ユーザーズマニュアル

ＡＣａｔ　ＲＳＬ
Ｖｅｒ．１．２０

目　次

ACat RSL の概要

第１章 アプリケーション開発

第２章 RSL 関数

第３章 付録

ACat RSL の概要

本 RT 共有ライブラリ(以下 RSL とする)「ACat.RSL」は、アルゴシステム EtherCAT マスタを利用する INtime アプリケーションを、容易に作成できるようにするために提供されます。

ユーザーは、Microsoft Visual Studio 等(※1)の開発言語から、RSL 関数をコールすることによって EtherCAT 通信や、各スレーブユニットへの入出力を処理するアプリケーションを作成することができます。

本 RSL は、上位と下位との複数からなっておりユーザーアプリケーションがアクセスする上位の「ACat.RSL」と EtherCAT マスタをアクセスする為の下位 RSL「ACMst.RSL」から構成されております。
ユーザーは下位 RSL および EtherCAT マスタを意識する必要はありません。
（EtherCAT マスタは使用するスレーブユニットの種類を config.xml ファイルで指定します。これにより、複数種類のスレーブユニットをサポートができるようになります。config.xml ファイルについては、「config.xml 設定マニュアル」を参照して下さい。また、下位 RSL「ACMst.RSL」の設定は「ACMst.ini 設定マニュアル」を参照して下さい。）

ユーザーは EtherCAT スレーブへのデータの入出力を、該当する関数をコールすることで簡単に行うことができます。

本バージョンより、EtherCAT 通信方法を変わっております。
以前のバージョンを使用している場合は、EtherCAT 通信開始手順の変更が必要です。「２－２－１　アプリケーション開始」のフローチャートを参考に、EtherCAT 通信開始手順を変更して下さい。

（※１）INtime6.2の開発環境にはVisual Studio 2008、Visual Studio 2010、Visual Studio 2012、Visual Studio 2013、Visual Studio 2015が使用可能です。

第１章　アプリケーション開発

１－１　EtherCAT 動作環境

ユーザーは作成するアプリケーション内でACat.RSLの関数をコールすることによりEtherCATスレーブの入出力を処理します。
EtherCAT マスタはシステムの情報としてネットワーク情報ファイル(config.xml)を参照しますので、EtherCAT マスタを使用したアプリケーションを動作させる前にネットワーク情報ファイル(config.xml)を作成する必要があります。
作成したアプリケーション、ACat.RSL、ACMst.RSL、ACatProc.rta、EcMaster.RSL、emlI8254x.RSL、ACMst.ini、config.xml は同一フォルダ（ディレクトリ）に格納してアプリケーションを動作させます。

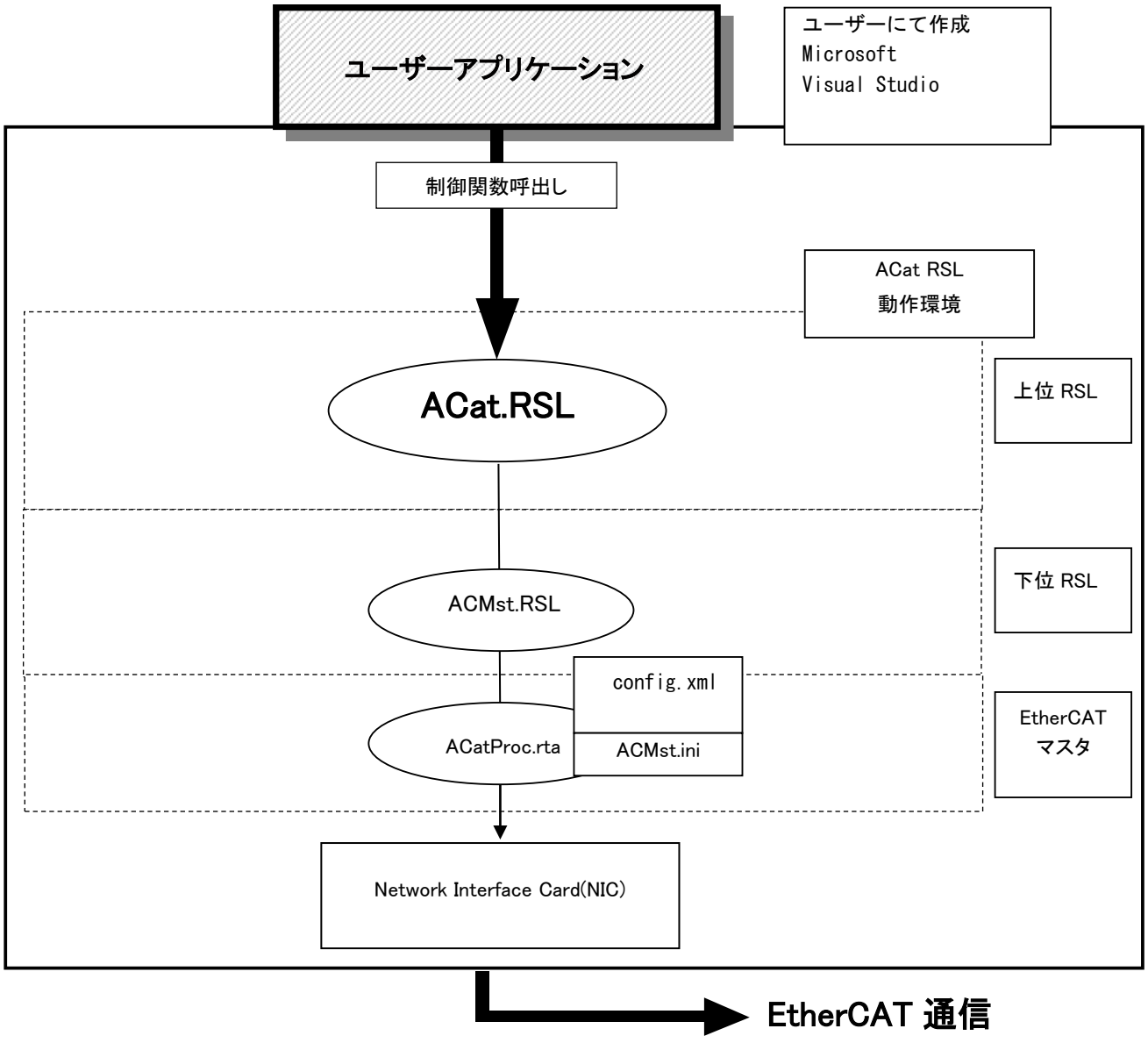

図1-1-1. EtherCAT 動作環境

１－２　アプリケーション開発の準備

開発アプリケーションから RSL をコールできるようにする為に、開発ユーザーは、下記の手順を実行します。

1）Microsoft Visual Studio
プロジェクトのソースファイルがあるフォルダに、AlgACat.CPP、AlgACat.H、ACatDef.H をコピーします。
RSL の関数をコールするソースファイルへ、AlgACat.H をインクルードします。
プロジェクトへ AlgACat.CPP を追加します。
プログラム起動時に実行する部分で、次の関数をコールして下さい。LoadACatRsl(“ACat.RSL”);
プログラム終了時に実行する部分で、次の関数をコールして下さい。UnloadACatRsl();

上記以外に C++のコンパイル設定で「プリコンパイルヘッダを使用しない」を指定して下さい。但し、「プリコンパイルヘッダを使用する」を指定する場合（ヘッダ指定が　stdafx.h　の時、つまりスイッチが /Yu"stdafx.h" の時）は、AlgACat.cpp の上部 ヘッダ指定に次の１行を追加して下さい。

```
例：　　#include "stdafx.h"　　＜－－追加行
　　#include "AlgACat.h"
```

＊　上記で使用されるヘッダファイル等は<省配線開発環境 CD-ROM>¥EtherCAT¥SDK ディレクトリに含まれています。また、開発環境 CD-ROM にはサンプルソースなども含まれますので合わせてご利用下さい。

第２章　RSL 関数

２－１　RSL 関数概要

ACat ライブラリはライブラリ関数、マスタ制御関数、汎用 PDO/SDO 制御関数、CiA402 ドライブプロファイル制御関数と各ユニットタイプ（別記号）ごとに対応する制御関数が用意されています。

各ユニットタイプごとに対応する制御関数の形式は、ACat_[タイプ別記号]_[コマンド](引数 1, 引数 2, ・・・) になります。

各関数の詳細は「ACat RSL リファレンスマニュアル」を参照して下さい。
ここでは、一般的な RSL 関数について説明します。

１）ライブラリ関数

ライブラリ関数はライブラリの初期化、終了を行います。

ACat ライブラリを使用するために必ずコールする必要があります。

ライブラリ関数

ACat_Create( )	ACat ライブラリの初期化を行います
ACat_Destroy( )	ACat ライブラリの終了を行います
ACat_Make_ENI( )	ネットワーク情報ファイル(config.xml)を生成します
ACat_Start( )	通信を開始します
ACat_StartEx( )	DC を有効にし、通信を開始します
ACat_Stop( )	通信を停止します
ACat_Dllname( )	ACat ライブラリの DLL 名称を取得します
ACat_Version( )	ACat ライブラリのバージョンを取得します

２）マスタ制御関数

マスタ制御関数には、マスタを直接制御するための関数と、マスタからスレーブを直接制御する関数が用意されています。マスタ制御関数では、関数の引数としてフィジカルアドレスを渡す必要があります。専用ユニットを制御するにはユニット制御関数を使用してください。

RSL 関数を使用するには、先ず各ユニットごとに Open 関数をコールする必要があります。正常リターンを確認してから、データ入出力関数をコールするようにして下さい。また、そのユニットのアクセスが不要になれば、Close 関数をコールするようにして下さい。

マスタ制御関数

ACat_Mst_Start( )	通信を開始します
ACat_Mst_Stop( )	通信を停止します
ACat_Mst_Get_State( )	マスタの State を取得します
ACat_Mst_Set_State( )	マスタの State を設定します
ACat_Mst_Get_Slave_Count( )	接続されるスレーブ数を取得します
ACat_Mst_Get_Slave_Info( )	接続されるスレーブ情報を取得します
ACat_Mst_Dc_Start( )	Distributed Clock を有効にします
ACat_Mst_Dc_Stop( )	Distributed Clock を無効にします
ACat_Mst_Dc_Config( )	Distributed Clock の設定を行います
ACat_Mst_RxPdo_Read( )	入力 PDO を読み込みます
ACat_Mst_TxPdo_Read( )	出力 PDO を読み込みます
ACat_Mst_TxPdo_Write( )	出力 PDO を書き込みます
ACat_Mst_Set_Notification_Hook( )	マスタ通知関数の登録を行います
ACat_Mst_Set_CycleEvent_Hook( )	マスタ周期関数の登録を行います
ACat_Mst_Get_Link_Connected( )	ポートのリンク状態を取得します
ACat_Mst_Get_Link_Connected_Main( )	ポート(メイン)のリンク状態を取得します
ACat_Mst_Get_Link_Connected_Sub( )	ポート(サブ)のリンク状態を取得します
ACat_Mst_Get_Connected_Slave_Count( )	通信しているスレーブ数を取得します
ACat_Mst_Get_Connected_Slave_Count_Main( )	ポート(メイン)の通信スレーブ数を取得します
ACat_Mst_Get_Connected_Slave_Count_Sub( )	ポート(サブ)の通信スレーブ数を取得します
ACat_Mst_Get_Configured_Slave_Count( )	コンフィギュレーションしているスレーブ数を取得します

マスタからスレーブを直接制御する関数

ACat_Slv_Read_Input( )	スレーブ入力を読み込みます
ACat_Slv_Read_Output( )	スレーブ出力を読み込みます
ACat_Slv_Write_Output( )	スレーブ出力を書き込みます
ACat_Slv_Get_State( )	スレーブの State を取得します
ACat_Slv_Set_State( )	スレーブの State を設定します
ACat_Slv_Get_Info( )	スレーブ情報を取得します
ACat_Slv_Get_Address( )	スレーブのフィジカルアドレスを取得します
ACat_Slv_Get_SlaveID( )	スレーブのスレーブアドレスを取得します
ACat_Slv_Reset( )	スレーブのソフトウェアリセットを行います
ACat_Slv_Sdo_Download( )	スレーブの SDO ダウンロードを行います
ACat_Slv_Sdo_DownloadReq( )	スレーブの SDO ダウンロードを開始し、すぐに戻ります
ACat_Slv_Sdo_Upload( )	スレーブの SDO アップロードを行います
ACat_Slv_Sdo_UploadReq( )	スレーブの SDO アップロードを開始し、すぐに戻ります
ACat_Slv_Foe_Download( )	スレーブの FoE ダウンロードを行います
ACat_Slv_Foe_DownloadReq( )	スレーブの FoE ダウンロードを開始し、すぐに戻ります
ACat_Slv_Foe_Upload( )	スレーブの FoE アップロードを行います
ACat_Slv_Foe_UploadReq( )	スレーブの FoE アップロードを開始し、すぐに戻ります
ACat_Slv_Read_Register( )	スレーブの ASIC のレジスタを読み込みます
ACat_Slv_Write_Register( )	スレーブの ASIC のレジスタを書き込みます
ACat_Slv_Read_Eeprom( )	スレーブの EEPROM を読み出します
ACat_Slv_Write_Eeprom( )	スレーブの EEPROM を書き込みます
ACat_Slv_Reload_Eeprom( )	スレーブの EEPROM をリロードします
ACat_Slv_Assign_Eeprom( )	スレーブの EEPROM アクセスを設定します
ACat_Slv_Active_Eeprom( )	スレーブの EEPROM アクセス設定を読み出します

３）汎用 PDO/SDO 制御関数

汎用 PDO/SDO 制御関数は、スレーブ PDO、SDO を直接制御するための関数が用意されています。関数の引数としてスレーブアドレス（スレーブ ID）を渡す必要があります。全てのスレーブはシステム全体で異なるスレーブアドレスをもつ必要があります。そのスレーブアドレスのユニットが実際にはどこに接続され、スレーブアドレスが何番であるかは、各スレーブのディップスイッチで指定します。

RSL 関数を使用するには、先ず Create 関数をコールする必要があります。正常リターンを確認してから、データ入出力関数をコールするようにして下さい。また、そのユニットのアクセスが不要になれば、Destroy 関数をコールするようにして下さい。

ACat_Gene_Create( )	汎用制御関数を初期化します
ACat_Gene_Destroy( )	汎用制御関数を終了します
ACat_Gene_Read_Input( )	入力 PDO を読み込みます
ACat_Gene_Read_Output( )	出力 PDO を読み込みます
ACat_Gene_Write_Output( )	出力 PDO を書き込みます
ACat_Gene_Sdo_Download( )	SDO ダウンロードを行います
ACat_Gene_Sdo_Upload( )	SDO アップロードを行います

４）CiA402 ドライブプロファイル制御関数

CiA402 ドライブプロファイル制御関数は、IEC 61800-7-201 および IEC 61800-7-301 に定められている、ドライブおよびモーションコントロールのデバイスプロファイルに対応したスレーブを制御するための関数群です。関数の引数にスレーブアドレス(スレーブ ID)を渡す必要があります。全てのスレーブはシステム全体で異なるスレーブアドレスをもつ必要があります。そのスレーブアドレスのユニットが実際にはどこに接続され、スレーブアドレスが何番であるかは、各スレーブのディップスイッチで指定します。

CiA402 ドライブプロファイル制御関数を使用するには、先ず Create 関数をコールする必要があります。正常リターンを確認してから、動作関数をコールするようにして下さい。また、終了時は Destroy 関数をコールするようにして下さい。

