

# f-palette-CANopen ソフトウェアマニュアル

**Release4.0**

**2011/12/05**

| 変更日付       | バージョン      | 変更内容                        | 変更者   |
|------------|------------|-----------------------------|-------|
| 2011/12/05 | Release4.0 | ソフトウェアをオープンソースで公開するに当たり、微修正 | 清水 正晴 |
|            |            |                             |       |
|            |            |                             |       |
|            |            |                             |       |

## 目次

|     |                            |    |
|-----|----------------------------|----|
| 1.  | 概要.....                    | 4  |
| 1.1 | 開発のベース.....                | 4  |
| 1.2 | 開発のターゲット.....              | 4  |
| 2.  | ソフトウェア構成と内容.....           | 5  |
| 2.1 | Driver 層.....              | 5  |
| 2.2 | Application 層.....         | 5  |
| 3.  | フォルダ構成.....                | 6  |
| 4.  | サンプルコードの説明.....            | 7  |
| 4.1 | Node ID, ビットレート.....       | 7  |
| 4.2 | I/O ポートの割り当て.....          | 7  |
| 4.3 | アナログ入力の値について.....          | 9  |
| 4.4 | ペリフェラルの割り当て.....           | 9  |
| 4.5 | DS401 の機能.....             | 9  |
| 5.  | Object Dictionary.....     | 10 |
| 5.1 | デフォルトの PDO マッピング.....      | 10 |
| 5.2 | トランスミット PDO パラメータについて..... | 11 |
| 5.3 | SDO による読み書き.....           | 13 |
| 5.4 | Object Dictionary の編集..... | 13 |
| 5.5 | TxPDO マッピングの変更時の追加事項.....  | 13 |
| 5.6 | objdictgen の補足説明.....      | 14 |
| 6.  | TI のコンパイラに固有の問題と対応.....    | 15 |
| 6.1 | 原因.....                    | 15 |
| 6.2 | 問題症状.....                  | 15 |
| 6.3 | 参考にした情報.....               | 15 |
| 6.4 | 対策方法.....                  | 16 |
| 6.5 | その他報告が必要と思われるバグ.....       | 17 |
| 7.  | CCS4 セットアップの手順.....        | 18 |

## 1. 概要

Open Source の公開を前提とした、f-palette 用の CANopen ソフトウェア f-palette-CANopen について説明する。

### 1.1 開発のベース

CANopen フレームワークには OpenSource の CAN Festival-v3.0 を使用した。

DS301 の機能はこれにより実装される。

CAN Festival のマニュアル(manual\_en.pdf)をあわせて参照のこと。

### 1.2 開発のターゲット

開発環境&コンパイラ : Texas Instruments Code Composer Studio V4.2.3

マイコン(DSP) : TMS320F28035

ターゲットボード : f-palette (製作 fuRo)

CPU ボード+IO ベースボード

## 2. ソフトウェア構成と内容

### 2.1 Driver 層

マイコン(DSP)のペリフェラルへアクセスのためのドライバ

can\_TI\_F2803x.c : CAN ペリフェラルへのアクセスのためのドライバ

timer\_TI\_F2803x.c : タイマ機能のためのドライバ

### 2.2 Application 層

開発ターゲットが動くためのサンプルのコード。

CANopen salve の DS401 として動作可能。

example フォルダ内にある。

### 3. フォルダ構成

#### 【src】

objacces\_path.patch :Objacces.c へのパッチファイル  
CANfestival-ML より入手、後術。  
objacces.c :上記パッチ適用済みコード

#### 【include→TI\_F2803x】

can\_TI\_F2803x.h  
timerscfg.h  
canfestival.h  
error.h  
config.h

#### 【driver→TI\_F2803x】

can\_TI\_F2803x.c  
timer\_TI\_F2803x.c

#### 【examples】

main.c :main 文、ds401 のためのドライバも含む  
ds401.c, ds401.h : AVR のものを参考に、主にアナログインプット部を追加  
ObjDict.c, ObjDict.h :サンプルに用意した、DS401 を含む ObjectDictionary  
ObjDict.od : objdictgen を使用して上記 ObjDict.c, ObjDict.h を生成するソース  
ObjDict.eds : サンプルの EDS ファイル、objdictgen より生成される

F28035\_CanFestival.cmd :本ソフトウェア用にカスタマイズされたリンカーファイル

DSP2803x\_headers と DSP2803x\_common の2つのフォルダは、

TI より下記の名前で提供される

SPRC892 : 2803x C/C++ヘッダー・ファイルとペリフェラルの例 v1.21

## 4. サンプルコードの説明

### 4.1 Node ID, ビットレート

main.c の define にて設定

```
#define      NODE_ID      0x05
#define      CAN_BAUDRATE  500      /* kbps */
```

