  DVLA_A         0x03f9      /* DEVISOR LATCH UPPER             */
#define  LCR            0x03fb      /* LINE CONTROL REGISTER           */
#define  LSR            0x03fd      /* LINE STATUS REGISTER            */

#define  CR             0x0d
#define  LF             0x0a
#define  DUMMY          0x00        /* DUMMY DATA                      */

#define  ADR1           0x01        /* C770AL ADDRESS                  */
#define  ADR2           0x03        /* CB08 ADDRESS                    */
#define  AXIS_X         0x00        /* C770AL AXIS=X                   */
#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     0x0000000c  /* CONNECT CHECK COMPARE DATA          */
                                    /* VALID SLAVE ADDRESS = 01, 03, MASTER */

UC trs_ptr[30];         /* TRANSMISSION BUFFER   */
UC rev_ptr[20];         /* RECEIVE 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(UC*, UC*);
void    al_initialize(void);
UL      adr_map(void);
void    cb08iowrite(US);
UL      cb08ioread(void);
void    error_op(void);
void    Rs232c_Initialize( void );
UC      ASCII_to_Binary( UC * );
void    Binary_to_ASCII( UC , UC* );
```

頻繁に現れるCOMポートのステータス確認(1byte送受信)、C-770ALのMCC05v2のRDYの確認をマクロ化し、プログラムの簡素化を図ります。

```
#define     reqprdy()     while( !( _inp( LSR ) & 0x20 ) )    /* TRANSMIT BUFFER EMPTY WAIT */
#define     ansprdy()     while( !( _inp( LSR ) & 0x01 ) )    /* 1BYTE RECEIVE 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。
当プログラム例で使用するRAMエリアを下記のように定義します。

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

尚、本章に示すプログラム例はあくまでも参考例であり、必ずしもこれに従う必要ありません。

12-4.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( UC *trs_ptr, UC *rev_ptr )
 {
   US data;

   do{
         data = *trs_ptr;           /* 1BYTE DATA SET        */
         reqprdy();                 /* REQUEST READY WAIT    */
         outp( THR, data );         /* 1BYTE DATA WRITE      */
         if( data == LF )
         break;
         trs_ptr++;
     }while( 1 );

   do{
         ansprdy();                /* ANSWERBACK READY WAIT */
         *rev_ptr = _outp( RBR );  /* 1BYTE DATA READ       */
         if( *rev_ptr == LF)
         break;
         rev_ptr++;
     }while( 1 );
 }
```

12-5.AL シリーズ ADDRESS CHECK 関数例

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=1F $_H$ となる仕様とします。

```
/*-------------------------------------------------------------------*/
/*              AL SERIES ADDRESS CHECK                              */
/*-------------------------------------------------------------------*/

 UL   adr_map( void )
 {
      UL    a;

      *trs_ptr              = '@';
      Binary_to_ASCII( 0x03 ,        trs_ptr + 1 );        /* REQUEST LENGTH            */
      Binary_to_ASCII( 0x00 ,        trs_ptr + 3 );        /* MASTER ADDRESS            */
      Binary_to_ASCII( TYPE_MASTER , trs_ptr + 5 );        /* SLAVE TYPE(INVALID)       */
      Binary_to_ASCII( 0xe0 ,        trs_ptr + 7 );        /* ADDRESS CHECK READ REQUEST */
      *( trs_ptr  + 9 )     =  CR;                         /* END CODE                  */
      *( trs_ptr  + 10 )    =  LF;

      request( trs_ptr, rev_ptr );

      *( ( UC * )&a + 3 ) = ASCII_to_Binary( rev_ptr + 5  );
      *( ( UC * )&a + 2 ) = ASCII_to_Binary( rev_ptr + 7  );
      *( ( UC * )&a + 1 ) = ASCII_to_Binary( rev_ptr + 9  );
      *( ( UC * )&a     ) = ASCII_to_Binary( rev_ptr + 11 );

  return a;
 }
```

12-6.AL シリーズ INITIALIZE PROGRAM 例

AL シリーズ通信動作の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
XID 機能　　… 0 回