ACat_Motion_Create( )	ユニットを初期化します
ACat_Motion_Destroy( )	ユニットを終了します
ACat_Motion_ServoOn( )	ユニットをサーボ ON します
ACat_Motion_ServoOff( )	ユニットをサーボ OFF します
ACat_Motion_MoveAbsolute( )	ユニットを絶対位置制御します
ACat_Motion_MoveRelative( )	ユニットを相対位置制御します
ACat_Motion_MoveVelocity( )	ユニットを速度制御します
ACat_Motion_MoveTorque( )	ユニットをトルク制御します
ACat_Motion_MoveParameter( )	ユニットの動作パラメータを設定します
ACat_Motion_Homing( )	ユニットを原点復帰動作します
ACat_Motion_HomingParameter( )	ユニットの原点復帰パラメータを設定します
ACat_Motion_Stop( )	ユニットを停止します
ACat_Motion_Reset( )	ユニットをエラーリセットします
ACat_Motion_ReadTargetReached( )	ユニットの目標到達を取得します
ACat_Motion_ReadHomingEnd( )	ユニットの原点復帰完了を取得します
ACat_Motion_ReadAxisError( )	ユニットのエラーコードを取得します
ACat_Motion_ReadActualPosition( )	ユニットの現在位置を取得します
ACat_Motion_ReadActualVelocity( )	ユニットの現在速度を取得します
ACat_Motion_ReadActualTorque( )	ユニットの現在トルクを取得します
ACat_Motion_ReadTargetPosition( )	ユニットの目標位置を取得します
ACat_Motion_ReadTargetVelocity( )	ユニットの目標速度を取得します
ACat_Motion_ReadTargetTorque( )	ユニットの目標トルクを取得します
ACat_Motion_ReadInputData( )	ユニットのデジタル入力を取得します
ACat_Motion_WriteOutputData( )	ユニットのデジタル出力を書込みます
ACat_Motion_SetParameter( )	ユニットのパラメータを書込みます
ACat_Motion_GetParameter( )	ユニットのパラメータを取得します
ACat_Motion_WriteControlword( )	ユニットのコントロールワードを書込みます
ACat_Motion_ReadStatusword( )	ユニットのステータスワードを取得します

ACat_Motion_WriteShutdown( )	ユニットに「Shutdown」を書込みます
ACat_Motion_WriteSwitchOn( )	ユニットに「SwitchOn」を書込みます
ACat_Motion_WriteEnableOperation( )	ユニットに「Enable Operation」を書込みます
ACat_Motion_WriteDisableOperation( )	ユニットに「Disable Operation」を書込みます
ACat_Motion_WriteQuickStop( )	ユニットに「QuickStop」を書込みます
ACat_Motion_WriteDisableVoltage( )	ユニットに「DisableVoltage」を書込みます
ACat_Motion_WriteFaultReset( )	ユニットに「FaultReset」を書込みます
ACat_Motion_ReadFSAState( )	ユニットの FSA ステータスを取得します

５）ユニット制御関数

ユニット制御関数の関数名には、ユニットのタイプ別記号が用いられています。複数個の同じユニットを区別する為に、関数の引数としてスレーブアドレス（スレーブ ID）を渡す必要があります。全てのスレーブはシステム全体で異なるスレーブアドレスをもつ必要があります。そのスレーブアドレスのユニットが実際にはどこに接続され、スレーブアドレスが何番であるかは、各スレーブのディップスイッチで指定します。

RSL 関数を使用するには、先ず各ユニットごとに Create 関数をコールする必要があります。正常リターンを確認してから、データ入出力関数をコールするようにして下さい。また、そのユニットのアクセスが不要になれば、Destroy 関数をコールするようにして下さい。

・タイプ別記号 Dio　　デジタル入出力ユニット制御関数

入出力として ON、OFF の 2 状態を持つ接点を扱うユニットのための関数

ACat_Dio_Create( )	ユニットを初期化します
ACat_Dio_Destroy( )	ユニットを終了します
ACat_Dio_Open( )	ユニットをオープンします
ACat_Dio_Close( )	ユニットをクローズします
ACat_Dio_Read( )	ユニットの PDO を読み込みます
ACat_Dio_Write( )	ユニットの PDO に書き込みます
ACat_Dio_GetParam( )	ユニットのパラメータを取得します
ACat_Dio_SetParam( )	ユニットにパラメータを設定します
ACat_Dio_SaveParam( )	ユニットにパラメータを保存します
ACat_Dio_LoadParam( )	ユニットのパラメータを初期化します
ACat_Dio_SoftReset( )	ユニットをソフトウェアリセットします

・タイプ別記号 Aio　　アナログ入出力ユニット制御関数

入出力としてアナログ値を扱うユニットのための関数

ACat_Aio_Create( )	ユニットを初期化します
ACat_Aio_Destroy( )	ユニットを終了します
ACat_Aio_Open( )	ユニットをオープンします
ACat_Aio_Close( )	ユニットをクローズします
ACat_Aio_Read( )	ユニットの PDO を読み込みます
ACat_Aio_Write( )	ユニットの PDO に書き込みます
ACat_Aio_AD_GetParam( )	AD ユニットのパラメータを取得します
ACat_Aio_AD_SetMode( )	AD ユニットにモードを設定します
ACat_Aio_AD_SetParam( )	AD ユニットにパラメータを設定します
ACat_Aio_AD_SetCalib( )	AD ユニットにキャリブレーション値を設定します
ACat_Aio_DA_GetParam( )	DA ユニットのパラメータを取得します
ACat_Aio_DA_SetMode( )	DA ユニットにモードを設定します
ACat_Aio_DA_SetParam( )	DA ユニットにパラメータを設定します
ACat_Aio_DA_SetCalib( )	DA ユニットにキャリブレーション値を設定します
ACat_Aio_SoftReset( )	ユニットをソフトウェアリセットします

・タイプ別記号 Axis　　モーションコントローラユニット制御関数

モータの制御を扱うユニットのための関数

ACat_Axis_Create( )	ユニットを初期化します
ACat_Axis_Destroy( )	ユニットを終了します
ACat_Axis_Open( )	ユニットをオープンします
ACat_Axis_Close( )	ユニットをクローズします
ACat_Axis_Read( )	ユニットの PDO を読み込みます
ACat_Axis_Write( )	ユニットの PDO に書き込みます
ACat_Axis_GetParam( )	ユニットのパラメータを取得します
ACat_Axis_SetParam( )	ユニットにパラメータを設定します
ACat_Axis_SoftReset( )	ユニットをソフトウェアリセットします

・タイプ別記号 Enc　　エンコーダユニット制御関数

エンコーダ・カウンタ値を扱うユニットのための関数

ACat_Enc_Create( )	ユニットを初期化します
ACat_Enc_Destroy( )	ユニットを終了します
ACat_Enc_Open( )	ユニットをオープンします
ACat_Enc_Close( )	ユニットをクローズします
ACat_Enc_Read( )	ユニットの PDO を読み込みます
ACat_Enc_Write( )	ユニットの PDO に書き込みます
ACat_Enc_GetParam( )	ユニットのパラメータを取得します
ACat_Enc_SetDirection( )	ユニットに回転方向を設定します
ACat_Enc_SetEdgeEvaluation( )	ユニットに入力方式を設定します
ACat_Enc_SetMaxRingCount( )	ユニットにリングカウンタ最大値を設定します
ACat_Enc_SaveParam( )	ユニットにパラメータを保存します
ACat_Enc_LoadParam( )	ユニットのパラメータを初期化します
ACat_Enc_SoftReset( )	ユニットをソフトウェアリセットします

・タイプ別記号 Sio　　SIO ゲートウェイユニット制御関数

シリアル通信データを扱うユニットのための関数

ACat_Sio_Create	ユニットを初期化します
ACat_Sio_Destroy	ユニットを終了します
Acat_Sio_Open( )	ユニットをオープンします
Acat_Sio_Close( )	ユニットをクローズします
Acat_Sio_Read( )	ユニットの PDO を読み込みます
Acat_Sio_Write( )	ユニットの PDO に書き込みます
Acat_Sio_GetCommInfo( )	ユニットの COMMINFO パラメータを取得します
Acat_Sio_SetCommInfo( )	ユニットに COMMINFO パラメータを設定します
Acat_Sio_GetCommState( )	ユニットの通信状態を取得します
ACat_Sio_ClearError( )	ユニットの通信エラーをクリアします
ACat_Sio_PurgeComm( )	ユニットの入出力バッファを破棄します
ACat_Sio_SaveParam( )	ユニットにパラメータを保存します
ACat_Sio_LoadParam( )	ユニットのパラメータを初期化します
ACat_Sio_SoftReset( )	ユニットをソフトウェアリセットします

・タイプ別記号 Alink　　A-Link ゲートウェイユニット制御関数

A-Link ゲートウェイの入出力を扱うユニットのための関数

ACat_Alink_Create( )	ユニットを初期化します
ACat_Alink_Destroy( )	ユニットを終了します
ACat_Alink_Open( )	ユニットをオープンします
ACat_Alink_Close( )	ユニットをクローズします
ACat_Alink_Start( )	ユニットを通信開始します
ACat_Alink_Stop( )	ユニットを通信停止します
ACat_Alink_Read( )	ユニットの PDO を読み込みます
ACat_Alink_Write( )	ユニットの PDO に書き込みます
ACat_Alink_ReadChk2Occur( )	ユニットの CHK2 発生状況を取得します
ACat_Alink_GetStatus( )	ユニットのステータスを取得します
ACat_Alink_GetSysSts( )	ユニットのシステムステータスを取得します
ACat_Alink_GetChk1Count( )	ユニットの CHK1 発生回数を取得します
ACat_Alink_GetChk2Count( )	ユニットの CHK2 発生回数を取得します
ACat_Alink_ClrChk1Count( )	ユニットの CHK1 発生回数をクリアします
ACat_Alink_ClrChk2Count( )	ユニットの CHK2 発生回数をクリアします
ACat_Alink_GetCommParam( )	ユニットの通信設定を取得します
ACat_Alink_SetCommParam( )	ユニットに通信設定を設定します
ACat_Alink_SoftReset( )	ユニットをソフトウェアリセットします

・タイプ別記号 Mbg　　Modbus ゲートウェイユニット関数

MODBUS ゲートウェイの入出力を扱うユニットのための関数

ACat_Mbg_Create( )	ユニットを初期化します
ACat_Mbg_Destroy( )	ユニットを終了します
ACat_Mbg_Open( )	ユニットをオープンします
ACat_Mbg_Close( )	ユニットをクローズします
ACat_Mbg_ReadByte( )	ユニットの入力データを Byte 単位で読込みます
ACat_Mbg_WriteByte( )	ユニットの出力データを Byte 単位で書き込みます
ACat_Mbg_ReadWord( )	ユニットの入力データを Word 単位で読込みます
ACat_Mbg_WriteWord( )	ユニットに出力データを Word 単位で書き込みます
ACat_Mbg_Start( )	ターゲット機器との通信を開始します
ACat_Mbg_Stop( )	ターゲット機器との通信を停止します
ACat_Mbg_GetTargetInfo( )	ユニットのターゲット機器設定パラメータを取得します
ACat_Mbg_SetTargetInfo( )	ユニットにターゲット機器設定パラメータを設定します
ACat_Mbg_GetCommInfo( )	ユニットのシリアル通信設定パラメータを取得します
ACat_Mbg_SetCommInfo( )	ユニットにシリアル通信設定パラメータを設定します
ACat_Mbg_GetMonCmd( )	ユニットのモニタデータコマンドを取得します
ACat_Mbg_SetMonCmd( )	ユニットにモニタデータコマンドを設定します
ACat_Mbg_GetSpotReqCmd( )	ユニットの即時要求データコマンドを取得します
ACat_Mbg_SetSpotReqCmd( )	ユニットに即時要求データコマンドを設定します
ACat_Mbg_GetManReqCmd( )	ユニットの手動要求データコマンドを取得します
ACat_Mbg_SetManReqCmd( )	ユニットに手動要求データコマンドを設定します
ACat_Mbg_GetMonCmdErr( )	ユニットのモニタデータコマンドエラーステータスを取得します
ACat_Mbg_GetSpotReqCmdErr( )	ユニットの即時要求データコマンドエラーステータスを取得します
ACat_Mbg_GetManReqCmdErr( )	ユニットの手動要求データコマンドエラーステータスを取得します
ACat_Mbg_GetSpotReqCmdRes( )	ユニットの即時要求データコマンドレスポンスを取得します
ACat_Mbg_GetManReqCmdRes( )	ユニットの手動要求データコマンドレスポンスを取得します
ACat_Mbg_ClrMonCmdErr( )	ユニットのモニタデータコマンドエラーステータスをクリアします
ACat_Mbg_ClrSpotReqCmdErr( )	ユニットの即時要求データコマンドエラーステータスをクリアします
ACat_Mbg_ClrManReqCmdErr( )	ユニットの手動要求データコマンドエラーステータスをクリアします
ACat_Mbg_ClrSpotReqCmdRes( )	ユニットの即時要求データコマンドレスポンスをクリアします
ACat_Mbg_ClrManReqCmdRes( )	ユニットの手動要求データコマンドレスポンスをクリアします
ACat_Mbg_GetErrorStatus( )	ユニットのエラー状況を取得します
ACat_Mbg_GetResponseStatus( )	ユニットのレスポンス状況を取得します
ACat_Mbg_SaveParam( )	ユニットにパラメータを保存します
ACat_Mbg_LoadParam( )	ユニットのパラメータを初期化します
ACat_Mbg_SoftReset( )	ユニットをソフトウェアリセットします

・タイプ別記号 CN　　CUnet ゲートウェイユニット制御関数

CUnet ゲートウェイの入出力を扱うユニットのための関数

ACat_CN_Create( )	ユニットを初期化します
ACat_CN_Destroy( )	ユニットを終了します
ACat_CN_open( )	ユニットをオープンします
ACat_CN_close( )	ユニットをクローズします
ACat_CN_start( )	ユニットを通信開始します
ACat_CN_start_lf( )	ユニットを LF モードで通信開始します
ACat_CN_start_gmm( )	ユニットを GMM モードで通信開始します
ACat_CN_stop( )	ユニットを通信停止します
ACat_CN_reset( )	ユニットをリセットします
ACat_CN_GetCnectStat( )	ユニットの接続状態を取得します
ACat_CN_config( )	ユニットの通信設定を行います
ACat_CN_Status( )	ユニットのステータスを取得します
ACat_CN_MailOpen( )	ユニットのメールを通信開始します
ACat_CN_MailClose( )	ユニットのメールを通信停止します
ACat_CN_MailSend( )	ユニットからメールを送信します
ACat_CN_MailReceive( )	ユニットのメールを受信します
ACat_CN_MailStatus( )	ユニットのメール状況を取得します
ACat_CN_Put( )	ユニットの GM エリアを読み込みます
ACat_CN_Get( )	ユニットの GM エリアに書き込みます
ACat_CN_GetReg64( )	ユニットのレジスタを 64 ビットサイズで読み込みます
ACat_CN_GetReg32( )	ユニットのレジスタを 32 ビットサイズで読み込みます
ACat_CN_GetReg16( )	ユニットのレジスタを 16 ビットサイズで読み込みます
ACat_CN_SetReg64( )	ユニットのレジスタを 64 ビットサイズで書き込みます
ACat_CN_SetReg32( )	ユニットのレジスタを 32 ビットサイズで書き込みます
ACat_CN_SetReg16( )	ユニットのレジスタを 16 ビットサイズで書き込みます
ACat_CN_GetMemByte( )	ユニットの GM エリアを 1 バイトサイズで読み込みます
ACat_CN_GetMemShort( )	ユニットの GM エリアを 2 バイトサイズで読み込みます
ACat_CN_GetMemLong( )	ユニットの GM エリアを 4 バイトサイズで読み込みます
ACat_CN_GetMemDLong( )	ユニットの GM エリアを 8 バイトサイズで読み込みます
ACat_CN_SetMemByte( )	ユニットの GM エリアを 1 バイトサイズで書き込みます
ACat_CN_SetMemShort( )	ユニットの GM エリアを 2 バイトサイズで書き込みます
ACat_CN_SetMemLong( )	ユニットの GM エリアを 4 バイトサイズで書き込みます
ACat_CN_SetMemDLong( )	ユニットの GM エリアを 8 バイトサイズで書き込みます
ACat_CN_HardReset( )	ユニットをハードリセットします

