リファレンスマニュアル

『MULTIPROG 用 CAN』

目　次

はじめに

この度は、アルゴシステム製品をお買い上げ頂きありがとうございます。
弊社製品を安全かつ正しく使用していただくために、お使いになる前に本書をお読みいただき、十分に理解していただくようお願い申し上げます。

１）　お願いと注意

本書では、下記の方法について説明します。
・MULTIPROG への CAN 機能の登録方法
・PLC プログラミング用 CAN ライブラリの使用方法

MULTIPROG や PLC プログラミングについての詳細は省略させていただきます。MULTIPROG および PLC プログラミングに関する資料および文献と併せて本書をお読みください。

第１章　MULTIPROG 用 CAN ライブラリ

本章では PHOENIX CONTACT 社製 MULTIPROG における CAN ライブラリについて、基本的な仕様,構成について説明します。

１－１　MULTIPROG とは

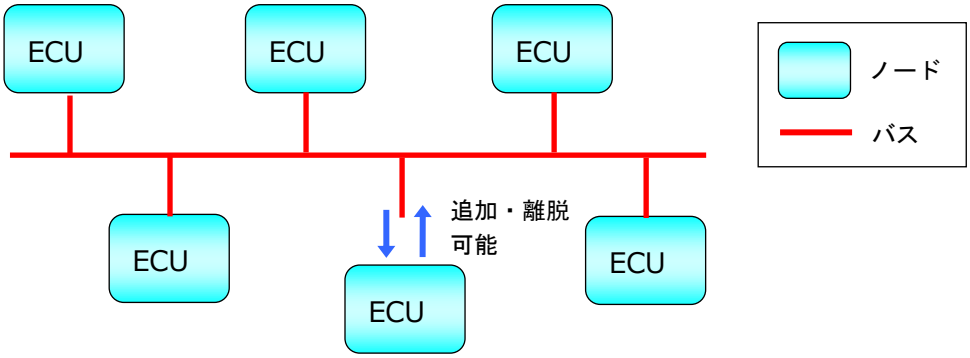
MULTIPROG とは、PHOENIX CONTACT 社が開発した、IEC に基づいて設計された PLC や従来からの PLC のための、標準的なプログラミングシステムです。
MULTIPROG は IEC61131-3 規格に基づいており、IEC の特徴を最大限含みます。

１－２　CAN とは

CAN は 1986 年、ドイツ Robert Bosch により仕様公開されたプロトコルです。後に、CAN プロトコルは国際標準化機構である ISO（International Organization for Standardization）により標準規格化（ISO11898／ISO11519）されました。現在ではほぼすべての自動車に採用され、さらに FA（Factory Automation）、産業機器など広く利用されています。

・マルチマスタ

CAN プロトコルではライン型構造で接続される各ノードに平等なバスアクセスが可能な「マルチマスタ方式」を採用しており、バスが空いているときに最初に送信を開始したノードが送信権を得ます。送信したいノードが複数ある場合はメッセージの衝突などが考えられるため、衝突回避策が必要となります。また、ネットワーク管理は各 ECU が個別に行っており、動的なノードの追加や削除が可能です。



図．CAN 通信構造図

CAN は本来マスタ、スレーブの関係はありません。しかし、弊社の CAN マスタモジュールを弊社の産業用 PC/産業用パネル PC/オールインワンコントローラに搭載することで、CAN を使ったマスタコントローラとして使用することができます。弊社で販売している CAN 接続の DI032 モジュールを最大 15 台まで接続でき、IN480 点、OUT480 点の制御が可能です。

また、CAN コントローラチップとして NXP(PHILIPS)製 SJA1000 を採用しており、通常の CAN としても制御することが可能です。

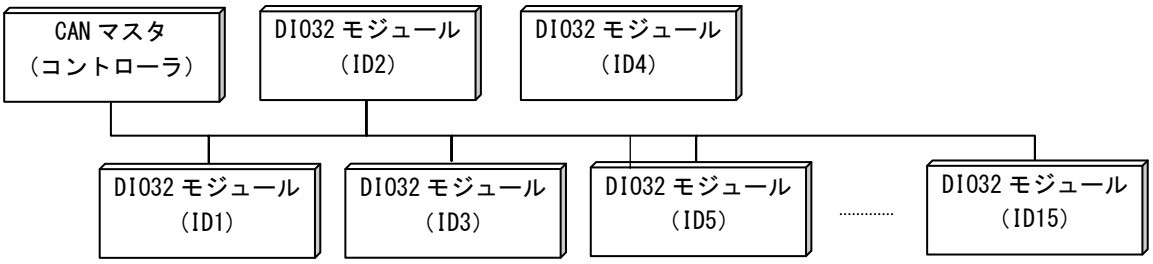


図 1-2-1．CAN 接続図

第２章　ファンクションブロック

本章では、ファンクションブロックについて説明します。
ファンクションブロックを使用する際のファームウェアライブラリ名称は「MP_FwLib_CANMst」になります。

２－１　　機能概要

１）マスタアクセス

ファンクション ブロック名	機　　能
CANMst_Open( )	CAN マスタをオープンします
CANMst_Close( )	CAN マスタをクローズします
CANMst_SetCommSetting( )	通信設定値を設定します
CANMst_GetCommSetting( )	通信設定値を読み出します
CANMst_SetAcceptFilter( )	CAN バス上に流れるデータのフィルタリング設定値の設定します
CANMst_GetAcceptFilter( )	CAN バス上に流れるデータのフィルタリング設定値を読み出します
CANMst_StartComm( )	CAN バス接続を開始します
CANMst_ChkStartComm( )	CAN バス接続完了を確認します
CANMst_ResetComm	CAN バス切断を開始します
CANMst_ChkResetComm	CAN バス切断完了を確認します
CANMst_GetMasterStatus( )	CAN マスタステータスを取得します
CANMst_ClearBuffer( )	指定したバッファをクリアします

