

ixxat INpact Ethernet/IP Slave Pcle Setup Guide on Ubuntu

By kuwano (@calm0815)





Overview

Ethernet/IPの構成と通信に関するざっとした説明

Requirements

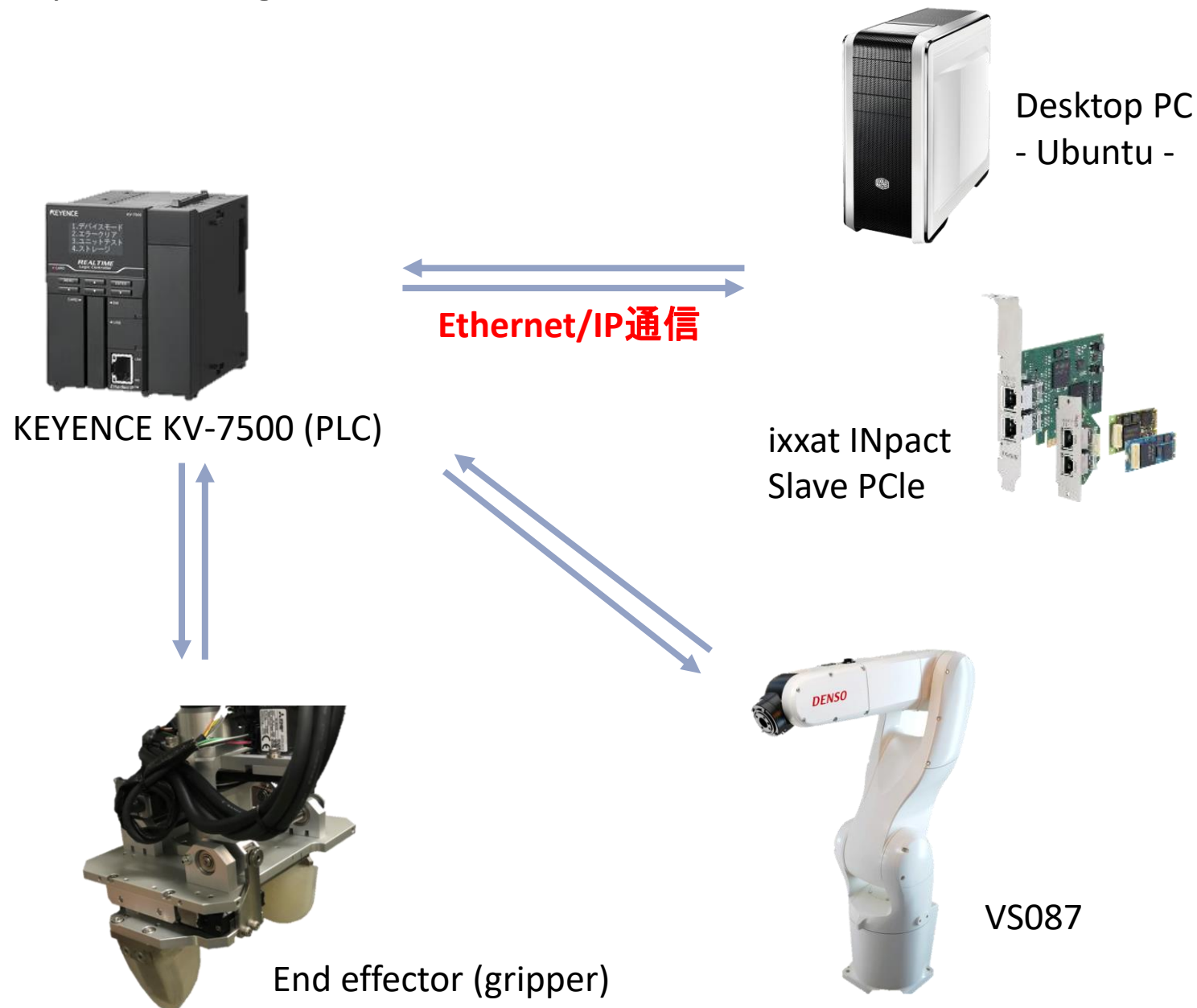
- INpact Slave PCIeが設置されたPC (Ubuntu)
- 設定時に使用するPC (windows)
- 通信の設定ファイル(.eds)
- 諦めない心