２－２　RSL 使用方法

２－２－１　アプリケーション開始

ライブラリを使用したアプリケーション開始のフローチャートを以下に示します。

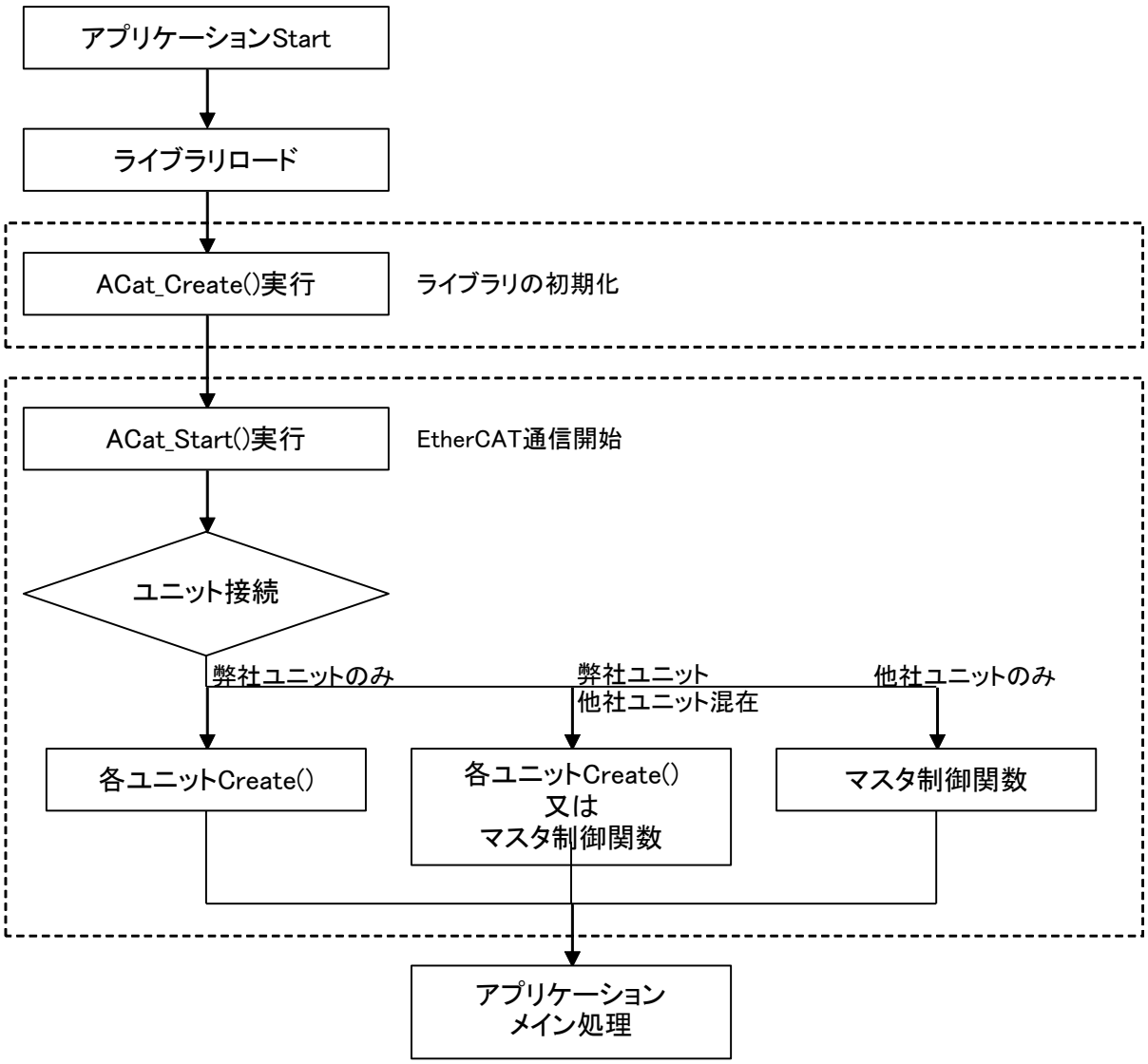

図2-2-1-1. アプリケーション開始フローチャート

アルゴシステム製品のスレーブユニットを使用する場合は、スレーブユニットの Create()関数を実行後、スレーブユニットのライブラリ関数を実行する事が出来ます。

他社製品へのアクセスは、マスタ制御関数や汎用 PDO/SDO 関数を使用します。

２－２－２　アプリケーション開始（DC モード有効のスレーブを使用する場合）

DC モード有効のスレーブが含まれる EtherCAT システム構成でライブラリを使用した
アプリケーション開始のフローチャートを以下に示します。

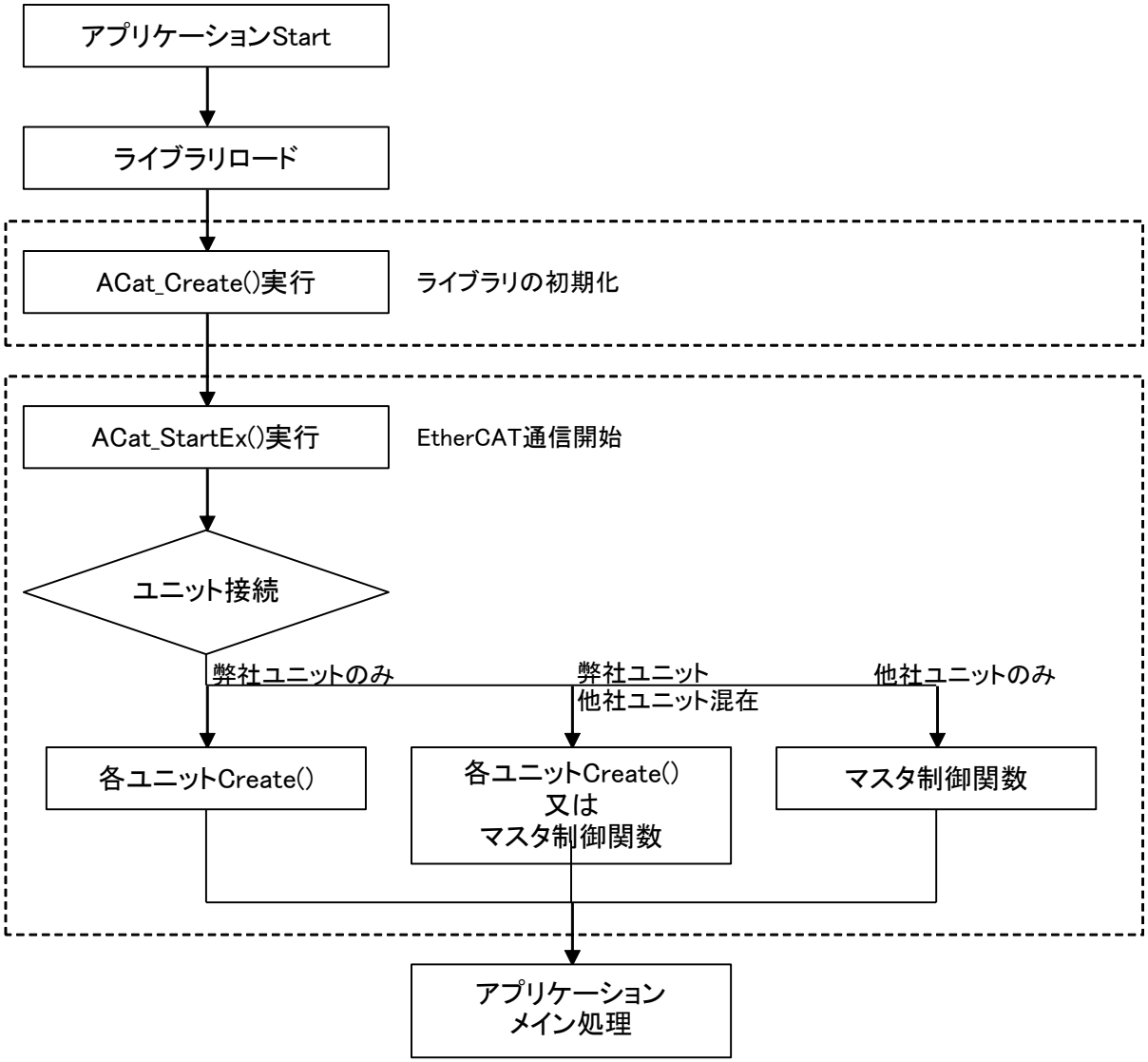

図 2-2-2-1. アプリケーション開始フローチャート

アルゴシステム製品のスレーブユニットを使用する場合は、スレーブユニットの Create()関数を実行後、スレーブユニットのライブラリ関数を実行する事が出来ます。

他社製品へのアクセスは、マスタ制御関数や汎用 PDO/SDO 関数、CiA402 ドライブプロファイル制御関数を使用します。

２－２－３　ネットワーク情報ファイル生成

ライブラリを使用したネットワーク情報ファイル作成を含むアプリケーション開始のフローチャートを以下に示します。

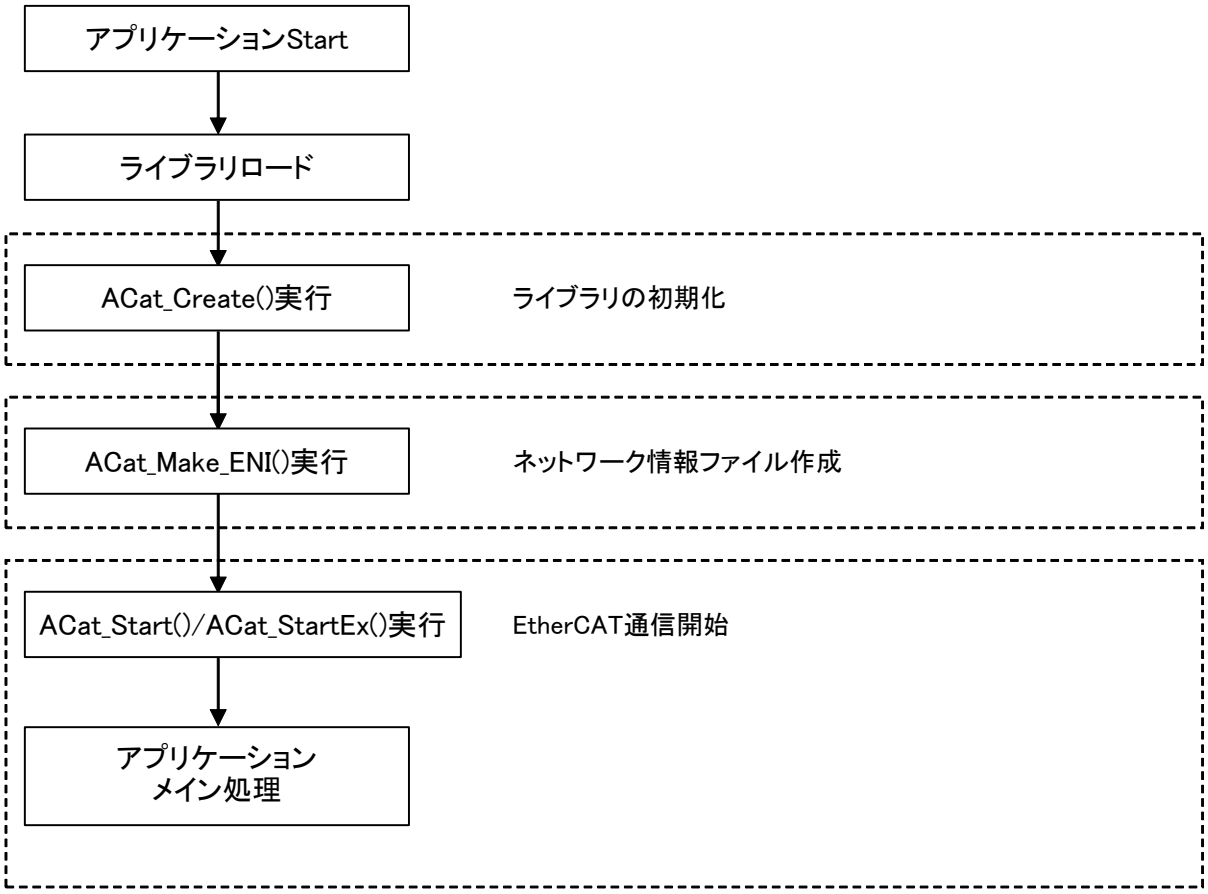

図 2-2-3-1. ネットワーク情報ファイル生成フローチャート

ネットワーク情報ファイルを作成する場合は、スレーブ設定ファイル、システム構成ファイルが別途必要になります。スレーブ設定ファイル、システム構成ファイルについては、別紙「EtherCAT ネットワーク情報ファイル config.xml 設定マニュアル」を参照してください。

２－２－４　アプリケーション終了

ライブラリを使用したアプリケーション終了のフローチャートを以下に示します。

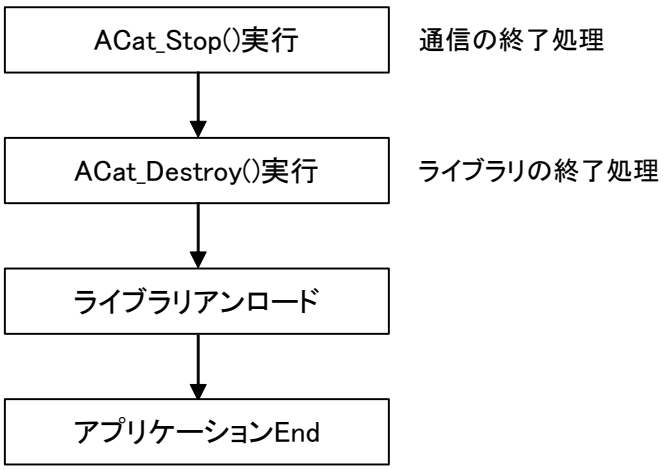

図 2-2-4-1. アプリケーション終了フローチャート

第３章　付録

３－１　サンプルコード

３－１－１　開発用ファイル

「EtherCAT マスタ開発基本ソフトキット SDK」に EtherCAT マスタライブラリにアクセスするためのヘッダファイルと EtherCAT マスタライブラリを使用したサンプルコードを用意しています。開発用ファイル、サンプルコードは INtime アプリケーションです。INtime6.2 開発環境で使用することが可能です。
開発環境に含まれる開発用ファイルの内容を表 3-1-1 に示します。

表3-1-1. EtherCAT マスタ開発基本ソフトキット SDK 開発用ファイル

開発環境のフォルダ	内容
¥SDK¥ACatRSLSample¥Develop	EtherCAT マスタライブラリへのアクセスに必要なヘッダファイルを格納しています。
¥SDK¥ACatRSLSample¥ACatRSLSample1	RSL リンク、ライブラリ初期化のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample2	マスタ制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample3	汎用 PDO/SDO 制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample4	CiA402 ドライブプロファイル制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample5	デジタル入出力ユニット制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample6	アナログ入出力ユニット制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample7	モーションコントロールユニット制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample8	エンコーダユニット制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample9	SIO ゲートウェイユニット制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample10	A-Link ゲートウェイユニット制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample11	Modbus ゲートウェイユニット制御関数のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample12	EtherCAT ネットワーク情報ファイルの動的生成のサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample13	FoE ダウンロードのサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample14	FoE アップロードのサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample15	CUnet ゲートウェイユニットの GM アクセスのサンプルコードです。
¥SDK¥ACatRSLSample¥ACatRSLSample16	CUnet ゲートウェイユニットのメール送受信のサンプルコードです。

３－１－２　RSL リンク、ライブラリ初期化サンプルコード

RSL とのリンク部分、ライブラリの初期化のサンプルコードを次に示します。
RSL リンクはプログラム開始時に必ず必要です。RSL リンクを実行しないとユニットへのアクセスができません。また、プログラム終了時には必ず RSL リンク解放を実行してください。

１）RSL リンク

```
//-----------------------------------------------------------------------------
//       RSL リンク
//-----------------------------------------------------------------------------
  int ret;

  // RSL リンク(この関数はサンプルファイル AlgACat.cpp/h に存在します)
  ret = LoadACatRsl();
  if (ret) {
      printf("RSL LOAD 失敗 RET=%08X", ret);
  } else {
      printf("RSL LOAD 完了");
  }

  // RSL 初期化
  ret = ACat_Create();
  if (ret) {
      printf("RSL 初期化失敗 RET=%08X", ret);
  } else {
      printf("RSL 初期化完了");
  }
```

２）RSL リンク解放

```
//-----------------------------------------------------------------------------
//      RSL リンク開放
//-----------------------------------------------------------------------------
  int ret;

  // RSL 終了
  ret = ACat_Destroy();
  if (ret) {
      printf("RSL 終了失敗 RET=%08X", ret);
  } else {
      printf("RSL 終了完了");
  }

  // RSL リンク開放(この関数はサンプルファイル AlgACat.cpp/h に存在します)
  UnLoadACatRsl();
```

３－１－３　マスタ制御関数サンプルコード

マスタ制御関数のサンプルコードを次に示します。

１）DC モード開始、EtherCAT 通信開始

```
//-----------------------------------------------------------------------------
//      DC モード開始、EtherCAT 通信開始
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_StartEx(100000, TRUE);                    // EtherCAT 通信開始(DC モード開始)
  if (ret) {
      printf("EtherCAT 通信開始失敗 RET=%08X", ret);
  } else {
      printf("EtherCAT 通信開始完了");
  }
```

