TOPPERS

TOPPERS BASE PLATFORM (ST) V1.4.6

担当者	竹内良輔 (Educational WG)

変更履歴

リリース	日付	作成者	変更内容
1.0.0	2016/05/28	竹内良輔	STM32F4xx 版
1.0.1	2016/06/23	竹内良輔	STM32F746 の機能を追加,446/746-nucleo-144 を追加
1.1.0	2016/08/26	竹内良輔	USB OTG の対応/I2C プロトシールドを追加
1.1.1	2017/01/20	竹内良輔	USB OTG プログラム修正による改訂、CV 版と分離
1.2.0	2017/07/07	竹内良輔	STM32F767/STM32F769 ボード対応,spi 変更
1.2.1	2017/08/25	竹内良輔	SM32F091/STM32L073/STM32L476 ボード対応
1.3.0	2018/05/31	竹内良輔	TOPPERS USB MEDDLEWARE の対応
1.4.0	2019/03/04	竹内良輔	バージョンアップ、変更履歴詳細に記載
1.4.1	2020/05/11	竹内良輔	gdic ドライバ追加/lwip 対応、新ボード対応
1.4.2	2021/06/01	竹内良輔	WIFI 拡張/OpenAMP の対応/stmcube 変更、6.6,6.7 章追加、BLE で不足のユーティリティを追記
1.4.3	2022/04/18	竹内良輔	PWM ドライバ追加、USB シールド変更、RTOS 隠蔽機能、STM32G0B1 ボード対応、WB の BLE スタックを 1.2.0 に対応、STM32WB15 nucleo ボード対応
1.4.4	2023/04/13	竹内良輔	PWM ドライバに deinit 関数を追加、STM-USB デバイスの CDC のエンドポイントの不具合を修正、USB ミドルウェアの最適化、gdic に esp32 を追加、lwip を 2.1.3 にバージョンアップ
1.4.5	2024/01/12	竹内良輔	lwIP の OPTTEST 用 TCPIP ロック機能を追加､標準出力処理の変更、ネットワークモニタ対応、USB デバイス MSC を追加、WB15 の P2Pserver の対応をやめる
1.4.6	2024/02/27	竹内良輔	STM32WB55 の ble_remote アプリを修正、ワーニング対応

目次

1　背景

TOPPERS 教育 WG では、組込み用ソフトウェアプラットフォーム用教材を基礎３コンテンツとして作成した。コンテンツとしては抽象的でわかりにくいものとなり、組込みプラットフォーム教材として満足のいくものではなかった。そこで新しい組込みプラットフォームの教材の作成に着手した。開発にあたり、アプローチ方法を大きく見直した。この教材を構築する前提として実際に商品レベルに使用可能な組込みソフトウェアプラットフォームとドキュメント（本リファレンスマニュアル）を開発し、それをベースとして、新基礎２、新基礎３のセミナー部と、実習部を作成する手順とした。

旧基礎１，２，３コンテンツから新基礎１，２，３コンテンツの改訂にあたり、セミナー部とリファレンスシステムで使用するターゲットボードと実際にターゲットボード上で動作する組込みソフトウェアプラットフォームを TOPPERS BASE PLATFORM と呼称し、仕様書はリファレンスマニュアル呼称することとなった。

2　目的

本リファレンスマニュアルは、ターゲットボードとターゲットボード上に作成した組込みソフトウェアプラットフォーム(TOPPERS BASE PLATFORM(ST))ついて記載する。

TOPPERS BASE PLATFORM(ST) は STM32F4SoC 対応のボードに対する機能を for STM32F4xx として記載する。同様に、STM32F7SoC 対応のボードに対する機能を for STM32F7xx、STM32F0SoC 対応ボードに対する記載を STM32F0xx、STM32L0SoC に対応する記載を STM32L0xx、STM32H7SoC に対応する記載を STM32H7xx、STM32G4SoC に対する記載を STM32G4xx、STM32WB55SoC に対する記載を STM32WBxx とする。

■ターゲットボード
・STM32F407 Discovery for STM32F4xx
・STM32F401 Nucleo-64 for STM32F4xx
・STM32F446 Nucleo-64 for STM32F4xx
・STM32F446 Nucleo-144 for STM32F4xx
・STM32F746 Discovery for STM32F7xx
・STM32F746 Nucleo-144 for STM32F7xx
・STM32F767 Nucleo-144 for STM32F7xx
・STM32F769 Discovery for STM32F7xx
・STM32F091 Nucleo-64 for STM32F0xx
・STM32L073 Nucleo-64 for STM32L0xx
・STM32L476 Nucleo-64 for STM32L4xx
・STM32L476 Discovery for STM32L4xx
・STM32F723 Discovery for STM32F7xx
・STM32L4R5 Nucleo-144 for STM32L4xx
・STM32H743 Nucleo-144 for STM32H7xx
・STM32G474 Nucleo-64 for STM32G4xx
・STM32G431 Nucleo-64 for STM32G4xx
・STM32G071 Nucleo-64 for STM32G0xx
・STM32G0B1 Nucleo-64 for STM32G0xx
・STM32WB55 Nucleo for STM32WBxx
・STM32WB15 Nucleo for STM32WBxx
・STM32MP157C DK2 for STM32MP15xx

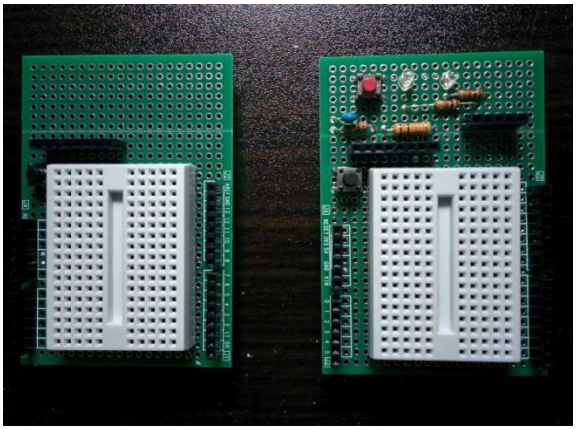
3　TOPPERS BASE PLATFORM (ST)仕様

本章では、TOPPERS BASE PLATFORM (ST)の仕様について記載する。

3.1　プラットフォームのターゲット

TOPPERS BASE PLATFORM(ST)は Arduino メインボードのようにシールドをつけて種々の形態

を持つようなシステムをターゲットとする。基礎１、２セミナー用の形態はプロトシールドをつけてセミナーを行う。新基礎３では adafruit 製１．８インチ　１８ｂｉｔカラーＴＦＴシールド with microSD & Joy stick をつけて、Arduino と同じように、これを制御する。



プロトシールド A-TYPE と B-TYPE

プロトシールド A-TYPE はブレッドボードとリセットＳＷのみのシールド、プロトシールド B-TYPE は A-TYPE に加え、基礎１、基礎２セミナーで使用するユーザースイッチと２つの LED 回路を追加したシールド。

セミナーや実装開発用に、FlatOak 社製のプロトタイプシールド(TEB001)の使用を推奨する。このシールドは ST マイクロエレクトロニクス社の nucleo ボード専用に設計ボードでハンダ付なしに LED やユーザースイッチを使用できる。

FlatOak 社 TEB001 プロトタイプシールド

I2C 用の適切なシールドがないため、プロトタイプシールドに I2C 対応の LCD、温度センサー、EEPROM を乗せた I2C プロトタイプシールド、及び、IWHI 1.54” LCD シールド(TEB002)の回路図を示す。