セット可能なビットレートは、10, 20, 50, 100, 125, 250, 500, 1000kbps

(can\_TI\_F2803x.c の canInit を参照)

### 4.2 I/O ポートの割り当て

f-palette に対して、DS401 の機能を表のように割り当てた。

これらは、main.c 内の以下の関数でポートの割り当ての変更が可能である。

```
デジタル入力 : 8ch   UNS8 get_inputs()
デジタル出力 : 8ch   void set_outputs(UNS8 set_data)
アナログ入力 : 8ch   void read_ADCinput(Uint16 setdata[])
```

| アナログ<br>ポート | f-palette<br>コネクタ | DS401<br>Analog Input |
|-------------|-------------------|-----------------------|
| ADCNA0      | CN2_16            | Analog Input 01       |
| ADCNA1      | CN2_14            | Analog Input 02       |
| ADCNA2      | CN2_12            | Analog Input 03       |
| ADCNA3      | CN2_10            | Analog Input 04       |
| ADCNA4      | CN2_8             | Analog Input 05       |
| ADCNA5      | CN2_6             | Analog Input 06       |
| ADCNA6      | CN2_4             | Analog Input 07       |
| ADCNA7      | CN2_2             | Analog Input 08       |

| No | ペリフェラル | f-palette<br>コネクタ | 備考   | In/Out | DS401<br>D_Out | DS401<br>D_In |
|----|--------|-------------------|------|--------|----------------|---------------|
| 0  | GPI00  | CN1_18            |      | In     |                | Bit0          |
| 1  | GPI01  | CN2_13            | LED1 | Out    | Bit0           |               |
| 2  | GPI02  | CN1_20            |      | In     |                | Bit1          |
| 3  | GPI03  | CN2_11            | LED2 | Out    | Bit1           |               |
| 4  | GPI04  | CN1_22            |      | In     |                | Bit2          |
| 5  | GPI05  | CN2_9             | LED4 | Out    | Bit3           |               |
| 6  | GPI06  | CN1_24            |      | In     |                | Bit3          |
| 7  | GPI07  | CN2_7             | LED3 | Out    | Bit2           |               |
| 8  | GPI08  | CN1_26            |      | In     |                | Bit4          |
| 9  | GPI09  | CN2_5             | GPI0 | Out    | Bit6           |               |
| 10 | GPI010 | CN1_28            |      | In     |                | Bit5          |
| 11 | GPI011 | CN2_3             | GPI0 | Out    | Bit7           |               |
| 40 | GPI040 | CN1_30            | GPI0 | Out    | Bit4           |               |
| 41 | GPI041 | CN2_1             | GPI0 | Out    | Bit5           |               |
| 42 | GPI042 | CN2_18            |      | In     |                | Bit6          |
| 44 | GPI044 | CN2_17            |      | In     |                | Bit7          |



### 4.3 アナログ入力の値について

ADCの変換は、f-palette AIN コネクタ入力部 0~5V (DSP 入力部で 0~3.3V) の 12bit である。8bit の OD 0x6400 に当てはめる際には 4bit 下位にシフトしている。(上位 8bit のみ使用) 16bit の OD 0x6401 に当てはめる際には 4bit 上位にシフトしている。DS401 では OD 0x6401 は integer16 型だが、unsigned int 型として 16bit フルレンジを使用している。

### 4.4 ペリフェラルの割り当て

CAN0 : can\_TI\_F2803x.c にて使用

割り込み : 受信割り込み ECAN0INTA を使用

Timr1 : timer\_TI\_F2803x.c にて使用

割り込み : timer period 割り込み、TINT1 を使用

Timer0 : main.c にて使用

割り込み : timer period 割り込み、TINT0 を使用

デジタル I/O や ADC の定期呼出に使用。

サンプルでは 1ms 周期に割り込みが発生。

Drive、main 文どちらも、DSP/BIOS を使用していない。

### 4.5 DS401 の機能

Timer0 にて 1ms の周期割り込み時に DS401 の機能のタスクを実行している。

CANFestival のフレームワークが持っている Object Dictionary、CO\_Data へのアクセスは、ds401.c にて行っている。

サンプルの ds401.c は、公開されている AVR マイコンのものに、アナログ部を追加したものである。

ds401.c のハンドラでは、PDO 送信のためのトリガ処理も行っている。(後術)