２）デジタル入出力ユニット

ファンクション ブロック名	機　　能
CANSlv_DIN32Read( )	DIO32 スレーブの IN データを読み出します
CANSlv_DOUT32Write( )	DIO32 スレーブの OUT データを書き込みます
CANSlv_DOUT32Read ( )	DIO32 スレーブの OUT データを読み出します

５）ライブラリフローチャート

ライブラリを使用する際のファンクションブロック呼び出しのフローチャートを以下に示します。

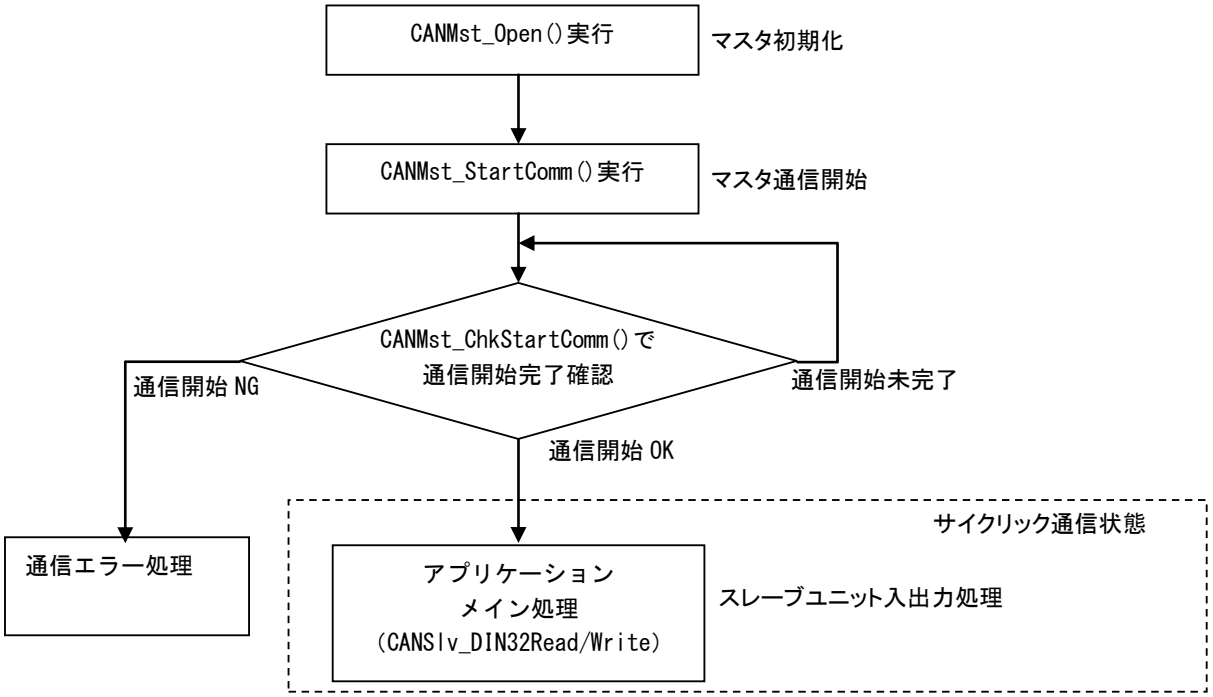


図 2-1-1. ライブラリフローチャート

PLC 開始後、マスタ初期化、マスタ通信開始を行うことでスレーブユニットへアクセス可能となります。通信が正常に開始されれば、スレーブユニットへの入出力を行うことができます。

２－２　　使用方法

MULTIPROG のプロジェクトでユニット毎のファンクションブロックを使用するためには、プロジェクト毎に登録が必要となります。
本項では登録方法について説明します。
ライブラリのインストールパスは「<MULTIPROG インストール先ディレクトリ>¥plc¥FW_LIB」になります。