あきらめるか！



諦めてない心くん⇒

System Configuration

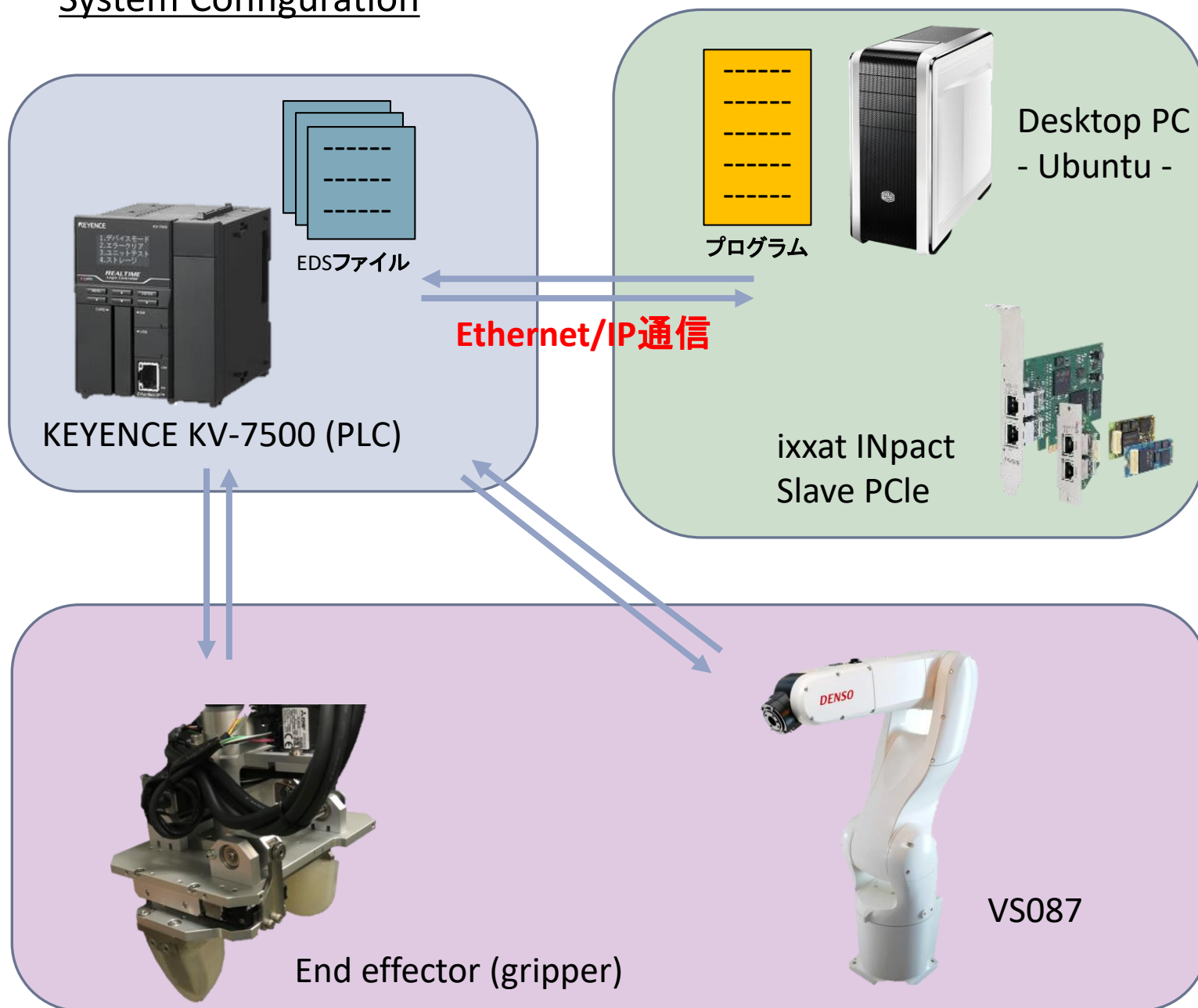


Ethernet/IP通信

Ethernet/IP通信はPLC側とPC側の
それぞれが

- 自分の持つ情報を頼りに相手の
機器を探す
 - PLCはEDSファイルを頼りに探す
 - PCは通信に用いる
プログラムに記載された
情報を頼りに探す
- 情報が一致する相手と通信を行う
の手順を踏んで行われる。

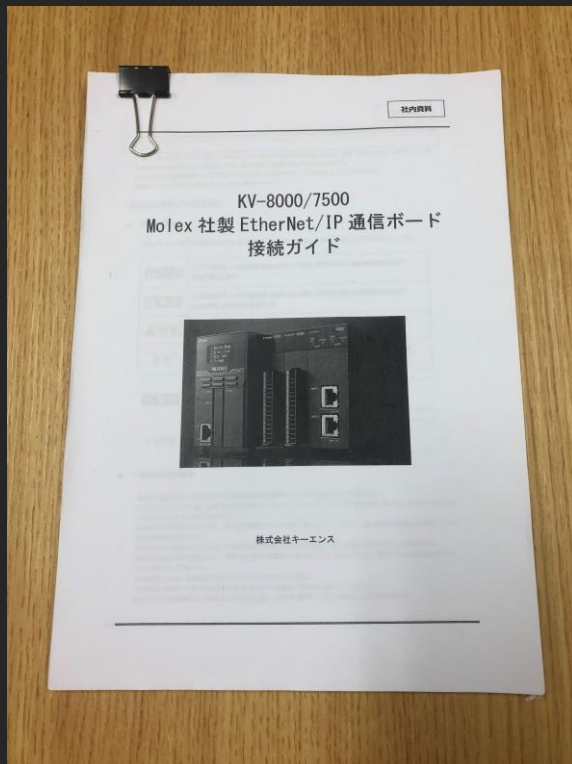
System Configuration



Ethernet/IP通信

通信する際のデータに関する
取り決めは右記の通り(2019/9/7)

↓ピンとこない場合は下記のP6-7を参照



Ethernet/IPの通信仕様

- INpactモジュールのIPアドレス: 192.168.2.100
- ノード数: 1
- インスタンスのアウトプット番号: 101
- アウトプットで扱うデータサイズ: 500 byte
- インスタンスのインプット番号: 102
- インプットで扱うデータサイズ: 500 byte
- インスタンスのconfig: 0 word

今回のセットアップで編集した設定ファイルはNASにバックアップを取ってます。

再セットアップ時はそれを再利用してください。
通信仕様変更時は後半の備忘録を参考にしてください。

Ethernet/IP通信

サンプルプログラムの内部
(`IdlLinux_amd64/src/IdlApp/example_app/appl_adimap_separate 16.c`)では、PLCから送られてきたデータを
`appl_aiUint16_10[]` (PLC⇒PC)と
`appl_aiUint16_11[]` (PC⇒PLC)の
2つの配列に値を格納し、これらを用いて値の送受信を行っている。

