fvIO SPI 汎用プラグイン
アプリケーションノート

Rev1.00

2019 年 07 月 23 日

シマフジ電機（株）

変　更　履　歴　表

版	変　更　内　容	変更日付
1.00	初版	2019/07/23

目次

1. はじめに

本書は、fvIO SPI 汎用プラグイン(以下プラグイン)のサンプルプログラムのアプリケーションノートである。プラグインはコネクタ 1ch 分を占有する 1 スロット版と、コネクタ 2ch 分を占有する 2 スロット版を使用する。本サンプルプログラムは、e2studio のバージョン 7.3.0 で開発している。サンプルプログラムは、以下のように RAM 実行版と SPI ブート版の 2 種類を開発している。2 種類のサンプルプログラムは同様の動作を行う。

■サンプルプログラムのプロジェクト

プロジェクト名	プロジェクト説明
RZ_T_ram_sample	RAM 実行版。 e2studio からマイコンの RAM へデータをロードして処理を実行する。
RZ_T_sflash_sample	SPI ブート版。 マイコンボードの FLASH へデータを書き込み、リセット時にマイコンへデータをロードして処理を実行する。

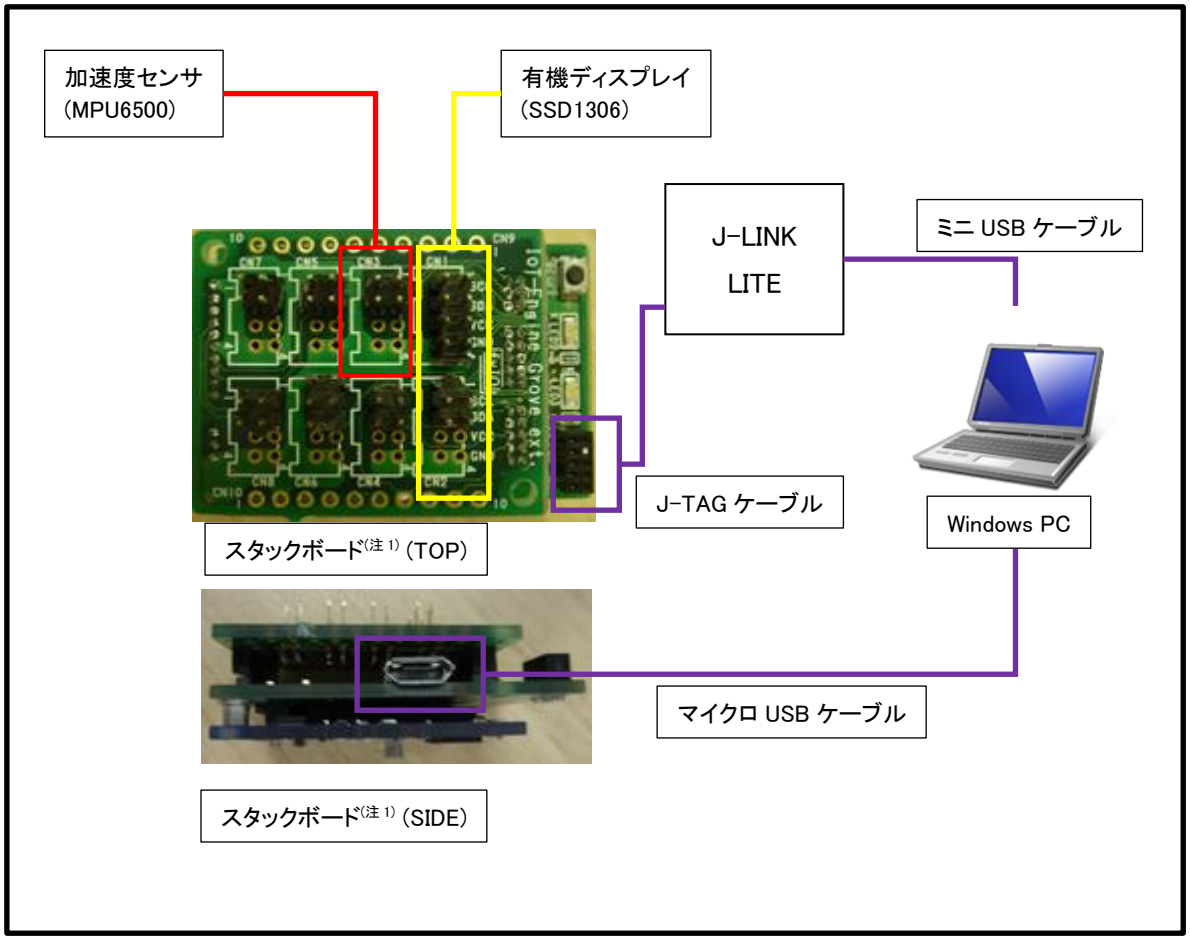
2. 動作環境

本サンプルプログラムが動作する CPU と対応するセンサは以下の通り。

項目	種類	品名等	備考
CPU	ルネサス製マイコン	RZ/T1	
センサ	加速度センサ	MPU6500	汎用の fvIO SPI(1 スロット版)で制御
	有機ＥＬディスプレイ	SSD1306(制御チップ型名)	汎用の fvIO SPI(2 スロット版)で制御

3. 環境接続図

e2studio から本サンプルプログラムの動作確認、プログラムの書き込みを行う時の接続例を以下に示す。



(注 1)スタックボードは以下のボードを組み合わせている。

- シマフジ電機製 SEMB1401 ボード
- シマフジ電機製 SEMB1311-1 ボード
- シマフジ電機製 Iot-Engine Sensor Ext Board

