タイトル: PCAN-Router FD CANFD_CAN変換

文書バージョン: 1.0 作成日: 2022年8月2日



メーカ: PEAK-System 製品: PCAN-Router FD OS: Windows 10,11 **ガイロジック株式会社** 〒180-0004 東京都武蔵野市吉祥寺本町2-5-11 松栄ビル5F Tel 0422-26-8211 Fax 0422-26-8212 http://WWW.gailogic.co.jp

1 はじめに

本アプリケーションノートでは、PCAN-Router FD を使って CANFD メッセージを CAN メッセージにして転送する方法 についてについて説明します。

PCAN-Router FD のサンプルには CAN から CANFD の変換は含まれていますが(サンプル名: 10_CAN_FD)、その逆は 含まれていません。

ここでは、PEAK-Systemの Forum にあるサンプルを使った方法を説明します。

CAN では 1 フレームのデータ長は最大 8byte ですが、CAN FD では最大 64byte に対応しています。そのため、8byte 以上のデータを持つ CAN FD メッセージを CAN メッセージに変換する場合、単純にプロトコルを変換するだけでは全てのデータを変換することができません。

本サンプルでは、CAN FD メッセージを複数の CAN メッセージに分割することでそれを解決しています。8byte 毎に1つの CAN メッセージに分割され、CAN ID は最初の CAN ID に 0x001 ずつインクリメントされます。

例: 変換前 CAN FD メッセージ

CANID: 0x500 データ: 0x00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 変換後 CAN メッセージ CANID: 0x500

データ:0x00 01 02 03 04 05 06 07 CANID:0x501 データ:0x08 09 0A 0B 0C 0D 0E 0F

2 準備

下記を準備します。

- PCAN-Router FD
- PEAK-Flash (PEAK-System 製品のフラッシュソフトウェア)
- Microsoft Visual Studio Code
- PCAN インターフェイス(例. PCAN-USB)
- 電源 (DC 8~26 V)
- CAN ケーブル (終端抵抗付)

2-1 開発環境の準備

下記の日本語資料および PCAN-Router FD マニュアルを参考に、開発環境を準備して下さい。

- ・PCAN-Router FD 日本語資料: https://www.gailogic.co.jp/pdf/AN_Download_PCAN-Router(_FD) jpdoc_2_0.pdf
- ・PCAN-Router FD 日本語マニュアル: https://www.gailogic.co.jp/pdf/PCAN-Router_FD_211_JP_v_2_0.pdf

3 ファームウェア開発

CAN FD メッセージを CAN メッセージにするためのファームウェアを作成します。 開発には 2-1 開発環境の準備 にてインストールした Microsoft Visual Studio Code を使用します。

3-1 プログラミング

CANFD メッセージを CAN メッセージにするコードをプログラミングします。

- 以下の PEAK-System Forum ページから、サンプルをダウンロードします。 ファイル名: main.c <u>https://forum.peak-system.com/viewtopic.php?f=199&t=5627</u>
- ダウンロードした main.c を以下のように編集します。
 2 行目 #include "datatypes.h" -> #include <stdint.h>
- 3. 編集した main.c を保存します。
- PCAN-Router FD の適当なプログラミングサンプルを一つフォルダごとコピーして、名前を変えて複製します。
 (例: PEAK-DevPack¥Hardware¥PCAN-Router_FD¥Examples¥01_ROUTING)
- 5. 複製したフォルダ内の main.c を、先ほど編集したものに置き換えます。
- 6. Visual Studio Code を開き、ファイル -> 「フォルダーを開く」から 4.で作成したフォルダを開きます。
- 7. ターミナル -> タスクの実行から "Make All" を実行します。

表示(V) 移	動(G) 実行(R)	ターミナル(T)	ヘルプ(H)	main.c - 10_CAN_FD - Visual Studio Code	
C main.c	1 ×		実行するタスクの選択		
src > C n	nain.c >		Make All		最近使用 🐯 🗙
78		nw_set	Compile File		構成済み
79	, I I		Compile File ARM (ARM7 bas	ed hardware only)	
80		/ catch (Flash Device		
81		f (RxMsg.	Make Clean		
82	{		🖻 grunt		
83			🔁 gulp		
84		//	E∃ iake		
85		HW	P1 npm		
80		// if	P1 typescript		
87		11	Pi conbuild		
89			すべてのタスクの表示		
90			Outmisg.butttype = CAN	BUFFEK_IX_MSG;	
91			OutMsg.msgtype = CAN_	MSGTYPE_STANDARD;	
92			OutMsg.id = RxMsg.id;		
93			OutMsg.dlc = RxMsg.dl	c;	
94			OutMsg.data32[0] = Rx	Msg.data32[0];	

8. 「out」フォルダに.bin ファイルが作成されます。

4 接続

図 4-1 のように接続します。



図 4-1

5 ファームウェア書き込み

- 1. PEAK-FLASH を起動します。
- 2. Welcome 画面が表示されたら Next > をクリックします。
- "Modules connected to the CAN-Bus" を選択し、接続している CAN インターフェイスが表示されていることが確認 出来たら Detect をクリックします。