## 5. Object Dictionary

### 5.1 デフォルトの PDO マッピング

サンプルの初期の PDO マッピングは次のようになっている。  
 ObjDict にて設定されていて、objdictgen にて編集が可能

| 通信方向   |                   | ID           | データ長     |      |  |
|--------|-------------------|--------------|----------|------|--|
| RPDO 1 |                   | 0x200+NodeID | 1Byte    |      |  |
| Byte   | データ               | Value        | 型        | 備考   |  |
| 0      | Digital Output 01 | 0x6200-01-08 | Integer8 | 8bit |  |

| 通信方向   |                  | ID           | データ長     |      |  |
|--------|------------------|--------------|----------|------|--|
| TPDO 1 |                  | 0x180+NodeID | 1Byte    |      |  |
| Byte   | データ              | Value        | 型        | 備考   |  |
| 0      | Digital Input 01 | 0x6000-01-08 | Integer8 | 8bit |  |

| 通信方向   |                 | ID           | データ長      |    |  |
|--------|-----------------|--------------|-----------|----|--|
| TPDO 2 |                 | 0x280+NodeID | 8yte      |    |  |
| Byte   | データ             | Value        | 型         | 備考 |  |
| 0, 1   | Analog Input 01 | 0x6401-01-10 | Integer16 |    |  |
| 2, 3   | Analog Input 02 | 0x6401-02-10 | Integer16 |    |  |
| 4, 5   | Analog Input 03 | 0x6401-03-10 | Integer16 |    |  |
| 6, 7   | Analog Input 04 | 0x6401-04-10 | Integer16 |    |  |

| 通信方向   |                 | ID           | データ長      |    |  |
|--------|-----------------|--------------|-----------|----|--|
| TPDO 3 |                 | 0x380+NodeID | 8yte      |    |  |
| Byte   | データ             | Value        | 型         | 備考 |  |
| 0, 1   | Analog Input 05 | 0x6401-05-10 | Integer16 |    |  |
| 2, 3   | Analog Input 06 | 0x6401-06-10 | Integer16 |    |  |
| 4, 5   | Analog Input 07 | 0x6401-07-10 | Integer16 |    |  |

|     |                 |              |           |  |
|-----|-----------------|--------------|-----------|--|
| 6,7 | Analog Input 08 | 0x6401-08-10 | Integer16 |  |
|-----|-----------------|--------------|-----------|--|

| 通信方向   |                 | ID           | データ長     |    |
|--------|-----------------|--------------|----------|----|
| TPDO 4 |                 | 0x480+NodeID | 8byte    |    |
| Byte   | データ             | Value        | 型        | 備考 |
| 0      | Analog Input 01 | 0x6400-01-08 | Integer8 |    |
| 1      | Analog Input 02 | 0x6400-02-08 | Integer8 |    |
| 2      | Analog Input 03 | 0x6400-03-08 | Integer8 |    |
| 3      | Analog Input 04 | 0x6400-04-08 | Integer8 |    |
| 4      | Analog Input 05 | 0x6400-05-08 | Integer8 |    |
| 5      | Analog Input 06 | 0x6400-06-08 | Integer8 |    |
| 6      | Analog Input 07 | 0x6400-07-08 | Integer8 |    |
| 7      | Analog Input 08 | 0x6400-08-08 | Integer8 |    |

## 5.2 トランスミット PDO パラメータについて

CANFestival のフレームワークにより、トランスミット PDO パラメータが有効である。

0x18xx-sub2 トランスミットタイプ

0x18xx-sub3 インヒビットタイム 単位 0.1ms

0x18xx-sub5 イベントタイマー 単位 1ms

トランスミットタイプを 0xFF (255) とせていることで、インヒビットタイムとイベントタイマーのタイミングによる PDO 送信となる。

マップされた PDO に変更がない場合、イベントタイマーの周期で送信される。

マップされた PDO に変更がある場合、インヒビットタイム以上の時間間隔を確保して送信される。

サンプルのプログラムでは、1ms 周期でデジタルやアナログ値を更新している。

インヒビットタイムに 0 をすると、「マップされた PDO に変更があったら、ただちに送信する」となる。アナログインプットのオブジェクトを PDO マッピングしている TxPDO でこの

設定を行うと、バス負荷が高くなるので注意が必要である。

### 5.3 SDO による読み書き

ターゲットに実装された状態で、Object Dictionary の値は、SDO を通じて読み書きできるが、保存はされない。

0x1000~0x1FFF のオブジェクトは、CANfestival のフレームワーク本体に依存する。  
CAN Festival のマニュアル(manual\_en.pdf)をあわせて参照のこと。