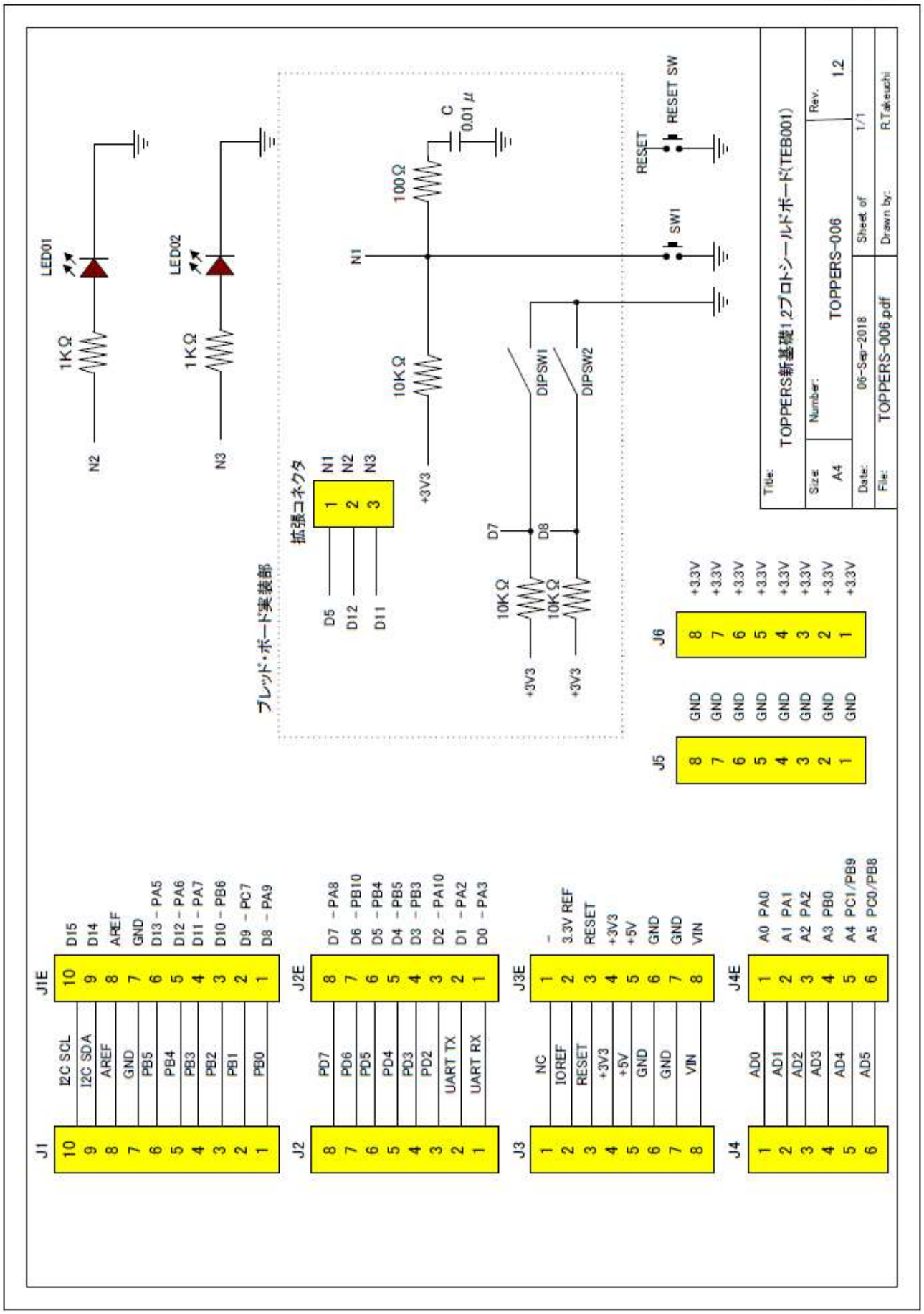


図 3.1 プロトシールド回路図

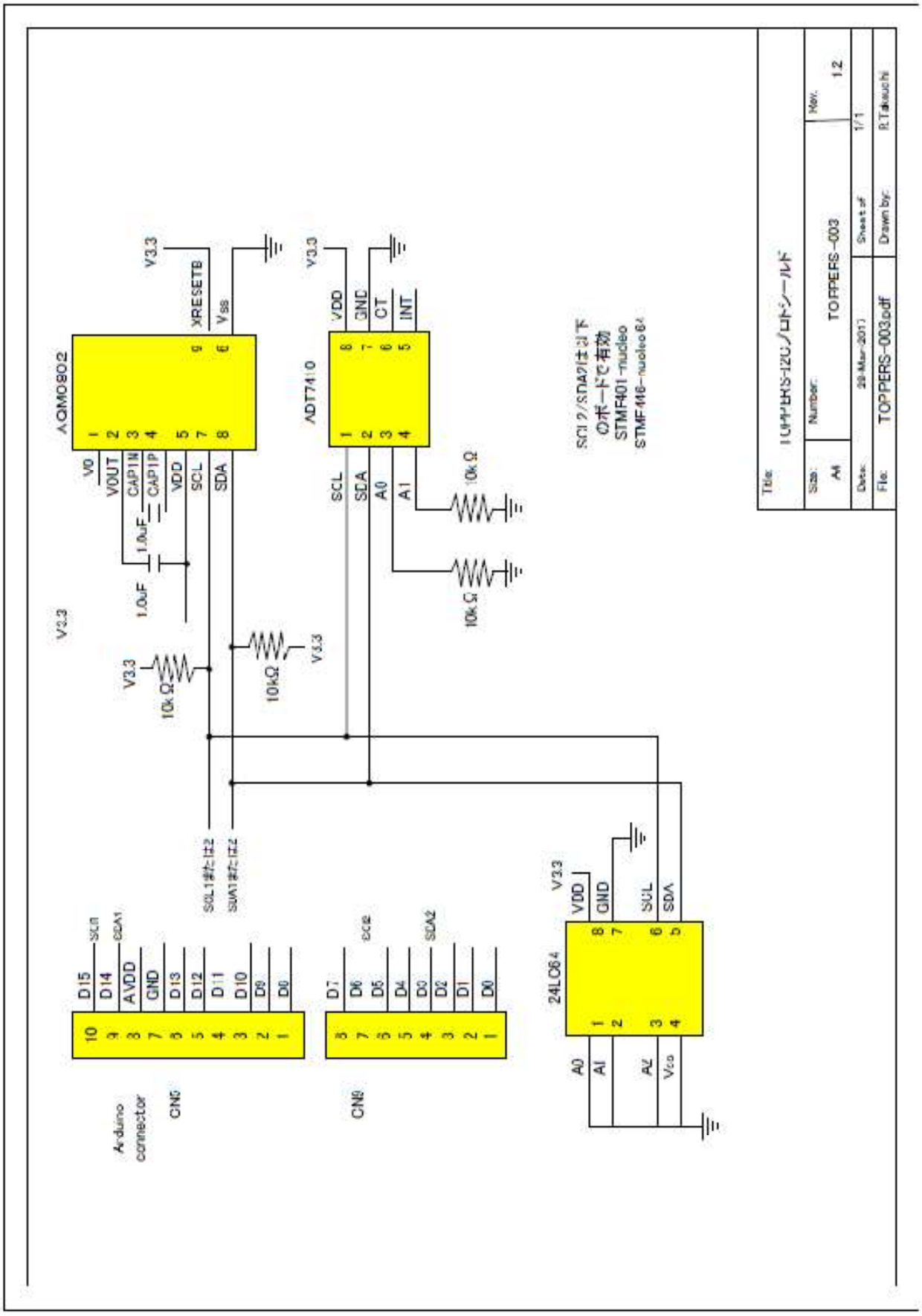


図 3.2 I2C プロトシールド

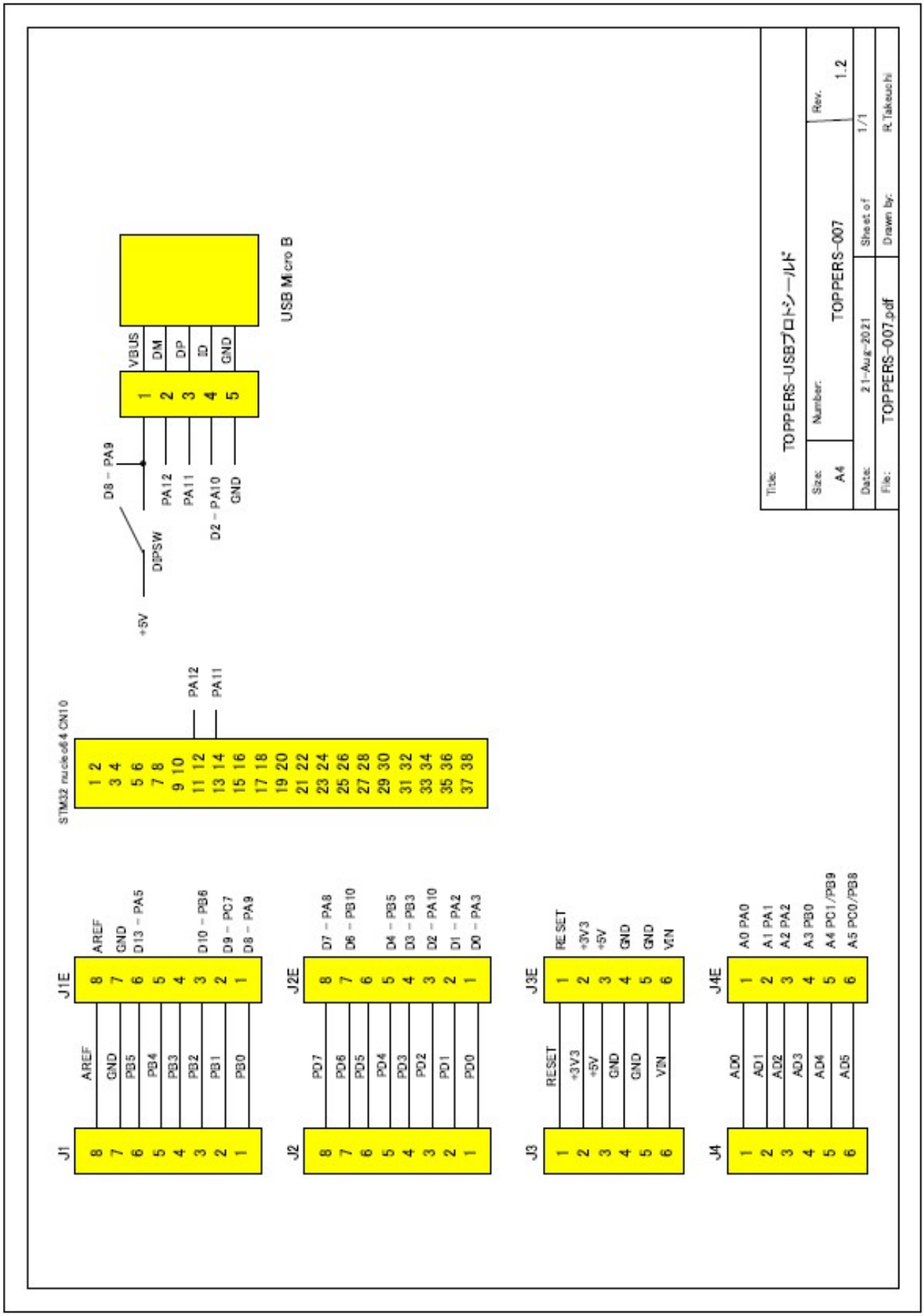

USB Micro B
VBUS
DM
DP
ID
GND
D8 – PA9
PA12
PA11
D2 – PA10
GND
+5V
DIPSW
STM32 nucleo64 CN10
PA12
PA11
J1E
AREF
GND
D13 – PA5
D10 – PB6
D9 – PC7
D8 – PA9
J2E
D7 – PA8
D6 – PB10
D5 – PB5
D4 – PB3
D3 – PA10
D2 – PA2
D1 – PA3
D0 – PA3
J3E
RESET
+3V3
+5V
GND
GND
VIN
J4E
A0 PA0
A1 PA1
A2 PA2
A3 PB0
A4 PC1/PB9
A5 PC0/PB8
J1
AREF
GND
PB5
PB4
PB3
PB2
PB1
PB0
J2
PD7
PD6
PD5
PD4
PD3
PD2
PD1
PD0
J3
RESET
+3V3
+5V
GND
GND
VIN
J4
A0
A1
A2
A3
A4
A5
Title: TOPPERS-USBプロトシールド
Size: A4
Number: TOPPERS-007
Rev. 1.2
Date: 21-Aug-2021
Sheet of 1/1
File: TOPPERS-007.pdf
Drawn by: R.Takeuchi