Ethernet/IP通信で扱う
通信プロトコルは右記の通り。

これに準拠してハンドの制御基板であるPLCとROSを扱うPCの通信を行う。

Ethernet/IPの通信プロトコル

PLC⇒PC

OUT	0	W11A	チャック動作可否	0=動作不可 1=動作許可		
OUT	1	W11B	チャック状態	0=未原点 1=開端(原点) 2=閉端		
OUT	2	W11C	チャック現在値	実数 0~60mm程度 ※片側の閉じ量を表示		入力桁は3桁入力 下位1桁目は小数点第一
OUT	3	W11D				
OUT	4	W11E				
OUT	5	W11F				
OUT	6	W120	現在締め付けトルク設定値	0~100%		
OUT	7	W121	チャック動作速度	mm/sec		
OUT	8	W122				
OUT	9	W123				
OUT	10	W124				
OUT	11	W125				
OUT	12	W126				
OUT	13	W127				
OUT	14	W128				
OUT	15	W129				
OUT	16	W12A	芯押し動作可否	0=動作不可 1=動作許可		
OUT	17	W12B	芯押し状態	0=未原点 1=開端(原点) 2=閉端		
OUT	18	W12C	芯押し現在値	実数 0~30mm程度 ※片側の閉じ量を表示		入力桁は3桁入力 下位1桁目は小数点第一
OUT	19	W12D				
OUT	20	W12E				
OUT	21	W12F				
OUT	22	W130	現在締め付けトルク設定値	0~100%		
OUT	23	W131	チャック動作速度	mm/sec		

PC⇒PLC

IN	0	W20	チャック動作指示	1=開き指令 2=閉じ指令		1secパルス
IN	1	W21	チャック保持トルク指定	0から100%で入力		
IN	2	W22	チャック動作速度指定	mm/secで入力		
IN	3	W23				
IN	4	W24				
IN	5	W25				
IN	6	W26				
IN	7	W27				
IN	8	W28				
IN	9	W29				
IN	10	W2A	芯押し動作指示	1=上昇指令 2=下降指令		1secパルス
IN	11	W2B	芯押し押し付けトルク指定	0から100%で入力		
IN	12	W2C	芯押し動作速度指定	mm/secで入力		



INpact Setup Guide

ドライバインストールから、通信アプリの動作確認までの手順を記してます。

この手順を踏んだファイルは既にNAS上にあります。再セットアップが必要な場合にのみ参考になしてください。

Step1.1

ubuntuにInpact Slave PCIeの KernelModuleをインストール

「(付属ディスク内)
/products/inpact/driver/Linux/」に
ある3つファイルのうち、使用PCの環境
に合ったものを解凍する。

解凍したファイルをローカルにコピーし
てくる。

(インストールした後のフォルダ移動と
かで文句言われたくないから)

解凍されたファイルのすぐ下にある
「ReadMe_Linux.txt」にある
「Compiling the Kernel Module」の項に
従ってモジュールをインストールする。

「Using IDL Application」にある
「./LinuxIdlApp」を起動し、「hogegege
WAIT_PROCESS」と表示されればOK



Step1.1

ubuntuにInpact Slave PCIeの KernelModuleをインストール

「(付属ディスク内)
/products/inpact/driver/Linux/」に
ある3つファイルのうち、使用PCの環境
に合ったものを解凍する。

解凍したファイルをローカルにコピーし
てくる。

(インストールした後のフォルダ移動と
かで文句言われたくないから)

解凍されたファイルのすぐ下にある
「ReadMe_Linux.txt」にある
「Compiling the Kernel Module」の項に
従ってモジュールをインストールする。

「Using IDL Application」にある
「./LinuxIdlApp」を起動し、「hogegege
WAIT_PROCESS」と表示されればOK

トラブルシューティング

- 「./LinuxIdlApp」の起動時に「abcc_OpenControl Failed」というエラーが表示された場合
 - このメッセージの意味は
「If the hardware is used by another application the call fails.」
別のターミナルなどで「./LinuxIdlApp」を起動している場合は
そちらを先に閉じること。起動していないのにこのエラーが出たら詰み
- 「cat /proc/idl」の実行時、「Killed」が返ってくる。
 - HMS社のSEによれば以下の要因で発生するとのこと
 - ixcat INpactモジュールが認識されていない。または
物理的に挿されていない。
 - カーネルがアンインストールされた
- 「./LinuxIdlApp」が起動するのであれば、デバイスの起動はでき
ているので、いったん無視していても構わない。

Step1.2

IdlAppのソースコードをEDSファイル
(Ethernet/IP通信の設定ファイル)
に合わせて編集する

*** 前項で「IdlLinux_amd64」を解凍したと仮定して説明する**

「IdlLinux_amd64/src/IdlApp/」に移動

統合開発環境eclipseに「.project」をimportする。(READMEでeclipseが使用されていた。Cのコンパイルができるなら、何を使ってもいいはず。)

Importしてきたら、通信の仕様が設定されたEDSファイルに沿ってファイルを編集していく。

【編集終了後】

ツールバーの「project」から「clear」⇒「build all」の順でビルドする。

*** EDSファイルと上記の作業済みファイルはNASにアップロードしておきます。***

編集内容(メールのやり取りのコピペです)

1. EDS ファイルの準備

INpactを挿したPC(ターゲット)の仕様にあわせて、EDSファイルを準備(作成)する必要があります。

本来は、ターゲットの設定にあわせてオリジネータ(PLC側)をあわせる必要がありますが、今回は「PLC業者さま(注: 前田機工)からの設定にINpact側をあわせる」という話で進めさせていただきます。

今回は事前の情報を基にEDSファイル[1]をご用意いたしました。本メールに添付いたしましたのでご利用いただければと思います。

弊社にて KEYENCE製PLC KV-7500 と Inpact をつなぎ、サイクリック通信接続できることを確認しております。

「lxxat_Inpact_PCle_EthernetIP_201608026.eds」を基に作成しております。元のファイルと比較していただき、変更点をご確認いただければと思います。