### 5.4 Object Dictionary の編集

CANFestival 付属の objdictgen を使用することで、Object Dictionary を編集することができる。

ハートビートの設定も Object Dictionary の編集(value の変更)で可能。

ObjDict.od : objdictgen を使用して上記 ObjDict.c, ObjDict.h を生成するソース  
ObjDict.c, ObjDict.h : サンプルに用意した、DS401 を含む ObjectDictionary  
ObjDict.eds : サンプルの EDS ファイル、objdictgen より生成される

ここで示した ObjDict.xx ファイル名は任意に変更可能。

編集にあたっては、現在の ObjDict.od を読み込み、それをベースにするのがよい。

### 5.5 TxPDO マッピングの変更時の追加事項

ds401.c では、PDO 送信のためのトリガ処理を行っているため、TxPDO マッピングを変更する際には、これも適切に変更する必要がある。

PDO 送信のためのトリガは、

```
d->PDO_status[x].last_message.cob_id = 0;
```

の形であり、x は 0 オリジンの TxPDO ナンバである。

## 5.6 objdictgen の補足説明

Objdictgenについては、CAN Festival のマニュアル(manual\_en.pdf)の  
P.28 9.1) Using Dictionary Editor GUI を参照のこと。

Windows 環境下での使用には、Python、wxPython をインストール後、Objdictgen フォルダ  
下に圧縮して添付してある gnosiss をインストールする必要がある。

GUI の起動には、objdictedit.py をダブルクリックすることで起動される。

## 6. TI のコンパイラに固有の問題と対応

### 6.1 原因

CANFestival では sizeof(int16) に対して、” 2” と返すことを期待しているが、TI のコンパイラでは sizeof(int16) を 1 と返してしまうことで問題が発生している。

### 6.2 問題症状

一部の 16bit のオブジェクトに対して、SD0 によるアクセスが適切にできない。

一部の 16bit のオブジェクトに対して、PDO マッピングが適切にできない。

一部のというのは、DS301 で規定されているオブジェクトに対しては、正常に動作するようである。

16bit のアナログ入力を使用しなければ、対策を行わなくても問題は発生せず動作可能だと思われる。

### 6.3 参考にした情報

CAN festival のメーリングリストに、TMS320F2812 ではあるが、解決例が示されていたので参考にした。objjaces\_path.patch のパッチはここで Luis Jiménez Gañán 氏より公開されているものをそのまま使用している。

参照元を抜粋して記す。

[http://sourceforge.net/mailarchive/forum.php?thread\\_name=4A03EC29.7020703%40sedecal.com&forum\\_name=canfestival-devel](http://sourceforge.net/mailarchive/forum.php?thread_name=4A03EC29.7020703%40sedecal.com&forum_name=canfestival-devel)

source forge - CanFestival - Mailing Lists

Email Archive : canfestival-devel

Title:uC driver, how about the TI' s tms320f2812 DSP

①- I changed the '.c' file generated by objdictgen python script to manually set "4" instead of "sizeof(UNS32)" and so on with INTEGER32, UNS16, INTEGER16, UNS8 and INTEGER8.

②- I made a patch to objjaces.c to avoid memcpy (the patch is in the file attached) when sending/receiving vars.

③- I didn't use the functions for local variable handle using index/subindex because of memcpy problems and for efficiency reasons.

④- The drivers attached are for TI DSP/BIOS rtos. CAN messages are sent through a DSP/BIOS MBX and there is another function that actually sends it and so on, but it

-

can be a starting point if DSP/BIOS is not used.

Luis Jiménez Gañán

- ①の内容は、次項の解決方法の「ObjDict.cの置換」に対応する。
- ②の内容を次項の解決方法の「objaccses.cのパッチの当て方」に具体的に示す。
- ③は関数ハンドラを用いたオブジェクトは使用しないようにした。
- ④は解決方法ではないが、今回 DSP/BIOS を用いていない、別の driver を用意した。

## 6.4 対策方法

問題解決には、2つの対策処理を両方行う必要がある。

ObjDict.cの置換

objaccses.cにパッチを当てる

### 【ObjDict.cの置換】

ObjDict.cに以下の置換を実行する。

多少ならば汎用のテキストエディタで十分だと思われる。

| 置換元               | 置換後 |
|-------------------|-----|
| sizeof(INTEGER16) | 2   |
| sizeof(INTEGER32) | 4   |
| sizeof(UNS16)     | 2   |
| sizeof(UNS32)     | 4   |