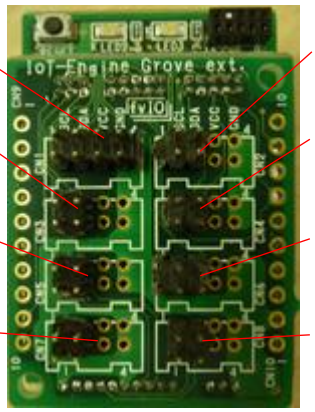
4.　fvIO インタフェース詳細

4.1　端子割り当て

端子割り当ては以下の通り。

CN1（fvIO スロット 0）端子番号

5	6	7	8
1	2	3	4



CN2(fvIO スロット 1)
CN3（fvIO スロット 2）
CN4（fvIO スロット 3）
CN5（fvIO スロット 4）
CN6（fvIO スロット 5）
CN7（fvIO スロット 6）
CN8（fvIO スロット 7）

■1 スロット版プラグインの端子割り当て

CN1～8 は同様のピン配となる。

CN	端子番号	シンボル
1～8	1	SCLK
	2	MOSI
	3	MISO
	4	CS0
	5	3.3V
	6	3.3V
	7	GND
	8	GND

CN	端子番号	シンボル
9	1	3.3V
	2	GND
	3	SCLK(CN1)
	4	MOSI(CN1)
	5	MISO(CN1)
	6	CS0(CN1)
	7	SCLK(CN3)
	8	MOSI(CN3)
	9	MISO(CN3)
	10	CS0(CN3)

CN	端子番号	シンボル
10	1	3.3V
	2	GND
	3	SCLK(CN2)
	4	MOSI(CN2)
	5	MISO(CN2)
	6	CS0(CN2)
	7	SCLK(CN4)
	8	MOSI(CN4)
	9	MISO(CN4)
	10	CS0(CN4)

■2 スロット版プラグインの端子割り当て

2 スロット版の場合、CN1-2,CN3-4,CN5-6,CN7-8 がペアになる。

CN	端子番号	シンボル
1,3,5,7	1	SCLK
	2	MOSI
	3	MISO
	4	CS0
	5	3.3V
	6	3.3V
	7	GND
	8	GND

CN	端子番号	シンボル
2,4,6,8	1	CS1
	2	RES
	3	D/C
	4	reserved
	5	3.3V
	6	3.3V
	7	GND
	8	GND

CN	端子番号	シンボル
9	1	3.3V
	2	GND
	3	SCLK(CN1)
	4	MOSI(CN1)
	5	MISO(CN1)
	6	CS0(CN1)
	7	SCLK(CN3)
	8	MOSI(CN3)
	9	MISO(CN3)
	10	CS0(CN3)

CN	端子番号	シンボル
10	1	3.3V
	2	GND
	3	CS1(CN2)
	4	RES(CN2)
	5	D/C(CN2)
	6	reserved
	7	CS1(CN2)
	8	RES(CN2)
	9	D/C(CN2)
	10	reserved

4.2　割り込みベクタ割り当て

割り込みベクタの割り当ては以下の通り。

2 スロットのプラグイン版、番号が小さい方のスロットの割り込み要因を使用する。

例）

0-1 スロットを使用するプラグイン→スロット 0 の割り込み要因(ベクタ 129、130)を使用。 2-3 スロットを使用するプラグイン→スロット 2 の割り込み要因(ベクタ 133、134)を使用。

fvIO スロット番号	シンボル	RZ/T1 ベクタ番号	機能
スロット 0	FIFO_PAF	129	fvIO　FIFO 送信割り込み
	FIFO_PAE	130	fvIO　FIFO 受信割り込み
	INT0	242 or 243(注 1)	STAT レジスタ割り込み
スロット 1	FIFO_PAF	131	fvIO　FIFO 送信割り込み
	FIFO_PAE	132	fvIO　FIFO 受信割り込み
	INT0	242 or 243(注 1)	STAT レジスタ割り込み
スロット 2	FIFO_PAF	133	fvIO　FIFO 送信割り込み
	FIFO_PAE	134	fvIO　FIFO 受信割り込み
	INT0	242 or 243(注 1)	STAT レジスタ割り込み
スロット 3	FIFO_PAF	135	fvIO　FIFO 送信割り込み
	FIFO_PAE	136	fvIO　FIFO 受信割り込み
	INT0	242 or 243(注 1)	STAT レジスタ割り込み
スロット 4	FIFO_PAF	137	fvIO　FIFO 送信割り込み
	FIFO_PAE	138	fvIO　FIFO 受信割り込み
	INT0	242 or 243(注 1)	STAT レジスタ割り込み
スロット 5	FIFO_PAF	139	fvIO　FIFO 送信割り込み
	FIFO_PAE	140	fvIO　FIFO 受信割り込み
	INT0	242 or 243(注 1)	STAT レジスタ割り込み
スロット 6	FIFO_PAF	141	fvIO　FIFO 送信割り込み
	FIFO_PAE	142	fvIO　FIFO 受信割り込み
	INT0	242 or 243(注 1)	STAT レジスタ割り込み
スロット 7	FIFO_PAF	143	fvIO　FIFO 送信割り込み
	FIFO_PAE	144	fvIO　FIFO 受信割り込み
	INT0	242 or 243(注 1)	STAT レジスタ割り込み

(注 1)INT0 割り込みは、RZ/T1 のイベントリンク機能を使用して割り込みを行う。イベントリンク機能の割り込みイベントはイベントリンク設定レジスタ(ELSRn)の ELSR18(割り込み 1、ベクタ番号 242)か ELSR19(割り込み 2、ベクタ番号 243)の 2 本のみなので注意すること。
ELSR18 または ELSR19 に以下の値を設定することでイベント設定ができる。

fvIO スロット番号	ELSR 設定値
スロット 0	0x2C
スロット 1	0x2B
スロット 2	0x2A
スロット 3	0x29
スロット 4	0x28
スロット 5	0x27
スロット 6	0x26
スロット 7	0x25