①MULTIPROG 画面の左ペインにある「ライブラリ」を選択します。

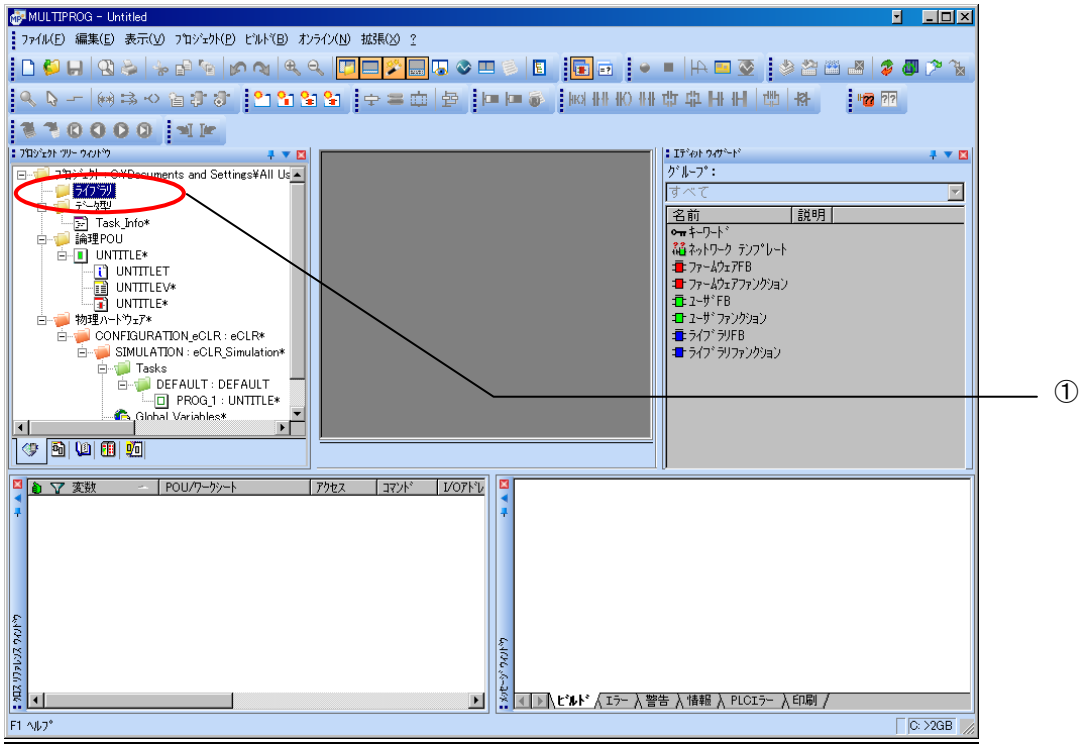


図 2-2-1. ライブラリ選択画面

②「ライブラリ」を右クリックし「挿入(I)」→「ファームウェアライブラリ(F)」と選択します。

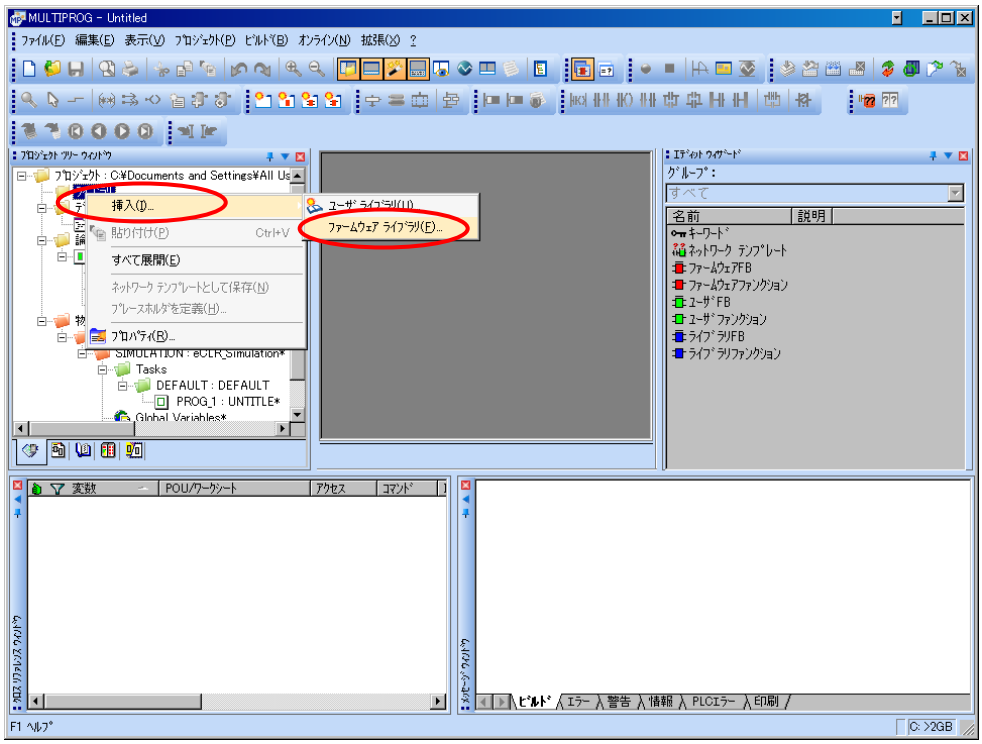


図 2-2-2.　ライブラリ挿入画面①

③図 2-2-3 の画面が表示されますので、使用するファームウェアライブラリと同じ名前のディレクトリを選択してください。

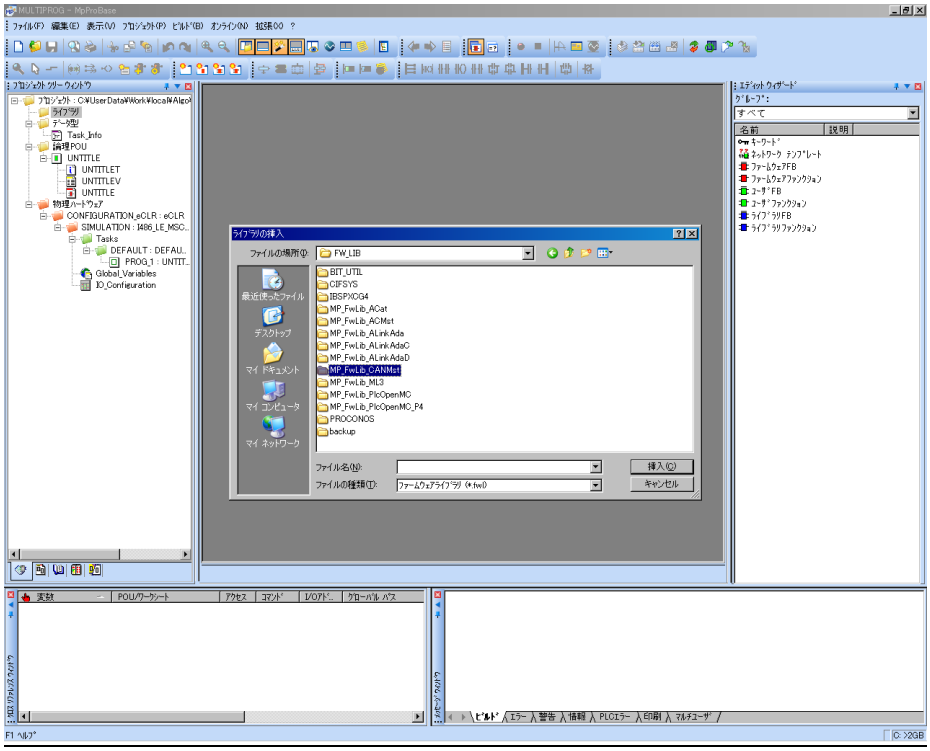


図 2-2-3.　ライブラリ挿入画面②

④図 2-2-4 の画面が表示されますので、選択したディレクトリと同じ名前のファイルを選択してください。

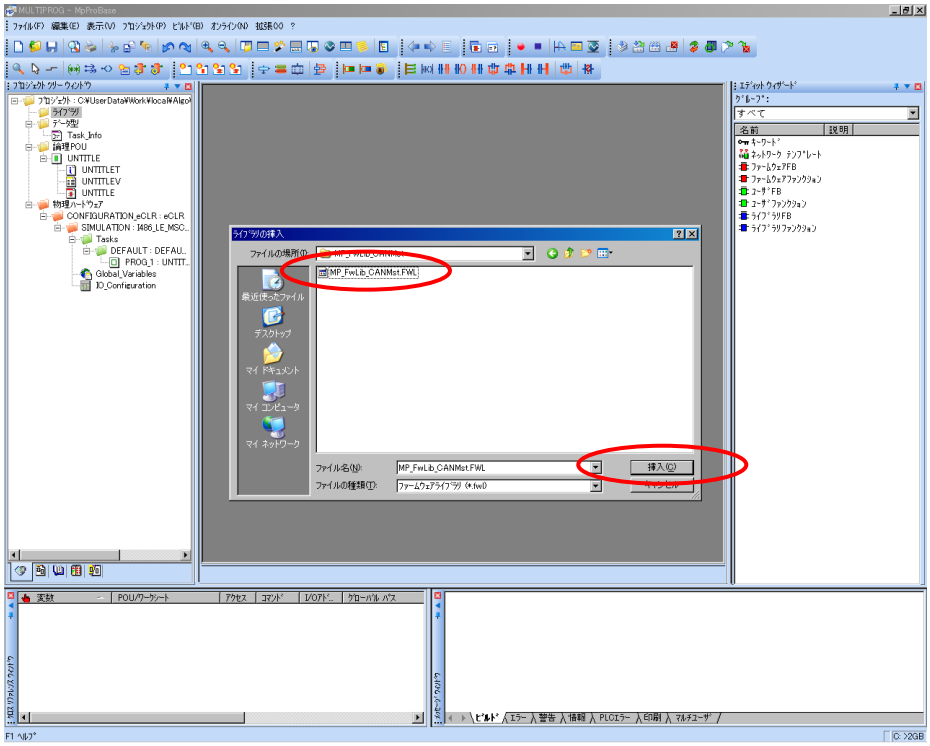


図 2-2-4.　ライブラリ挿入画面③

④最後に「挿入(C)」ボタンを押すことで、登録が完了します。
　ライブラリの項に選択したライブラリが追加されている事を確認してください。

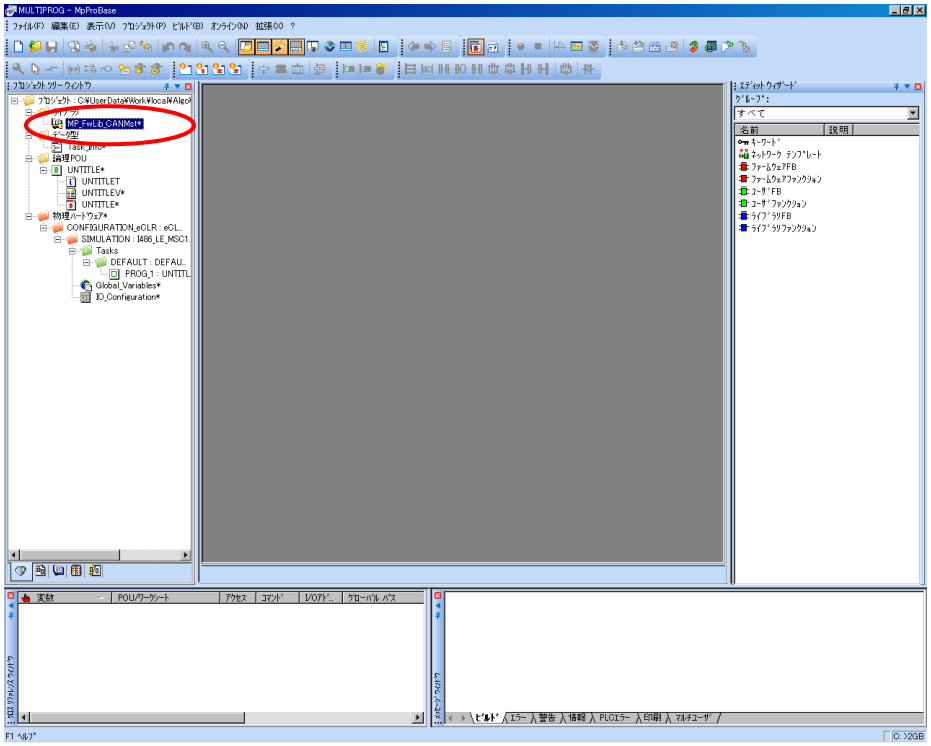


図 2-2-5. ライブラリ挿入画面④

２－３　　ファンクションブロックリファレンス

本項では MULTIPROG 用として用意した CAN のユニット毎のファンクションブロックについて、ファンクションブロックに共通の入出力パラメータについて説明します。

各ファンクションブロックにはコマンドを実行するための入力として「Act」、コマンドの応答結果を知らせるための出力として「Reply」「Error」を用意しています。全てのファンクションブロックに共通の制御を行っています。これらの制御について以下で説明します。

Act

立ち上がりエッジによりコマンドを実行します。以降は応答待ち状態になります。
Reply 出力が True となる前に本パラメータを False にする事で応答待ちを停止し、全ての出力を初期化します。応答確認による終了確認は処理されなくなりますが、コマンドの実行が可能な状態であれば実行されます。コマンドのキャンセル処理ではありません。
ファンクションブロックの入力パラメータを変更した場合は、この入力を False→True と変化させてコマンドを再度実行してください。
*)信号の入出力ファンクションブロックでは Act=True の間、信号の入出力を続けます。
　このため､信号の入出力ファンクションブロックでは､上記説明内の下線部については適用されません。