```
/*-------------------------------------------------------------------*/
/*              AL SERIES INITIALIZE                                 */
/*-------------------------------------------------------------------*/

 void    al_initialize( void )
 {
      *trs_ptr              = '@';
      Binary_to_ASCII( 0x04,         trs_ptr + 1 );        /* REQUEST LENGTH            */
      Binary_to_ASCII( 0x00 ,        trs_ptr + 3 );        /* MASTER ADDRESS            */
      Binary_to_ASCII( TYPE_MASTER , trs_ptr + 5 );        /* SLAVE TYPE(INVALID)       */
      Binary_to_ASCII( 0xe8 ,        trs_ptr + 7 );        /* AL INITIALIZE REQUEST     */
      Binary_to_ASCII( 0x18 ,        trs_ptr + 9 );        /* BAUD RATE, RETRY, XID SET */
      *( trs_ptr + 11 )     =  CR;                         /* END CODE                  */
      *( trs_ptr + 12 )     =  LF;                         /* END CODE                  */

      request( trs_ptr, rev_ptr );

      if( adr_map() != ADDRESS_MAP )                       /* CONNECT CHEACK            */
          error_op();

  }
```

12-7.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          = '@';
      Binary_to_ASCII( 0x08,         trs_ptr + 1  );    /* REQUEST LENGTH                      */
      Binary_to_ASCII( ADR1,         trs_ptr + 3  );    /* SLAVE ADDRES SET                    */
      Binary_to_ASCII( TYPE_C770AL,  trs_ptr + 5  );    /* SLAVE TYPE SET                      */
      Binary_to_ASCII( 0x10,         trs_ptr + 7  );    /* DRIVE COMMAND ALL WRITE REQUEST SET */
      Binary_to_ASCII( AXIS_X,       trs_ptr + 9  );    /* X AXIS SET                          */
      Binary_to_ASCII( com   ,       trs_ptr + 11 );    /* MCC DRIVE COMMAND SET               */
      Binary_to_ASCII( dt1 ,         trs_ptr + 13 );    /* MCC DRIVE DATA1 SET                 */
      Binary_to_ASCII( dt2 ,         trs_ptr + 15 );    /* MCC DRIVE DATA2 SET                 */
      Binary_to_ASCII( dt3 ,         trs_ptr + 17 );    /* MCC DRIVE DATA3 SET                 */
      *( trs_ptr + 19 )    =  CR;                       /* END CODE                            */
      *( trs_ptr + 20 )    =  LF;

      request( trs_ptr, rev_ptr );                      /* 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             = '@';
      Binary_to_ASCII( 0x05,         trs_ptr + 1  );   /* REQUEST LENGTH                      */
      Binary_to_ASCII( ADR1,         trs_ptr + 3  );   /* SLAVE ADDRES SET                    */
      Binary_to_ASCII( TYPE_C770AL,  trs_ptr + 5  );   /* SLAVE TYPE SET                      */
      Binary_to_ASCII( 0x11,         trs_ptr + 7  );   /* DRIVE COMMAND PORT WRITE REQUEST SET */
      Binary_to_ASCII( AXIS_X,       trs_ptr + 9  );   /* X AXIS SET                          */
      Binary_to_ASCII( com   ,       trs_ptr + 11 );   /* MCC DRIVE COMMAND SET               */
      *( trs_ptr + 13 )    =  CR;                      /* END CODE                            */
      *( trs_ptr + 14 )    =  LF;

      request( trs_ptr, rev_ptr );                     /* 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= '@';
      Binary_to_ASCII( 0x08,         trs_ptr+ 1  );    /* REQUEST LENGTH                        */
      Binary_to_ASCII( ADR1,         trs_ptr+ 3  );    /* SLAVE ADDRES SET                      */
      Binary_to_ASCII( TYPE_C770AL,  trs_ptr+ 5  );    /* SLAVE TYPE SET                        */
      Binary_to_ASCII( 0x20,         trs_ptr+ 7  );    /* COUNTER COMMAND ALL WRITE REQUEST SET */
      Binary_to_ASCII( AXIS_X,       trs_ptr+ 9  );    /* X AXIS SET                            */
      Binary_to_ASCII( com   ,       trs_ptr+ 11 );    /* MCC COUNTER COMMAND SET               */
      Binary_to_ASCII( dt1 ,         trs_ptr+ 13 );    /* MCC COUNTER DATA1 SET                 */
      Binary_to_ASCII( dt2 ,         trs_ptr+ 15 );    /* MCC COUNTER DATA2 SET                 */
      Binary_to_ASCII( dt3 ,         trs_ptr+ 17 );    /* MCC COUNTER DATA3 SET                 */
      *( trs_ptr + 19 )    =  CR;                      /* END CODE                              */
      *( trs_ptr + 20 )    =  LF;

      request( trs_ptr, rev_ptr );                     /* 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( void )
 {
      *trs_ptr             = '@';
      Binary_to_ASCII( 0x04,         trs_ptr + 1 );        /* REQUEST LENGTH               */
      Binary_to_ASCII( ADR1 ,        trs_ptr + 3 );        /* SLAVE ADDRESS                */
      Binary_to_ASCII( TYPE_C770AL , trs_ptr + 5 );        /* SLAVE TYPE SET               */
      Binary_to_ASCII( 0x40 ,        trs_ptr + 7 );        /* STATUS1 PORT READ REQUEST SET */
      Binary_to_ASCII( AXIS_X ,      trs_ptr + 9 );        /* X AXIS SET                   */
      *( trs_ptr + 11 )    =  CR;                          /* END CODE                     */
      *( trs_ptr + 12 )    =  LF;                          /* END CODE                     */

      request( trs_ptr, rev_ptr );                         /* REQUEST START                */

      return ASCII_to_Binary( rev_ptr + 5 );

}
```