図 3.3 USB シールド

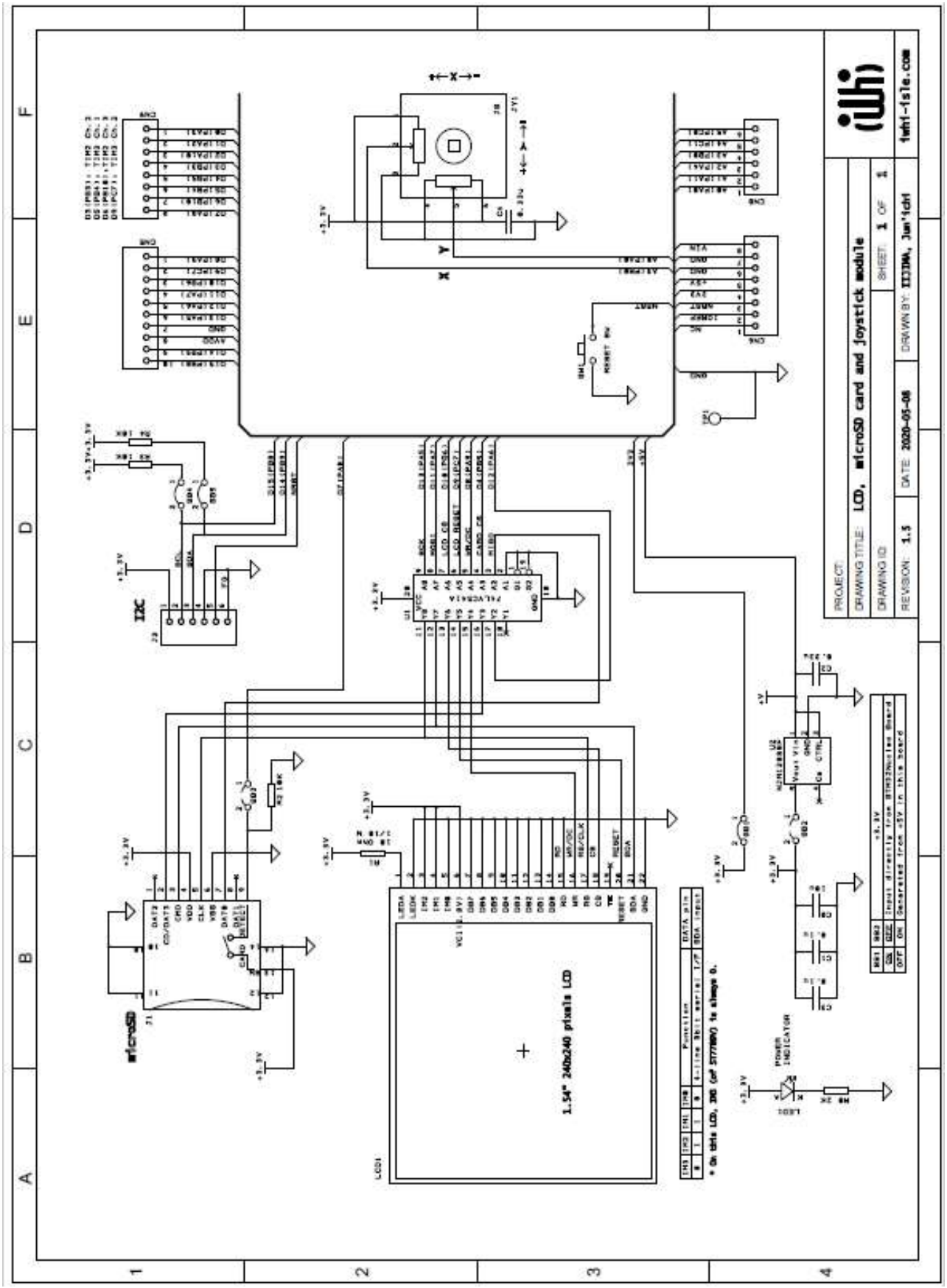
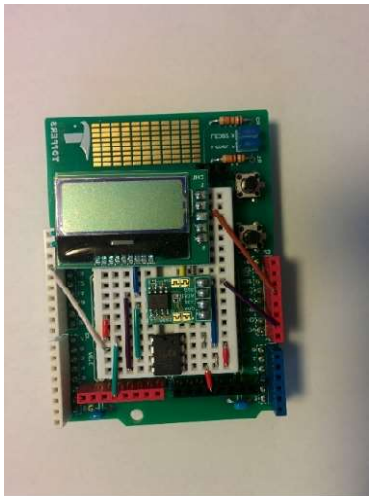
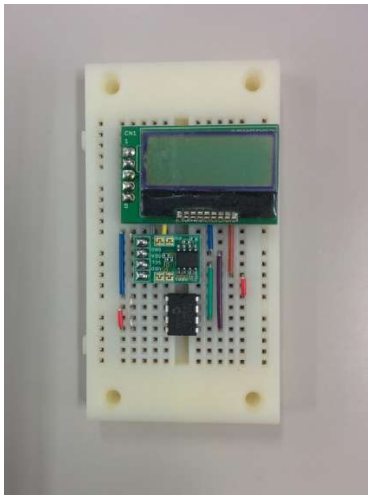


図 3.4 IWHI 1.54" LCD シールド(TEB002)


I2C プロトタイプシールド


I2C ブレッドボード(D15,14 を対応)


IWHI 1.54LCD シールド

3.2 レイア構造

PLATFORM のハードウェアドライバは下層から 3 層のレイヤ構造を持ち、その上に API 層、ライブラリの I/F 層をもつ。これらの PLATFORM は、Realtime Kernel TOPPERS ASP-1.9.3 上に構築している。また、ASP のもつデバッグ機能以外に、教育WGで提供するタスクモニタを標準に装備し、システムデバッグ用にデバッグコマンドを用いて開発補助を行う。

V1.4.3 以上では RTOS のサービス・コールの隠蔽機能を追加する。ディレクトリ pdic 直下の base_platform.h にて、サービス・コールのマクロ化を行い、μITRON 系の RTOS ならばマクロを書き換えることにより、種々の RTOS への対応が可能となる。対象は pdic/gdic のデバイスドライバとミドルウェアである。

図 3.2.1 に BASE PLATFORM(STM) for STM32F4xx/STM32L4xx/STM32F0xx/STM32L0xx の構造図を示す。Standard Driver、SPI xDriver、File Library により、アプリケーションから以下の機能が使用可能になる。

① SD card File system(SPI)
② SPI
③ Wire(I2C)
④ UART
⑤ ADC
⑥ QSPI
⑦ RTC
⑧ USB OTG

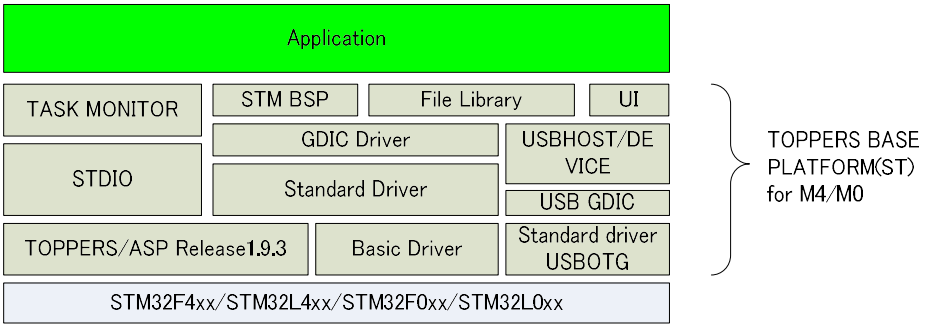


図 3.2.1 TOPPERS BASE PLATFORM(ST) for STM32F4xx/L4xx/F0xx/L0xx 構造図

図 3.2.2 に TOPPERS BASE PLATFORM(ST) for STM32F7xx の構造図を示す。F7 Depend driver の拡張により以下の機能が使用可能になる。

① jpeg-9b(JPEG ライブラリ)
② libmad-0.15.1b(mp3 デコーダライブラリ)

③ lwip(TCP/IP プロトコルスタック)
④ SD card File system(SDMMC)
⑤ SDRAM
⑥ GLCD/touch panel
⑦ RTC
⑧ AUDIO
⑨ SPI
⑩ Wire(I2C)
⑪ ADC
⑫ QSPI
⑬ USB OTG