[1] 作成していただいたEDSファイルはNASに保存しています。

Step1.2

IdlAppのソースコードをEDSファイル
(Ethernet/IP通信の設定ファイル)
に合わせて編集する

*** 前項で「IdlLinux_amd64」を解凍したと仮定して説明する**

「IdlLinux_amd64/src/IdlApp/」に移動

統合開発環境eclipseに「.project」をimportする。(READMEでeclipseが使用されていた。Cのコンパイルができるなら、何を使ってもいいはず。)

Importしてきたら、通信の仕様が設定されたEDSファイルに沿ってファイルを編集していく。

【編集終了後】

ツールバーの「project」から「clear」⇒「build all」の順でビルドする。

*** EDSファイルと上記の作業済みファイルはNASにアップロードしておきます。***

編集内容(メールのやり取りのコピペです)

2. instance番号の変更

instance 番号は、ホストアプリケーションで設定ができます。
INpactモジュールは、ホストアプリケーションからなにも設定しない場合

> Master(PLC)⇒PC(INpact) : input (instance = 150)
> PC(INpact)⇒Master(PLC) : output (instance = 100)

として動作する仕様になっています。

instance番号を指定する場合、abcc_adapt¥abcc_obj_cfg.h ファイル内で以下のように変更します。

- EIP_OBJ_ENABLE 定義を TRUE に変更 → 同ファイル内「EIP_IA_XXXX」定義が有効になります
- EIP_IA_PROD_INSTANCE_ENABLE 定義を TRUE に変更
- EIP_IA_PROD_INSTANCE_VALUE 定義で output 側のinstance 番号を指定
- EIP_IA_CONS_INSTANCE_ENABLE 定義を TRUE に変更
- EIP_IA_CONS_INSTANCE_VALUE 定義で input 側のinstance 番号を指定

INpact 起動時に、上記設定をINpactに返す⇒PLCは設定されたinstance番号で接続できる状態になります

Step1.2

IdlAppのソースコードをEDSファイル
(Ethernet/IP通信の設定ファイル)
に合わせて編集する

*** 前項で「IdlLinux_amd64」を解凍したと仮定して説明する**

「IdlLinux_amd64/src/IdlApp/」に移動

統合開発環境eclipseに「.project」をimportする。(READMEでeclipseが使用されていた。Cのコンパイルができるなら、何を使ってもいいはず。)

Importしてきたら、通信の仕様が設定されたEDSファイルに沿ってファイルを編集していく。

【編集終了後】

ツールバーの「project」から「clear」⇒「build all」の順でビルドする。

*** EDSファイルと上記の作業済みファイルはNASにアップロードしておきます。***

編集内容(メールのやり取りのコピペです)

3. 実行する「サイクリック通信のプログラム」を切り替え

ホストアプリケーションは、そのままの状態ですと
example_app¥appl_adimap_speed.c のプログラムを動かす仕組みになっています。このプログラムは4バイトのサイクリック通信となっております。

他にもappl_adimap_xxxx.c というファイルがサイクリック通信のサンプルプログラムになっており、実際のプログラムを実行するかはexample_app¥appl_adi_config.h 内 APPL_ACTIVE_ADI_SETUP 定義で決まります。

appl_adimap_speed.c のプログラムは4バイトのデータを複数の変数を使って構成しており、少し複雑なところがあります。
500バイトの通信とのことですので、以降 appl_adimap_separate16.c という比較的わかりやすいプログラムをもとにお話します。

上記より、appl_adimap_separate16.c を実行する場合は

example_app¥appl_adi_config.h 内 APPL_ACTIVE_ADI_SETUP 定義を
APPL_ADI_SETUP_SEPARATE_16 にします。

- ✓ 参考までに、appl_adimap_simple16.c は、受け取ったデータをappl_aiUint16配列に格納・内容をそのままPLC側に出力する、という構成になっています

Step1.2

IdlAppのソースコードをEDSファイル
(Ethernet/IP通信の設定ファイル)
に合わせて編集する

* 前項で「IdlLinux_amd64」を解凍したと仮
定して説明する

「IdlLinux_amd64/src/IdlApp/」に移動

統合開発環境eclipseに「.project」を
importする。(READMEでeclipseが使用
されていた。Cのコンパイルができるな
ら、何を使ってもいいはず。)

Importしてきたら、通信の仕様が設定
されたEDSファイルに沿ってファイルを
編集していく。

【編集終了後】

ツールバーの「project」から「clear」⇒
「build all」の順でビルドする。

* EDSファイルと上記の作業済みファイルは
NASにアップロードしておきます。*

編集内容(メールのやり取りのコピペです)

4. サイクリック通信のデータバイト数を変更

ネットワーク側(PLC)からアクセスできるデータは、APPL_asAdiEntryList 構造体で定義
されています。

appl_adimap_separate16.c では、この構造体の定義内容は以下のようになります。

- ABP_UINT16_SET の行:
 - Master(PLC)⇒PC(INpact) のサイクリック通信の定義
 - 16bit変数 x 32 = 64 バイトの通信設定
 - 受け取ったデータは appl_aiUint16_10 配列に格納
 - PLCからデータを受け取ったら SetAdi10Value 関数を実行
- ABP_UINT16_GET の行:
 - PC(INpact)⇒Master(PLC) のサイクリック通信の定義
 - 16bit変数 x 32 = 64 バイトの通信設定
 - appl_aiUint16_11 配列の内容をPLC側に出力
 - GetAdi11Value 関数を実行
- ABP_UINT16_COUNTER の行:
 - サイクリック通信ではなく、PLCから単発メッセージを使ってアクセスできる変数の定義
 - 16bit変数 x 1 = 2 バイトの通信設定
 - appl_Uint16_12 変数にアクセス可能

appl_adimap_separate16.c は、APPL_asAdObjDefaultMap 構造体により上記のうち
最初の2つ(ABP_UINT16_SET と ABP_UINT16_GET)がサイクリック通信として
定義されています。