4.3 レジスタアドレス

レジスタのベースアドレス、FIFO0 レジスタのアドレスは以下の通り。

2 スロットのプラグインの場合、番号が小さい方のスロットをベースアドレスとする。

例）

0-1 スロットを使用するプラグイン→スロット 0 のアドレスをベースとする。
2-3 スロットを使用するプラグイン→スロット 2 のアドレスをベースとする。

fvIO スロット番号	レジスタベースアドレス	FIFO0 アドレス
スロット 0	0xB011C100	0xB0FF4000
スロット 1	0xB051C100	0xB0FF4800
スロット 2	0xB091C100	0xB0FF5000
スロット 3	0xB0D1C100	0xB0FF5800
スロット 4	0xB031C100	0xB0FF6000
スロット 5	0xB071C100	0xB0FF6800
スロット 6	0xB0B1C100	0xB0FF7000
スロット 7	0xB0F1C100	0xB0FF7800

5. サンプルプログラム

本サンプルプログラムは、ルネサスエレクトロニクス社製の以下のサンプルプログラムにファイルの修正または追加を行うことで作成している。本章では、本サンプルプログラムの動作と、ベースとなったサンプルプログラムに修正または追加を行った箇所について説明を行う。RAM 実行版、SPI ブート版ともに特に記述がなければ同様の修正または追加を行っている。

■サンプルのベースプログラム

プロジェクト名	概要
an-r01an2554jj0141-rzt1-initial-settings	RZ/T1 グループ 初期設定(ver.1.4.1)

5.1 ファイル構成

本サンプルプログラムでは、以下のファイルを修正または追加している。

フォルダ	ファイル名	追加/修正	概要
inc	r_ecl_rzt1_if.h	追加	ルネサスエレクトロニクス製 EC-lib ヘッダファイル(注 1)
lib/ecl	r_ecl_rzt1.a	追加	ルネサスエレクトロニクス製 EC-lib ライブラリファイル(注 1)
	s_fvIO_rzt1_comn_spi_0.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 0)
	s_fvIO_rzt1_comn_spi_1.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 1)
	s_fvIO_rzt1_comn_spi_2.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 2)
	s_fvIO_rzt1_comn_spi_3.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 3)
	s_fvIO_rzt1_comn_spi_4.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 4)
	s_fvIO_rzt1_comn_spi_5.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 5)
	s_fvIO_rzt1_comn_spi_6.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 6)
	s_fvIO_rzt1_comn_spi_7.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 7)
	s_fvIO_rzt1_comn_spi_01.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 0-1)
	s_fvIO_rzt1_comn_spi_23.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 2-3)
	s_fvIO_rzt1_comn_spi_45.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 4-5)
	s_fvIO_rzt1_comn_spi_67.dat	追加	fvIO 汎用 SPI プラグインコンフィグデータ(スロット 6-7)
src/common/slfash_boot	spibsc_ioset_userdef.c	修正	SPIBSC I/O 設定ユーザー定義ファイル(注 2)
src/fvio/fvio_if	fvIO_if.c	追加	fvIO I/F ソース
	fvIO_if.h	追加	fvIO I/F ヘッダ

src/fvio/fvio_driver	fvIO_cmn_if.h	追加	fvIO 汎用プラグイン共通ヘッダ
	fvIO_cmn_2s_if.h	追加	fvIO 汎用プラグイン共通ヘッダ(2 スロット版)
	fvIO_dat.asm	追加	リンカ設定ファイル
	fvIO_rzt1_dma.c	追加	fvIO　DMA 設定処理ソース
	fvIO_rzt1_dma.h	追加	fvIO　DMA 設定処理ヘッダ
	fvIO_rzt1_spi_cmn.c	追加	fvIO 汎用プラグイン処理ソース
	fvIO_rzt1_spi_cmn.h	追加	fvIO 汎用プラグイン処理ヘッダ
	fvIO_rzt1_spi_cmn_2s.c	追加	fvIO 汎用プラグイン処理ソース(2 スロット版)
	fvIO_rzt1_spi_cmn_2s.h	追加	fvIO 汎用プラグイン処理ヘッダ(2 スロット版)
src/fvio/dev_driver/mpu6500	fvIO_rzt1_spi_mpu6500.c	追加	fvIO 汎用プラグイン・MPU6500 専用処理ソース
	fvIO_rzt1_spi_mpu6500.h	追加	fvIO 汎用プラグイン・MPU6500 専用処理ヘッダ
src/fvio/dev_driver/ssd1306	fvIO_rzt1_spi_ssd1306.c	追加	fvIO 汎用プラグイン・SSD1306 専用処理ソース
	fvIO_rzt1_spi_ssd1306.h	追加	fvIO 汎用プラグイン・SSD1306 専用処理ヘッダ
src/sample	main.c	修正	メインファイル
	utility.c	追加	ユーティリティソース
	utility.h	追加	ユーティリティヘッダ

(注 1)ルネサスエレクトロニクス製の Encoder I/F Configuration Library のファイルを使用。

(注 2) SPI ブート版(RZ_T_sflash_sample)のみ以下を修正

修正前:

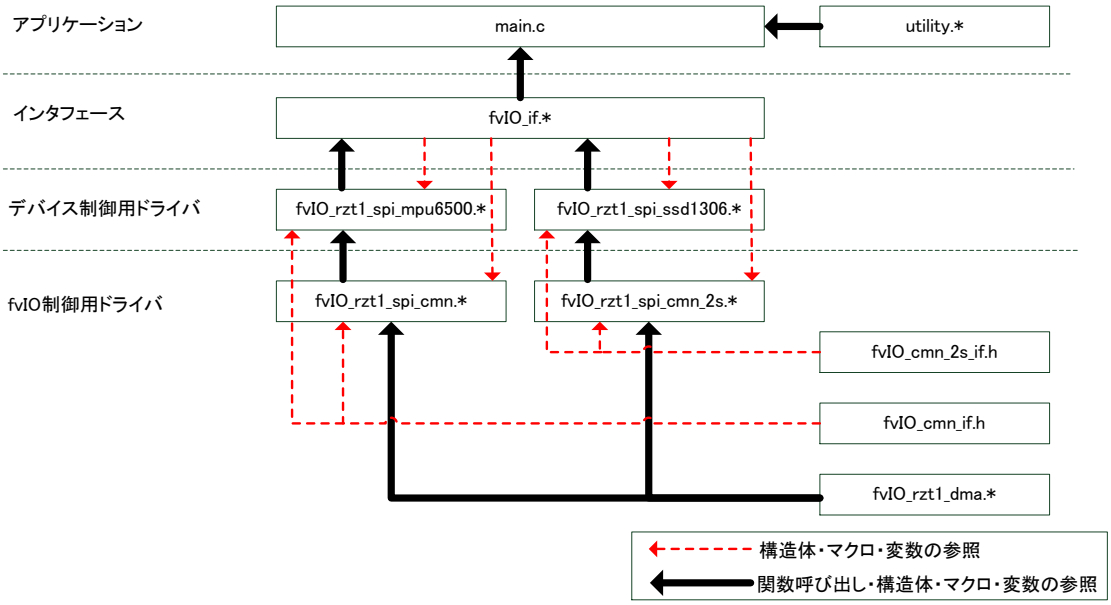
```
#define SPIBSC_BUS_WITDH        (4)
#define SPIBSC_OUTPUT_ADDR  (SPIBSC_OUTPUT_ADDR_32)
```

修正後:

```
#define SPIBSC_BUS_WITDH        (1)
#define SPIBSC_OUTPUT_ADDR  (SPIBSC_OUTPUT_ADDR_24)
```

5.2　ファイルの関連

src/sample, src/fvio 以下にあるファイルの関連図を以下に示す。



5.3　I/F 関数

I/F 関数を以下に示す。I/F 関数では、fvio ID，スロット ID，プラグイン ID を入力することでアクセスする fvIO の判別を行う。

関数名	概要
fvio_sys_init	プラグイン内部で使用する変数やモジュール等の初期化を行う。
fvio_entry	fvIO の I/F 関数群(assign 以下)をプラグインに登録(malloc でメモリを確保)
fvio_release	entry で登録した I/F 関数群をプラグインから削除する。
fvio_assign	プラグインとスロットの紐づけを行う。(一部、ポート等の初期化も実行)
fvio_unassign	プラグインとスロットの紐づけを解除する。
fvio_sys_start	プラグインを紐づけたスロットにロードし、fvIO を一斉に起動する。(全 fvIO は同時に起動)
fvio_stop	fvIO を停止する。(スロット毎に可能)
fvio_write	fvIO に対してライト制御を実行。(プラグイン内部の設定に対しても可能)
fvio_read	fvIO に対してリード制御を実行。(プラグイン内部の設定に対しても可能)

I/F 関数の詳細を以下に示す。

関数名	void fvio_sys_init(void)
機能	fvIO システム初期化処理
引数	なし
戻り値	なし
処理概要	1.DMA モジュールのストップ解除を実行。 2.fvIO プラグインで使用する内部変数を初期化する。
備考	

関数名	int32_t fvio_entry(ST_FVIO_IF_LIST *new_entry, uint32_t mode, int32_t *fvio_id)	
機能	fvIO エントリ処理	
引数	ST_FVIO_LIST new_entry	新規追加する fvIO I/F 関数リスト
	uint32_t mode	処理モード(reserved)
	int32_t *fvio_id	fvIO ID
戻り値	int32_t	0=正常, それ以外=異常
処理概要	1.I/F 関数リストのチェックを行う。 2.プラグインに関数リストを登録する。 3.*fvio_id に I/F 関数リストの fvio_id を設定する。	
備考		

関数名	int32_t fvio_release(int32_t fvio_id)	
機能	fvIO リリース処理	
引数	ST_FVIO_LIST new_entry	新規追加する fvIO I/F 関数リスト
	int32_t fvio_id	fvIO ID
戻り値	int32_t	0=正常, それ以外=異常
処理概要	1.リリース可能かチェックを行う。 2.fvio_id で指定された I/F 関数をプラグインから削除する。	
備考		

関数名	int32_t fvio_assign(int32_t fvio_id, int32_t slot_id, int32_t *plug_id, void *attr)
機能	fvIO アサイン処理
引数	int32_t fvio_id　　fvIO ID
	int32_t slot_id　　スロット ID(0～7)
	int32_t *plug_id　　プラグイン ID
	void *attr　　実装依存の引数
戻り値	int32_t　　0=正常, それ以外=異常
処理概要	1.アサイン可能かチェックを行う。 2.デバイス制御用ドライバから個別のアサイン処理を呼び出す。 3.*plug_id にプラグイン ID を設定する。
備考	2 スロット版のプラグインをアサインする場合、スロット ID は 0,2,4,6 のいずれかを指定すること。

関数名	int32_t fvio_unassign(int32_t plug_id)
機能	fvIO アンアサイン処理
引数	int32_t fvio_id　　fvIO ID
戻り値	int32_t　　0=正常, それ以外=異常
処理概要	1.アンアサイン可能かチェックを行う。 2.デバイス制御用ドライバから個別のアンアサイン処理を呼び出す。
備考	