Reply

この出力パラメータが True になる事でコマンドの応答確認まで完了します。

Error

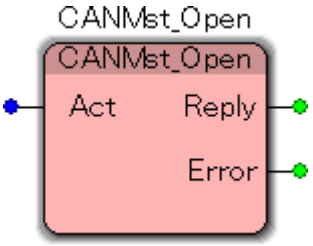
ファンクションブロックが正常終了の場合、正常時は 0 出力となります。
Reply が False の応答があった場合は、この出力パラメータを確認してください。

２－４　　マスタアクセス

CANMst_Open 関数

機能　CAN マスタをオープンします。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)

出力

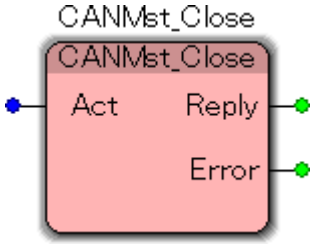
BOOL	Reply	：リターン(0:応答なし，1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　CAN マスタをオープンします。
CAN マスタをオープンし、CAN マスタプロセスとの接続を行います。
CAN マスタをオープンしていない場合、以降のファンクションブロックはエラーとなります。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。

CANMst_Close 関数

機能　CAN マスタをクローズします。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)

出力

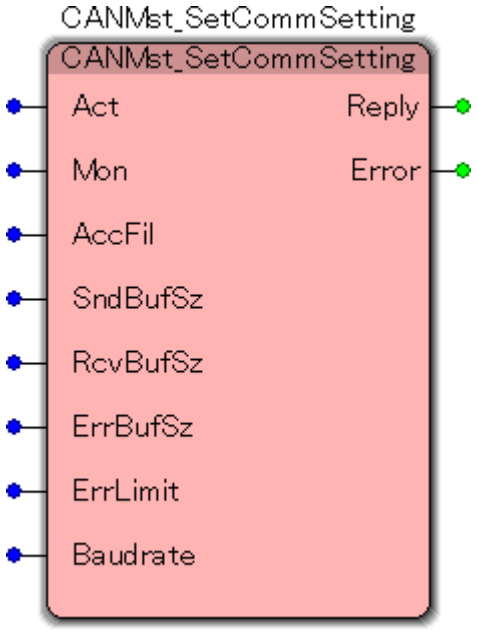
BOOL	Reply	：リターン(0:応答なし，1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　リソースを開放し、CAN マスタとの接続を切断します。
再度、CAN マスタへのアクセスを行う場合には CANMst_Open を実行する必要があります。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。

CANMst_SetCommSetting 関数

機能　CAN インターフェースの各種設定を行います。

書式



入力

BOOL	Act	：アクションフラグ(0:停止, 1:実行)
BYTE	Mon	：モニタモード
BYTE	AccFil	：アクセプタンスフィルタモード
UINT	SndBufSz	：送信バッファサイズ
UINT	RcvBufSz	：受信バッファサイズ
UINT	ErrBufSz	：エラーバッファサイズ
UINT	ErrLimit	：エラーリミット数
WORD	Baudrate	：通信速度設定

出力

BOOL	Reply	：リターン(0:応答なし, 1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　CAN インターフェースの設定を行います。
この設定は、CAN バスが切断されているときのみ実行できます。
CAN バスが接続されている場合、CN_ER_ALREADYCOMM が返ります。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。

通信設定

Mon　：モニタモード設定　（初期値：0）
[0：通常モード]
[1：モニタモード]

AccFil　：アクセプタンスフィルタモード設定　（初期値：1）
[0：Dual フィルタ]
[1：Single フィルタ]

SndBufSz　：送信バッファサイズ　（初期値：64）
[64 ～ 65536] 単位：【20Byte】

RcvBufSz　：受信バッファサイズ　（初期値：64）
[64 ～ 65536] 単位：【20Byte】

ErrBufSz　：エラーバッファサイズ　（初期値：64）
[64 ～ 65536] 単位：【3Byte】

ErrLimit　：エラーリミット数（SJA1000 の EWLR レジスタ設定値）　（初期値：96）
[0 ～255]

Baudrate　：通信ボーレート　（初期値：0x1C05）

通信速度	設定値
10【Kbps】	0x6F31
20【Kbps】	0x6F18
50【Kbps】	0x1C0E
100【Kbps】	0x2F05
125【Kbps】	0x1C05
250【Kbps】	0x1C02
500【Kbps】	0x6F00
800【Kbps】	0x1B00
1【Mbps】	0x0900

上記の表以外の通信ボーレートを設定する場合は、下記の計算方法を使って設定値を計算します。

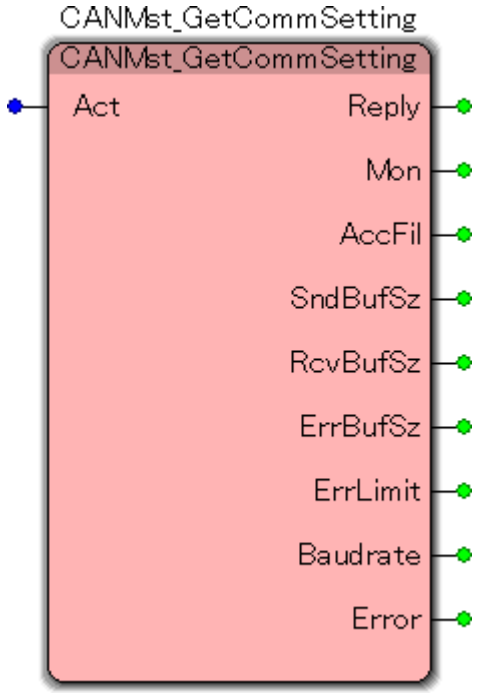
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAM	TSEG2			TSEG1				SJW		BRP					