２）DC モード停止、EtherCAT 通信終了

```
//-----------------------------------------------------------------------------
//      DC モード停止、EtherCAT 通信終了
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Stop(100000);                             // EtherCAT 通信停止(DC モード停止)
  if (ret) {
      printf("EtherCAT 通信停止失敗 RET=%08X", ret);
  } else {
      printf("EtherCAT 通信停止完了");
  }
```

３）スレーブ数取得、スレーブ情報取得

```
//-----------------------------------------------------------------------
//      スレーブ数取得、スレーブ情報取得
//-----------------------------------------------------------------------
  int SlaveCount;
  ACMST_SLAVE_INFO SlaveInfo[32];

  ret = ACat_Mst_Get_Slave_Count(&SlaveCount);
  if (ret) {
      printf("スレーブカウント取得失敗 RET=%08X", ret);
  } else {
      printf("スレーブカウント取得完了 Slavecount=%d", SlaveCount);
  }

  for (int i=0; i<SlaveCount; i++) {
      ret = ACat_Mst_Get_Slave_Info(i, &SlaveInfo[i]);
      if (ret) {
          printf("スレーブ情報取得失敗 RET=%08X", ret);
      } else {
          printf("スレーブ情報取得完了 スレーブ NO=%d", i);
          printf("  VendorID      : %08x", SlaveInfo[i].VendorID);
          printf("  ProductCode   : %08x", SlaveInfo[i].ProductCode);
          printf("  RevisionNo    : %08x", SlaveInfo[i].RevisionNo);
          printf("  SerialNo      : %08x", SlaveInfo[i].SerialNo);
          printf("  AliasAddress  : %d",   SlaveInfo[i].AliasAddress);
          printf("  PhysAddress   : %d",   SlaveInfo[i].PhysAddress);
          printf("  PdInputOffset : %d",   SlaveInfo[i].PdInputOffset);
          printf("  PdInputSize   : %d",   SlaveInfo[i].PdInputSize);
          printf("  PdOutputOffset: %d",   SlaveInfo[i].PdOutputOffset);
          printf("  PdOutputSize  : %d",   SlaveInfo[i].PdOutputSize);
      }
  }
```

４）入力 PDO 読出し(PDO オフセットでアクセスする場合)

```
//-----------------------------------------------------------------------
//      入力 PDO 読出し(PDO オフセット=30, サイズ=4 バイト)
//-----------------------------------------------------------------------
  int ret;
  DWORD rbuf;

  ret = ACat_Mst_RxPdo_Read(30, 4, (char *)&rbuf);
  if (ret) {
      printf("ACat_Mst_RxPdo_Read 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Mst_RxPdo_Read 完了: %08X", rbuf);
  }
```

５）出力 PDO 読出し(PDO オフセットでアクセスする場合)

```
//-----------------------------------------------------------------------
//      出力 PDO 読出し(PDO オフセット=10, サイズ=1 バイト)
//-----------------------------------------------------------------------
  int ret;
  BYTE rbuf;

  ret = ACat_Mst_TxPdo_Read(10, 1, (char *)&rbuf);
  if (ret) {
      printf("ACat_Mst_TxPdo_Read 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Mst_TxPdo_Read 完了: %08X", rbuf);
  }
```

６）出力 PDO 書込み(PDO オフセットでアクセスする場合)

```
//-----------------------------------------------------------------------
//       出力 PDO 書込み(PDO オフセット=20, サイズ=2 バイト)
//-----------------------------------------------------------------------
  int ret;
  WORD wbuf;

  wbuf = 0x1234;
  ret = ACat_Mst_TxPdo_Write(20, 2, (char *)&wbuf);
  if (ret) {
      printf("ACat_Mst_TxPdo_Write 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Mst_TxPdo_Write 完了");
  }
```

７）入力 PDO 読出し(スレーブ ID でアクセスする場合)

```
//-----------------------------------------------------------------------
//      入力 PDO 読出し(スレーブ ID=1, スレーブオフセット=2, サイズ=4 バイト)
//-----------------------------------------------------------------------
  int ret, addr;
  DWORD rbuf;

  ret = ACat_Slv_Get_Address(1, &addr);
  if (ret) {
      printf("ACat_Slv_Get_Address 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Get_Address 完了: %d", addr);
  }

  ret = ACat_Slv_Read_Input(addr, 2, 4, (char *)&rbuf);
  if (ret) {
      printf("ACat_Slv_Read_Input 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Read_Input 完了: %08X", rbuf);
  }
```

８）出力 PDO 読出し(スレーブ ID でアクセスする場合)

```
//-------------------------------------------------------------------------
//      出力 PDO 読出し(スレーブ ID=1, スレーブオフセット=4, サイズ=1 バイト)
//-------------------------------------------------------------------------
  int ret, addr;
  BYTE rbuf;

  ret = ACat_Slv_Get_Address(1, &addr);
  if (ret) {
      printf("ACat_Slv_Get_Address 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Get_Address 完了: %d", addr);
  }

  ret = ACat_Slv_Read_Output(addr, 4, 1, (char *)&rbuf);
  if (ret) {
      printf("ACat_Slv_Read_Output 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Read_Output 完了: %02X", rbuf);
  }
```

９）出力 PDO 書込み(スレーブ ID でアクセスする場合)

```
//-------------------------------------------------------------------------
//       出力 PDO 書込み(スレーブ ID=1, スレーブオフセット=6, サイズ=2 バイト)
//-------------------------------------------------------------------------
  int ret, addr;
  WORD wbuf;

  ret = ACat_Slv_Get_Address(1, &addr);
  if (ret) {
      printf("ACat_Slv_Get_Address 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Get_Address 完了: %d", addr);
  }

  wbuf = 0x1234;
  ret = ACat_Slv_Write_Output(addr, 6, 2, (char *)&wbuf);
  if (ret) {
      printf("ACat_Slv_Write_Output 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Write_Output 完了");
  }
```

１０）　SDO アップロード/ダウンロード

```
//-----------------------------------------------------------------------------
//      SDO ダウンロード(スレーブ ID=1、Index=0x6000、SubIndex=0x01、サイズ=2 バイト)
//-----------------------------------------------------------------------------
  int ret, addr;
  WORD rbuf;

  ret = ACat_Slv_Get_Address(1, &addr);
  if (ret) {
      printf("ACat_Slv_Get_Address 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Get_Address 完了: %d", addr);
  }

  ret = ACat_Slv_Sdo_Upload(addr, 0x7000, 0x01, (char *)&rbuf, 2);
  if (ret) {
      printf("ACat_Slv_Sdo_Upload 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Sdo_Upload 完了 : %04X", rbuf);
  }

//-----------------------------------------------------------------------------
//       SDO アップロード(スレーブ ID=1、Index=0x7000、SubIndex=0x01、サイズ=2 バイト)
//-----------------------------------------------------------------------------
  int ret, addr;
  WORD wbuf;

  ret = ACat_Slv_Get_Address(1, &addr);
  if (ret) {
      printf("ACat_Slv_Get_Address 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Get_Address 完了: %d", addr);
  }

  wbuf = 0x1234;
  ret = ACat_Slv_Sdo_Download(addr, 0x6000, 0x01, (char *)&wbuf, 2);
  if (ret) {
      printf("ACat_Slv_Sdo_Download 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Slv_Sdo_Download 完了");
  }
```

１１）　通知関数

```
//-----------------------------------------------------------------------------
//      通知関数の登録
//-----------------------------------------------------------------------------
ret = ACat_Mst_Set_Notification_Hook(&ACatNotificationHook, NULL);
if (ret) {
      printf("ACat_Mst_Set_Notification_Hook 失敗 RET=%08X", ret);
} else {
      printf("ACat_Mst_Set_Notification_Hook 完了");
}

//-----------------------------------------------------------------------------
//      通知コールバック関数(例：ESM の STATECHANGED 時に通知を受け取る)
//-----------------------------------------------------------------------------
BOOL ACatNotificationHook(void *pInst, DWORD Code, void *pJobData)
{
    if (Code == ACAT_NOTIFY_STATECHANGED) {
        ACAT_STATECHANGE* pStateInfo = (ACAT_STATECHANGE *)pJobData;
        printf("State change OldState=%04x NewState=%04x ",
                pStateInfo->oldState, pStateInfo->newState);
    }
    return TRUE;
}
```

１２）　サイクルイベント関数

```
//-----------------------------------------------------------------------------
//      サイクルイベント関数の登録
//-----------------------------------------------------------------------------
ret = ACat_Mst_Set_CycleEvent_Hook(&ACatCycleEventHook, NULL);
if (ret) {
      printf("ACat_Mst_Set_CycleEvent_Hook 失敗 RET=%08X", ret);
} else {
      printf("ACat_Mst_Set_CycleEvent_Hook 完了");
}

//-----------------------------------------------------------------------------
//      サイクルイベントコールバック関数(例：イベント発生時に入力 PDO を読込む)
//-----------------------------------------------------------------------------
BOOL ACatCycleEventHook(void *pInst, QWORD qwStartCount)
{
    int ret;
    BYTE rbuf;

    ret = ACat_Mst_RxPdo_Read(0, 1, (char *)&rbuf);
    return TRUE;
}
```

３－１－４　汎用 PDO/SDO 制御関数サンプルコード

汎用 PDO/SDO 制御関数のオープン部と実際の PDO/SDO アクセス部分のサンプルコードを次に示します。

１）汎用 PDO/SDO 制御ライブラリ初期化

```
//-------------------------------------------------------------------------
//      汎用 PDO/SDO 制御ライブラリ 初期化
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Gene_Create();                          // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }
```

２）PDO 入出力

```
//-------------------------------------------------------------------------
//       PDO 入力読み出し(スレーブ ID=1, オフセット=0, サイズ=4 バイト)
//-------------------------------------------------------------------------
  int ret;
  DWORD rbuf;

  ret = ACat_Gene_Read_Input(1, 0, 4, (char *)&rbuf);
  if (ret) {
      printf("ACat_Gene_Read_Input 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Gene_Read_Input 完了: %08X", rbuf);
  }

//-------------------------------------------------------------------------
//       PDO 出力書き込み(スレーブ ID=1, オフセット=0, サイズ=4 バイト)
//-------------------------------------------------------------------------
  int ret;
  DWORD wbuf;

  wbuf = 0x12345678;
  ret = ACat_Gene_Write_Output(1, 0, 4, (char *)&wbuf);
  if (ret) {
      printf("ACat_Gene_Write_Output 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Gene_Write_Output 完了");
  }
```

３）SDO アップロード/ダウンロード

```
//-------------------------------------------------------------------------
//       SDO アップロード(スレーブ ID=1、Index=0x6000、SubIndex=0x01、サイズ=2 バイト)
//-------------------------------------------------------------------------
  int ret;
  WORD rbuf;

  ret = ACat_Gene_Sdo_Upload(1, 0x6000, 0x01, (char *)&rbuf, 2);
  if (ret) {
      printf("ACat_Gene_Sdo_Upload 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Gene_Sdo_Upload 完了: %04X", rbuf);
  }

//-------------------------------------------------------------------------
//       SDO ダウンロード(スレーブ ID=1、Index=0x7000、SubIndex=0x01、サイズ=2 バイト)
//-------------------------------------------------------------------------
  int ret;
  WORD wbuf;

  wbuf = 0x1234;
  ret = ACat_Gene_Sdo_Download(1, 0x7000, 0x01, (char *)&wbuf, 2);
  if (ret) {
      printf("ACat_Gene_Sdo_Download 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Gene_Sdo_Download 完了");
  }
```

４）汎用 PDO/SDO 制御ライブラリ終了

```
//-------------------------------------------------------------------------
//      汎用 PDO/SDO 制御ライブラリ終了
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Gene_Destroy();                          // 終了
  if (ret) {
      printf("DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("DESTROY 完了");
  }
```

３－１－５　CiA402 ドライブプロファイル制御関数サンプルコード

CiA402 ドライブプロファイル制御関数のサンプルコードを次に示します。

１）初期化、サーボ ON

```
//-------------------------------------------------------------------------
//      モーションユニット 初期化、サーボ ON(スレーブ ID=1、チャンネル=1、タイムアウト=100000)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_Create();                          // 初期化
  if (ret) {
      printf("MOTION CREATE 失敗 RET=%08X", ret);
  } else {
      printf("MOTION CREATE 完了");
  }

  ret = ACat_Motion_ServoOn(1, 1, 100000);             // サーボ ON
  if (ret) {
      printf("サーボ ON 失敗 RET=%08X", ret);
  } else {
      printf("サーボ ON 完了");
  }
```

２）サーボ OFF、終了

```
//-------------------------------------------------------------------------
//      モーションユニット 終了、サーボ OFF(スレーブ ID=1、チャンネル=1、タイムアウト=50000)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_ServoOff(1, 1, 50000);             // サーボ OFF
  if (ret) {
      printf("サーボ OFF 失敗 RET=%08X", ret);
  } else {
      printf("サーボ OFF 完了");
  }

  ret = ACat_Motion_Destroy();                         // 終了
  if (ret) {
      printf("MOTION DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("MOTION DESTROY 完了");
  }
```

３）パラメータ設定

```
//-------------------------------------------------------------------------
//       パラメータ設定(スレーブ ID=1、チャンネル=1、プロファイル速度=50000、
// プロファイル加速度=5000、プロファイル減速度=7000、タイムアウト=1000000)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_MoveParameter(1, 1, 50000, 5000, 7000, 1000000);
  if (ret) {
      printf("ACat_Motion_MoveParameter 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_MoveParameter 完了");
  }
```

４）プロファイル位置制御モード　絶対位置制御

```
//-------------------------------------------------------------------------
//       絶対位置制御(スレーブ ID=1、チャンネル=1、目標位置=10000、
// 即時実行フラグ=0、タイムアウト=1000000)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_MoveAbsolute(1, 1, 10000, 0, 1000000);
  if (ret) {
      printf("ACat_Motion_MoveAbsolute 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_MoveAbsolute 完了");
  }
```

５）プロファイル位置制御モード　相対位置制御

```
//-------------------------------------------------------------------------
//       相対位置制御(スレーブ ID=1、チャンネル=1、目標位置=10000、
// 即時実行フラグ=0、タイムアウト=1000000)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_MoveRelative(1, 1, 10000, 0, 1000000);
  if (ret) {
      printf("ACat_Motion_MoveRelative 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_MoveRelative 完了");
  }
```

６）プロファイル速度制御モード　速度制御

```
//-------------------------------------------------------------------------
//       速度制御(スレーブ ID=1、チャンネル=1、目標速度=10000、タイムアウト=1000000)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_MoveVelocity(1, 1, 10000, 1000000);
  if (ret) {
      printf("ACat_Motion_MoveVelocity 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_MoveVelocity 完了");
  }
```

７）プロファイルトルク制御モード　トルク制御

```
//-------------------------------------------------------------------------
//       トルク制御(スレーブ ID=1、チャンネル=1、目標トルク=10000、タイムアウト=1000000)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_MoveTorque(1, 1, 10000, 1000000);
  if (ret) {
      printf("ACat_Motion_MoveTorque 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_MoveTorque 完了");
  }
```

８）目標値読出し

```
//-------------------------------------------------------------------------
//       目標値読出し(スレーブ ID=1、チャンネル=1)
//-------------------------------------------------------------------------
  int ret, TargetReached;

  ret = ACat_Motion_ReadTargetReached(1, 1, &TargetReached);
  if (ret) {
      printf("ACat_Motion_ReadTargetReached 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_ReadTargetReached 完了 TargetReached=%d", TargetReached);
  }
```

９）ホーミングモード　パラメータ設定

```
//--------------------------------------------------------------------------
//       パラメータ設定(スレーブ ID=1、チャンネル=1、スイッチサーチ速度=1000、
// ゼロサーチ速度=500、ホーミング加速度=100、タイムアウト=1000000)
//--------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_HomingParameter(1, 1, 1000, 500, 100, 1000000);
  if (ret) {
      printf("ACat_Motion_HomingParameter 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_HomingParameter 完了");
  }
```

１０）　ホーミングモード　原点復帰

```
//--------------------------------------------------------------------------
//       原点復帰(スレーブ ID=1、チャンネル=1、homemethod=35、タイムアウト=1000000)
//--------------------------------------------------------------------------
  ret =  ACat_Motion_Homing(1, 1, 35, 1000000);
  if (ret) {
      printf("ACat_Motion_Homing 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_Homing 完了");
  }
```