よってこれらの行でサイクリック通信のデータサイズを決めることができます。
例えば 32 ⇒ 50 にすることで 100 バイトのサイクリック通信とすることができます。

Step1.2

IdlAppのソースコードをEDSファイル
(Ethernet/IP通信の設定ファイル)
に合わせて編集する

* 前項で「IdlLinux_amd64」を解凍したと仮
定して説明する

「IdlLinux_amd64/src/IdlApp/」に移動

統合開発環境eclipseに「.project」を
importする。(READMEでeclipseが使用
されていた。Cのコンパイルができるな
ら、何を使ってもいいはず。)

Importしてきたら、通信の仕様が設定
されたEDSファイルに沿ってファイルを
編集していく。

【編集終了後】

ツールバーの「project」から「clear」⇒
「build all」の順でビルドする。

* EDSファイルと上記の作業済みファイルは
NASにアップロードしておきます。*

編集内容(メールのやり取りのコピペです)

5. デバイス情報の設定

INpactを挿したPC(ターゲット)とEDSファイルに記載されているデバイスの情
報を一致させる必要があります。不一致の場合、PLCがINpactと通信できな
い場合があります。

デバイス情報の設定は以下のように行います。

- abcc_adapt¥abcc_obj_cfg.h ファイル内 EIP_OBJ_ENABLE 定義を
TRUE に変更
- abcc_adapt¥abcc_identification.h ファイル内 の以下の定義を変更
 - EIP_IA_VENDOR_ID_VALUE:EDSファイル内 VendCode キー値
 - EIP_IA_DEVICE_TYPE_VALUE:EDSファイル内 ProdType キー値
 - EIP_IA_PRODUCT_CODE_VALUE:EDSファイル内 ProdCode キー値
 - EIP_IA_REVISION_MAJOR_VALUE:EDSファイル内 MajRev キー値
 - EIP_IA_REVISION_MINOR_VALUE:EDSファイル内 MinRev キー値
 - EIP_IA_SERIAL_NUMBER_VALUE:任意です。
 - EIP_IA_PRODUCT_NAME_VALUE:EDSファイル内 ProdName キー値
(Ethernet/IPの規格で33文字以内と定められているので、
設定する文字数に気を付けること)

EDSファイルではバージョン1.35 と定義してありますが、このバージョンもお
お客様の管理するバージョン番号に設定していただくことが可能です。

EDSファイル・プログラムともに 1.1 などに変更していただいても結構です。

Step1.2

IdlAppのソースコードをEDSファイル
(Ethernet/IP通信の設定ファイル)
に合わせて編集する

* 前項で「IdlLinux_amd64」を解凍したと仮
定して説明する

「IdlLinux_amd64/src/IdlApp/」に移動

統合開発環境eclipseに「.project」を
importする。(READMEでeclipseが使用
されていた。Cのコンパイルができるな
ら、何を使ってもいいはず。)

Importしてきたら、通信の仕様が設定
されたEDSファイルに沿ってファイルを
編集していく。

【編集終了後】

ツールバーの「project」から「clear」⇒
「build all」の順でビルドする。

* EDSファイルと上記の作業済みファイルは
NASにアップロードしておきます。*

編集内容(メールのやり取りのコピペです)

6. そのほか、アプリケーションの変更の依頼

これは本件には関連してありませんが、ある機能が実行されたときに、
期待通りに動作しない箇所があります。

abcc_obj¥ad_obj.c - AD_ProcObjectRequest()関数内の
case ABP_APPD_IA_NUM_SUB_ELEM:の分岐におきまして、
以下ブルーハッチングの箇所

```
1482 |         case ABP_APPD_IA_NUM_SUB_ELEM:↓
1483 | #if( ABCC_CFG_STRUCT_DATA_TYPE )↓
1484 |     if( psAdiEntry->psStruct != NULL )↓
1485 |     {↓
1486 |         UINT16 i;↓
1487 |         ↓
1488 |         iDataSize = ABP_APPD_IA_NUM_SUB_ELEM_DS * psAdiEntry->bNumOfElements;↓
1489 |         for ( i = 0; i < psAdiEntry->bNumOfElements; i++ )↓
1490 |         {↓
1491 |             ABCC_SetMsgData16( psMsgBuffer,↓
1492 |                                 psAdiEntry->psStruct[i].iNumSubElem,↓
1493 |                                 ( i * ABP_APPD_IA_NUM_SUB_ELEM_DS ) );↓
1494 |             ↓
1495 |         }↓
1496 |     }↓
1497 | #endif↓
1498 |     {↓
1499 |         ABCC_SetMsgData16( psMsgBuffer, AD_DEFAULT_NUM_SUB_ELEMS, 0 );↓
1500 |         iDataSize = ABP_APPD_IA_NUM_SUB_ELEM_DS;↓
1501 |     }↓
1502 |     break;↓
1503 | default:↓
1504 |     ↓
```

を

Step1.2

IdlAppのソースコードをEDSファイル
(Ethernet/IP通信の設定ファイル)
に合わせて編集する

* 前項で「IdlLinux_amd64」を解凍したと仮
定して説明する

「IdlLinux_amd64/src/IdlApp/」に移動

統合開発環境eclipseに「.project」を
importする。(READMEでeclipseが使用
されていた。Cのコンパイルができるな
ら、何を使ってもいいはず。)

Importしてきたら、通信の仕様が設定
されたEDSファイルに沿ってファイルを
編集していく。

【編集終了後】

ツールバーの「project」から「clear」⇒
「build all」の順でビルドする。

* EDSファイルと上記の作業済みファイルは
NASにアップロードしておきます。*

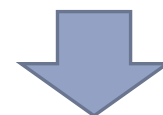
編集内容(メールのやり取りのコピペです)