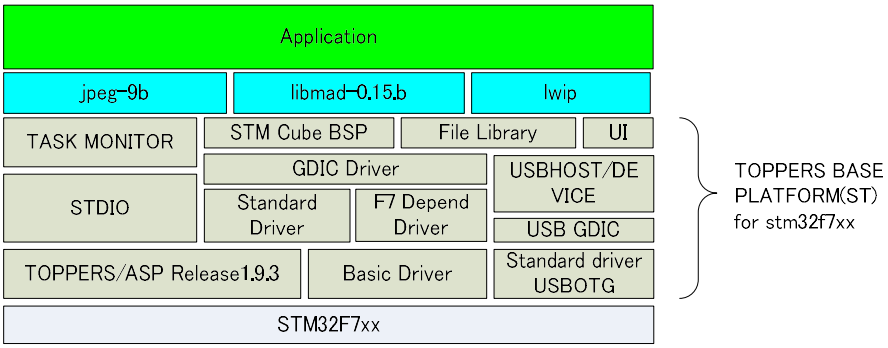


図 3.2.2 TOPPERS BASE PLATFORM(ST) for STM32F7xx

以下に、レイア構造の概要を示す。

(1)basic driver ハードウェア仕様により API が変わるドライバ層
(2)standard driver 拡張ボードの標準化により、API がある程度標準化されているドライバ層
(3)GDIC driver 下位の driver を使用して構築する特定のデバイス用 driver
(4)F7depend driver STM32F7xx のハード構成に従うドライバ層(F7 のみ)
(5)BASE PLATFORM で標準化している API 層
(6)open source のライブラリ等

PLATFORM(ST)		STM32F4/F0xx STM32L4/L0xx	STM32F7xx	detail	対象の講座
Device Driver (pdic)	Basic Driver	gpio	gpio		新基礎 2
		dma	dma		
		watch doc	watch doc		
		(timer)	(timer)		
		uart	uart		
	Standard Driver	i2c	i2c	CLCD/senser/eeprom	新基礎 3 (Reference Manual)
		spi	spi	GLCD/SD card	
		adc	adc	Joy stick	
		(qspi)	qspi	FLASH	
		rtc	rtc	Clock 機能	
		usb OTG	usbo	MSC/HID	
	F7 Depend Driver		ltdc/dsi	GLCD	新基礎 3 Reference System (Application Manual Reference Manual)
			tp		
			sdmmc	SD card	
			audio/dfsdm		
			emac	有線 LAN	
			wifi	無線 LAN	
gdic	high driver			Shield 依存	
api middleware	File System	fatfs	fatfs	FAT	
		file libreary	file library	C language file	

		stdio	stdio	stdio	
	Graphic UI	ui	ui	文字グラフィック描画	
	(Open source) library	USB	usb meddle	USB host/dev	
			libjpeg(*1)	JPEG	
			libmad(*1)	MP3	
		LAN	lwip(*2)	TCP/IP	

*1：api(posix 互換)のためソース未修正で実装：バージョンアップに追従
*2：問題点対応用に patch ファイルを用意：バージョンは固定となる

3.3 開発環境

TOPPERS BASE PLATFORM は Windows7 以降のパソコンで cygwin または MSYS2 をインストールし、コンパイラ環境は GCC-ARM の以下のバージョンをインストールし開発を行っている。
(1) gcc version 5.4(GCC ARM:2016q2-20160622)
(2) gcc version 5.4(GCC ARM-2016q2-20160926)
(3) gcc version 10.2.1(GCC ARM-2020q4-major)

4 Device Driver 仕様

ハードウェア用デバイスドライバの仕様について記載を行う。本 PLATFORM では、3 種類のデバイスドライバを提供する。Basic Driver と Standard Driver は STM32Fxxx と STM32Lxxx で共通の仕様となる。M7 Depend Driver は STM32Fxx でサポートするハードウェア専用のデバイスドライバである。

4.1 Basic Driver

4.1.1 概要

Basic Driver は、ハードウェアを制御する基本的なドライバ群である。制御は簡単な制御手順ですが、初期化や拡張機能は、SoC によってまちまちの実装が行われており、標準的な API では作成できないものが多い。また、Basic Driver は直接ミドルウェアやアプリから制御を行うより、上位のドライバから使用機能として呼び出すケースが多い。逆に Basic Driver は他のドライバの呼び出しは行わない。Basic Driver はハードウェアの依存性が大きいため、PLATFORM を別の SoC にポーティングする場合、別の API の実装となる。

TIMER と UART は、asp カーネルで使用されている。基本的には asp カーネルのドライバを使用する。差分のみを Basic Driver として記載する。

4.1.2 ドライバー覧

Basic Driver として分類するドライバは以下の４つである。
(1) gpio　　汎用ＩＯドライバ
(2) dma　　ダイナミック・メモリ・アクセスドライバ
(3) timer　　タイマードライバ
(4) uart　　シリアルドライバ

4.1.3 GPIO

GPIO は汎用のＩＯを制御するドライバである。GPIO はピン設定を入力または出力に設定し、ピンに対してデータを読み込むまたは書き込みことにより、外部のロジックとのデータ交換を行う機能を持つ。機能的には単純であるが、ピンアサイン、ベースの電圧設定、出力モード設定、割込みの対応等、初期化に関して SoC の設計により、設定仕様がまちまちであり、標準的な初期化手順を作ることが難しい。

4.1.3.1 データ仕様

STM 社 Cortex-M 系の GPIO の初期化に用いるデータと構造体について記載する。GPIO の初期化には表 4.1.3.1 の GPIO_Init_t 型を使用する。

番号	項目	型	機能
1	mode	uint32_t	対象のピンアサインの設定モード
2	pull	uint32_t	出力 Pull-Up/Pull-Down 設定
3	otype	uint32_t	出力タイプ Open-drain/Push-Pull 設定
4	speed	uint32_t	出力スピード設定
5	alternate	uint32_t	アルタネート設定

表 4.1.3.1 GPIO_Init_t 型

① mode

モードはピンアサインの GPIO モードを設定する

定義	値	内容
GPIO_MODE_INPUT	0x00000000	GPIO 入力モード
GPIO_MODE_IT_RISING	0x10110000	GPIO 入力 EXTI 立ち上がりエッジ割込み
GPIO_MODE_IT_FALLING	0x10210000	GPIO 入力 EXTI 立下りエッジ割込み
GPIO_MODE_IT_RISING_FALLING	0x10310000	GPIO 入力 EXTI 両エッジ割込み
GPIO_MODE_EVT_RISING	0x10120000	GPIO 入力 EXTI 立ち上がりエッジ EVENT
GPIO_MODE_EVT_FALLING	0x10220000	GPIO 入力 EXTI 立下りエッジ EVENT
GPIO_MODE_EVT_RISING_FALLING	0x10320000	GPIO 入力 EXTI 両エッジ EVENT
GPIO_MODE_OUTPUT_PP	0x00000001	GPIO 出力モード
GPIO_MODE_AF_PP	0x00000002	アルタネートピン設定
GPIO_MODE_ANALOG	0x00000003	アナログピン設定
GPIO_MODE_ANALOG_AD	0x00000007	ADC アナルグピン(STM32L4xx のみ)

表 4.1.3.2 mode 設定値

② pull

pull はピンアサインが出力モード設定の場合、Pull-Up/Pull-Down の端子設定を行う。

定義	値	内容
GPIO_NOPULL	0x00000000	Pull-Up/Down 設定を行わない
GPIO_PULLUP	0x00000001	Pull-Up 設定
GPIO_PULLDOWN	0x00000002	Pull-Down 設定

表 4.1.3.3 pull 設定値

③ otype

otype はピンアサインが出力モードの場合、Open-Drain/Push-Pull の端子設定を行う

定義	値	内容
GPIO_OTYPE_PP	0x00000000	Push-Pull 設定
GPIO_OTYPE_OD	0x00000001	Open-Drain 設定

表 4.1.3.4 otype 設定値

④ speed

speed はピンアサインが出力モードの場合、出力周波数の設定を行う。出力周波数はデバイス毎に最適値がある。

定義	値	内容
GPIO_SPEED_LOW	0x00000000	低速
GPIO_SPEED_MEDIUM	0x00000001	中間

GPIO_SPEED_FAST	0x00000002	速い
GPIO_SPEED_HIGH	0x00000003	最高

表 4.1.3.5 speed 設定値

⑤ altanate
アルタネートはピン設定をデバイスの入出力ピンとして使用する場合のアルタネート値を設定する。モードが GPIO_MODE_AF_PP の場合のみ有効となる。

4.1.3.2 インターフェイス仕様

GPIO を初期設定するドライバ関数を以下に示す。

関数名	型	引数	機能	備考
gpio_setup	void	uint32_t base GPIO_Init_t *init uint32_t pin	base アドレスと pin 番号で指定されたポートを初期化する。	アルタネートの設定デバイスピンの初期化の場合もある

表 4.1.3.6 GPIO 設定関数

4.1.3.3 フローチャート

基本的な GPIO の出力設定のフローチャートを以下に示す。

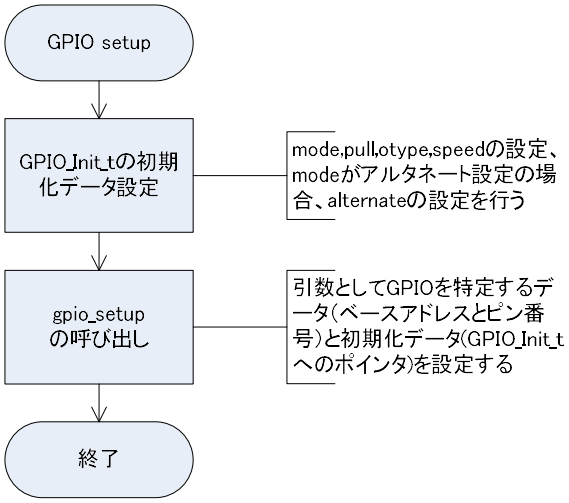


図 4.1.1.3.1 GPIO の設定フローチャート

4.1.4 DMA

DMA は CPU を通さず、直接メモリとデバイスまたはメモリ間でデータ転送を行う機構である。デバイスとメモリ間で高速にデータ通信した場合、オーバーラン・エラーやアンダーラン・エラーが発生するケースが多く、Standard Driver では DMA を使用している。