### 【objaccses.cのパッチの当て方】

obcaccses\_path.patchをsrcフォルダに置く

obcaccses.cとobcaccses\_path.patchが同じフォルダにあることを確認する。

Cygwin等(写真はMINGW)のコマンドラインでsrcフォルダに降りて行き

```
patch obcaccses.c < obcaccses_path.patch
```

を実行。以下のメッセージが出れば成功。

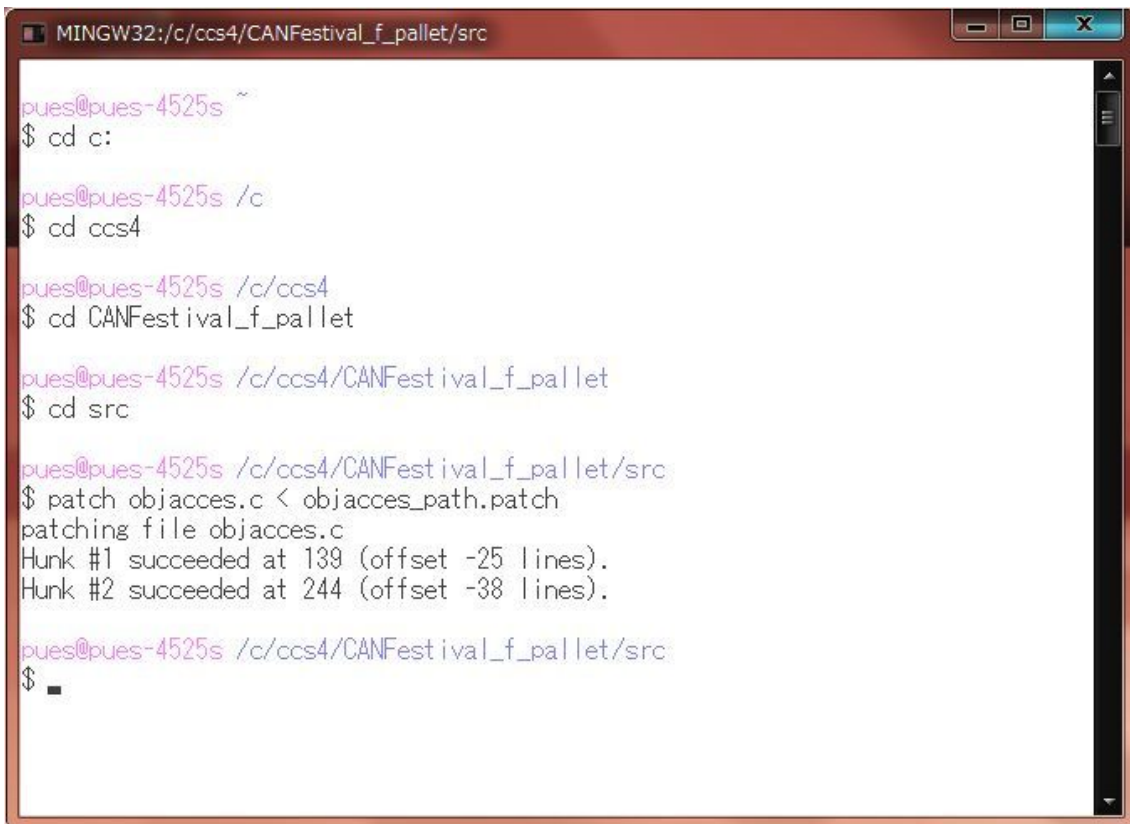
```
Patching file obcaccses.c
```

```
Hunk #1 succed at 139 (offset -25 lines).
```

```
Hunk #2 succed at 244 (offset -38 lines).
```

実行後、obcaccses.c.orgとobcaccses\_path.patchは不要となる。





```
MINGW32:/c/ccs4/CANFestival_f_pallet/src
pues@pues-4525s ~
$ cd c:

pues@pues-4525s /c
$ cd ccs4

pues@pues-4525s /c/ccs4
$ cd CANFestival_f_pallet

pues@pues-4525s /c/ccs4/CANFestival_f_pallet
$ cd src

pues@pues-4525s /c/ccs4/CANFestival_f_pallet/src
$ patch objaccs.c < objaccs_path.patch
patching file objaccs.c
Hunk #1 succeeded at 139 (offset -25 lines).
Hunk #2 succeeded at 244 (offset -38 lines).

pues@pues-4525s /c/ccs4/CANFestival_f_pallet/src
$
```

## 6.5 その他報告が必要と思われるバグ

src フォルダ内の timer.c の 39 行目に

```
s_timer_entry timers[MAX_NB_TIMER] = {TIMER_FREE, NULL, NULL, 0, 0, 0};
```