関数名	int32_t fvio_sys_start(int32_t *result, void *attr)
機能	fvIO スタート処理
引数	int32_t *result　　各スロットの処理結果　0=正常、 0 以外エラー
	void *attr　　実装依存の引数
戻り値	int32_t　　0=正常, それ以外=異常
処理概要	1.fvIO スタート可能かチェックを行う。 2.アサインした fvIO のコンフィグをロードする。 3.デバイス制御用ドライバから個別のスタート処理を呼び出す。 4.*result にスタートの結果を格納する。
備考	スタートは全スロット一括で行う。

関数名	int32_t fvio_stop(int32_t plug_id, void *attr)	
機能	fvIO ストップ処理	
引数	int32_t plug_id	プラグイン ID
	void *attr	実装依存の引数
戻り値	int32_t	0=正常, それ以外=異常
処理概要	1.fvIO ストップ可能かチェックを行う。 2.デバイス制御用ドライバから個別のストップ処理を呼び出す。	
備考	ストップはスロット個別で行う。	

関数名	int32_t fvio_write(int32_t plug_id, uint32_t mode, void *attr)	
機能	fvIO ライト処理	
引数	int32_t plug_id	プラグイン ID
	uint32_t mode	動作モード
	void *attr	実装依存の引数
戻り値	int32_t	0=正常, それ以外=異常
処理概要	1.fvIO,fvIO プラグインにライト可能かチェックを行う。 2.デバイス制御用ドライバから個別のライト処理を呼び出す。	
備考		

関数名	int32_t fvio_read(int32_t plug_id, uint32_t mode, void *attr)	
機能	fvIO リード処理	
引数	int32_t plug_id	プラグイン ID
	uint32_t mode	動作モード
	void *attr	実装依存の引数
戻り値	int32_t	0=正常, それ以外=異常
処理概要	1.fvIO,fvIO プラグインがリード可能かチェックを行う。 2.デバイス制御用ドライバから個別のリード処理を呼び出す。	
備考		

5.4 リード/ライト関数の動作モード

fvio_read 関数, fvio_write 関数の動作モードについて以下に示す。

5.4.1 加速度センサ(MPU6500)の動作モード

加速度センサ(MPU6500)に対して以下のリード/ライトの動作モードをサポートしている。

定数(引数 mode で指定)	R/W	処理概要
FVIO_SPI_MPU6500_MODE_INFO	R	デバイスの I/O タイプ、入出力サイズの取得
FVIO_SPI_MPU6500_MODE_STOP	W	fvIO シーケンスの強制停止
FVIO_SPI_MPU6500_MODE_CREG	R/W	制御レジスタの R/W
FVIO_SPI_MPU6500_MODE_DMA	R	DMA を使用して 3 軸データを取得
FVIO_SPI_MPU6500_MODE_CNF	R/W	fvIO の動作設定を行う。
FVIO_SPI_MPU6500_MODE_INT	W	DMA 完了割り込みのハンドラを登録する。
FVIO_SPI_MPU6500_MODE_BSY	R	busy ステータスを取得
FVIO_SPI_MPU6500_MODE_RDMA	W	FVIO_MPU6500_MODE_DMA と同じ設定で DMA を再実行する。

(1)FVIO_SPI_MPU6500_MODE_INFO

引数*attr に対して以下の構造体を入力し、デバイスの I/O タイプ、入出力サイズを取得する。

```
typedef struct {
        uint8_t    io_type;
        uint32_t in_sz;
        uint32_t out_sz;
}ST_FVIO_IF_INFO;
```

パラメータ	説明
io_type	fvIO の I/O タイプ FVIO_IF_INFO_TYPE_I : CPU からは入力(データ取得)のみ可能 FVIO_IF_INFO_TYPE_O : CPU からは出力(データ設定)のみ可能 FVIO_IF_INFO_TYPE_IO : 入出力どちらも可能
in_sz	fvIO プラグインを使用するのに最低限必要な入力バッファサイズ(1byte 単位)
out_sz	fvIO プラグインを使用するのに最低限必要な出力バッファサイズ(1byte 単位)

(2) FVIO_SPI_MPU6500_MODE_STOP

fvIO シーケンスの連続実行を停止する。引数*attr には NULL を入力する。
シーケンスが停止するまでアプリケーション側に制御が返らないので注意すること。

(3) FVIO_SPI_MPU6500_MODE_CREG

引数*attr に対して以下の構造体を入力し、レジスタの R/W を実行する。

```
typedef struct {
        uint8_t trg;
        uint8_t *sdata;
        uint8_t *rdata;
        uint8_t sz;
}ST_FVIO_SPI_MPU6500_CREG;
```

■リード時

パラメータ	説明
trg	トリガオプション FVIO_CMN_REG_TRG_SYNC　：ELC と同期して動作 FVIO_CMN_REG_TRG_REP　：リピート実行
*sdata	送信データ 送信データの先頭アドレスを設定。フォーマットは以下を参照
*rdata	受信データ 受信データの先頭アドレスを設定。*rdata には受信データの RAW データが 1byte 目から順に格納される。(1byte 目は先頭アドレスの
sz	通信サイズ 受信データ数-1 を設定。設定範囲は 0～7。

リード時の送信データフォーマット：

*sdata	送信データ
1byte 目	アクセスするレジスタの先頭アドレス

リード時の受信データフォーマット：

*sdata	送信データ
1byte 目	不定値
2byte 目	先頭アドレスのレジスタ値
3byte 目	2byte 目のアドレスのレジスタ値
4byte 目	3byte 目のアドレスのレジスタ値
5byte 目	4byte 目のアドレスのレジスタ値
6byte 目	5byte 目のアドレスのレジスタ値
7byte 目	6byte 目のアドレスのレジスタ値
8byte 目	7byte 目のアドレスのレジスタ値

■ライト時

パラメータ	説明
trg	トリガオプション FVIO_CMN_REG_TRG_SYNC　：ELC と同期して動作 FVIO_CMN_REG_TRG_REP　：リピート実行
*sdata	送信データ 送信データの先頭アドレスを設定。フォーマットは以下を参照
*rdata	受信データ 設定なし(受信は実行しない)
sz	通信サイズ 送信データ数-1 を設定。設定範囲は 0～7。