4.1.4.1 データ仕様

DMA ドライバは初期設定用に DMA_Init_t 型、制御を行うためにハンドラとして使用する DMA_Handle_t 型の二つの型を持つ。STM32F4xx / STMF7xx と STM32L4xx / STM32F0xx / STM32L0xx では IP が異なるため、構造体に若干の差異がある。

番号	項目	型	機能
1	Channel	uint32_t	DMA のチャネル番号
2	Direction	uint32_t	転送方向

3	PeriphInc	uint32_t	ペリフェラルインクリメントモード
4	MemInc	uint32_t	メモリインクリメントモード
5	PeriphDataAlignment	uint32_t	ペリフェラルのデータアラインメントを設定する
6	MemDataAlignment	uint32_t	メモリのデータアラインメントを設定する
7	Mode	uint32_t	転送モード設定
8	Priority	uint32_t	優先順位設定
9	FIFOMode	uint32_t	FIFO の有効、無効を設定
10	FIFOThreshold	uint32_t	FIFO のスレッシュホールドを設定
11	MemBurst	uint32_t	メモリ側のバースト設定
12	PeriphBurst	uint32_t	ペリフェラル側のバースト設定

表 4.1.4.1 DMA_Init_t 型(STM32F4xx/STM32F7xx)

番号	項目	型	機能
1	Request	uint32_t	リクエストチャネル番号
2	Direction	uint32_t	転送方向
3	PeriphInc	uint32_t	ペリフェラルインクリメントモード
4	MemInc	uint32_t	メモリインクリメントモード
5	PeriphDataAlignment	uint32_t	ペリフェラルのデータアラインメントを設定する
6	MemDataAlignment	uint32_t	メモリのデータアラインメントを設定する
7	Mode	uint32_t	転送モード設定
8	Priority	uint32_t	優先順位設定

表 4.1.4.2 DMA_Init_t 型(STM32F0xx/STM32L0xx/STM32L4xx)

① Channel

STMF4/F7 では２つの DMA で各 8 つのチャネルが使用可能である。

定義	値	内容
DMA_CHANNEL_0	0x00000000	チャネル０
DMA_CHANNEL_1	0x02000000	チャネル１
DMA_CHANNEL_2	0x04000000	チャネル２
DMA_CHANNEL_3	0x06000000	チャネル３
DMA_CHANNEL_4	0x08000000	チャネル４
DMA_CHANNEL_5	0x0A000000	チャネル５
DMA_CHANNEL_6	0x0C000000	チャネル６
DMA_CHANNEL_7	0x0E000000	チャネル７

表 4.1.4.3 Channel 設定値

② Dirction

Dirction はデータ転送の種別と転送方向を指定する。

定義	値	内容
DMA_PERIPH_TO_MEMORY	0x00000000	メモリからペリフェラルへの転送
DMA_MEMORY_TO_PERIPH	DMA_SxCR_DIR_0	ペリフェラルからメモリへの転送
DMA_MEMORY_TO_MEMORY	DMA_SxCR_DIR_1	メモリからメモリへの転送

表 4.1.4.4 Direction 設定値

③ PeriphInc

ペリフェラル転送時、インクリメントモード設定。

定義	値	内容
DMA_PINC_ENABLE	DMA_SxCR_PINC	ペリフェラルインクリメントモード有効
DMA_PINC_DISABLE	0x00000000	ペリフェラルインクリメントモード無効

表 4.1.4.5 PeriphInc 設定値

④ MemInc
メモリ転送時インクリメントモード

定義	値	内容
DMA_MINC_ENABLE	DMA_SxCR_MINC	メモリインクリメントモード有効
DMA_MINC_DISABLE	0x00000000	メモリインクリメントモード無効

表 4.1.4.6 MemInc 設定値

⑤ PeriphDataAlignment
PeriphDataAlignment はペリフェラル側のデータアラインメントを設定する。

定義	値	内容
DMA_PDATAALIGN_BYTE	0x00000000	アラインメントバイト設定
DMA_PDATAALIGN_HALFWORD	DMA_SxCR_PSIZE_0	アラインメント２バイト設定
DMA_PDATAALIGN_WORD	DMA_SxCR_PSIZE_1	アラインメント４バイト設定

表 4.1.4.7 PeriphDataAlignment 設定値

⑥ MemDataAlignment
MemDataAlignment はメモリ側のデータアラインメントを設定する。

定義	値	内容
DMA_MDATAALIGN_BYTE	0x00000000	アラインメントバイト設定
DMA_MDATAALIGN_HALFWORD	DMA_SxCR_MSIZE_0	アラインメント２バイト設定
DMA_MDATAALIGN_WORD	DMA_SxCR_MSIZE_1	アラインメント４バイト設定

表 4.1.4.8 MemDataAlignment 設定値

⑦ Mode
アルタネートはピン設定をデバイスの入出力ピンとして使用する場合のアルタネート値を設定する。モードが GPIO_MODE_AF_PP の場合のみ有効となる。

定義	値	内容
DMA_NORMAL	0x00000000	ノーマルモード
DMA_CIRCULAR	DMA_SxCR_CIRC	サーキュラーモード
DMA_PFCTRL	DMA_SxCR_PFCTRL	ペリフェラルフロー制御モード

表 4.1.4.9 Mode 設定値

⑧ Priority
DMA の実行優先度設定を行う。

定義	値	内容
DMA_PRIORITY_LOW	0x00000000	低い
DMA_PRIORITY_MEDIUM	DMA_SxCR_PL_0	中間
DMA_PRIORITY_HIGH	DMA_SxCR_PL_1	高い
DMA_PRIORITY_VERY_HIGH	DMA_SxCR_PL	非常に高い

表 4.1.4.10 Priority 設定値

⑨ FIFOMode
DMA の FIFO の有効、無効を設定する。

定義	値	内容
DMA_FIFOMODE_DISABLE	0x00000000	DMA FIFO 無効
DMA_FIFOMODE_ENABLE	DMA_SxFCR_DMDIS	DMA FIFO 有効

表 4.1.4.11 FIFOmode 設定値

⑩ FIFOThreshold
FIFO のスレシュホールドを設定する。

定義	値	内容
DMA_FIFO_THRESHOLD_1QUARTERFULL	0x00000000	1/4
DMA_FIFO_THRESHOLD_HALFFULL	DMA_SxFCR_FTH_0	1/2
DMA_FIFO_THRESHOLD_3QUARTERSFULL	DMA_SxFCR_FTH_1	3/4
DMA_FIFO_THRESHOLD_FULL	DMA_SxFCR_FTH	フル

表 4.1.4.12 FIFOthreshold 設定値

⑪ MemBurst
メモリ側の DMA バースト設定

定義	値	内容
DMA_MBURST_SINGLE	0x00000000	シングル
DMA_MBURST_INC4	DMA_SxCR_MBURST_0	インクリメント４
DMA_MBURST_INC8	DMA_SxCR_MBURST_1	インクリメント８
DMA_MBURST_INC16	DMA_SxCR_MBURST	インクリメント１６

表 4.1.4.13 MemBurst 設定値

⑫ PeriphBurst
ペリフェラル側の DMA バースト設定

定義	値	内容
DMA_PBURST_SINGLE	0x00000000	シングル
DMA_PBURST_INC4	DMA_SxCR_PBURST_0	インクリメント４
DMA_PBURST_INC8	DMA_SxCR_PBURST_1	インクリメント８
DMA_PBURST_INC16	DMA_SxCR_PBURST	インクリメント１６

表 4.1.4.14 PeriphBurst 設定値

番号	項目	型	機能
1	base	uint32_t	DMA ストリームコントローラのベースアドレス
2	Init	DMA_Init_t	DMA 初期化情報
3	sdid	uint32_t	DMA ストリーム ID
4	xfercallback	void *func	転送終了時のコールバック関数
5	xferhalfcallback	void *func	半分転送終了時のコールバック関数
6	xferm1callback	void *func	Mem1 転送用コールバック関数
7	errorcallback	void *func	エラー発生時のコールバック関数
8	ErrorCode	uint32_t	エラーコード
9	localdata	void*	ローカル領域へのポインタ

表 4.1.4.15 DMA_Handle_t 型(STM32Fxx/STM32F7xx

番号	項目	型	機能
1	base	uint32_t	DMA ポートのベースアドレス（自動設定）

2	cbase	uint32_t	DMA チャネルのベースアドレス
3	Init	DMA_Init_t	DMA 初期化情報
4	chid	uint32_t	DMA チャネル ID（自動設定）
5	status	uint32_t	DMA ステータス
6	xfercallback	void *func	転送終了時のコールバック関数
7	xferhalfcallback	void *func	半分転送終了時のコールバック関数
8	xferm1callback	void *func	Mem1 転送用コールバック関数
9	errorcallback	void *func	エラー発生時のコールバック関数
10	ErrorCode	uint32_t	エラーコード
11	localdata	void*	ローカル領域へのポインタ

表 4.1.4.16 DMA_Handle_t 型(STM32L40xx/STM32F0xx/STM32L0xx)

① sdid/chid

STM32F4xx/STM32F7xx では DMA ストリーム ID を使用する。これの値は初期化時 base より自動設定される。DMA ストリームコントローラのシーケンシャルな番号である。