(5)PULSE COUNTER DATA READ PROGRAM 例

ここでは読み出した PULSE COUNTER の COUNT 値を RETURN 値とする関数例です。

```
/*-------------------------------------------------------------------*/
/*              X-AXIS COUNTER READ                                  */
/*-------------------------------------------------------------------*/

 long xcntred( void )
 {
      long a;

      xdcom( 0xfc );                                   /* PULSE COUNTER PORT SELECT COMMAND OUT */
      *trs_ptr             = '@';
      Binary_to_ASCII( 0x04,         trs_ptr + 1 );    /* REQUEST LENGTH               */
      Binary_to_ASCII( ADR1 ,        trs_ptr + 3 );    /* SLAVE ADDRESS                */
      Binary_to_ASCII( TYPE_C770AL , trs_ptr + 5 );    /* SLAVE TYPE SET               */
      Binary_to_ASCII( 0x30 ,        trs_ptr + 7 );    /* DRIVE DATA PORT ALL READ REQUEST SET */
      Binary_to_ASCII( AXIS_X ,      trs_ptr + 9 );    /* X AXIS SET                   */
      *( trs_ptr + 11 )    =  CR;                      /* END CODE                     */
      *( trs_ptr + 12 )    =  LF;                      /* END CODE                     */

      request( trs_ptr, rev_ptr );                     /* REQUEST START                */


      *( ( UC * )&a + 2 ) = ASCII_to_Binary( rev_ptr+ 5 );   /* COUNTER MSB IN       */
      *( ( UC * )&a + 1 ) = ASCII_to_Binary( rev_ptr+ 7 );
      *( ( UC * )&a     ) = ASCII_to_Binary( rev_ptr+ 9 );   /* COUNTER LSB IN       */

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

      return a;

 }
```

12-8.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              = '@';
      Binary_to_ASCII( 0x03 ,      trs_ptr + 1 );          /* REQUEST LENGTH SET          */
      Binary_to_ASCII( ADR2 ,      trs_ptr + 3 );          /* SLAVE ADDRESS SET           */
      Binary_to_ASCII( TYPE_CB08 , trs_ptr + 5 );          /* SLAVE TYPE SET              */
      Binary_to_ASCII( 0x60 ,      trs_ptr + 7 );          /* CB08 I/O READ REQUEST       */
      *( trs_ptr  + 9 )     =  CR;                         /* END CODE                    */
      *( trs_ptr  + 10 )    =  LF;

      request( trs_ptr, rev_ptr );