ライト時の送信データフォーマット：

*sdata	送信データ
1byte 目	アクセスするレジスタの先頭アドレス
2byte 目	設定データ(1byte 目)
3byte 目	設定データ(2byte 目)
4byte 目	設定データ(3byte 目)
5byte 目	設定データ(4byte 目)
6byte 目	設定データ(5byte 目)
7byte 目	設定データ(6byte 目)
8byte 目	設定データ(7byte 目)

(4)FVIO_SPI_MPU6500_MODE_DMA

引数*attr に対して以下の構造体を入力し、MPU6500 のレジスタ 0x3B(ACCEL_XOUT_H)～0x40(ACCEL_ZOUT_L) の 6byte を DMA で取得する。

DMA 転送は 1 回の転送で終了するので、連続実行する時は後述 FVIO_MPU6500_MODE_RDMA を使用して再実行する必要がある。この時、ソースアドレスは以下のように切り替わる。

```
data1→data2→data1→data2→...
```

また、FVIO_MPU6500_MODE_DMA を使用して DMA を再実行した場合、受信データが格納されるデスティネーションは、data1 となる。

```
typedef struct {
        uint8_t   trg;
        uint8_t   *data1;
        uint8_t   *data2;
        uint8_t   sz;
        uint32_t num;
}ST_FVIO_SPI_MPU6500_DMA;
```

パラメータ	説明
trg	トリガオプション FVIO_CMN_REG_TRG_SYNC　：ELC と同期して動作 FVIO_CMN_REG_TRG_REP　：リピート実行
*data1	DMA のデスティネーションアドレスデータ 1 受信データを格納する領域の先頭アドレスを設定する。
*data2	DMA のデスティネーションアドレスデータ 2 受信データを格納する領域の先頭アドレスを設定する。
sz	1 パケット当たりの受信データ数 6 を設定する。
num	1 回の DMA 転送で通信するパケット数 sz*num の値が DMA 転送 1 回当たりの総 byte 数となる。

(5)FVIO_SPI_MPU6500_MODE_CNF

引数*attr に対して以下の構造体を入力し、fvIO プラグイン設定の R/W を行う。

```
typedef struct {
        uint8_t    cwait;
        uint32_t lwait;
}ST_FVIO_MPU6500 _CNF;
```

パラメータ	説明
cwait	通信用のクロック周期設定 クロック周期[MHz] = 20/((設定値+1)*4)
lwait	fvIO シーケンスのトリガ無効期間(ウェイト) ウェイト期間中、シーケンスは実行できない。 ウェイト期間はシーケンス開始時からカウントする。 0 : ウェイト期間[ns]=0 それ以外: ウェイト期間[ns]=(設定値+1)×50

(6)FVIO_SPI_MPU6500_MODE_INT

引数*attr に対して以下の構造体を入力し、割り込みコールバック関数の設定を行う。

```
typedef struct {
        void (*paf_callback)(int32_t);
        void (*pae_callback)(int32_t);
}ST_FVIO_SPI_MPU6500_INT;
```

パラメータ	説明
*paf_callback	reserved
*pae_callback	割り込みコールバック(FVIO_SPI_MPU6500_MODE_DMA の DMA 転送完了) FVIO_SPI_MPU6500_MODE_DMA で実行した DMA 転送完了時にコールバックする割り込みハンドラのアドレスを登録する。

(7)FVIO_SPI_MPU6500 _MODE_BSY

fvIO の動作ステータスを取得する。

返り値	説明
0	fvIO アイドル中(シーケンス完了)
それ以外	fvIO 動作中

(8)FVIO_SPI_MPU6500_MODE_RDMA

直前に実行した DMA 転送を同じ設定で再実行する。引数*attr には NULL を入力。

このモードは必ず FVIO_SPI_MPU6500 _MODE_DMA を実行した後で行う。

5.4.2有機ディスプレイ(SSD1306)の動作モード

有機 EL ディスプレイ(SSD1306)に対して以下のリード/ライトの動作モードをサポートしている。

定数(引数 mode で指定)	R/W	処理概要
FVIO_SPI_SSD1306_MODE_INFO	R	fvIO の I/O タイプ、入出力サイズの取得
FVIO_SPI_SSD1306_MODE_STOP	W	fvIO シーケンスの強制停止
FVIO_SPI_SSD1306_MODE_CREG	W	制御レジスタの R/W
FVIO_SPI_SSD1306_MODE_DMA	W	DMA を使用して OLED に対して連続出力
FVIO_SPI_SSD1306_MODE_CNF	R/W	fvIO の動作設定を行う。
FVIO_SPI_SSD1306_MODE_INT	W	DMA 完了割り込みのハンドラを登録する。
FVIO_SPI_SSD1306_MODE_BSY	R	busy ステータスを取得
FVIO_SPI_SSD1306_MODE_RDMA	W	FVIO_SPI_SSD1306_MODE_DMA と同じ設定で DMA を再実行する。

(1)FVIO_SSD1306_MODE_INFO

引数*attr に対して以下の構造体を入力し、デバイスの I/O タイプ、入出力サイズを取得する。

```
typedef struct {
        uint8_t    io_type;
        uint32_t in_sz;
        uint32_t out_sz;
}ST_FVIO_IF_INFO;
```

パラメータ	説明
io_type	fvIO の I/O タイプ FVIO_IF_INFO_TYPE_I : CPU からは入力(データ取得)のみ可能 FVIO_IF_INFO_TYPE_O : CPU からは出力(データ設定)のみ可能 FVIO_IF_INFO_TYPE_IO : 入出力どちらも可能
in_sz	fvIO プラグインを使用するのに最低限必要な入力バッファサイズ(1byte 単位)
out_sz	fvIO プラグインを使用するのに最低限必要な出力バッファサイズ(1byte 単位)