定義	値	内容
DMA1STM0_SID	(0)	DMA1 STREAM0
DMA1STM1_SID	(1)	DMA1 STREAM1
DMA1STM2_SID	(2)	DMA1 STREAM2
DMA1STM3_SID	(3)	DMA1 STREAM3
DMA1STM4_SID	(4)	DMA1 STREAM4
DMA1STM5_SID	(5)	DMA1 STREAM5
DMA1STM6_SID	(6)	DMA1 STREAM6
DMA1STM7_SID	(7)	DMA1 STREAM7
DMA2STM0_SID	(8+0)	DMA2 STREAM0
DMA2STM1_SID	(8+1)	DMA2 STREAM1
DMA2STM2_SID	(8+2)	DMA2 STREAM2
DMA2STM3_SID	(8+3)	DMA2 STREAM3
DMA2STM4_SID	(8+4)	DMA2 STREAM4
DMA2STM5_SID	(8+5)	DMA2 STREAM5
DMA2STM6_SID	(8+6)	DMA2 STREAM6
DMA2STM7_SID	(8+7)	DMA2 STREAM7

表 4.1.4.17 sdid 設定値

STM32L4xx/STM32F0xx/STM32L0xx ではチャネル ID を使用する。この値は初期化時 cbase より自動設定される。

定義	値	内容
DMA1CH1_ID	(0)	DMA1 CHANNEL1
DMA1CH2_ID	(1)	DMA1 CHANNEL2
DMA1CH3_ID	(2)	DMA1 CHANNEL3
DMA1CH4_ID	(3)	DMA1 CHANNEL4
DMA1CH5_ID	(4)	DMA1 CHANNEL5
DMA1CH6_ID	(5)	DMA1 CHANNEL6
DMA1CH7_ID	(6)	DMA1 CHANNEL7
DMA2CH1_ID	(7+0)	DMA2 CHANNEL1
DMA2CH2_ID	(7+1)	DMA2 CHANNEL2
DMA2CH3_ID	(7+2)	DMA2 CHANNEL3
DMA2CH4_ID	(7+3)	DMA2 CHANNEL4
DMA2CH5_ID	(7+4)	DMA2 CHANNEL5
DMA2CH6_ID	(7+5)	DMA2 CHANNEL6
DMA2CH7_ID	(7+6)	DMA2 CHANNEL7

表 4.1.4.18 chid 設定値

② xfercallback
DMA Mem0 転送終了時のコールバック関数

③ xferhalfcallback
DMA Mem0 half 転送終了時のコールバック関数

④ xferm1callback
DMA Mem1 half 転送終了時のコールバック関数

⑤ errorcallback
DMA 転送エラー発生時のコールバック関数

⑥ ErrorCode
DMA のエラー状態

定義	値	内容
DMA_STATUS_BUSY	0x00000001	DMA BUSY 状態
DMA_STATUS_READY_HMEM0	0x00000002	DMA Mem0 half 転送終了
DMA_STATUS_READY_HMEM1	0x00000004	DMA Mem1 half 転送終了
DMA_STATUS_READY_MEM0	0x00000008	DMA Mem0 成功終了
DMA_STATUS_READY_ERROR	0x00000100	DMA エラー終了

表 4.1.4.19 ErrorCode 設定値

⑦ localdata
上位のドライバが自由に設定可能なポインタ領域

4.1.4.2 インターフェイス仕様

DMA を制御するドライバ関数を以下に示す。

関数名	型	引数	機能	備考
dma_init	ER	DMA_Handler_t *hdma	ストリーム DMA の初期化を行う。初期値として DMA_Init_t と base の設定を行う。初期化後、必要に応じてコールバック関数を設定する。	
dma_deinit	ER	DMA_Handler_t *hdma	DMA を未使用状態に戻す	
dma_start	ER	DMA_Handler_t *hdma uint32_t SrcAddr uint32_t DstAddr uint32_t Length	ストリーム DMA をスタートさせる。	
dma_end	ER	DMA_Handler_t *hdma	ストリーム DMA を停止させる。	
dma_inthandler	void	DMA_Handler_t *hdma	ストリーム DMA の割込みハンドラ	asp の割込みサービスルーチンからコールする

表 4.1.4.17 DMA 設定関数

4.1.4.3　　フローチャート

DMA 処理は初期化と転送に分けられる。STM32 の場合、DMA は２つ用意されており、各 DMA に８つまたは７つのチャネルが割り当てられている。DMA を使用するペレフェラルごとに、どの DMA のどのチャネルを使用するかはハード的に決められている。

初期化のフローチャートを図 4.1.4.3.1 に示す。まず、静的 API を用いて DMA 割込みを設定する。上記の通り、DMA 2×チャネル 8 = 16（または 7 = 14）の割込みが設定可能となる。割込みハンドラは割込みサービスルーチン（stream_dma_isr/channel_dma_isr）を使用する。割込みサービスルーチンから割込みハンドラを取り出すために引数に sdid/chid を設定する必要がある。注１次に C 言語等による記載で、静的なデータとして DMA ハンドラを用意して、これを用いて DMA の設定を行う。ベースアドレスとして Stream DMA のベースアドレスをセットし、ハンドラ内の DMA_Init_t の項目を設定したあと、dma_init 関数で初期設定を行う。対象の DMA を使用しない場は、ハンドラへのポインタを引数として dma_deinit 関数コールでペリフェラルを未動作状態に戻す。

注１）割込みの設定は静的 API を使用するため、C 言語に記載ではない。

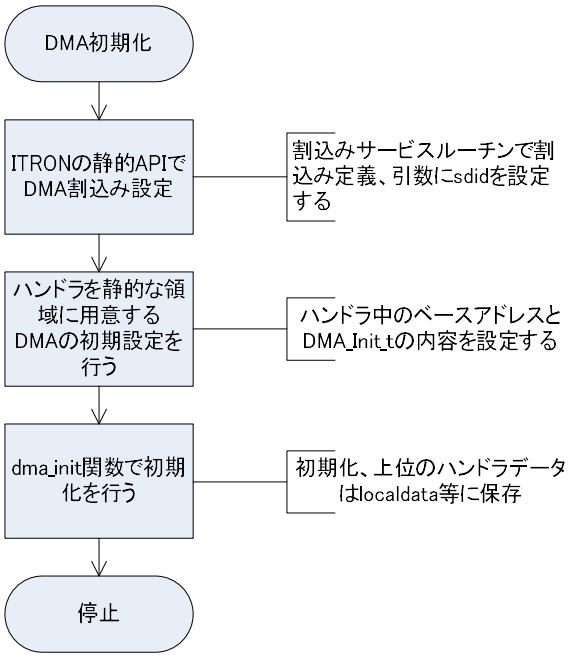


図 4.1.4.3.1 DMA 初期化

DMA 転送開始のフローチャートを図 4.1.4.3.2 に示す。DMA の転送情報やエラー情報は転送開始後の割込みにて通知される。そのため、DMA ハンドラ中にコールバックルーチン設定領域が用意されている。割込みによる状態遷移が発生した場合、割込みサービスルーチンからコールバック関数が呼び出される。コールバック関数中からセマフォ等を用いて DMA を使用しているタスクに対して通知を行う。DMA 転送開始は dma_start 関数を用いて、ペリフェラルにキックをかける。DMA の停止はコールバック関数からの通知を受けてタスク側で転送終了やエラー発生を判定して、DMA の停止処理 dma_end 関数を用いて処理する。

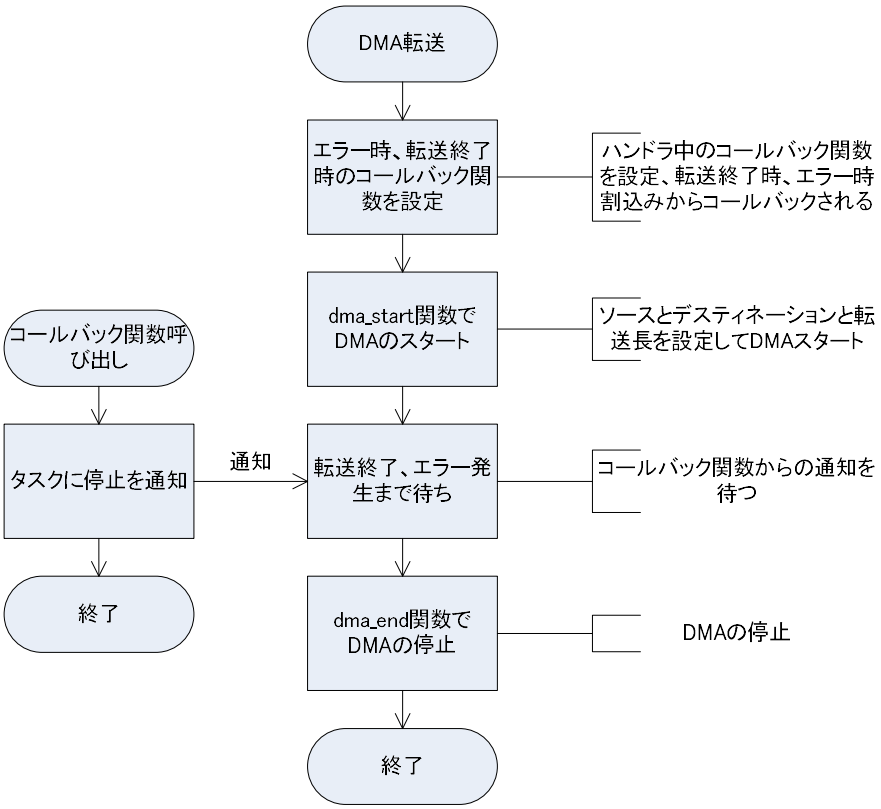


図 4.1.4.3.2 DMA 転送開始

4.1.5 WATCH DOG

WATCH DOG はシステムに異常が発生した場合、リセットを行う機能である。システムは WATCH DOG TIMER を起動し、一定周期でタイマーのリセットを行うことにより WATCH DOG に正常動作を通知する。単一の Window WATCH DOG 機能を持ち、このドライバは Window WATCH DOG Timer の制御を行う。