１１）　原点復帰完了読出し

```
//--------------------------------------------------------------------------
//       原点復帰完了読出し(スレーブ ID=1、チャンネル=1)
//--------------------------------------------------------------------------
  int ret, HomingEnd;

  ret = ACat_Motion_ReadHomingEnd(1, 1, &HomingEnd);
  if (ret) {
      printf("ACat_Motion_ReadHomingEnd 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_ReadHomingEnd 完了 HomingEnd=%d", HomingEnd);
  }
```

１２）　動作停止

```
//--------------------------------------------------------------------------
//       動作停止(スレーブ ID=1、チャンネル=1、タイムアウト=100000)
//--------------------------------------------------------------------------
  int ret;

  ret = ACat_Motion_Stop(1, 1, 100000);
  if (ret) {
      printf("ACat_Motion_Stop 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_Stop 完了");
  }
```

１３）　エラーコード読出し

```
//-------------------------------------------------------------------------
//       エラーコード読出し(スレーブ ID=1、チャンネル=1)
//-------------------------------------------------------------------------
  int ret, ErrorCode;

  ret = ACat_Motion_ReadAxisError(1, 1, &ErrorCode);
  if (ret) {
      printf("ACat_Motion_ReadAxisError 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Motion_ReadAxisError 完了 ErrorCode =%d", ErrorCode);
  }
```

３－１－６　デジタル入出力ユニット制御関数サンプルコード

デジタル入出力ユニット制御関数のサンプルコードを次に示します。

１）ユニットオープン

```
//-------------------------------------------------------------------------
//      デジタル入出力ユニット オープン(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Dio_Create();                          // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }

  ret = ACat_Dio_Open(1);                           // スレーブ ID=1 をオープン
  if (ret) {
      printf("OPEN 失敗 RET=%08X", ret);
  } else {
      printf("OPEN 完了");
  }
```

２）ユニット入出力データ読込み/書込み

```
//-------------------------------------------------------------------------
//       入力データ読込み(OUT32 ユニット接続時：スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  DWORD rbuf_out32;

  ret = ACat_Dio_Read(1, 4, (char *)&rbuf_out32);
  if (ret) {
      printf("ACat_Dio_Read 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Dio_Read 完了: %08X", rbuf_out32);
  }

//-------------------------------------------------------------------------
//       出力データ書込み(OUT32 ユニット接続時：スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  DWORD wbuf_out32;

  wbuf_out32 = 0x12345678;
  ret = ACat_Dio_Write(1, 4, (char *)&wbuf_out32);
  if (ret) {
      printf("ACat_Dio_Write 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Dio_Write 完了");
  }
```

３）ユニット設定読込み/書込み

```
//-------------------------------------------------------------------------
//       ユニット設定読込み(OUT32 ユニット接続時：スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  WORD filter, erroroutput;

  ret = ACat_Dio_GetParam(1, &filter, &erroroutput);
  if (ret) {
      printf("ACat_Dio_GetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Dio_GetParam 完了: filter=%04X erroroutput=%04x", filter, erroroutput);
  }

//-------------------------------------------------------------------------
//       ユニット設定書込み(OUT32 ユニット接続時：スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  WORD filter, erroroutput;

  // 通信異常時出力設定に 1 を設定
  filter = 0;
  erroroutput = 1;
  ret = ACat_Dio_SetParam(1, filter, erroroutput);
  if (ret) {
      printf("ACat_Dio_SetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Dio_SetParam 完了");
  }
```

４）ユニットソフトリセット

```
//-------------------------------------------------------------------------
//       ユニットソフトウェアリセット(OUT32 ユニット接続時：スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Dio_SoftReset(1);
  if (ret) {
      printf("ACat_Dio_SoftReset 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Dio_SoftReset 完了");
  }
```

５）ユニットクローズ

```
//-------------------------------------------------------------------------
//      デジタル入出力ユニット クローズ(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Dio_Close(1);                          // クローズ
  if (ret) {
      printf("CLOSE 失敗 RET=%08X", ret);
  } else {
      printf("CLOSE 完了");
  }

  ret = ACat_Dio_Destroy();                         // 終了
  if (ret) {
      printf("DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("DESTROY 完了");
  }
```

３－１－７　アナログ入出力ユニット制御関数サンプルコード

アナログ入出力ユニット制御関数のサンプルコードを次に示します。

１）ユニットオープン

```
//-------------------------------------------------------------------------
//      アナログ入出力ユニット オープン(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_Create();                          // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }

  ret = ACat_Aio_Open(1);                           // スレーブ ID=1 をオープン
  if (ret) {
      printf("OPEN 失敗 RET=%08X", ret);
  } else {
      printf("OPEN 完了");
  }
```

２）ユニット入出力データ読込み/書込み

```
//-------------------------------------------------------------------------
//       入力データ読込み(A/D ユニット：スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  TADDATA rbuf_ad;

  ret = ACat_Aio_Read(1, 1, 4, (char *)&rbuf_ad);
  if (ret) {
      printf("ACat_Aio_Read 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_Read 完了: 入力データ=%04X 断線検出=%d", rbuf_ad.data, rbuf_ad.conn);
  }

//-------------------------------------------------------------------------
//       出力データ書込み(D/A ユニット：スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  TDADATA wbuf_da;

  wbuf_da.data = 0x5678;
  ret = ACat_Aio_Write(1, 1, 2, (char *)&wbuf_da);
  if (ret) {
      printf("ACat_Aio_Write 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_Write 完了");
  }
```

３）A/D ユニット設定読込み/書込み

```
//-----------------------------------------------------------------------------
//       A/D ユニット設定読込み(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;
  WORD Mode, Filter, MinMaxSw, SamplingNum, CalibState;

  ret = ACat_Aio_AD_GetParam(1, 1, &Mode, &Filter, &MinMaxSw, &SamplingNum, &CalibState);
  if (ret) {
      printf("ACat_Aio_AD_GetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_AD_GetParam 完了");
      printf("Mode=%04X Filter=%04x MinMaxSw=%04X SamplingNum=%04x CalibState=%04X",
          Mode, Filter, MinMaxSw, SamplingNum, CalibState);
  }

//-----------------------------------------------------------------------------
//       A/D ユニットモード設定書込み(スレーブ ID=1)
//       モード設定(設定モードに変更)
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_AD_SetMode(1, 1, AIO_AD_MODE_CONFIG);
  if (ret) {
      printf("ACat_Aio_AD_SetMode 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_AD_SetMode 完了");
  }

//-----------------------------------------------------------------------------
//       A/D ユニットパラメータ設定書込み(スレーブ ID=1)
//       パラメータ設定(単純平均、最大最少除去しない、サンプリング回数 8 回)
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_AD_SetParam(1, 1, AIO_AD_FILTYPE_SIMPLE, AIO_AD_ENABLE_MINMAX,
                             AIO_AD_SAMPLING_NUM_8);
  if (ret) {
      printf("ACat_Aio_AD_SetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_AD_SetParam 完了");
  }
```

```
//-----------------------------------------------------------------------------
//       A/D ユニットキャリブレーション設定書込み(スレーブ ID=1)
//       キャリブレーション設定(最小値登録)
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_AD_SetCalib(1, 1, AIO_AD_MIN_CALIBRATION);
  if (ret) {
      printf("ACat_Aio_AD_SetCalib 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_AD_SetCalib 完了");
  }
```

４）D/A ユニット設定読込み/書込み

```
//-----------------------------------------------------------------------------
//       D/A ユニット設定読込み(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;
  WORD Mode, ErrorOutput, ErrorOutputData, CalibState;

  ret = ACat_Aio_DA_GetParam(1, 1, &Mode, &ErrorOutput, &ErrorOutputData, &CalibState);
  if (ret) {
      printf("ACat_Aio_DA_GetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_DA_GetParam 完了");
      printf("Mode=%04X ErrorOutput=%04x ErrorOutputData=%04X CalibState=%04X",
          Mode, ErrorOutput, ErrorOutputData, CalibState);
  }

//-----------------------------------------------------------------------------
//       D/A ユニットモード設定書込み(スレーブ ID=1)
//       モード設定(設定モードに変更)
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_DA_SetMode (1, 1, AIO_DA_MODE_CONFIG);
  if (ret) {
      printf("ACat_Aio_DA_SetMode 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_DA_SetMode 完了");
  }
```

```
//-------------------------------------------------------------------------
//       D/A ユニットパラメータ設定書込み(スレーブ ID=1)
//       パラメータ設定(異常時出力：ユーザー設定データ出力、異常時出力データ：0x1234)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_DA_SetParam(1, 1, AIO_DA_ERROUTPUT_USERNUM, 0x1234);
  if (ret) {
      printf("ACat_Aio_DA_SetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_DA_SetParam 完了");
  }

//-------------------------------------------------------------------------
//       D/A ユニットキャリブレーション設定書込み(スレーブ ID=1)
//       キャリブレーション設定(最小値登録)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_DA_SetCalib(1, 1, AIO_DA_MIN_CALIBRATION);
  if (ret) {
      printf("ACat_Aio_DA_SetCalib 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_DA_SetCalib 完了");
  }
```

５）ユニットソフトリセット

```
//-------------------------------------------------------------------------
//       ユニットソフトウェアリセット(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_SoftReset(1);
  if (ret) {
      printf("ACat_Aio_SoftReset 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Aio_SoftReset 完了");
  }
```

６）ユニットクローズ

```
//-------------------------------------------------------------------------
//      アナログ入出力ユニット クローズ(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Aio_Close(1);                          // クローズ
  if (ret) {
      printf("CLOSE 失敗 RET=%08X", ret);
  } else {
      printf("CLOSE 完了");
  }

  ret = ACat_Aio_Destroy();                         // 終了
  if (ret) {
      printf("DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("DESTROY 完了");
  }
```

３－１－８　モーションコントローラユニット制御関数サンプルコード

モーションコントローラユニット制御関数のサンプルコードを次に示します。

１）ユニットオープン

```
//-------------------------------------------------------------------------
//      モーションコントローラユニット オープン(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Axis_Create();                          // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }

  ret = ACat_Axis_Open(1);                           // スレーブ ID=1 をオープン
  if (ret) {
      printf("OPEN 失敗 RET=%08X", ret);
  } else {
      printf("OPEN 完了");
  }
```

２）ユニット入出力データ読込み/書込み

```
//-------------------------------------------------------------------------
//       入力データ読込み(スレーブ ID=1、オフセット=2、読込データサイズ=4)
//-------------------------------------------------------------------------
  int ret;
  DWORD rbuf_out32;

  ret = ACat_Axis_Read(1, 2, 4, (char *)&rbuf_out32);
  if (ret) {
      printf("ACat_Axis_Read 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Axis_Read 完了: %08X", rbuf_out32);
  }

//-------------------------------------------------------------------------
//       出力データ書込み(スレーブ ID=1、オフセット=5、書込データサイズ=4)
//-------------------------------------------------------------------------
  int ret;
  DWORD wbuf_out32;

  wbuf_out32 = 0x12345678;
  ret = ACat_Axis_Write(1, 5, 4, (char *)&wbuf_out32);
  if (ret) {
      printf("ACat_Dio_Write 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Dio_Write 完了");
  }
```

３）ユニット設定読込み/書込み

```
//-------------------------------------------------------------------------
//       ユニット設定読込み(スレーブ ID=1、Index=0x603F、SubIndex=0x00)
//-------------------------------------------------------------------------
  int ret;
  WORD errorcode;

  // エラーコード(603Fh)を読込む場合
  ret = ACat_Axis_GetParam(1, 0x603F, 0x00, 2, (char*)&errorcode);
  if (ret) {
      printf("ACat_Axis_GetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Axis_GetParam 完了: errorcode=%04X", errorcode);
  }

//-------------------------------------------------------------------------
//       ユニット設定書込み(スレーブ ID=1、Index=0x6081、SubIndex=0x00)
//-------------------------------------------------------------------------
  int ret;
  DWORD velocity;

  // プロファイル速度(6081h)を書込む
  velocity = 10000;
  ret = ACat_Axis_SetParam(1, 0x6081, 0x00, 4, (char*)&velocity);
  if (ret) {
      printf("ACat_Axis_SetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Axis_SetParam 完了");
  }
```

４）ユニットソフトリセット

```
//-------------------------------------------------------------------------
//       ユニットソフトウェアリセット(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Axis_SoftReset(1);
  if (ret) {
      printf("ACat_Axis_SoftReset 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Axis_SoftReset 完了");
  }
```

５）ユニットクローズ

```
//-------------------------------------------------------------------------
//      モーションコントロールユニット クローズ(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Axis_Close(1);                          // クローズ
  if (ret) {
      printf("CLOSE 失敗 RET=%08X", ret);
  } else {
      printf("CLOSE 完了");
  }

  ret = ACat_Axis_Destroy();                         // 終了
  if (ret) {
      printf("DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("DESTROY 完了");
  }
```

３－１－９　エンコーダユニット制御関数サンプルコード

エンコーダユニット制御関数のサンプルコードを次に示します。

１）ユニットオープン

```
//-------------------------------------------------------------------------
//      エンコーダユニット オープン(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Enc_Create();                          // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }

  ret = ACat_Enc_Open(1);                           // スレーブ ID1 をオープン
  if (ret) {
      printf("OPEN 失敗 RET=%08X", ret);
  } else {
      printf("OPEN 完了");
  }
```

２）ユニット入出力データ読込み/書込み

```
//-------------------------------------------------------------------------
//       入力データ読込み(スレーブ ID=1、チャンネル=1)
//-------------------------------------------------------------------------
  int ret;
  TENCSTATE EncState;

  ret = ACat_Enc_Read(1, 1, &EncState);
  if (ret) {
      printf("ACat_Enc_Read 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_Read 完了: CurCount=%08X Latch_A=%08X Latch_B=%08X Status=%04X",
            EncState.CurCount, EncState.Latch_A, EncState.Latch_B, EncState.Status);
  }
```

```
//-------------------------------------------------------------------------
//       出力データ書込み(スレーブ ID=1、チャンネル=1)
//-------------------------------------------------------------------------
  int ret;
  TENCCOMMAND EncCmd;

  EncCmd.SoftSw = 0x1;
  EncCmd.Preset = 0;
  ret = ACat_Enc_Write(1, 1, &EncCmd);
  if (ret) {
      printf("ACat_Enc_Write 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_Write 完了");
  }
```

３）ユニット設定読込み/書込み

```
//-------------------------------------------------------------------------
//       ユニット設定読込み(スレーブ ID=1、チャンネル=1)
//-------------------------------------------------------------------------
  int ret;
  TENCPARA EncPara;

  ret = ACat_Enc_GetParam(1, 1, &EncPara);
  if (ret) {
      printf("ACat_Enc_GetParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_GetParam 完了: SoftSw=%04X Preset=%08x Direction=%02X Evaluation=%02x
            MaxCount=%08X", EncPara.SoftSw, EncPara.Preset, EncPara.Direction,
            EncPara.Evaluation, EncPara.MaxCount);
  }

//-------------------------------------------------------------------------
//       回転方向設定書込み(スレーブ ID=1、チャンネル=1)
//-------------------------------------------------------------------------
  int ret;

  // 回転方向設定に ENC_RND_CW を設定
  ret = ACat_Enc_SetDirection(1, 1, ENC_RND_CW);
  if (ret) {
      printf("ACat_Enc_SetDirection 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_SetDirection 完了");
  }
```

```
//-------------------------------------------------------------------------
//       入力方式(逓倍)設定書込み(スレーブ ID=1、チャンネル=1)
//-------------------------------------------------------------------------
  int ret;

  //入力方式(逓倍)設定に IN_TYPE_1TMS を設定
  ret = ACat_Enc_SetEdgeEvaluation(1, 1, IN_TYPE_1TMS);
  if (ret) {
      printf("ACat_Enc_SetEdgeEvaluation 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_SetEdgeEvaluation 完了");
  }

//-------------------------------------------------------------------------
//       リングカウンタ最大値設定書込み(スレーブ ID=1、チャンネル=1)
//-------------------------------------------------------------------------
  int ret;

  //リングカウンタ最大値設定に ENC_COUNT_MAX(0xFFFF)を設定
  ret = ACat_Enc_SetMaxRingCount(1, 1, ENC_COUNT_MAX);
  if (ret) {
      printf("ACat_Enc_SetMaxRingCount 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_SetMaxRingCount 完了");
  }
```