(2)FVIO_SPI_SSD1306_MODE_STOP

fvIO シーケンスの連続実行を停止する。引数*attr には NULL を入力する。

シーケンスが停止するまでアプリケーション側に制御が返らないので注意すること。

(3)FVIO_SPI_SSD1306_MODE_CREG

引数*attr に対して以下の構造体を入力し、レジスタのライトを実行する。

```
typedef struct {
        uint8_t trg;
        uint8_t *sdata;
        uint8_t *rdata;
        uint8_t sz;
}ST_FVIO_SPI _SSD1306_CREG;
```

■ライト時

パラメータ	説明
trg	トリガオプション FVIO_CMN_2S_REG_TRG_SYNC　: ELC と同期して動作 FVIO_CMN_2S_REG_TRG_REP　: リピート実行
*sdata	送信データ 送信データの先頭アドレスを設定。フォーマットは以下を参照
*rdata	受信データ 設定なし
sz	通信サイズ 送信データ数-1 を設定。設定範囲は 0～7。

ライト時の送信データフォーマット:

*sdata	送信データ
1byte 目	アクセスするレジスタの先頭アドレス
2byte 目	設定データ(1byte 目)
3byte 目	設定データ(2byte 目)
4byte 目	設定データ(3byte 目)
5byte 目	設定データ(4byte 目)
6byte 目	設定データ(5byte 目)
7byte 目	設定データ(6byte 目)
8byte 目	設定データ(7byte 目)

(4)FVIO_SPI_SSD1306_MODE_DMA

引数*attr に対して以下の構造体を入力し、SSD1306 の GDDRAM に対して pixel データを送信する。

DMA 転送は 1 回の転送で終了するので、連続実行する時は後述の FVIO_SPI_SSD1306_MODE_RDMA を使用して再実行する必要がある。この時、ソースアドレスは以下のように切り替わる。

```
data1→data2→data1→data2→...
```

また、FVIO_SPI_SSD1306_MODE_DMA を使用して DMA を再実行した場合、送信データとして使用するソースは、data1 となる。

```
typedef struct {
        uint8_t   trg;
        uint8_t   *data1;
        uint8_t   *data2;
        uint8_t   sz;
        uint32_t num;
}ST_FVIO_SPI_SSD1306_DMA;
```

パラメータ	説明
trg	トリガオプション FVIO_CMN_2S_REG_TRG_SYNC　: ELC と同期して動作 FVIO_CMN_2S_REG_TRG_REP　: リピート実行
*data1	DMA のデスティネーションアドレスデータ 1 受信データを格納する領域の先頭アドレスを設定する。 下記の構造体データを使用することを推奨
*data2	DMA のデスティネーションアドレスデータ 2 受信データを格納する領域の先頭アドレスを設定する。 下記の構造体データを使用することを推奨
sz	1 パケット当たりの送信データ数 1 パケット当たりの送信データ数-1 を設定する。 下記の構造体データを使用する場合は 3 を設定する。
num	1 回の DMA 転送で通信するパケット数 sz*num の値が DMA 転送 1 回当たりの総 byte 数となる。

SSD1306 通信用のパケット構造体:

```
typedef struct {
        uint8_t data[4];
}ST_FVIO_SPI_SSD1306_PACKET;
```

パラメータ	説明
data[4]	dot データ ディスプレイの dot データを設定する。1 度の通信で設定する dot 数は 32dot(1bit=1dot)。描画方向は SSD1306 のレジスタ設定に依存。

(5)FVIO_SPI_SSD1306_MODE_CNF

引数*attr に対して以下の構造体を入力し、fvIO プラグイン設定の R/W を行う。

```
typedef struct {
        uint8_t   cwait;
        uint32_t lwait;
}ST_FVIO_SPI_SSD1306_CNF;
```

パラメータ	説明
cwait	通信用のクロック周期設定 クロック周期[MHz] = 20/((設定値+1)*4)
lwait	fvIO シーケンスのトリガ無効期間(ウェイト) ウェイト期間中、シーケンスは実行できない。 ウェイト期間はシーケンス開始時からカウントする。 0 : ウェイト期間[ns]=0 それ以外: ウェイト期間[ns]=(設定値+1)×50

(6)FVIO_SPI_SSD1306_MODE_INT

引数*attr に対して以下の構造体を入力し、割り込みコールバック関数の設定を行う。

```
typedef struct {
        void (*paf_callback)(int32_t);
        void (*pae_callback)(int32_t);
}ST_FVIO_SPI_SSD1306_INT;
```

パラメータ	説明
*paf_callback	割り込みコールバック(FVIO_SPI_SSD1306_MODE_DMA の DMA 転送完了) FVIO_SPI_SSD1306_MODE_DMA で実行した DMA 転送完了時にコールバックする割り込みハンドラのアドレスを登録する。
*pae_callback	reserved