入力クロック = 24【MHz】= 24000000【Hz】
fscl = 入力クロック【Hz】/ ((BRP + 1) * 2)
ボーレート = fscl / (1 + (TSEG1 + 1) + (TSEG2 + 1))
SAM：サンプリング回数 0：1 回 1：3 回

CANMst_GetCommSetting 関数

機能　CAN インターフェースの各種設定を読み出します。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)

出力

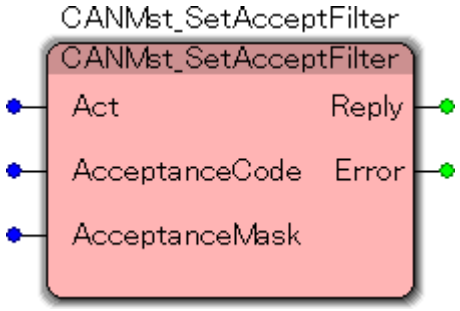
BOOL	Reply	：リターン(0:応答なし，1:実行終了)
BYTE	Mon	：モニタモード
BYTE	AccFil	：アクセプタンスフィルタモード
UINT	SndBufSz	：送信バッファサイズ
UINT	RcvBufSz	：受信バッファサイズ
UINT	ErrBufSz	：エラーバッファサイズ
UINT	ErrLimit	：エラーリミット数
WORD	Baudrate	：通信速度設定
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　CAN インターフェースの各種設定を読み出します。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。
通信設定値の詳細については、「CANMst_SetCommSetting」を参照してください。

CANMst_SetAcceptFilter 関数

機能　CAN バス上に流れるデータのフィルタリング設定値を設定します。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)
DWORD	AcceptanceCode	：ビットフィルタ値
DWORD	AcceptanceMask	：ビットマスク値

出力

BOOL	Reply	：リターン(0:応答なし，1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明

AcceptanceMask で 0 を指定された bit がフィルタの対象となります。
AcceptanceCode の値と一致したデータのみが、受信バッファに取り込まれます。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。

●通常フォーマット時のフィルタ適用範囲

ID(11bit)							RTR	1バイト目データ								2バイト目データ								3バイト目データ							
28	27	26	・・・	20	19	18	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

フィルタ適用範囲

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	RTR	リザーブ			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB1:7	DB1:6	DB1:5	DB1:4	DB1:3	DB1:2	DB1:1	DB1:0	DB2:7	DB2:6	DB2:5	DB2:4	DB2:3	DB2:2	DB2:1	DB2:0

●拡張フォーマット時のフィルタ適用範囲

ID(29bit)																RTR	1バイト目データ							
28	27	26	25	・・・								3	2	1	0	0	7	6	5	4	3	2	1	0

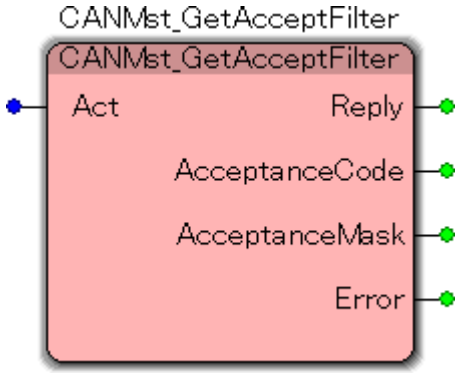
フィルタ適用範囲

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR	リザーブ	

CANMst_GetAcceptFilter 関数

機能　CAN バス上に流れるデータのフィルタリング設定値を読み出します。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)

出力

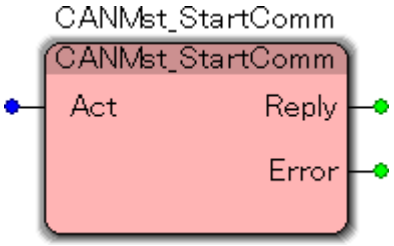
BOOL	Reply	：リターン(0:応答なし，1:実行終了)
DWORD	AcceptanceCode	：ビットフィルタ値
DWORD	AcceptanceMask	：ビットマスク値
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　CAN バス上に流れるデータのフィルタリング設定値を読み出します。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。
フィルタ設定値の詳細については、「CANMst_SetAcceptFilter」を参照してください。

CANMst_StartComm 関数

機能　CAN バス接続を開始します。

書式



入力

BOOL	Act	：アクションフラグ(0:停止, 1:実行)

出力

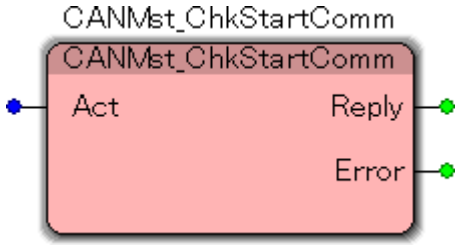
BOOL	Reply	：リターン(0:応答なし, 1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　CAN バスの通信設定を確定し、通信開始します。
送信バッファ、受信バッファ、エラーバッファのサイズはこの関数実行時に決まります。
通信中は、通信設定およびフィルタ設定は実行することができません。
CANMst_ChkStartComm を使用し、通信確立完了を待つ必要があります。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。

CANMst_ChkStartComm 関数

機能　CAN バス接続開始の確認を行います。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)

出力

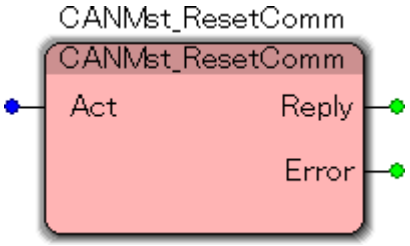
BOOL	Reply	：リターン(0:応答なし，1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　「CANMst_StartComm」の接続状況を確認します。
CN_ER_NOTCOMM が返答される間は、通信接続処理中です。
通信確立時に Reply=1 になります。
本ファンクションブロックは入力 Act が True の間、処理を続けます。