6. そのほか、アプリケーションの変更の依頼

```
1462  case ABP_APPD_IA_NUM_SUB_ELEM:↓
1463  #if( ABCC_CFG_STRUCT_DATA_TYPE )↓
1464      if( psAdiEntry->psStruct != NULL )↓
1465          {↓
1466              UINT16 i;↓
1467              ↓
1468              iDataSize = ABP_APPD_IA_NUM_SUB_ELEM_DS * psAdiEntry->bNumOfElements;↓
1469              for ( i = 0; i < psAdiEntry->bNumOfElements; i++ )↓
1470                  {↓
1471                      ABCC_SetMsgData16( psMsgBuffer,↓
1472                                          psAdiEntry->psStruct[i].iNumSubElem,↓
1473                                          ( i * ABP_APPD_IA_NUM_SUB_ELEM_DS ) );↓
1474                  }↓
1475              }↓
1476          else↓
1477      #endif↓
1478          {↓
1479              bErrCode = ABP_ERR_INV_CMD_EXT_0;↓
1480          }↓
1481          break;↓
1482      default:↓
1483          ...
```

bErrCode = ABP_ERR_INV_CMD_EXT_0;

に変更していただけますでしょうか。



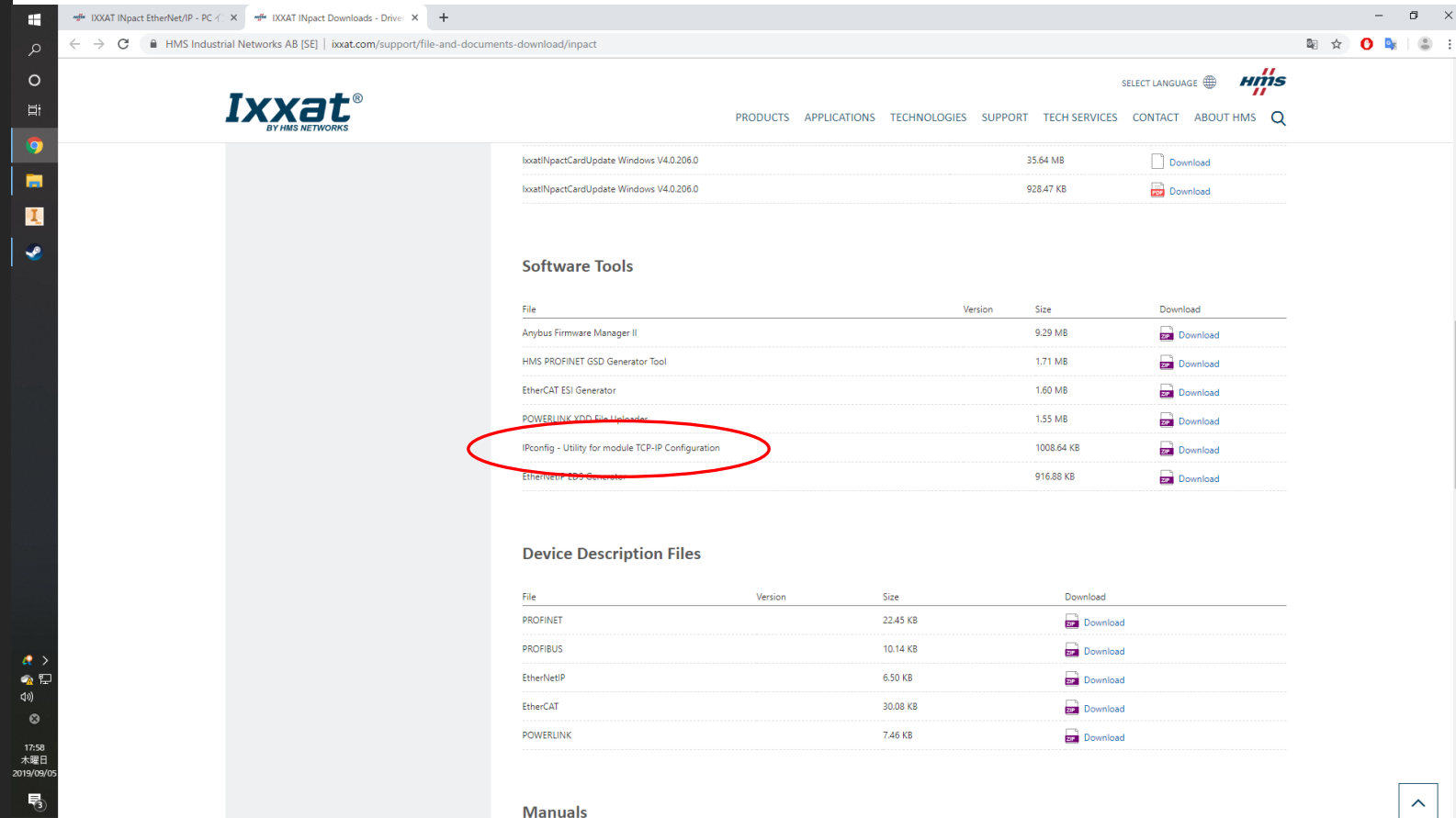
編集終わり

※編集したファイルはNASに保存してます

Step2.1 windowsにIpconfig.exeを インストール

ixxat社のサイトへ行き、
「Software Tools」の項目から
「IPconfig - Utility for module TCP-IP
Configuration」をダウンロード