(7)FVIO_SPI_SSD1306_MODE_BSY

fvIO の動作ステータスを取得する。引数*attr には NULL を入力。

返り値	説明
0	fvIO アイドル中(シーケンス完了)
それ以外	fvIO 動作中

(8)FVIO_SPI_SSD1306_MODE_RDMA

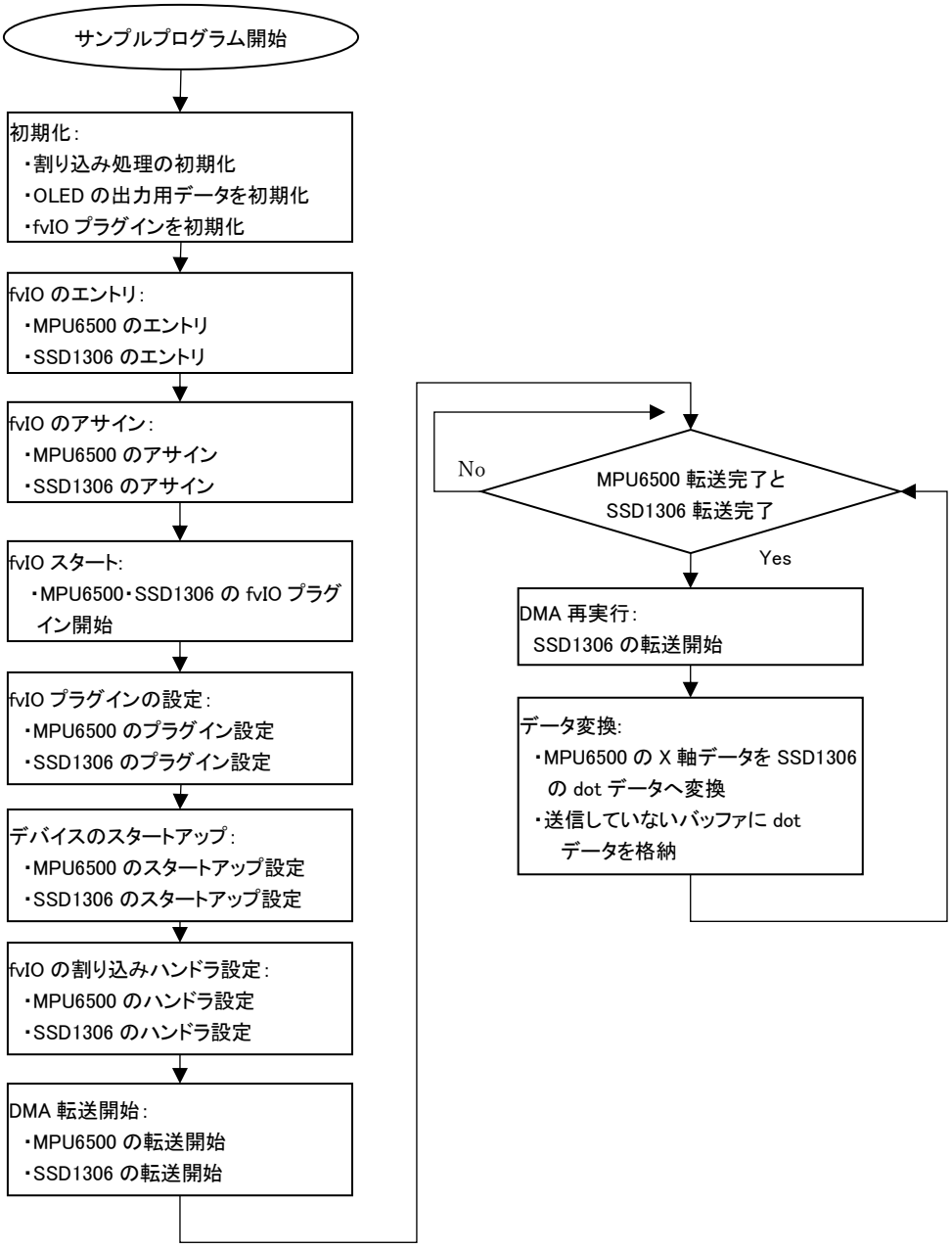
直前に実行した DMA 転送を同じ設定で再実行する。引数*attr には NULL を入力。
このモードは必ず FVIO_SPI_SSD1306_MODE_DMA を実行した後で行う。

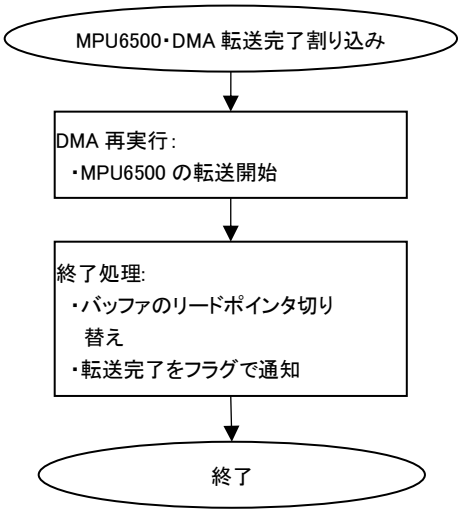
5.5 サンプルプログラム概要

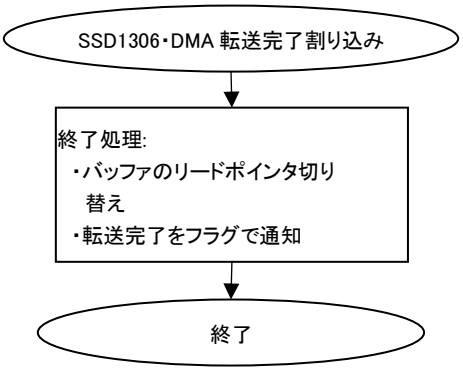
MPU6500 から 3 軸加速度データを DMA 転送で取得する。
取得したデータの内、X 軸データを SSD1306 用の折れ線グラフの dot データへ変換後、
DMA 転送で出力する。3 軸加速度データは 1 回の DMA 転送で 1 画面分取得し、
SSD1306 の出力も 1 画面分出力する。

5.6　サンプルプログラムフローチャート

サンプルプログラムのフローチャートを以下に示す。




MPU6500・DMA 転送完了割り込み
DMA 再実行:
・MPU6500 の転送開始
終了処理:
・バッファのリードポインタ切り
替え
・転送完了をフラグで通知
終了


SSD1306・DMA 転送完了割り込み
終了処理:
・バッファのリードポインタ切り
替え
・転送完了をフラグで通知
終了