４）ユニットパラメータセーブ/ロード

```
//-------------------------------------------------------------------------
//       ユニットパラメータセーブ(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Enc_SaveParam(1);
  if (ret) {
      printf("ACat_Enc_SaveParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_SaveParam 完了");
  }

//-------------------------------------------------------------------------
//       ユニットパラメータロード(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Enc_LoadParam(1);
  if (ret) {
      printf("ACat_Enc_LoadParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_LoadParam 完了");
  }
```

５）ユニットソフトリセット

```
//--------------------------------------------------------------------------
//       ユニットソフトウェアリセット(スレーブ ID=1)
//--------------------------------------------------------------------------
  int ret;

  ret = ACat_Enc_SoftReset(1);
  if (ret) {
      printf("ACat_Enc_SoftReset 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Enc_SoftReset 完了");
  }
```

６）ユニットクローズ

```
//--------------------------------------------------------------------------
//      エンコーダユニット クローズ(スレーブ ID=1)
//--------------------------------------------------------------------------
  int ret;

  ret = ACat_Enc_Close(1);                          // クローズ
  if (ret) {
      printf("CLOSE 失敗 RET=%08X", ret);
  } else {
      printf("CLOSE 完了");
  }

  ret = ACat_Enc_Destroy();                         // 終了
  if (ret) {
      printf("DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("DESTROY 完了");
  }
```

３－１－１０ SIO ゲートウェイユニット制御関数サンプルコード

SIO ゲートウェイユニット制御関数のサンプルコードを次に示します。

１）ユニットオープン

```
//-------------------------------------------------------------------------
//      SIO ゲートウェイユニット オープン(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Sio_Create();                          // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }

  ret = ACat_Sio_Open(1, 1, SIO_CONF_COMMTYPE_RS422);   // スレーブ ID=1 をオープン
  if (ret) {
      printf("OPEN 失敗 RET=%08X", ret);
  } else {
      printf("OPEN 完了");
  }
```

２）ユニット入出力データ読込み/書込み

```
//-------------------------------------------------------------------------
//       入力データ読込み(OUT32 ユニット接続時：スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  unsigned long len;
  char rcvbuf[16];

  ret = ACat_Sio_Read(1, 1, 16, &rcvbuf[0], &len, 10000);
  if (ret) {
      printf("ACat_Sio_Read 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_Read 完了：buf=%s", &rcvbuf[0]);
  }
```

```
//--------------------------------------------------------------------------
//       出力データ書込み(OUT32 ユニット接続時：スレーブ ID=1)
//--------------------------------------------------------------------------
  int ret,
  unsigned long len;
  char sndbuf[16] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

  ret = ACat_Sio_Write(1, 1, 16, &sndbuf[0], &len, 10000);
  if (ret) {
      printf("ACat_Sio_Write 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_Write 完了");
  }
```

３）ユニット設定読込み/書込み

```
//--------------------------------------------------------------------------
//       シリアル通信設定読込み(スレーブ ID=1)
//--------------------------------------------------------------------------
  int ret;
  TPORTINFO PortInfo;

  ret = ACat_Sio_GetCommInfo(1, 1, &PortInfo);
  if (ret) {
      printf("ACat_Sio_GetCommInfo 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_GetCommInfo 完了");
  }

//--------------------------------------------------------------------------
//       シリアル通信設定書込み(スレーブ ID=1)
//--------------------------------------------------------------------------
  int ret;
  TPORTINFO PortInfo;

  ret = ACat_Sio_SetCommInfo(1, 1, &PortInfo);
  if (ret) {
      printf("ACat_Sio_SetCommInfo 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_SetCommInfo 完了");
  }
```

```
//-------------------------------------------------------------------------
//       通信ステータス読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  TRSSTATUS RsState;

  ret = ACat_Sio_GetCommState(1, 1, &RsState);
  if (ret) {
      printf("ACat_Sio_GetCommState 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_GetCommState 完了");
  }

//-------------------------------------------------------------------------
//       通信エラー状態解除(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Sio_ClearError(1, 1);
  if (ret) {
      printf("ACat_Sio_ClearError 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_ClearError 完了");
  }

//-------------------------------------------------------------------------
//       送受信バッファクリア(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Sio_PurgeComm(1, 1, SIO_CMD_CLEAR_SENDBUF| SIO_CMD_CLEAR_RECVBUF);
  if (ret) {
      printf("ACat_Sio_PurgeComm 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_PurgeComm 完了");
  }
```

４）ユニットパラメータセーブ/ロード

```
//-------------------------------------------------------------------------
//       ユニットパラメータセーブ(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Sio_SaveParam(1);
  if (ret) {
      printf("ACat_Sio_SaveParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_SaveParam 完了");
  }
```

```
//-------------------------------------------------------------------------
//       ユニットパラメータロード(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Sio_LoadParam(1);
  if (ret) {
      printf("ACat_Sio_LoadParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_LoadParam 完了");
  }
```

５）ユニットソフトリセット

```
//-------------------------------------------------------------------------
//       ユニットソフトウェアリセット(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Sio_SoftReset(1);
  if (ret) {
      printf("ACat_Sio_SoftReset 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Sio_SoftReset 完了");
  }
```

６）ユニットクローズ

```
//-------------------------------------------------------------------------
//      SIO ゲートウェイユニット クローズ(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Sio_Close(1, 1);                       // クローズ
  if (ret) {
      printf("CLOSE 失敗 RET=%08X", ret);
  } else {
      printf("CLOSE 完了");
  }

  ret = ACat_Sio_Destroy();                         // 終了
  if (ret) {
      printf("DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("DESTROY 完了");
  }
```

３－１－１１ A-Link ゲートウェイユニット制御関数サンプルコード

A-Link ゲートウェイユニット制御関数のサンプルコードを次に示します。

１）ユニットオープン

```
//-------------------------------------------------------------------------
//      A-Link ゲートウェイユニット オープン(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Alink_Create();                        // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }

  ret = ACat_Alink_Open(1);                         // スレーブ ID=1 をオープン
  if (ret) {
      printf("OPEN 失敗 RET=%08X", ret);
  } else {
      printf("OPEN 完了");
  }
```

２）A-Link 通信開始/停止

```
//-------------------------------------------------------------------------
//       A-Link 通信開始(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Alink_Start(1, 1, 63, ALINK_CONSTANT_SCAN_START);
  if (ret) {
      printf("ACat_Alink_Start 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_Start 完了");
  }

//-------------------------------------------------------------------------
//       A-Link 通信停止(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Alink_Stop(1, 1);
  if (ret) {
      printf("ACat_Alink_Stop 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_Stop 完了");
  }
```

３）A-Link DI/DO データ読込み/書込み

```
//-------------------------------------------------------------------------
//       A-Link DI データ読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  WORD rbuf;

  ret = ACat_Alink_Read(1, 1, 1, 1, &rbuf);
  if (ret) {
      printf("ACat_Alink_Read 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_Read 完了: %04X", rbuf);
  }

//-------------------------------------------------------------------------
//       A-Link DO データ書込み(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  WORD wbuf;

  wbuf = 0x1234;
  ret = ACat_Alink_Write(1, 1, 1, 1, &wbuf);
  if (ret) {
      printf("ACat_Alink_Write 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_Write 完了");
  }
```

４）A-Link 通信異常/ステータス読込み

```
//-------------------------------------------------------------------------
//       A-Link 通信異常(CHK2 発生状況)読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  TALINK_CHK2_OCCUR chk2;

  ret = ACat_Alink_ReadChk2Occur(1, 1, &chk2);
  if (ret) {
      printf("ACat_Alink_ReadChk2Occur 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_ReadChk2Occur 完了 :  %04X %04X %04X %04X",
                chk2.sa[0], chk2.sa[1], chk2.sa[2], chk2.sa[3]);
  }
```

```
//-----------------------------------------------------------------------------
//       ステータスデータ読込み(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;
  WORD status;

  ret = ACat_Alink_GetStatus(1, 1, 1, &status);
  if (ret) {
      printf("ACat_Alink_GetStatus 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_GetStatus 完了: status=%04x", status);
  }

//-----------------------------------------------------------------------------
//       A-Link システムステータス読込み(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;
  TALINK_SYS_STATUS syssts;

  ret = ACat_Alink_GetSysSts(1, 1, &syssts);
  if (ret) {
      printf("ACat_Alink_GetSysSts 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_GetSysSts 完了: syssts=%04x", syssts.Word);
  }

//-----------------------------------------------------------------------------
//       A-Link 通信異常(CHK1)発生回数読込み(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;
  WORD chk1cnt;

  ret = ACat_Alink_GetChk1Count(1, 1, &chk1cnt);
  if (ret) {
      printf("ACat_Alink_GetChk1Count 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_GetChk1Count 完了: chk1cnt=%04x", chk1cnt);
  }
```

```
//-----------------------------------------------------------------------------
//       A-Link 通信異常(CHK2)発生回数読込み(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;
  WORD chk2cnt;

  ret = ACat_Alink_GetChk2Count(1, 1, &chk2cnt);
  if (ret) {
      printf("ACat_Alink_GetChk2Count 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_GetChk2Count 完了: chk2cnt=%04x", chk2cnt);
  }

//-----------------------------------------------------------------------------
//       A-Link 通信異常(CHK1)発生回数クリア(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Alink_ClrChk1Count(1, 1);
  if (ret) {
      printf("ACat_Alink_ClrChk1Count 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_ClrChk1Count 完了");
  }

//-----------------------------------------------------------------------------
//       A-Link 通信異常(CHK2)発生回数クリア(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Alink_ClrChk2Count(1, 1);
  if (ret) {
      printf("ACat_Alink_ClrChk2Count 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_ClrChk2Count 完了");
  }
```

５）A-Link 通信設定読込み/書込み

```
//-------------------------------------------------------------------------
//       A-Link 通信設定読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;
  WORD Duplex, Baudrate, HubNum;

  ret = ACat_Alink_GetCommParam(1, 1, &Duplex, &Baudrate, &HubNum);
  if (ret) {
      printf("ACat_Alink_GetCommParam 敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_GetCommParam 完了 : Duplex=%d Baudrate=%d HubNum=%d",
                Duplex, Baudrate, HubNum);
  }

//-------------------------------------------------------------------------
//       A-Link 通信設定書込み(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Alink_SetCommParam(1, 1, ALINK_DUPLEX_HALF, ALINK_BAUDRATE_6M, 7);
  if (ret) {
      printf("ACat_Alink_SetCommParam 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_SetCommParam 完了");
  }
```

６）ユニットソフトリセット

```
//-------------------------------------------------------------------------
//       ユニットソフトウェアリセット(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_Alink_SoftReset(1);
  if (ret) {
      printf("ACat_Alink_SoftReset 失敗 RET=%08X", ret);
  } else {
      printf("ACat_Alink_SoftReset 完了");
  }
```

７）ユニットクローズ

```
//--------------------------------------------------------------------------
//      A-Link ゲートウェイユニット クローズ(スレーブ ID=1)
//--------------------------------------------------------------------------
  int ret;

  ret = ACat_Alink_Close(1);                          // クローズ
  if (ret) {
      printf("CLOSE 失敗 RET=%08X", ret);
  } else {
      printf("CLOSE 完了");
  }

  ret = ACat_Alink_Destroy();                         // 終了
  if (ret) {
      printf("DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("DESTROY 完了");
  }
```

３－１－１２ Modbus ゲートウェイユニット制御関数サンプルコード

Modbus ゲートウェイユニット制御関数のサンプルコードを次に示します。

１）ユニットオープン

```
//-------------------------------------------------------------------------
//      MODBUS ゲートウェイユニット オープン(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;

ret = ACat_Mbg_Create();                            // 初期化
if (ret) {
    printf("CREATE 失敗 RET=%08X¥n", ret);
} else {
    printf("CREATE 完了¥n");
}

ret = ACat_Mbg_Open(1);                             // スレーブ ID=1 をオープン
if (ret) {
    printf("OPEN 失敗 RET=%08X¥n", ret);
} else {
    printf("OPEN 完了¥n");
}
```

２）ユニット入出力データ BYTE 読込み/書込み

```
//-------------------------------------------------------------------------
//       Byte データ読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
WORD  Offset   = 0;
BYTE  readbuf[4];

ret = ACat_Mbg_ReadByte(1, Offset, 4, readbuf);
if (ret) {
    printf("ACat_Mbg_ReadByte 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_ReadByte 完了¥n");
    printf("readbuf[0]=0x%02x readbuf[1]=0x%02x readbuf[2]=0x%02x readbuf[3]=0x%02x¥n",
        readbuf[0], readbuf[1], readbuf[2], readbuf[3]);
}
```

```
//-------------------------------------------------------------------------
//       Byte データ書込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
WORD  Offset   = 0;
BYTE  writebuf[4] = {0x00, 0x10, 0x20, 0x30};

ret = ACat_Mbg_WriteByte(1, Offset, 4, writebuf);
if (ret) {
    printf("ACat_Mbg_WriteByte 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_WriteByte 完了¥n");
}
```

３）ユニット入出力データ WORD 読込み/書込み

```
//-------------------------------------------------------------------------
//       Word データ読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
WORD  Offset   = 0;
WORD  readbuf[4];

ret = ACat_Mbg_ReadWord(1, Offset, 4, readbuf);
if (ret) {
    printf("ACat_Mbg_ReadWord 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_ReadWord 完了¥n");
    printf("readbuf[0]=0x%04x readbuf[1]=0x%04x readbuf[2]=0x%04x readbuf[3]=0x%04x¥n",
        readbuf[0], readbuf[1], readbuf[2], readbuf[3]);
}

//-------------------------------------------------------------------------
//       Word データ書込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
WORD  Offset   = 0;
WORD  writebuf[4] = {0x0000, 0x1000, 0x2000, 0x3000};

ret = ACat_Mbg_WriteWord(1, Offset, 4, writebuf);
if (ret) {
    printf("ACat_Mbg_WriteWord 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_WriteWord 完了¥n");
}
```

４）Modbus 通信開始/停止

```
//-----------------------------------------------------------------------------
//       通信開始(スレーブ ID=1, ターゲット機器 ID=1)
//-----------------------------------------------------------------------------
int ret;
DWORD TargetID        = 1;

ret = ACat_Mbg_Start(1, TargetID);
if (ret) {
    printf("ACat_Mbg_Start 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_Start 完了¥n");
}

//-----------------------------------------------------------------------------
//       通信停止(スレーブ ID=1, ターゲット機器 ID=1)
//-----------------------------------------------------------------------------
int ret;
DWORD TargetID        = 1;

ret = ACat_Mbg_Stop(1, TargetID);
if (ret) {
    printf("ACat_Mbg_Stop 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_Stop 完了¥n");
}
```

５）COM ポート設定**読込み/書込み**

```
//-------------------------------------------------------------------------
//       COM ポート設定読込み(スレーブ ID=1, COM ポート=1)
//-------------------------------------------------------------------------
int ret;
DWORD ComPort    = 1;
WORD  BaudRate, DataLength, StopBit, Parity, FormatType, Interface;

ret = ACat_Mbg_GetCommInfo(1, ComPort, &BaudRate, &DataLength, &StopBit, &Parity, &FormatType,
&Interface);
if (ret) {
    printf("ACat_Mbg_GetCommInfo 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetCommInfo 完了¥n");
    printf("BaudRate=%d DataLength=%d StopBit=%d Parity=%d FormatType=%d Interface=%d¥n",
        BaudRate, DataLength, StopBit, Parity, FormatType, Interface);
}

//-------------------------------------------------------------------------
//       COM ポート設定書込み(スレーブ ID=1, COM ポート=1)
//-------------------------------------------------------------------------
int ret;
DWORD ComPort    = 1;
WORD  BaudRate   = MBG_COMM_BAUDRATE_115200;       // ボーレート        115200bps
WORD  DataLength = MBG_COMM_DATALEN_8BIT;          // データ長          8bit
WORD  StopBit    = MBG_COMM_STOPBIT_1BIT;          // ストップビット    1bit
WORD  Parity     = MBG_COMM_PARITY_NON;            // パリティ          なし
WORD  FormatType = MBG_COMM_PARAM_MODE_RTU;        // 伝送モード        RTU
WORD  Interface  = MBG_COMM_PARAM_IF_RS232C;       // インターフェイス  RS232C

ret = ACat_Mbg_SetCommInfo(1, ComPort, BaudRate, DataLength, StopBit, Parity, FormatType,
Interface);
if (ret) {
    printf("ACat_Mbg_SetCommInfo 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_SetCommInfo 完了¥n");
}
```