その後インストールを行う。



URL : <https://www.ixxat.com/support/file-and-documents-download/inpact>

Step2.2

INpact Slave PCleに固定IPを セットする

- ◆ PC(windows)とPC(ubuntu/PCle)をLANケーブルで繋いでおくこと

Step2.1でインストールしたIpconfig.exeを起動する。

起動したら「scan」を押し、対象機器が認識されることを確認する。
(最初はすべての値が0になっている)

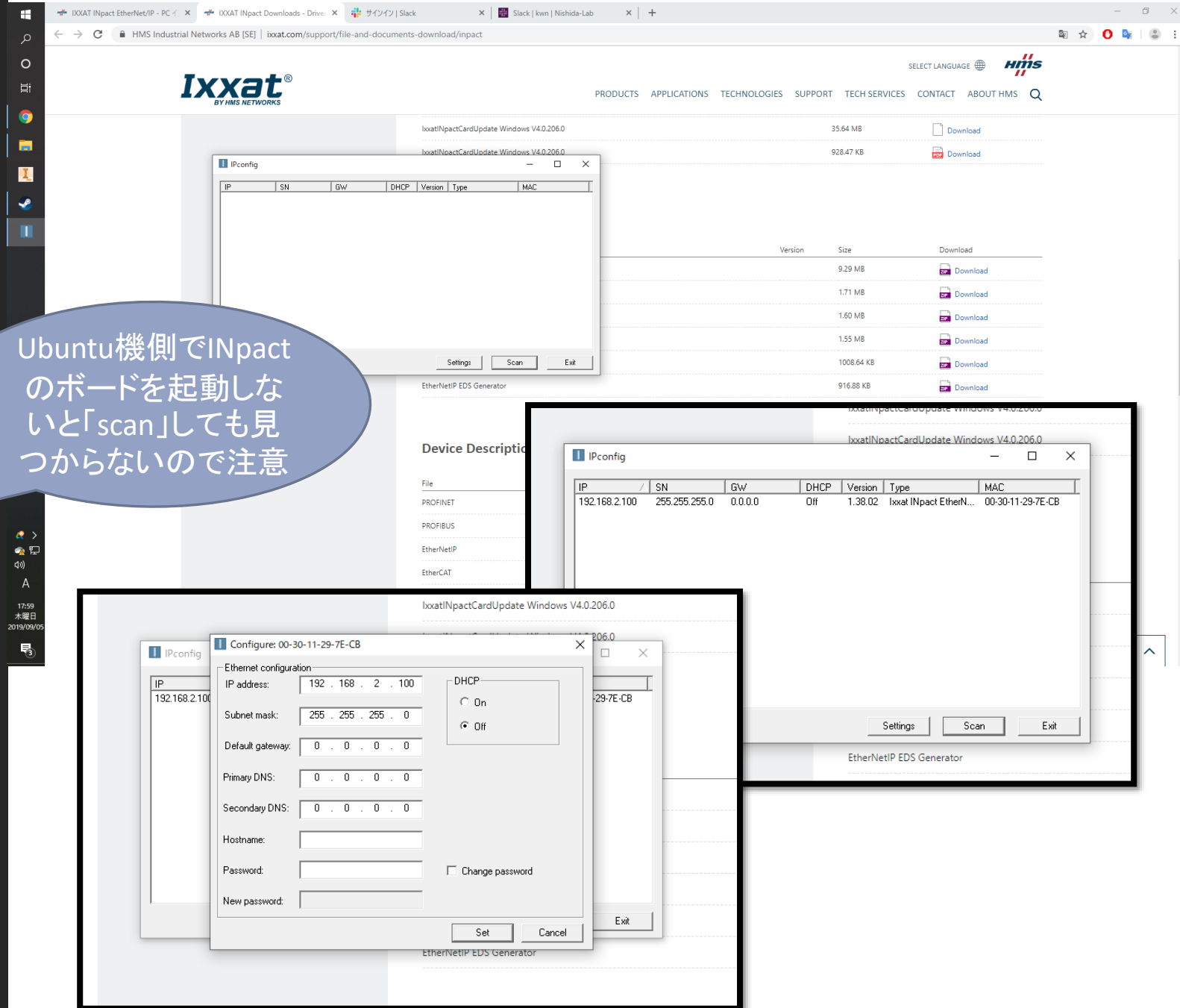
「DHCP」の項目(画像中で「off」になっている箇所)をダブルクリックして、選択画面を出す。