とあるが、これでは要素が 2 以上の配列を初期化できない。

初期化されていない場合、TPDO 用にタイマが使用できなくなる。

今回は、driver の timer\_TI\_F2803x.c の initTimer にて初期化を行っている。

## 7. CCS4 セットアップの手順

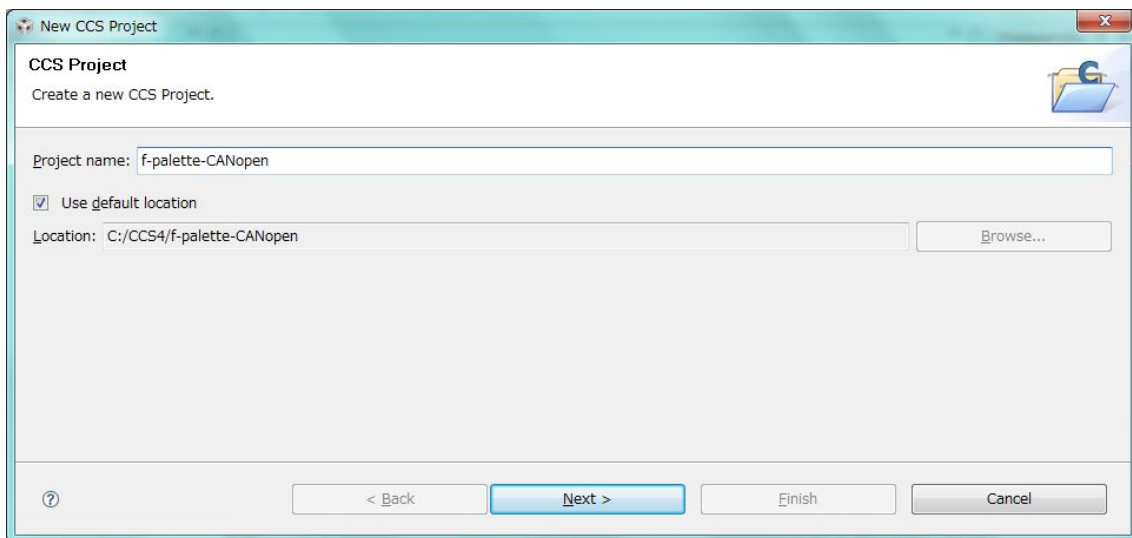
CCS4 のプロジェクトファイルがない状態からの、セットアップ方法を記述する。

File->new->CCS Project

Project name: 任意