      *( ( UC * )&a + 3 ) = ASCII_to_Binary( rev_ptr + 5  );
      *( ( UC * )&a + 2 ) = ASCII_to_Binary( rev_ptr + 7  );
      *( ( UC * )&a + 1 ) = ASCII_to_Binary( rev_ptr + 9  );
      *( ( UC * )&a     ) = ASCII_to_Binary( rev_ptr + 11 );

       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              = '@';
      Binary_to_ASCII( 0x05 ,                 trs_ptr + 1  );  /* REQUEST LENGTH SET          */
      Binary_to_ASCII( ADR2 ,                 trs_ptr + 3  );  /* SLAVE ADDRESS SET           */
      Binary_to_ASCII( TYPE_CB08 ,            trs_ptr + 5  );  /* SLAVE TYPE SET              */
      Binary_to_ASCII( 0x50 ,                 trs_ptr + 7  );  /* CB08 I/O READ REQUEST       */
      Binary_to_ASCII( *( ( UC *)&data     ) , trs_ptr + 9  );  /* SLAVE TYPE SET              */
      Binary_to_ASCII( *( ( UC *)&data + 1 ) , trs_ptr + 11 );  /* CB08 I/O READ REQUEST       */
      *( trs_ptr  + 13 )     =  CR;                             /* END CODE                    */
      *( trs_ptr  + 14 )     =  LF;

      request( trs_ptr, rev_ptr );
  }
```

12-9.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*/

  }
```

12-10.C-770AL(MCC05 V2) INITIALIZE PROGRAM 例

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 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 STATUS 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, 0x08, DUMMY, DUMMY );            /* SPEC INITIALIZE1 COMMAND OUT */

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

      /** ADDRESS INITIALIZE COMMAND **/
         xmccrdy();
         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。

12-11.C-770AL(MCC05 V2) 実動作 PROGRAM 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

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, DUMMY, 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 Hz 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 ABSORUTE INDEX DRIVE                          */
/*-------------------------------------------------------------------*/

   void xabsindex( void )
  {
      /** RATE SET COMMAND **/
      xmccrdy();                                /* X-AXIS MCC05 RDY WAIT */
      xdall( 0x06, DUMMY, 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 */
      xdall( 0x15,  *( ( UC *)&adsdt + 1 ), *( ( UC *)&adsdt + 2 ), *( ( UC *)&adsdt + 3 )
                                                /* ABSOLUTE INDEX DRIVE COMMAND OUT  */
  }
```

(注)RAM AREA urate,drate には RATE DATA TABLE の No.が、lspd,hspd には Hz 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

ORIGIN DRIVE には URATE,DRATE,LSPD,HSPD,CSPD,OFFSET PULSE,LIMIT DELAY,SCAN DELAY,JOG DELAY 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, DUMMY, 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, DUMMY, DUMMY, offset );    /*OFFSET PULSE SET COMMAND OUT */

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

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

  }
```

(注)RAM エリア urate,drate には RATE DATA TABLE の No.が、lspd,hspd,cspd には Hz 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。

※特に必要なければデフォルト値をそのまま設定して下さい。

12-12.CB-08 実動作 PROGRAM 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 ){          /*  SENSOR 1 ON ? */
           if( ( a & IN11 ) != 0x00 )       /*  SENSOR 2 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-14 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	*アンサーバックが返ってこない。	*CB-14 の電源が ON になっていることを確認して下さい。 *上位の RS232C 通信仕様と CB-14 の RS232C 通信仕様は合っていますか？　通信速度等、RS232C 通信の仕様を確認してください。 *CB-14 の RS232C 通信速度設定が自動合わせ(ディップスイッチの設定がすべて OFF)になっていませんか？ RS232C 通信速度設定を自動合わせにした場合、CB-14 の電源 ON 後にまず自動合わせ実行シーケンスを行わなければなりません。
2	*アンサーバックのエラー判定結果が未初期化エラー(07 H)である。	*CB-14 に対し、初期化リクエストで正しく初期化を行いましたか？ 初期化リクエストを実行しない限り、CB-14 は他のリクエストを受け付けません。 初期化は CB-14、スレーブの電源 ON → CB-14 の初期化リクエストの順序で行って下さい。
3	*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。	*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。 CB-14 に対し、初期化リクエストで再度、初期化を実行して下さい。
4	*アンサーバックのエラー判定結果がタイムアウトエラー(82 H)である。	*スレーブの電源が ON になっていることを確認して下さい。 スレーブの電源が ON になっていないとスレーブが応答せず、タイムアウトエラーが発生します。