正常に PCAN-Router FD が検出されると図 4-1 のように表示されます。

DEAK-Flash 3/2022			84 <u>—</u>		×
1. Welcome	Select Hardware Please select the hardware to	o be flashed.			\$
2. Select Hardware	 Locally connected CAN/LIN- Modules connected to the C 	nterfaces AN-Bus			
3. Select Firmware	Channels of connected CAN har PCAN-USB Pro FD, Device ID: 1	dware 1712h, Channel: 1			v
4. Ready to Flash	Bit rate: Nominal Bit Rate: 500	kbit/s. Data Bit Rate: 2 Mbi	t/s		
5. Flashing	Detect				
6. Finish	Hardware name	Module ID	Firmware ver	sion	-
	PCAN-Router FD	15	1.2		
AN					
© 2022 PEAK-System Techn	ik GmbH	< Back	Next >	Can	cel
	図 5-1				

4. Next> をクリックすると、Select Firmware の画面が開きます。

Firmware File: を選択し、Browse... をクリックして3章で作成した .bin ファイルを開きます。

0 PEAK-Flash 3/2022		_	
1. Welcome	Select Firmware Please select a firmware file.		*
2. Select Hardware	Selected hardware: PCAN-Router FD Current version: 1.2		
3. Select Firmware	Embedded Firmware		
4. Ready to Flash	Release Date:		
5. Flashing	Firmware File: example_routing.bin		Browse
6. Finish	Version: unknown File date: 2022/07/21 Compatibility: ✔		
© 2022 PEAK-System Technik	GmbH < Back Next >		Cancel
	図 5-2		

5. Next> をクリックすると、 Ready to Flash の画面に進みます。

Start をクリックしてファームウェアを PCAN-Router FD に書き込みます。

🙆 PEAK-Flash 3/2022	- 🗆 X
1. Welcome	Ready to Flash Review your selected options.
2. Select Hardware	Selected hardware: PCAN-Router FD
3. Select Firmware	Firmware File: example_routing.bin
4. Ready to Flash	Current version: 1.2
5. Flashing	New firmware version unknown
6. Finish	
	Please do not disconnect or power down the device while flashing, to prevent possible damage to the hardware.
© 2022 PEAK-System Technik G	mbH < Back Start Cancel

🗵 5-3

6. Progress: バーが 100%になり、ログに "Please disconnect and reconnect the device" と表示されたら書き込み完了 です。

Next> をクリックし、Exit で PEAK-Flash を終了します。

🙆 PEAK-Flash 3/2022		_		×
1. Welcome	Flashing Please wait for the flash process to finish.			*
2. Select Hardware	Time Description			
3. Select Firmware	14:11:48 Beginning flash: PCAN-Router FD 14:11:49 Erasing 14:11:50 Flashing new firmware 14:11:50 Done. Duration: 2.0 s			
4. Ready to Flash	14:11:50 Please disconnect and reconnect the device.			
5. Flashing				
6. Finish				
Printer 1	Reset Module Reset All Mo	dules on th	e Bus	
6/14/4	Progress:		1	00%
© 2022 PEAK-System Technik	mbH < Back Next	>	Cance	el
	図 5-4			

7. PCAN-Router FD の電源を切ります。

その後、Boot ピンを H レベルにせず電源を投入することで CANFD -> CAN の動作が開始します。

6 動作確認

正常にファームウェアが書き込めていた場合以下の様に動作します。

・CAN1ch に入力される CAN FD メッセージ

	CAN-ID^	Туре	Length	Data		Cycle Ti	Count	Trigger	Comment
	500h	FO	32	00 01 02 03 04 08 09 0A 0B 00 10 11 12 13 14 18 19 1A 1B 10	4 05 06 07 C 0D 0E 0F 4 15 16 17 C 1D 1E 1F	300	23	Time	
Transmit									
				a	D'I			- 01	
V	Connected to hardware PCAN-USB Pro FD, Channel T 😽 Bit rate: 500 kbit/s / 4 Mibit/s Status: OK								
	図 6-1								

・CAN 2ch から出力される CAN メッセージ

	Receive / Transmit	🚥 Trace	et PCA	AN-USB Pro FD 🛛 💀 Bus Load 🛛 🛕 Error (Generator	
	CAN-ID ^	Туре	Length	Data	Cycle Time	Count
	500h		8	00 01 02 03 04 05 06 07	300.7	7
	501h		8	08 09 0A 0B 0C 0D 0E 0F	300.7	7
	502h		8	10 11 12 13 14 15 16 17	300.7	7
ø	503h		8	18 19 1A 1B 1C 1D 1E 1F	300.7	7
.ei						
Še						
-						