注意：この機能を使用するには、2018 年 11 月以降にリリースされた TOPPERA ASP カーネルの CortexM 用の個別パッケージと組み合わせて使用する必要がある。

4.1.5.1 データ仕様

Window WATCH DOG は単一であるためポート番号 1 のみ有効である。ドライバ I/F はポート番号を用いて制御を行う。ポートとは別に WATCH DOG コンフィギュレーション型を持ち、初期設定を行う。

番号	項目	値	機能
1	WDOG1_PORTID	1	Window Watch Dog IP を示す

表 4.1.5.1.1 WATCH DOG ポート番号

MCounter は、WATCH DOG で設定したタイムアウトを MCounter 分延長するためのカウンタ。

番号	項目	型	機能
1	Window	uint32_t	ダウンカウンタのコンペア値(0x40～0x7F)
2	Counter	uint32_t	ダウンカウンタの初期値(0x40～0x7F)
3	MCounter	uint32_t	ダウンカウンタ拡張カウント

表 4.1.5.1.2 WATCH DOG コンフィギュレーション型

4.1.5.2　　インターフェイス仕様

WATCH DOG を制御するドライバ関数は以下の通りである。WATCH DOG カウンタのプリスケール設定は 8(PCLK1/8)に固定とする。

関数名	型	引数	機能	備考
wdog_init	ER	ID portid WDOG_Init_t *pini	指定ポート ID の WATCH DOG ペリフェラルを初期化する	
wdog_deinit	ER	ID portid	WATCH DOG を未使用状態に戻す	
wdog_reset	ER	ID portid	タイマーのリセットを行う	

表 4.1.5.2.1 WATCH DOG ドライバ関数

4.1.6 TIMER

TIMER は一般的に使用される周期タイマー割込み以外に特殊な機能を持つものもある。例えば PWM 出力の設定のようなクロック生成にも使用されるため、ドライバの API を統一することは難しい。RTOS のシステムタイマーとして使用する場合は、asp-1.9.3 カーネル内に記述があるため、それを参照して頂きたい。周期タイマーの例として基礎１，２講座で使用するタイマードライバを参照して頂きたい。

4.1.7 UART

UART 用のデバイスドライバは asp-1.9.2 で実装済のデバイスドライバを使用しているため、ここでは記載しない。

4.2　Standard Driver

スタンダードドライバは拡張ボード（シールド）の標準化により、ドライバ API がデファクトスタンダードとなっているドライバを指す。但し、ペリフェラルの実装は標準化されたドライバ API 以上の機能を持つものが多く、ハードウェアを最大限に利用するのは拡張インターフェイスにて拡張を行う必要がある。

4.2.1 概要

TOPPERS BASE PLATFORM として、スタンダードドライバとしたのは、以下の 7 つのペリフェラルである。いずれもインターフェイス用のペリフェラルであり、接続先にセンサー、LCD、GLCD、SD card、ネットワークハードウェア等の機器の制御用に用いられる。個々のハードウェアのドライバは別途上位に GDIC ドライバを用意しなければならない。

① I2C
② SPI
③ ADC
④ RTC
⑤ USB OTG
⑥ QSPI
⑦ USB

スタンダードドライバは、基本的にポート ID を指定してハンドラを取り出しハンドラを用いてペリフェラルを制御する構成を取る。

4.2.2 I2C

I2C（アイ・スクエア・シー）は周辺機との通信用にフィリップス社が開発した低速なシリアルバスである。メインボード側がマスタ、周辺機側がスレーブとなり、スレーブアドレスをキーにデータの送受信を行う。基本的な通信速度は 100kbit/sec の通常モードと 10kbit/sec の低速モードがあるが、基本以上、または、基本以下の速度で通信を行う場合も多い。スレーブアドレスは通常は 7 ビットであるが、拡張として１０ビットのスレーブアドレスも通信可能となっている。

I2C は SCL（クロック）と SDA（データ）の２つの線で通信を行う、周辺機が複数ある場合はこの２つの線を共有する形となる。マスタ側が常に制御権を持っており基本のクロック SCL はマスタ側が設定する。但し、スレーブ側で待ちが必要な場合は、スレーブ側 SCL 信号を Low に落として待ち状態を作る。送信を行う場合は、送信側がクロックに合わせて SDA 上にデータ信号を乗せる。最後の８ビット目で受信側が SDA を Low にした場合は ACK となり、Hi のままならば NACK となる。スレーブアドレス 7bit の一般的なデータ転送を図 4.2.2.1 に示す。

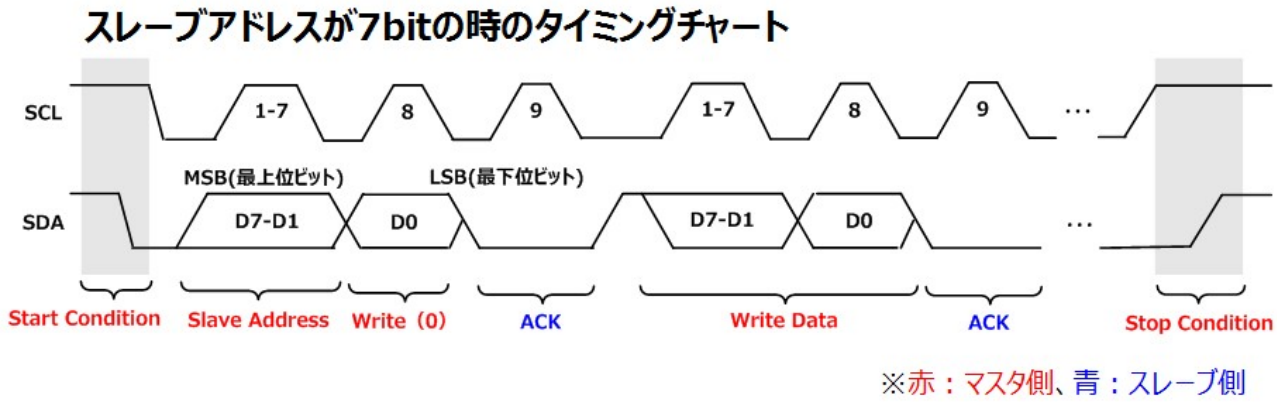


図 4.2.2.1 スレーブアドレス 7bit のデータ転送

4.2.2.1 データ仕様

I2C ドライバは初期化用の型として、表 4.2.2.1.1 と表 4.2.2.1.2 の I2C コンフィギュレーション型と、ハンドラとして表 4.2.2.1.3 の I2C ハンドラ型を持つ。

番号	項目	型	機能
1	ClockSpeed	uint32_t	通信クロックスピード(bps)
2	DutyCycle	uint32_t	ディーティサイクル設定
3	OwnAddress1	uint32_t	スレーブアドレス１(スレーブの場合のみ)
4	AddressingMode	uint32_t	アドレスモード(7bit or 10bit)
5	DualAddressMode	uint32_t	２つのスレーブアドレスを持つかの設定(スレーブの場合のみ)
6	OwnAddress2	uint32_t	スレーブアドレス２(スレーブの場合のみ)
7	GenealCallMode	uint32_t	ジェネラルコールモード設定
8	NoStretchMode	uint32_t	ノーストレッジモード設定
9	semid	int	通信用セマフォ ID（0 でセマフォなし）
10	smlock	int	排他制御用セマフォ ID(0 で排他制御なし)

表 4.2.2.1.1 STM32F4xx I2C コンフィギュレーション型

番号	項目	型	機能
1	Timing	uint32_t	通信クロックタイミング設定
2	OwnAddress1	uint32_t	スレーブアドレス１(スレーブの場合のみ)
3	AddressingMode	uint32_t	アドレスモード(7bit or 10bit)
4	DualAddressMode	uint32_t	２つのスレーブアドレスを持つかの設定(スレーブの場合のみ)
5	OwnAddress2	uint32_t	スレーブアドレス２(スレーブの場合のみ)
6	GenealCallMode	uint32_t	ジェネラルコールモード設定
7	NoStretchMode	uint32_t	ノーストレッジモード設定
8	semid	int	通信用セマフォ ID（0 でセマフォなし）
9	smlock	int	排他制御用セマフォ ID(0 で排他制御なし)

表 4.2.2.1.2 STM32F7xx I2C コンフィギュレーション型

番号	項目	型	機能
1	Timing	uint32_t	通信クロックタイミング設定
2	OwnAddress1	uint32_t	スレーブアドレス 1 (スレーブの場合のみ)
3	AddressingMode	uint32_t	アドレスモード(7bit or 10bit)
4	DualAddressMode	uint32_t	2 つのスレーブアドレスを持つかの設定(スレーブの場合のみ)
5	OwnAddress2Masks	uint32_t	スレーブアドレス 2 のマスク設定
6	OwnAddress2	uint32_t	スレーブアドレス 2 (スレーブの場合のみ)
7	GenealCallMode	uint32_t	ジェネラルコールモード設定
8	NoStretchMode	uint32_t	ノーストレッジモード設定
9	semid	int	通信用セマフォ ID（0 でセマフォなし）
10	smlock	int	排他制御用セマフォ ID(0 で排他制御なし)