６）ターゲット機器設定**読込み/書込み**

```
//-----------------------------------------------------------------------------
//       ターゲット機器設定読込み(スレーブ ID=1, ターゲット機器 ID=1)
//-----------------------------------------------------------------------------
int ret;
DWORD TargetID        = 1;
WORD  TargetAddress, ComPort;

ret = ACat_Mbg_GetTargetInfo(1, TargetID, &TargetAddress, &ComPort);
if (ret) {
    printf("ACat_Mbg_GetTargetInfo 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetTargetInfo 完了¥n");
    printf("TargetAddress=%d ComPort=%d¥n",
        TargetAddress, ComPort);
}

//-----------------------------------------------------------------------------
//       ターゲット機器設定書込み(スレーブ ID=1, ターゲット機器 ID=1)
//-----------------------------------------------------------------------------
int ret;
DWORD TargetID        = 1;
WORD  TargetAddress   = 1;
WORD  ComPort         = MBG_TARGET_PARAM_COM_1CH;

ret = ACat_Mbg_SetTargetInfo(1, TargetID, TargetAddress, ComPort);
if (ret) {
    printf("ACat_Mbg_SetTargetInfo 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_SetTargetInfo 完了¥n");
}
```

７）モニタデータコマンド設定**読込み/書込み**

```
//-------------------------------------------------------------------------
//       モニタデータコマンド設定読込み(スレーブ ID=1, コマンド ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID       = 1;
WORD  TargetID, FunctionCode, StartAddress, Size, Offset, Cycle;

ret = ACat_Mbg_GetMonCmd(1, CommandID, &TargetID, &FunctionCode, &StartAddress, &Size, &Offset,
&Cycle);
if (ret) {
    printf("ACat_Mbg_GetMonCmd 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetMonCmd 完了¥n");
    printf("TargetID=%d FunctionCode=%d StartAddress=%d Size=%d Offset=%d Cycle=%d¥n",
        TargetID, FunctionCode, StartAddress, Size, Offset, Cycle);
}

//-------------------------------------------------------------------------
//       モニタデータコマンド設定書込み(スレーブ ID=1, コマンド ID=1)
//                  【機能】
//                    入力レジスタの開始アドレス=0 から 4Word のデータを読み込み、
//                    取得したデータを PDO オフセット=0～8 にセットする
//-------------------------------------------------------------------------
int ret;
DWORD CommandID       = 1;
WORD  TargetID        = 1;
WORD  FunctionCode    = MBG_CMD_FUNCCODE_READ_INPUTREG;
WORD  StartAddress    = 0;
WORD  Size            = 4;
WORD  Offset          = 0;
WORD  Cycle           = 1000;

ret = ACat_Mbg_SetMonCmd(1, CommandID, TargetID, FunctionCode, StartAddress, Size, Offset, Cycle);
if (ret) {
    printf("ACat_Mbg_SetMonCmd 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_SetMonCmd 完了¥n");
}
```

８）即時要求データコマンド設定**読込み/書込み**

```
//-----------------------------------------------------------------------
//       即時要求データコマンド設定読込み(スレーブ ID=1, コマンド ID=1)
//-----------------------------------------------------------------------
int ret;
DWORD CommandID       = 1;
WORD  TargetID, FunctionCode, StartAddress, Size, Offset;

ret = ACat_Mbg_GetSpotReqCmd(1, CommandID, &TargetID, &FunctionCode, &StartAddress, &Size,
&Offset);
if (ret) {
    printf("ACat_Mbg_GetSpotReqCmd 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetSpotReqCmd 完了¥n");
    printf("TargetID=%d FunctionCode=%d StartAddress=%d Size=%d Offset=%d¥n",
        TargetID, FunctionCode, StartAddress, Size, Offset);
}

//-----------------------------------------------------------------------
//       即時要求データコマンド設定書込み(スレーブ ID=1, コマンド ID=1)
//                  【機能】
//                    PDO オフセット=0～8 にセットされている 4Word データを
//                    保持レジスタの開始アドレス=0 へ書き込みする
//-----------------------------------------------------------------------
int ret;
DWORD CommandID       = 1;
WORD  TargetID        = 1;
WORD  FunctionCode    = MBG_CMD_FUNCCODE_FORCE_MULTIREG;
WORD  StartAddress    = 0;
WORD  Size            = 4;
WORD  Offset          = 0;

ret = ACat_Mbg_SetSpotReqCmd(1, CommandID, TargetID, FunctionCode, StartAddress, Size, Offset);
if (ret) {
    printf("ACat_Mbg_SetSpotReqCmd 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_SetSpotReqCmd 完了¥n");
}
```

９）手動要求データコマンド設定**読込み/書込み**

```
//-------------------------------------------------------------------------
//       手動要求データコマンド設定読込み(スレーブ ID=1, コマンド ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID       = 1;
WORD  TargetID, FunctionCode, StartAddress, Size, Offset, WriteSw;

ret = ACat_Mbg_GetManReqCmd(1, CommandID, &TargetID, &FunctionCode, &StartAddress, &Size, &Offset,
 &WriteSw);
 if (ret) {
     printf("ACat_Mbg_GetManReqCmd 失敗 RET=%08X¥n", ret);
 } else {
     printf("ACat_Mbg_GetManReqCmd 完了¥n");
     printf("TargetID=%d FunctionCode=%d StartAddress=%d Size=%d Offset=%d WriteSw =%d ¥n",
         TargetID, FunctionCode, StartAddress, Size, Offset, WriteSw);
 }

 //-------------------------------------------------------------------------
 //       手動要求データコマンド設定書込み(スレーブ ID=1, コマンド ID=1)
 //                  【機能】
 //                    PDO オフセット=0～8 にセットされている 4Word データを
 //                    保持レジスタの開始アドレス=0 へ書き込みする
 //-------------------------------------------------------------------------
 int ret;
 DWORD CommandID       = 1;
 WORD  TargetID        = 1;
 WORD  FunctionCode    = MBG_CMD_FUNCCODE_FORCE_MULTIREG;
 WORD  StartAddress    = 0;
 WORD  Size            = 4;
 WORD  Offset          = 0;
 WORD  WriteSw         = 1;

 ret = ACat_Mbg_SetManReqCmd(1, CommandID, TargetID, FunctionCode, StartAddress, Size, Offset,
 WriteSw);
 if (ret) {
     printf("ACat_Mbg_SetManReqCmd 失敗 RET=%08X¥n", ret);
 } else {
     printf("ACat_Mbg_SetManReqCmd 完了¥n");
 }
```

１０）　エラー状況/レスポンス状況読込み

```
//-------------------------------------------------------------------------
//       エラー状況読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
TMODBUS_ERROR_STATUS status;
memset(&status, '¥0', sizeof(TMODBUS_ERROR_STATUS));

ret = ACat_Mbg_GetErrorStatus(1, &status);
if (ret) {
    printf("ACat_Mbg_GetErrorStatus 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetErrorStatus 完了¥n");
    printf("MonCmdErr=0x%08x SpotCmdErr=0x%08x ManCmdErr=0x%08x¥n",
        status.MonCmdErr1, status.SpotCmdErr1, status.ManCmdErr);
}

//-------------------------------------------------------------------------
//       レスポンス状況読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
TMODBUS_RESPONSE_STATUS response;
memset(&response, '¥0', sizeof(TMODBUS_RESPONSE_STATUS));

ret = ACat_Mbg_GetResponseStatus(1, &response);
if (ret) {
    printf("ACat_Mbg_GetResponseStatus 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetResponseStatus 完了¥n");
    printf("SpotCmdRes=0x%08x ManCmdRes=0x%08x¥n",
        response.SpotCmdRes1, response.ManCmdRes);
}
```

１１）　エラーステータス読込み

```
//-------------------------------------------------------------------------
//       モニタデータコマンド エラーステータス読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;
WORD Status, CommError, ModbusError;

ret = ACat_Mbg_GetMonCmdErr(1, CommandID, &Status, &CommError, &ModbusError);
if (ret) {
    printf("ACat_Mbg_GetMonCmdErr 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetMonCmdErr 完了¥n");
    printf("Status=0x%04x CommError=0x%04x ModbusError=0x%04x¥n",
        Status, CommError, ModbusError);
}

//-------------------------------------------------------------------------
//       即時要求データコマンド エラーステータス読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;
WORD Status, CommError, ModbusError;

ret = ACat_Mbg_GetSpotReqCmdErr(1, CommandID, &Status, &CommError, &ModbusError);
if (ret) {
    printf("ACat_Mbg_GetSpotReqCmdErr 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetSpotReqCmdErr 完了¥n");
    printf("Status=0x%04x CommError=0x%04x ModbusError=0x%04x¥n",
        Status, CommError, ModbusError);
}

//-------------------------------------------------------------------------
//       手動要求データコマンド エラーステータス読込み(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;
WORD Status, CommError, ModbusError;

ret = ACat_Mbg_GetManReqCmdErr(1, CommandID, &Status, &CommError, &ModbusError);
if (ret) {
    printf("ACat_Mbg_GetManReqCmdErr 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetManReqCmdErr 完了¥n");
    printf("Status=0x%04x CommError=0x%04x ModbusError=0x%04x¥n",
        Status, CommError, ModbusError);
}
```

１２）　レスポンス読込み

```
//-----------------------------------------------------------------------
//       即時要求データコマンド レスポンス読込み(スレーブ ID=1)
//-----------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;
WORD Status, CommError, ModbusError;

ret = ACat_Mbg_GetSpotReqCmdRes(1, CommandID, &Status, &CommError, &ModbusError);
if (ret) {
    printf("ACat_Mbg_GetSpotReqCmdRes 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetSpotReqCmdRes 完了¥n");
    printf("Status=0x%04x CommError=0x%04x ModbusError=0x%04x¥n",
        Status, CommError, ModbusError);
}

//-----------------------------------------------------------------------
//       手動要求データコマンド レスポンス読込み(スレーブ ID=1)
//-----------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;
WORD Status, CommError, ModbusError;

ret = ACat_Mbg_GetManReqCmdRes(1, CommandID, &Status, &CommError, &ModbusError);
if (ret) {
    printf("ACat_Mbg_GetManReqCmdRes 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_GetManReqCmdRes 完了¥n");
    printf("Status=0x%04x CommError=0x%04x ModbusError=0x%04x¥n",
        Status, CommError, ModbusError);
}
```

１３）　エラークリア

```
//-------------------------------------------------------------------------
//       モニタデータコマンド エラークリア(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;

ret = ACat_Mbg_ClrMonCmdErr(1, CommandID);
if (ret) {
    printf("ACat_Mbg_ClrMonCmdErr 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_ClrMonCmdErr 完了¥n");
}

//-------------------------------------------------------------------------
//       即時要求データコマンド エラークリア(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;

ret = ACat_Mbg_ClrSpotReqCmdErr(1, CommandID);
if (ret) {
    printf("ACat_Mbg_ClrSpotReqCmdErr 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_ClrSpotReqCmdErr 完了¥n");
}

//-------------------------------------------------------------------------
//       手動要求データコマンド エラークリア(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;

ret = ACat_Mbg_ClrManReqCmdErr(1, CommandID);
if (ret) {
    printf("ACat_Mbg_ClrManReqCmdErr 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_ClrManReqCmdErr 完了¥n");
}
```

１４）　レスポンスクリア

```
//-------------------------------------------------------------------------
//       即時要求データコマンド レスポンス(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;

ret = ACat_Mbg_ClrSpotReqCmdRes(1, CommandID);
if (ret) {
    printf("ACat_Mbg_ClrSpotReqCmdRes 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_ClrSpotReqCmdRes 完了¥n");
}

//-------------------------------------------------------------------------
//       手動要求データコマンド レスポンス(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;
DWORD CommandID        = 1;

ret = ACat_Mbg_ClrManReqCmdRes (1, CommandID);
if (ret) {
    printf("ACat_Mbg_ClrManReqCmdRes 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_ClrManReqCmdRes 完了¥n");
}
```

１５）　ユニットパラメータセーブ/ロード

```
//-------------------------------------------------------------------------
//       ユニットパラメータセーブ(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;

ret = ACat_Mbg_SaveParam(1);
if (ret) {
    printf("ACat_Mbg_SaveParam 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_SaveParam 完了¥n");
}



//-------------------------------------------------------------------------
//       ユニットパラメータロード(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;

ret = ACat_Mbg_LoadParam(1);
if (ret) {
    printf("ACat_Mbg_LoadParam 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_LoadParam 完了¥n");
}
```

１６）　ユニットソフトリセット

```
//-------------------------------------------------------------------------
//       ユニットソフトウェアリセット(スレーブ ID=1)
//-------------------------------------------------------------------------
int ret;

ret = ACat_Mbg_SoftReset(1);
if (ret) {
    printf("ACat_Mbg_SoftReset 失敗 RET=%08X¥n", ret);
} else {
    printf("ACat_Mbg_SoftReset 完了¥n");
}
```

１７）　ユニットクローズ

```
//-----------------------------------------------------------------------
//      Modbus ゲートウェイユニット クローズ(スレーブ ID=1)
//-----------------------------------------------------------------------
int ret;

ret = ACat_Mbg_Close(1);                          // クローズ
if (ret) {
    printf("CLOSE 失敗 RET=%08X¥n", ret);
} else {
    printf("CLOSE 完了¥n");
}

ret = ACat_Mbg_Destroy();                         // 終了
if (ret) {
    printf("DESTROY 失敗 RET=%08X¥n", ret);
} else {
    printf("DESTROY 完了¥n");
}
```

３－１－１３ EtherCAT ネットワーク情報ファイル動的生成サンプルコード

ネットワーク情報ファイルの動的生成のサンプルコードを次に示します。
ネットワーク情報ファイルの動的生成は、EtherCAT 初期化の後、EtherCAT 通信開始前に実行してください。
EtherCAT 通信開始後の動的生成はできませんので注意してください。

１）EtherCAT 初期化

```
//-----------------------------------------------------------------------------
//       EtherCAT 初期化
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Create();          // EtherCAT 初期化
  if (ret) {
      printf("EtherCAT 初期化失敗 RET=%08X", ret);
  } else {
      printf("EtherCAT 初期化完了");
  }
```

２）ネットワーク情報ファイル動的生成

```
//-----------------------------------------------------------------------------
//      EtherCAT ネットワーク情報ファイル作成
//-----------------------------------------------------------------------------
int ret;

ret = ACat_Make_ENI(100000);        // タイムアウト=100000 ミリ秒
if (ret) {
    printf("ENI ファイル作成失敗 RET=%08X¥n", ret);
} else {
    printf("ENI ファイル作成完了¥n");
}
```

３）EtherCAT 通信開始

```
//-----------------------------------------------------------------------------
//      EtherCAT 通信開始
//-----------------------------------------------------------------------------
int ret;

ret = ACat_Start(100000);       // EtherCAT 通信開始(タイムアウト=100000 ミリ秒)
if (ret) {
    printf("EtherCAT 通信開始失敗 RET=%08X¥n", ret);
} else {
    printf("EtherCAT 通信開始完了¥n");
}
```

３－１－１４ FoE ダウンロード　サンプルコード

FoE ダウンロードのサンプルコードを次に示します。
FoE ダウンロードは、EtherCAT 初期化の後、EtherCAT ステートマシーンを Bootstrap(スレーブのファームウェアを書き換えるための特殊な状態)に変更後、実行してください。

１）EtherCAT 初期化

```
//-----------------------------------------------------------------------------
//       EtherCAT 初期化
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Create();          // EtherCAT 初期化
  if (ret) {
      printf("EtherCAT 初期化失敗 RET=%08X", ret);
  } else {
      printf("EtherCAT 初期化完了");
  }
```

２）EtherCAT ステートマシンを Bootstrap(BOOT)に遷移

```
//-----------------------------------------------------------------------------
//      EtherCAT 通信開始
//-----------------------------------------------------------------------------
  Int ret;

  ret = ACat_Start(10000);      // EtherCAT 通信開始
  if (ret) {
      printf("EtherCAT 通信開始失敗 RET=%08X¥n", ret);
  } else {
      printf("EtherCAT 通信開始完了¥n");
  }

//-----------------------------------------------------------------------------
//      ESM を INIT に遷移
//-----------------------------------------------------------------------------
  ret = ACat_Mst_Set_State(ACAT_STATE_INIT, 5000);      // State Machine=INIT
  if (ret) {
      printf("EtherCAT State Machine 設定失敗(INIT) RET=%08X¥n", ret);
    } else {
      printf("EtherCAT State Machine 設定完了(INIT)¥n");
    }

//-----------------------------------------------------------------------------
//      対象スレーブ(1001)の ESM を BOOT に遷移
//-----------------------------------------------------------------------------
  ret = ACat_Slv_Set_State(1001, ACAT_STATE_BOOTSTRAP, 5000);   // State Machine=BOOT
  if (ret) {
      printf("EtherCAT State Machine 設定失敗(BOOT) RET=%08X¥n", ret);
  } else {
      printf("EtherCAT State Machine 設定完了(BOOT)¥n");
  }
```