図 6-2

付録 A 編集後サンプルコード

```
#include <stdint.h>
#include "can.h"
#include "can_user.h"
#include "hardware.h"
// This example shows how to conver/split a CAN FD frame into multiple
// CAn2.0 franmes with ascending CAN-IDs
// identifier is needed by PCANFlash.exe -> do not delete
const char Ident[] __attribute__ ((used)) = { "PCAN-Router_FD"};
// variables for LED toggle
static uint8_t LED_toggleCAN1;
static uint8_t LED_toggleCAN2;
//Vriables need for conversion into CAN2.0
static int i;
int MessageLength;
uint32_t DLCValues[16] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 20, 24, 32, 48, 64};
// main_greeting()
// transmit a message at module start
static void main_greeting (void)
{
              CANTxMsg_t Msg;
Msg.bufftype = CAN_BUFFER_TX_MSG;
Msg.dlc = CAN_LEN8_DLC;
              Msg.msgtype = CAN_MSGTYPE_STANDARD;
              Msg.id
                            = 0x050;
              Msg.data32[0] = 0x67452301;
Msg.data32[1] = 0xEFCDAB89;
              // Send Msg
CAN_Write ( CAN_BUS1, &Msg);
}
// main()
// entry point from cr_startup_lpc40xx.c
int main (void)
{
              // init hardware and timer 0. Timer 0 is free running
              // with 1 us resolution without any IRQ.
              HW Init();
              // init CAN
              CAN_UserInit();
              // set green LEDs for CAN1 and CAN2
              HW_SetLED ( HW_LED_CAN1, HW_LED_GREEN);
HW_SetLED ( HW_LED_CAN2, HW_LED_GREEN);
              // send the greeting message
              // main_greeting();
              // main loop
              while (1)
              {
                            CANRxMsg t RxMsg;
                            CANTxMsg_t OutMsg;
                            // process messages from CAN1
                            if (CAN_UserRead (CAN_BUS1, &RxMsg) == CAN_ERR_OK)
                            {
                                          // message received from CAN1
LED_toggleCAN1 ^= 1;
                                          if (LED_toggleCAN1)
                                          {
                                                        HW_SetLED (HW_LED_CAN1, HW_LED_ORANGE);
                                          }
                                          else
                                          {
                                                        HW_SetLED ( HW_LED_CAN1, HW_LED_GREEN);
                                          }
                                          // catch CAN FD and convert it to CAN 2.0
                                          if (RxMsg.msgtype == CAN_MSGTYPE_FDF)
                                          {
```

```
//Now we have a real FD Frame - set the LED to RED to show it!
                                                                               HW_SetLED ( HW_LED_CAN1, HW_LED_RED);
                                                                               //If the message is 8 bytes long or less, it will only be converted.
                                                                               if(RxMsg.dlc <= 8)
                                                                               {
                                                                                               //Send Msg 1:1
OutMsg.bufftype = CAN_BUFFER_TX_MSG;
OutMsg.msgtype = CAN_MSGTYPE_STANDARD;
OutMsg.id = RxMsg.id;
                                                                                                OutMsg.dlc = RxMsg.dlc;
                                                                                               OutMsg.data32[0] = RxMsg.data32[0];
OutMsg.data32[1] = RxMsg.data32[1];
CAN_Write (CAN_BUS2, &OutMsg);
                                                                               }
                                                                               else
                                                                               {
                                                                                               //if it exceeds 8 bytes it will divided by 8 and split into multiple messages
MessageLength = DLCValues[RxMsg.dlc];
                                                                                                for(i = 0; i < (MessageLength/8); i++)
                                                                                                {
                                                                                                                OutMsg.bufftype = CAN_BUFFER_TX_MSG;
OutMsg.msgtype = CAN_MSGTYPE_STANDARD;
OutMsg.id = RxMsg.id + i;
                                                                                                                OutMsg.dlc = 8;
                                                                                                                OutMsg.data32[0] = RxMsg.data32[(i*2)];
                                                                                                                OutMsg.data32[1] = RxMsg.data32[(i*2+1)];
CAN_Write ( CAN_BUS2, &OutMsg);
                                                                                                // If the devision by 8 does leave a remainder, the message will be split
                                                                                               // into multiple messages, where one messages is shorter than 8 bytes, so
all data is
                                                                                                // transmitted properly.
                                                                                                if(MessageLength%8)
                                                                                                {
                                                                                                                OutMsg.bufftype = CAN_BUFFER_TX_MSG;
OutMsg.msgtype = CAN_MSGTYPE_STANDARD;
OutMsg.id = RxMsg.id + i;
                                                                                                                OutMsg.dlc = MessageLength%8;
                                                                                                                OutMsg.data32[0] = RxMsg.data32[(i)*2];
                                                                                                                OutMsg.data32[1] = RxMsg.data32[(i)*2+1];
                                                                                                                CAN_Write (CAN_BUS2, &OutMsg);
                                                                                               }
                                                                               }
                                               }
                               }
                                // process messages from CAN2
                               if ( CAN_UserRead ( CAN_BUS2, &RxMsg) == CAN_ERR_OK)
                               {
                                               // message received from CAN2
LED_toggleCAN2 ^= 1;
                                                if (LED_toggleCAN2)
                                                {
                                                                HW_SetLED ( HW_LED_CAN2, HW_LED_ORANGE);
                                               }
                                                else
                                                {
                                                                HW_SetLED ( HW_LED_CAN2, HW_LED_GREEN);
                                                // forward message to CAN1
                                                CAN_Write ( CAN_BUS1, &RxMsg);
                               }
               }
```

}