設定項目を埋めて「Set」

PC(windows)での作業はこれで終了
windowsは撤去してヨシ



Ubuntu機側でINpact
のボードを起動しな
いと「scan」しても見
つからないので注意

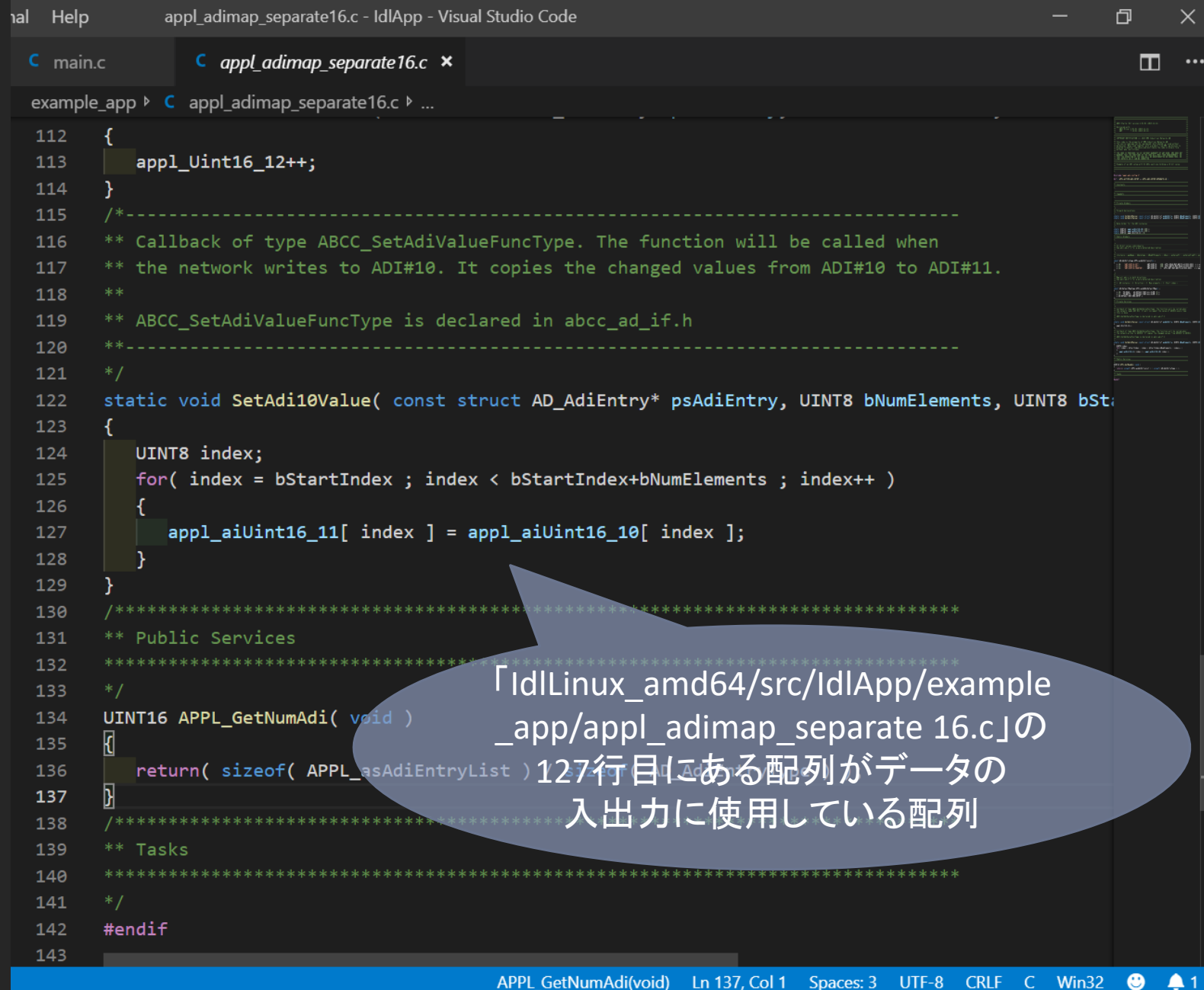


Step3.1 PLCとPCの接続確認

「IdlLinux_amd64/src/IdlApp/debug/
IdlApp」を起動する。

PCとPLCの通信が正常に行われ
れば、「WAIT_PROCESS」から
「ACTIVE_PROCESS」に移行する。

いつまで経っても移行しない場合は
PLC側に読み込ませたEDSファイル
の内容と、PC側のプログラムの内容
が一致しない可能性あり



```
112 {
113     appl_Uint16_12++;
114 }
115 /*-----
116 ** Callback of type ABCC_SetAdiValueFuncType. The function will be called when
117 ** the network writes to ADI#10. It copies the changed values from ADI#10 to ADI#11.
118 **
119 ** ABCC_SetAdiValueFuncType is declared in abcc_ad_if.h
120 **-----
121 */
122 static void SetAdi10Value( const struct AD_AdiEntry* psAdiEntry, UINT8 bNumElements, UINT8 bStartIndex )
123 {
124     UINT8 index;
125     for( index = bStartIndex ; index < bStartIndex+bNumElements ; index++ )
126     {
127         appl_aiUint16_11[ index ] = appl_aiUint16_10[ index ];
128     }
129 }
130 /*-----
131 ** Public Services
132 **-----
133 */
134 UINT16 APPL_GetNumAdi( void )
135 {
136     return( sizeof( APPL_asAdiEntryList ) / sizeof( AD_AdiEntry ) );
137 }
138 /*-----
139 ** Tasks
140 **-----
141 */
142 #endif
143
```

「IdlLinux_amd64/src/IdlApp/example_app/appl_adimap_separate16.c」の
127行目にある配列がデータの
入出力に使用している配列

APPL_GetNumAdi(void) Ln 137, Col 1 Spaces: 3 UTF-8 CRLF C Win32

Step3.1 PLCとPCの接続確認

「IdlLinux_amd64/src/IdlApp/debug/IdlApp」を起動する。

PCとPLCの通信が正常に行われれば、「WAIT_PROCESS」から「PROCESS_ACTIVE」に移行する。

いつまで経っても移行しない場合はPLC側に読み込ませたEDSファイルの内容と、PC側のプログラムの内容が一致しない可能性あり

WAIT_PROCESS

```
nishidalab@nishidalab-desktop: ~/apps/IdlLinux_amd64/src/IdlApp/debug$ ./IdlApp

Network type:      EtherNet/IP(TM)

Firmware version: 1.38 build 2
Network type 9b
Serial number:     A0:3C:43:50

ANB_STATUS: ABP_ANB_STATE_NW_INIT
ANB_STATUS: ABP_ANB_STATE_WAIT_PROCESS
```

PROCESS_ACTIVE

```
nishidalab@nishidalab-desktop: ~/apps/IdlLinux_amd64/src/IdlApp/debug$ ./IdlApp

Network type:      EtherNet/IP(TM)

Firmware version: 1.38 build 2
Network type 9b
Serial number:     A0:3C:43:50

ANB_STATUS: ABP_ANB_STATE_NW_INIT
ANB_STATUS: ABP_ANB_STATE_WAIT_PROCESS
ANB_STATUS: ABP_ANB_STATE_PROCESS_ACTIVE
```