１４．CB-14 COMMAND 一覧表

14-1.初期化機能一覧表

初期化リクエストで設定するパラメータです。

$D^7D^6D^5D^4D^3D^2D^1D^0$	HEX CODE	XID機能	ボーレート	リトライ回数	参照ページ
0 0 0 0 0 0 0 0	００	無 効	9765bps	０回	１０
0 0 0 0 0 0 0 1	０１	無 効	9765bps	１回	１０
0 0 0 0 0 0 1 0	０２	無 効	9765bps	２回	１０
0 0 0 0 0 0 1 1	０３	無 効	9765bps	３回	１０
0 0 0 0 1 0 0 0	０８	無 効	39062bps	０回	１０
0 0 0 0 1 0 0 1	０９	無 効	39062bps	１回	１０
0 0 0 0 1 0 1 0	０Ａ	無 効	39062bps	２回	１０
0 0 0 0 1 0 1 1	０Ｂ	無 効	39062bps	３回	１０
0 0 0 1 0 0 0 0	１０	無 効	156250bps	０回	１０
0 0 0 1 0 0 0 1	１１	無 効	156250bps	１回	１０
0 0 0 1 0 0 1 0	１２	無 効	156250bps	２回	１０
0 0 0 1 0 0 1 1	１３	無 効	156250bps	３回	１０
0 0 0 1 1 0 0 0	１８	無 効	625000bps	０回	１０
0 0 0 1 1 0 0 1	１９	無 効	625000bps	１回	１０
0 0 0 1 1 0 1 0	１Ａ	無 効	625000bps	２回	１０
0 0 0 1 1 0 1 1	１Ｂ	無 効	625000bps	３回	１０
0 0 1 0 0 0 0 0	２０	無 効	9765bps	０回	１０
0 0 1 0 0 0 0 1	２１	有 効	9765bps	１回	１０
0 0 1 0 0 0 1 0	２２	有 効	9765bps	２回	１０
0 0 1 0 0 0 1 1	２３	有 効	9765bps	３回	１０
0 0 1 0 1 0 0 0	２８	有 効	39062bps	０回	１０
0 0 1 0 1 0 0 1	２９	有 効	39062bps	１回	１０
0 0 1 0 1 0 1 0	２Ａ	有 効	39062bps	２回	１０
0 0 1 0 1 0 1 1	２Ｂ	有 効	39062bps	３回	１０
0 0 1 1 0 0 0 0	３０	有 効	156250bps	０回	１０
0 0 1 1 0 0 0 1	３１	有 効	156250bps	１回	１０
0 0 1 1 0 0 1 0	３２	有 効	156250bps	２回	１０
0 0 1 1 0 0 1 1	３３	有 効	156250bps	３回	１０
0 0 1 1 1 0 0 0	３８	有 効	625000bps	０回	１０
0 0 1 1 1 0 0 1	３９	有 効	625000bps	１回	１０
0 0 1 1 1 0 1 0	３Ａ	有 効	625000bps	２回	１０
0 0 1 1 1 0 1 1	３Ｂ	有 効	625000bps	３回	１０

14-2.対 CB-14 リクエスト一覧表

CB-14 に対するリクエストです。

$D^7D^6D^5D^4D^3D^2D^1D^0$	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	１３	
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	１３	
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	１３	
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	１３	
1 1 1 0 0 1 1 1	Ｅ７	使用禁止	１３	
1 1 1 0 1 0 0 0	Ｅ８	初期化リクエスト	１３	
1 1 1 0 1 0 0 1	Ｅ９	使用禁止	１３	
1 1 1 0 1 0 1 0	ＥＡ	232C 通信チェックリクエスト	１３	

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