CANMst_ResetComm 関数

機能　CAN バスを切断します。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)

出力

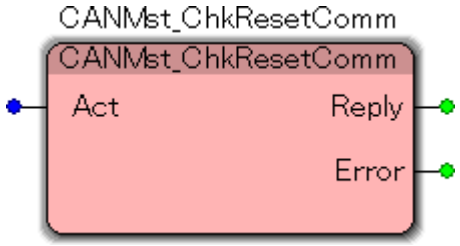
BOOL	Reply	：リターン(0:応答なし，1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　CAN バスを切断します。
送信バッファ、受信バッファ、エラーバッファのリソースを開放します。
CANMst_ChkResetComm を使用し、通信停止完了を待つ必要があります。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。

CANMst_ChkResetComm 関数

機能　CAN バス切断の確認を行います。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)

出力

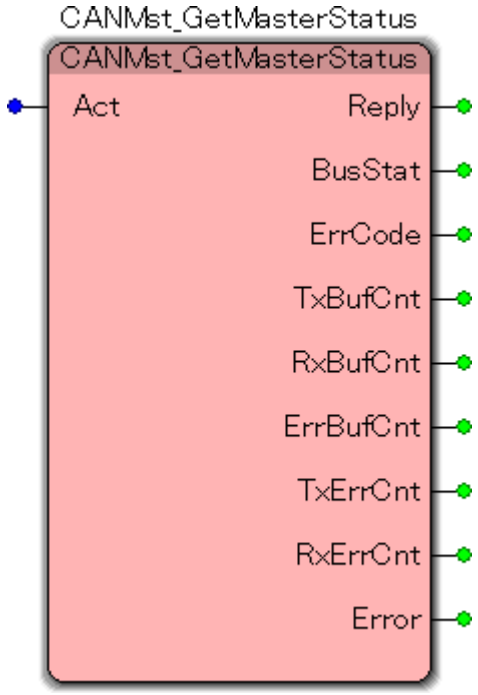
BOOL	Reply	：リターン(0:応答なし，1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　「CANMst_ResetComm」の接続状況を確認します。
CN_ER_NOTRESET が返答される間は、通信切断処理中です。
通信停止時に Reply=1 になります。
本ファンクションブロックは入力 Act が True の間、処理を続けます。

CANMst_GetMasterStatus 関数

機能　CAN バスのステータスを取得します。

書式



入力

BOOL	Act	：アクションフラグ(0:停止, 1:実行)

出力

BOOL	Reply	：リターン(0:応答なし, 1:実行終了)
INT	BusStat	：CAN バスステータス
INT	ErrCode	：マスタエラーコード
INT	TxBufCnt	：送信バッファにある未送信 CAN メッセージ数
INT	RxBufCnt	：受信バッファにある受信済み CAN メッセージ数
INT	ErrBufCnt	：エラーバッファにあるエラー情報数
SINT	TxErrCnt	：送信エラーカウンタ値
SINT	RxErrCnt	：受信エラーカウンタ値
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　CAN バスのステータスを取得します。
本ファンクションブロックは入力 Act が True の間、処理を続けます。

ステータス

BusStat　：　CAN バスステータス

BusStat	内容
CANMSTPROC_STS_INIT	0: マスタプロセス初期状態
CANMSTPROC_STS_WAIT	1: マスタプロセス通信待機状態
CANMSTPROC_STS_ACT	2: マスタプロセス通信処理中
CANMSTPROC_STS_CONNECT	3: マスタプロセス通信状態
CANMSTPROC_STS_RACT	4: マスタプロセス通信リセット処理中

ErrCode　：　マスタエラーコード

ErrCode	内容
CANMSTPROC_ERR_OK	0: エラー無し
CANMSTPROC_ERR_NOTWAIT	1: マスタプロセス通信待機状態でない
CANMSTPROC_ERR_INIT	2: マスタプロセス通信初期化失敗
CANMSTPROC_ERR_NOSEND	3: 送信失敗
CANMSTPROC_ERR_NOSENDMSG	4: 送信メッセージがない
CANMSTPROC_ERR_COMMFAIL	5: 通信異常

TxBufCnt　：　送信バッファにある未送信 CAN メッセージ数

RxBufCnt　：　受信バッファにある受信済み CAN メッセージ数

ErrBufCnt　：　エラーバッファにあるエラー情報数

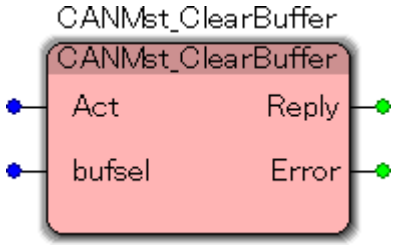
TxErrCnt　：　送信エラーカウンタ値

RxErrCnt　：　受信エラーカウンタ値

CANMst_ClearBuffer 関数

機能　指定したバッファやカウント値をクリアします。

書式



入力

BOOL	Act	：アクションフラグ(0:停止，1:実行)
BYTE	bufsel	：クリアするバッファを選択

出力

BOOL	Reply	：リターン(0:応答なし，1:実行終了)
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　指定したバッファやカウンタをクリアします。
本ファンクションブロックは入力 Act の False→True と変化した際にのみ実行されます。

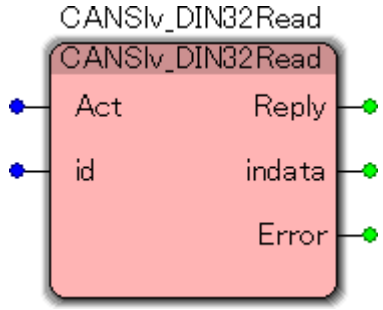
bufsel	内容
CLR_SNDBUF	0x01：送信バッファクリア
CLR_RCVBUF	0x02：受信バッファクリア
CLR_SNDERRCNT	0x04：送信エラーカウンタクリア
CLR_RCVERRCNT	0x08：受信エラーカウンタクリア
CLR_ERRBUF	0x10：エラーバッファクリア