３）FoE ダウンロード

```
//-----------------------------------------------------------------------
//      FoE ダウンロード
//      （アドレス=1001、ファイル名=[ECATFW__B1_FoE.efw]、データサイズ=0x90000、
//       パスワード=0x00000000、タイムアウト=300000）
//-----------------------------------------------------------------------
  int ret;

  ret = ACat_Slv_Foe_Download(1001, "ECATFW__B1_FoE.efw", 0x90000, 0x00000000, 300000);
  if (ret) {
      printf("FoE ダウンロード失敗 RET=%08X¥n", ret);
  } else {
      printf("FoE ダウンロード成功¥n");
  }
```

３－１－１５ FoE アップロード　サンプルコード

FoE アップロードのサンプルコードを次に示します。
FoE アップロードは、EtherCAT 初期化の後、EtherCAT ステートマシーンを Bootstrap(スレーブのファームウェアを書き換えるための特殊な状態)に変更後、実行してください。

１）EtherCAT 初期化

```
//-----------------------------------------------------------------------------
//       EtherCAT 初期化
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_Create();          // EtherCAT 初期化
  if (ret) {
      printf("EtherCAT 初期化失敗 RET=%08X", ret);
  } else {
      printf("EtherCAT 初期化完了");
  }
```

２）EtherCAT ステートマシンを Bootstrap(BOOT)に遷移

```
//-----------------------------------------------------------------------------
//      EtherCAT 通信開始
//-----------------------------------------------------------------------------
  int ret;

ret = ACat_Start(10000);        // EtherCAT 通信開始
if (ret) {
    printf("EtherCAT 通信開始失敗 RET=%08X¥n", ret);
} else {
    printf("EtherCAT 通信開始完了¥n");
}

//-----------------------------------------------------------------------------
//      ESM を INIT に遷移
//-----------------------------------------------------------------------------
ret = ACat_Mst_Set_State(ACAT_STATE_INIT, 5000);        // State Machine=INIT
if (ret) {
    printf("EtherCAT State Machine 設定失敗(INIT) RET=%08X¥n", ret);
} else {
    printf("EtherCAT State Machine 設定完了(INIT)¥n");
}

//-----------------------------------------------------------------------------
//      対象スレーブ(1001)の ESM を BOOT に遷移
//-----------------------------------------------------------------------------
ret = ACat_Slv_Set_State(1001, ACAT_STATE_BOOTSTRAP, 5000); // State Machine=BOOT
if (ret) {
    printf("EtherCAT State Machine 設定失敗(BOOT) RET=%08X¥n", ret);
} else {
    printf("EtherCAT State Machine 設定完了(BOOT)¥n");
}
```

３）FoE アップロード

```
//-----------------------------------------------------------------------
//      FoE アップロード
//      （アドレス=1001、ファイル名=[ECATFW__B1_UP.efw]、データサイズ=0x90000、
//       パスワード=0x00000000、タイムアウト=300000）
//-----------------------------------------------------------------------
  int ret;

  ret = ACat_Slv_Foe_Upload(1001, "ECATFW__B1_UP.efw", 0x90000, 0x00000000, 300000);
  if (ret) {
      printf("FoE アップロード失敗 RET=%08X¥n", ret);
  } else {
      printf("FoE アップロード成功¥n");
  }
```

３－１－１６ CUnet ゲートウェイユニット GM アクセス制御関数サンプルコード

CUnet ゲートウェイユニットの GM アクセスの制御関数のサンプルコードを次に示します。

１）ユニットオープン

```
//-----------------------------------------------------------------------------
//      CUnet ゲートウェイユニット 初期化、オープン(スレーブ ID=1)
//-----------------------------------------------------------------------------
  int ret;

  ret = ACat_CN_Create();                       // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }

  ret = ACat_CN_open(1);                              // スレーブ ID=1 をオープン
  if (ret) {
      printf("OPEN 失敗 RET=%08X", ret);
  } else {
      printf("OPEN 完了");
  }
```

２）CUnet 通信開始/停止

```
//-----------------------------------------------------------------------------
//      通信開始(スレーブ ID=1、CH=1)
//-----------------------------------------------------------------------------
  int ret;
  int StationAdr  = 0;          // ステーションアドレス  SA0
  int StationSize = 4;          // 占有幅                4
  int ByteBusSize = 2;          // バスサイズ            2

  ret = ACat_CN_start(1, 1, StationAdr, StationSize, ByteBusSize);
  if (ret) {
      printf("ACat_CN_start 失敗 RET=%08X¥n", ret);
  } else {
      printf("ACat_CN_start 完了¥n");
  }

//-----------------------------------------------------------------------------
//       通信停止(スレーブ ID=1、CH=1)
//-----------------------------------------------------------------------------
  ret = ACat_CN_stop(1, 1);
  if (ret) {
      printf("ACat_CN_stop 失敗 RET=%08X¥n", ret);
  } else {
      printf("ACat_CN_stop 完了¥n");
  }
```

３）CUnet GM エリア データ読込み/書込み

```
//--------------------------------------------------------------------------
//       DLong データ書込み(スレーブ ID=1、CH=1、SA=0)
//--------------------------------------------------------------------------
  int ret;
  WORD writebuf[4];

  writebuf[0] = 0x1234;
  writebuf[1] = 0x5678;
  writebuf[2] = 0x9ABC;
  writebuf[3] = 0xDEF0;
  ret = ACat_CN_SetMemDLong(1, 1, (unsigned long long *)writebuf, 0, 0, 1);
  if (ret) {
      printf("ACat_CN_SetMemDLong 失敗 RET=%08X¥n", ret);
  } else {
      printf("ACat_CN_SetMemDLong 完了¥n");
  }

//--------------------------------------------------------------------------
//       DLong データ読込み(スレーブ ID=1、CH=1、SA=4)
//--------------------------------------------------------------------------
  int ret;
  WORD readbuf[4];

  ret = ACat_CN_GetMemDLong(1, 1, (unsigned long long *)readbuf, 4, 0, 1);
  if (ret) {
      printf("ACat_CN_GetMemDLong 失敗 RET=%08X¥n", ret);
  } else {
      printf("ACat_CN_GetMemDLong 完了¥n");
      printf("readbuf[0]=0x%04x readbuf[1]=0x%04x readbuf[2]=0x%04x readbuf[3]=0x%04x¥n",
              readbuf[0], readbuf[1], readbuf[2], readbuf[3]);
  }
```

４）CUnet 自己占有エリア クリア

```
//--------------------------------------------------------------------------
//       自己占有エリアクリア(スレーブ ID=1、CH=1)
//--------------------------------------------------------------------------
  int ret;
  WORD data = 0x0000;

  ret = ACat_CN_reset(1, 1, data);
  if (ret) {
      printf("ACat_CN_reset 失敗 RET=%08X¥n", ret);
  } else {
      printf("ACat_CN_reset 完了¥n");
  }
```

５）CUnet 通信異常/ステータス読込み

```
//--------------------------------------------------------------------------
//       接続ステータス読込み(スレーブ ID=1、CH=1)
//--------------------------------------------------------------------------
  int ret;
  int StationAdr, StationSize;

  ret = ACat_CN_GetCnectStat(1, 1, &StationAdr, &StationSize);
  if (ret) {
      printf("ACat_CN_GetCnectStat 失敗 RET=%08X", ret);
  } else {
      printf("ACat_CN_GetCnectStat 完了: StationAdr=%d StationSize=%d",
          StationAdr, StationSize);
  }

//--------------------------------------------------------------------------
//       ステータス読込み(スレーブ ID=1、CH=1)
//--------------------------------------------------------------------------
  Int ret;
  CN_STATUS status;
  memset(&status, '¥0', sizeof(CN_STATUS));

  ret = ACat_CN_Status(1, 1, &status);
  if (ret) {
      printf("ACat_CN_Status 失敗 RET=%08X¥n", ret);
  } else {
      printf("ACat_CN_Status 完了¥n");
      printf("status=%d sa=%d slen=%d¥n",   status.status, status.sa, status.slen);
  }
```

６）CUnet 通信設定書込み

```
//--------------------------------------------------------------------------
//       CUnet 通信設定書込み(スレーブ ID=1、CH=1)
//--------------------------------------------------------------------------
  int ret;
  int baud = 6;          // 6Mbps

  ret = ACat_CN_config(1, 1, baud);
  if (ret) {
      printf("ACat_CN_config 失敗 RET=%08X", ret);
  } else {
      printf("ACat_CN_config 完了");
  }
```

７）ユニットハードリセット

```
//--------------------------------------------------------------------------
//       ユニットハードウェアリセット(スレーブ ID=1、CH=1)
//--------------------------------------------------------------------------
  int ret;

  ret = ACat_CN_HardReset(1, 1);
  if (ret) {
      printf("ACat_CN_HardReset 失敗 RET=%08X", ret);
  } else {
      printf("ACat_CN_HardReset 完了");
  }
```

８）ユニットクローズ

```
//-------------------------------------------------------------------------
//      CUnet ゲートウェイユニット クローズ(スレーブ ID=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_CN_close(1);                          // クローズ
  if (ret) {
      printf("CLOSE 失敗 RET=%08X", ret);
  } else {
      printf("CLOSE 完了");
  }

  ret = ACat_CN_Destroy();                         // 終了
  if (ret) {
      printf("DESTROY 失敗 RET=%08X", ret);
  } else {
      printf("DESTROY 完了");
  }
```

３－１－１７ CUnet ゲートウェイユニット メール通信制御関数サンプルコード

CUnet ゲートウェイユニットのメール通信の制御関数のサンプルコードを次に示します。

１）メールのオープン

```
//-------------------------------------------------------------------------
//      CUnet ゲートウェイユニット オープン(スレーブ ID=1、CH=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_CN_Create();                       // 初期化
  if (ret) {
      printf("CREATE 失敗 RET=%08X", ret);
  } else {
      printf("CREATE 完了");
  }

  ret = ACat_CN_open(1);                                  // スレーブ ID=1 をオープン
  if (ret) {
      printf("OPEN 失敗 RET=%08X¥n", ret);
  } else {
      printf("OPEN 完了¥n");
  }

//-------------------------------------------------------------------------
//      通信開始(スレーブ ID=1、CH=1)
//-------------------------------------------------------------------------
  int StationAdr  = 0;          // ステーションアドレス  SA0
  int StationSize = 4;          // 占有幅                4
  int ByteBusSize = 2;          // バスサイズ            2
  ret = ACat_CN_start(1, 1, StationAdr, StationSize, ByteBusSize);
  if (ret) {
      printf("ACat_CN_start 失敗 RET=%08X¥n", ret);
  } else {
      printf("ACat_CN_start 完了¥n");
  }

//-------------------------------------------------------------------------
//      メールオープン(スレーブ ID=1、CH=1)
//-------------------------------------------------------------------------
  ret = ACat_CN_MailOpen(1, 1);                           // メールオープン
  if (ret) {
      printf("MAIL OPEN 失敗 RET=%08X", ret);
  } else {
      printf("MAIL OPEN 完了");
  }
```

２）メールの送信

```
//-------------------------------------------------------------------------
//      CUnet ゲートウェイユニット メール送信(スレーブ ID=1、CH=1)
//-------------------------------------------------------------------------
  int ret;
  int StationAdr=3;     // 送信先 SA=3
  BYTE Mailbuf[256];

  ret = ACat_CN_MailSend(1, 1, StationAdr, &Mailbuf, 256);      // メール送信
  if (ret) {
      printf("ACat_CN_MailSend 失敗 RET=%08X", ret);
  } else {
      printf("ACat_CN_MailSend 完了");
  }
```

３）メールの受信(別スレッド)

```
//-------------------------------------------------------------------------
//      CUnet ゲートウェイユニット メール受信(スレーブ ID=1、CH=1)
//-------------------------------------------------------------------------
  int ret;
  int StationAdr;
  short Status;
  BYTE Mailbuf[256];

  while (fEnd) {
    ret = ACat_CN_MailWaitObject(1, 1, &Status);        // メール受信待ち
    if (ret == ACAT_ER_CN_MAILSENDRESULT) {
      if ((Status & 0x00ff) == 0x0001) {
        // メール送信完了(正常)
      }
      if ((Status & 0x00ff) == 0x0002) {
        // 相手先が BUSY
      }
      if ((Status & 0x00ff) == 0x0004) {
        // 相手先が不在
      }
    } else if (ret == ACAT_ER_CN_MAILRECEIVE) {
      ret = ACat_CN_MailReceive(1, 1, &Mailbuf, 256, &StationAdr);
      if (ret) {
        printf("ACat_CN_ MailReceive 失敗 RET=%08X", ret);
      } else {
        printf("ACat_CN_ MailReceive 完了");
      }
    }
  }
```

４）メールの受信(ステータスのポーリング)

```
//-------------------------------------------------------------------------
//      CUnet ゲートウェイユニット メール受信(スレーブ ID=1、CH=1)
//-------------------------------------------------------------------------
  int ret;
  int StationAdr;
  short Status;
  BYTE Mailbuf[256];

  while (Status == 0) {
    ACat_CN_MailStatus(1, 1, &Status);      // メール受信待ち
    if ((Status & 0xff00) == 0x0100) {
      // メール受信
      ret = ACat_CN_MailReceive(1, 1, &Mailbuf, 256, &StationAdr);
      if (ret) {
        printf("ACat_CN_ MailReceive 失敗 RET=%08X", ret);
      } else {
        printf("ACat_CN_ MailReceive 完了");
      }
    }
  }
```

５）メールのクローズ

```
//-------------------------------------------------------------------------
//      CUnet ゲートウェイユニット メールクローズ(スレーブ ID=1、CH=1)
//-------------------------------------------------------------------------
  int ret;

  ret = ACat_CN_MailClose (1, 1);       // メールクローズ
  if (ret) {
      printf("CLOSE 失敗 RET=%08X", ret);
  } else {
      printf("CLOSE 完了");
  }
```

３－２　EtherCAT 用語説明

用語説明一覧

EtherCAT Technology Group(ETG)	: EtherCAT をオープン化し普及促進をはかる団体
EtherCAT State Machine(ESM)	: EtherCAT の通信状態遷移
EtherCAT Specification	: EtherCAT 仕様
EtherCAT Slave Information(ESI)	: EtherCATスレーブの設定情報が記述されたXML形式のファイル
EtherCAT Notification	: EtherCAT マスタ通知
ベンダーID	: ETG に登録される EtherCAT のベンダーID
プロダクトコード	: EtherCAT ユニットの製品コード
スレーブアドレス(スレーブ ID)	: ステーションエイリアスアドレス。 スレーブによって DipSW 等で設定されるアドレス。
フィジカルアドレス	: スレーブユニットのステーションアドレス。 マスタから設定されるアドレス。
コンフィグファイル(config.xml)	: 接続される全スレーブユニットの設定情報が記述された XML 形式のファイル(EtherCAT Network Information ファイルと同意)
ディストリビュートクロック(DC)	: EtherCAT スレーブとマスタを同期させるための分配クロック
オブジェクト	: デバイス内の特定の構成体
オブジェクトディクショナリ(OD)	: オブジェクトの記述が入ったデータ構造
サービスデータオブジェクト(SDO)	: ODにアクセスできるCoEの非同期メールボックス通信
インデックス	: OD に含まれるオブジェクトのインデックス
サブインデックス	: OD に含まれるオブジェクトのサブインデックス
プロセスデータ	: 周期的あるいは非周期的に転送されるアプリケーションオブジェクト
プロセスデータオブジェクト(PDO)	: 1 つ以上のプロセスデータエンティティを持つパラメータをマッピングすることによって記述される構造
入力 PDO(Receive PDO)	: EtherCAT スレーブで受信する PDO
出力 PDO(Transmit PDO)	: EtherCAT スレーブで送信する PDO

このユーザーズマニュアルについて

（１）本書の内容の一部また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下さい。
（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もしくは、営業所までご連絡下さい。その際、巻末記載の書籍番号も併せてお知らせ下さい。

７６ＤＬＨ００４６Ｊ　　２０２０年　１０月　第１０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/