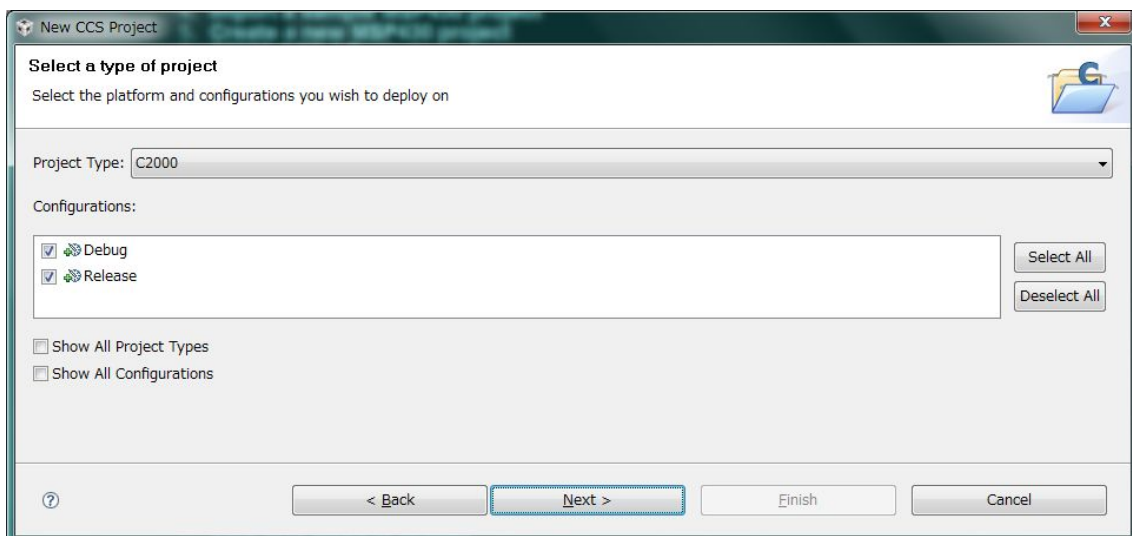
Location:work space の下にフォルダが作成される

next



Project type:C2000

next



Additional Project Setting:何もしない

next

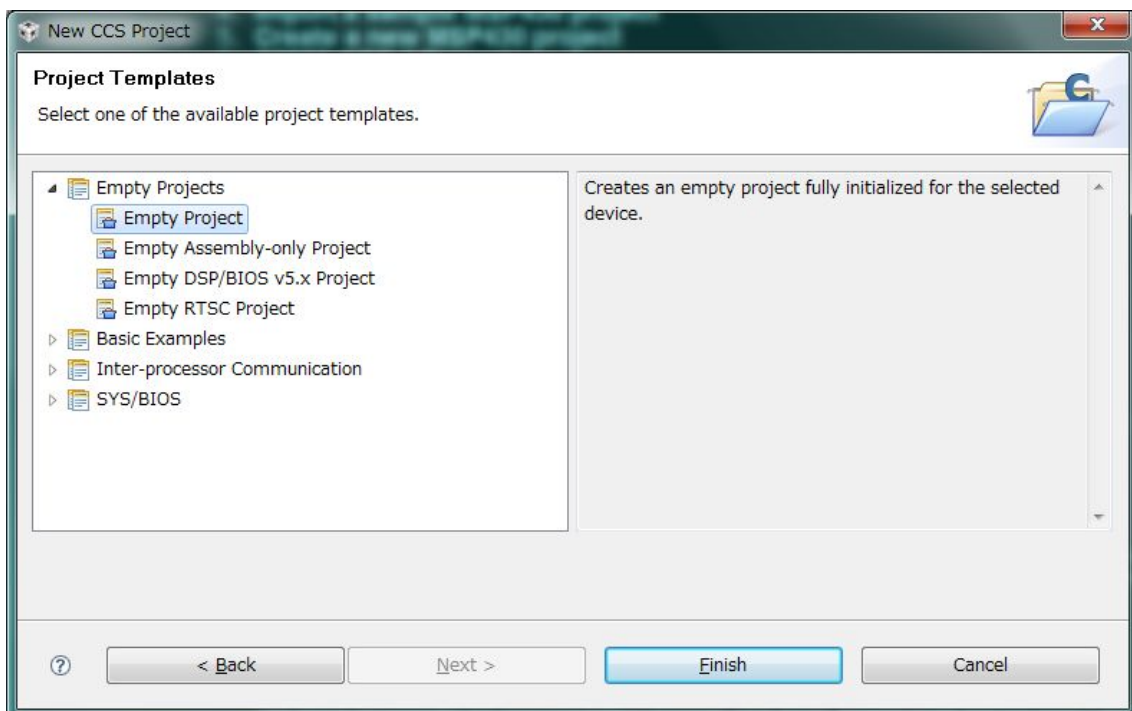
Device Variant: TMS320F28035

LinkerCommand Files: <none>(重要)