２－５　　デジタル入出力ユニット

CANSlv_DIN32Read 関数

機能　　DIO32 スレーブの IN32bit データを読み出します。

書式



入力

BOOL	Act	：アクションフラグ(0:停止, 1:実行)
UDINT	id	：スレーブ番号

出力

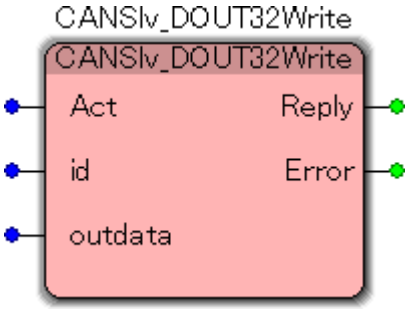
BOOL	Reply	：リターン(0:応答なし, 1:実行終了)
DWORD	indata	：読込データ
DINT	Error	：エラーコード(エラーコード一覧を参照)

説明　　指定された DIO32 スレーブの IN32bit データを読み出します。
スレーブ番号はスレーブユニットに設定した DipSW の値を指定する必要があります。
本ファンクションブロックは入力 Act が True の間、処理を続けます。

CANSlv_DOUT32Write 関数

機能　DIO32 スレーブの OUT32bit データを書き込みます。

書式



入力

BOOL	Act	: アクションフラグ(0:停止, 1:実行)
UDINT	id	: スレーブ番号
DWORD	outdata	: 書込データ

出力

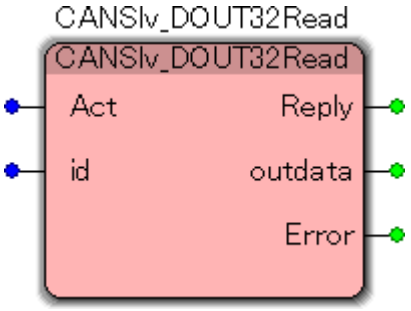
BOOL	Reply	: リターン(0:応答なし, 1:実行終了)
DINT	Error	: エラーコード(エラーコード一覧を参照)

説明　指定された DIO32 スレーブの OUT32bit データを書き込みます。
スレーブ番号はスレーブユニットに設定した DipSW の値を指定する必要があります。
本ファンクションブロックは入力 Act が True の間、処理を続けます。

CANSlv_DOUT32Read 関数

機能　DIO32 スレーブの OUT32bit データを読み出します。

書式



入力

BOOL	Act	: アクションフラグ(0:停止, 1:実行)
UDINT	id	: スレーブ番号

出力

BOOL	Reply	: リターン(0:応答なし, 1:実行終了)
DWORD	outdata	: 出力中データ
DINT	Error	: エラーコード(エラーコード一覧を参照)

説明　指定された DIO32 スレーブの OUT32bit データを読み出します。
スレーブ番号はスレーブユニットに設定した DipSW の値を指定する必要があります。
本ファンクションブロックは入力 Act が True の間、処理を続けます。

２－６　　エラーコード

エラーコード一覧

表 2-6-1. エラーコード一覧

エラーコード定義名	エラーコード	内容
CN_ER_OK	0x0000	正常です。
CN_ER_ALREADYOPEN	0x0001	すでにオープンしています。
CN_ER_NOMSTPROC	0x0002	CAN マスタプロセスが起動していません。
CN_ER_INVALIDPARAM	0x0003	無効な引数です。
CN_ER_NOTOPEN	0x0004	オープンしていません。
CN_ER_ALREADYCOMM	0x0005	すでに通信開始されています。
CN_ER_NOTCOMM	0x0006	通信していません。
CN_ER_NOTRESET	0x0007	リセットされていません。
CN_ER_COMMINITERR	0x0009	通信初期化エラーです。
CN_ER_ERROR	0x0101	内部エラーです。
CN_ER_CREATE	0x0102	各種デバイス生成失敗
CN_ER_TIMEOUT	0x0103	タイムアウトエラーです。
CN_ER_LOADDEVICE	0x0201	デバイスドライバロードエラーです。

第３章　付録

３－１　参考文献

●「IEC61131-3 を用いた PLC プログラミング」

著者　K.-H.John　／　M.Tiegelkamp
監訳者　PLCopen Japan
発行者　深田　良治
発行所　シュプリンガー・フェアラーク東京株式会社
発行年　2006 年

本 CD には PHOENIX CONTACT 社提供の MULTIPROG に関するマニュアルも収録しております。
MULTIPROG の使用方法に関する詳細などはそちらを参照してください。
各マニュアルは<開発環境 CD-ROM>¥CAN¥MAN¥に収録されています。
また、サンプルコードも<開発環境 CD-ROM>¥CAN¥SAMPLE¥に収録されています。こちらも参考にしてください。

このユーザーズマニュアルについて

(1)本書の内容の一部又は全部を当社からの事前の承諾を得ることなく、無断で複写、複製、掲載することは固くお断りします。
(2)本書の内容に関しては、製品改良のためお断りなく、仕様などを変更することがありますのでご了承ください。
(3)本書の内容に関しては万全を期しておりますが、万一ご不審な点や誤りなどお気付きのことがございましたらお手数ですが巻末記載の弊社までご連絡ください。その際、巻末記載の書籍番号も併せてお知らせください。

７７KW１００１６D　　２０１７年　４月　第４版
７７KW１００１６A　　２０１２年　１１月　初版

株式会社アルゴシステム

本社
〒５８７－００２１　大阪府堺市美原区小平尾６５６番地
ＴＥＬ(０７２)３６２－５０６７
ＦＡＸ(０７２)３６２－４８５６

ホームページ　http://www.algosystem.co.jp