Next

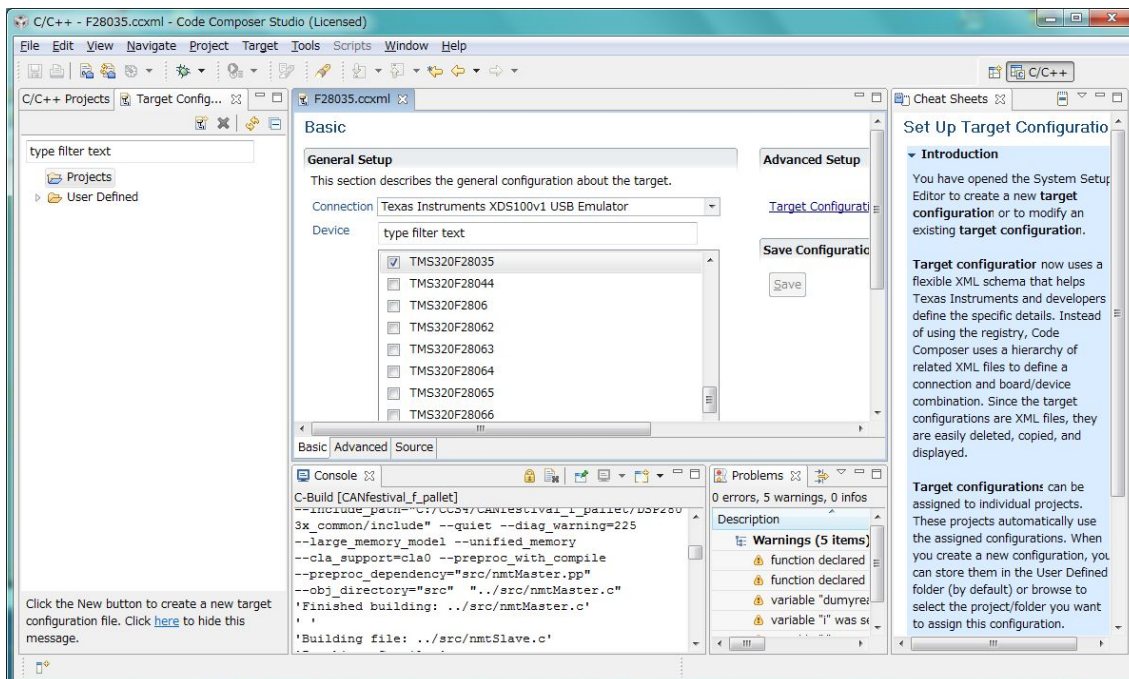
Project Templates:Empty Project

Finish



これでコンパイル・ビルドが実行可能になる。

ターゲットの設定は下記を参考に接続環境に合わせた設定をする。



View->C/C++Projects

Drag&Drop でファイルをプロジェクトに入れる  
(フォルダにそのファイルがコピーされる)

本件で開発したファイル

F28035\_CANFestival.cmd

drivers

include

examples->

ds401.c

ds401.h

ObjDict.c

ObjDict.h

main.c

TI のサンプル、SPRC892 : 2803x C/C++ヘッダー・ファイルとペリフェラルの例 より

DSP2803x\_common

DSP2803x\_headers

CanFestival3 より

Src

以下を右クリックで menu から下記をビルド対象から除外する。

Remove or Exclude File(s) from Build

src->lss.c

src->symbols.c

F28035\_RAM\_Ink.cmd

DSP2803x\_common->cmd

DSP2803x\_common->src

DSP2803x\_Comp.c

DSP2803x\_CSMPasswords.asm

DSP2803x\_DBGIER.asm

DSP2803x\_DisInt.asm

DSP2803x\_ECap.c

DSP2803x\_EPwm.c

DSP2803x\_EQep.c

DSP2803x\_Gpio.c

DSP2803x\_I2C.c

DSP2803x\_Lin.c

DSP2803x\_OscComp.c

DSP2803x\_Sci.c

DSP2803x\_Spi.c

DSP2803x\_SWPrioritizedDefaultIsr.c

DSP2803x\_SWPrioritizedPieVect.c

DSP2803x\_TempSensorConv.c

DSP2803x\_headers->cmd->DSP2803x\_Headers\_BIOS.cmd

include Pass の設定  
Project→Properties

C/C++ Buil

C2000 Compiler

Include Options

+から下記のフォルダを指定・追加 workspace...から選ぶと容易

```
"${workspace_loc:/CANfestival_f_pallet/examples}"
```

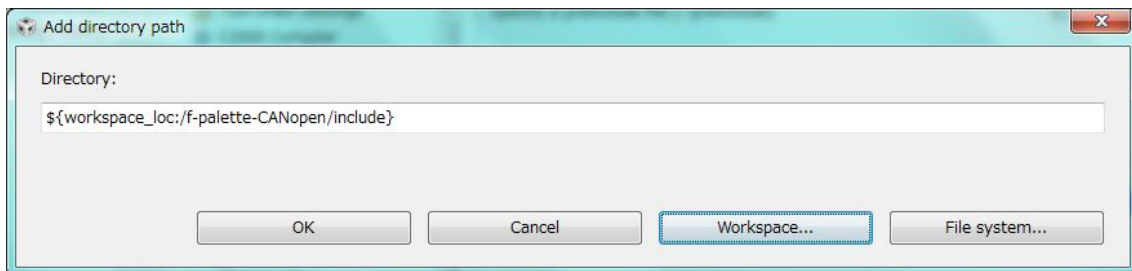
```
"${workspace_loc:/CANfestival_f_pallet/include}"
```

```
"${workspace_loc:/CANfestival_f_pallet/include/TI_F2803x}"
```

```
"${workspace_loc:/CANfestival_f_pallet/DSP2803x_headers/include}"
```

```
"${workspace_loc:/CANfestival_f_pallet/DSP2803x_common/include}"
```

workspace\_loc:/CANfestival\_f\_pallet を {PROJECT\_ROOT} としても可



C2000 linker

Basic Options→ Set C system stack size 0x3B0

File Serach Path 上段に"libc.a"があったら削除"

CCS Build

LinkerCommand Files: F28035\_CANFestival.cmd を設定。

うまくいかない場合は、<none>もしくは空欄を試す。