表 4.2.2.1.3 STM32F0xx/STM32L0xx/STML4xx I2C コンフィギュレーション型

コンフュギュレーション型は、SoC により若干異なる。クロックスピードの設定を STM32F4xx では、ClockSpeed と DutyCycle で設定するが、STM32F7xx では、ベースクロックからの分周値で設定するように変更されているためである。STM32F0xx/STM32L0xx/STML4xx では、OwnAddress2Mask 設定が追加されている。

semid はセマフォ通信用のセマフォ番号、ゼロで設定なし。このセマフォは割込みとドライバ間の伝達用に使用するため、設定なしの場合、通信遅延が発生する。smlock は、ドライバの排他制御に使用するセマフォ番号を指定する。ゼロの設定で排他制御なしとなる。

番号	項目	型	機能
1	base	uint32_t	I2C ベースアドレス
2	Init	I2C_Init_t	I2C コンフィギュレーション型
3	pBuffPtr	uint8_t *	通信データ領域へのポインタ
4	XferSize	uint16_t	通信バイト数
5	XferCount	volatile uint16_t	通信済みバイト数
6	writecallback	void (*)()	送信終了コールバック
7	readcallback	void (*)()	受信終了コールバック
8	errorcallback	void (*)()	エラーコールバック
9	i2cid	ID	I2C ポート ID
10	status	volatile uint32_t	I2C ドライバの状態
11	ErrorCode	volatile uint32_t	I2C エラーコード

表 4.2.2.1.4 I2C ハンドラ型(STM32F4xx/STM32F7xx)

番号	項目	型	機能
1	base	uint32_t	I2C ベースアドレス
2	Init	I2C_Init_t	I2C コンフィギュレーション型
3	pBuffPtr	uint8_t *	通信データ領域へのポインタ
4	XferOptions	uint32_t	通信オプション
5	XferSize	uint16_t	通信バイト数
6	XferCount	volatile uint16_t	通信済みバイト数
7	XferCount2	volatile uint16_t	通信済みバイト数
8	AddrEventCount	uint32_t	ADDR イベントカウンタ
9	writecallback	void (*)()	送信終了コールバック
10	readcallback	void (*)()	受信終了コールバック

11	listencallbak	void (*)()	LISTEN コールバック（スレーブ専用）
12	addrcallback	void (*)()	ADDR コールバック（スレーブ専用）
13	errorcallback	void (*)()	エラーコールバック
14	i2cid	ID	I2C ポート ID
15	status	volatile uint32_t	I2C ドライバの状態
16	ErrorCode	volatile uint32_t	I2C エラーコード

表 4.2.2.1.5 I2C ハンドラ型(STM32F0xx/STM32L0xx/STM32L4xx)

① DutyCycle
デューティサイクル設定。(STM32F4xx のみ)

定義	値	内容
I2C_DUTYCYCLE_2	0x00000000	
I2C_DUTYCYCLE_16_9	I2C_CCR_DUTY	

表 4.2.2.1.6 DutyCycle 設定値

② AddressingMode
I2C のアドレッシング・モード、STM32F4xx と STM32F7xx では値が異なる。

定義	値(32f4xx)	内容
I2C_ADDRESSINGMODE_7BIT	0x00004000	7 ビットモード
I2C_ADDRESSINGMODE_10BIT	0x0000C000	10 ビットモード

表 4.2.2.1.7 AddressingMode 設定値

③ DualAddressMode
デュアルアドレスモード設定、STM32F4xx と STM32F746 では値が異なる。

定義	値(32f4xx)	内容
I2C_DUALADDRESS_DISABLE	0x00000000	デュアルモード無効
I2C_DUALADDRESS_ENABLE	I2C_OAR2_ENDUAL	デュアルモード有効

表 4.2.2.1.8 DualAddressMode 設定値

④ GeneralCallMode
ジェネラルコールモード設定、STM32F4xx と STM32F746 では値が異なる。

定義	値(32f4xx)	内容
I2C_GENERALCALL_DISABLE	0x00000000	ジェネラルコールモード無効
I2C_GENERALCALL_ENABLE	I2C_CR1_ENGC	ジェネラルコールモード有効

表 4.2.2.1.9 GeneralCallMode 設定値

⑤ NoStretchMode
ノーストレッジモード設定、STM32F4xx と STM32F746 では値が異なる。

定義	値	内容
I2C_NOSTRETCH_DISABLE	0x00000000	ノーストレッジモード無効
I2C_NOSTRETCH_ENABLE	I2C_CR1_NOSTRETCH	ノーストレッジモード有効

表 4.2.2.1.10 NoStretchMode 設定値

4.2.2.2 インターフェイス仕様

I2C を制御するドライバ関数は以下の通りである。

関数名	型	引数	機能	備考
i2c_init	I2C_Handler*	ID portid I2C_Init_t *ii2c	指定ポート ID の I2C ペリフェラルを初期化し、ハンドラへのポインタを返す	
i2c_deinit	ER	I2C_Handler* hi2c	I2C を未使用状態に戻す	
i2c_slavewrite	ER	I2C_Handler* hi2c uint8_t *pData uint16_t Size	スレーブモードのデータ送信	
i2c_slaveread	ER	I2C_Handler* hi2c uint8_t *pData uint16_t Size	スレーブモードのデータ受信	
i2c_memwrite	ER	I2C_Handler* hi2c uint16_t DevAddr uint16_t MemAddr uint16_t MemAddSize uint8_t *pData uint16_t Size	マスターモードのデータ送信、MemAddSize をゼロにするとアドレス設定を行わない	
i2c_memread	ER	I2C_Handler* hi2c uint16_t DevAddr uint16_t MemAddr uint16_t MemAddSize uint8_t *pData uint16_t Size	マスターモードのデータ受信、MemAddSize をゼロにするとアドレス設定を行わない	
i2c_ev_handler	void	I2C_Handler* hi2c	I2C イベント割込みハンドラ関数	
i2c_er_handler	void	I2C_Handler* hi2c	I2C エラー割込みハンドラ関数	
i2c_ev_isr	void	intptr_t exinf	I2C イベント割込みサービスルーチン	F7/F4
i2c_er_isr	void	intptr_t exinf	I2C エラー割込みサービスルーチン	F7/F4
i2c_isr	void	intptr_t exinf	I2C 割込み	F0/L0

表 4.2.2.2.1 I2C ドライバ関数

4.2.2.3 フローチャート

I2C の初期設定は、i2c_init 関数を指定して対象ポート番号と初期設定したコンフィギュレーション構造体のポインタを指定します。BASE PLATFORM を使用する環境ではマスタとして使用しますので、マスタからペリフェラルとの通信方法について記載します。基本的に、このドライバでは割込みを使用してデータの送受信を行う。また、スタート、ストップ、ACK 処理はペリフェラル側で処理するので、マスタの場合、スレーブアドレスと送受信するデータ領域へのポインタと転送サイズを指定する。スレーブの場合、初期化でスレーブアドレスをセットして、送受信関数でバッファの設定を行う。PLATFORM はスレーブなることはないと思われますのでスレーブの説明は行いません。

図 4.2.2.3.1 に初期化のフローチャートを示します。i2c_init で取得した I2C ハンドラへのポインタは以後、I2C の制御用に使用する。

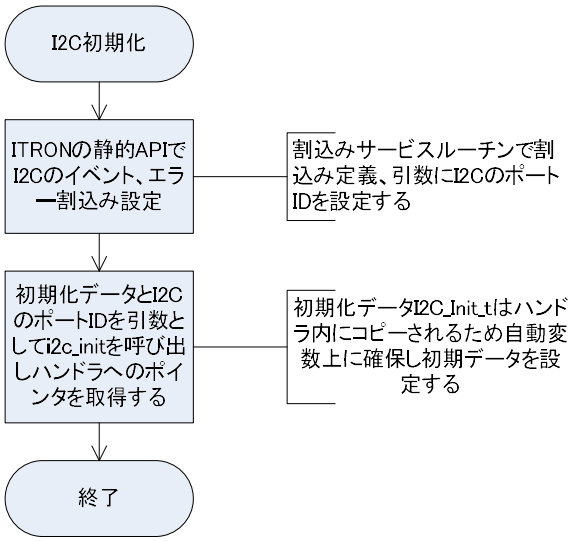


図 4.2.2.3.1 初期化フローチャート

図 4.2.2.3.2 にマスタのデータ送信のフローチャート、図 4.2.2.3.3 に受信のフローチャートを示す。送信ならば i2c_memwrite、受信ならば i2c_memread 関数を呼び出せば指定サイズの送受信が行える。送受信の結果は、関数の戻り値確認できる。E_OK 以外の戻り値の場合エラー処理を行ってください。ペリフェラルには、データアドレスを持つもの（EEPROM や RTC など）があり、送受信のとき、データアドレスの設定を行う必要がある。この場合、MemAddr でデータアドレス、MemAddSize でデータアドレスのバイトサイズを指定する。データアドレスの設定を行わない場合は、MemAddrSize をゼロして、送受信関数を呼び出す。

I2C ペリフェラルを終了させたい場合は、引数として I2C ハンドラへのポインタを指定して i2c_deinit 関数を呼び出せば、ペリフェラルとハンドラは未使用状態に戻ります。

I2C の割込みはイベント割込みとエラー割込みがあり、イベント割込みは I2C 内部のデータ遷移用に使用され、エラー発生時のみエラー割込みが発生します。エラー内容は、ハンドラの ErrorCode に設定される。ErrorCode がゼロの場合はエラーなしで、エラーが発生した場合の ErrorCode の値は表 4.2.2.3.1 の ErrorCode の内容に示す。

定義	値	内容
I2C_ERROR_NONE	0x00000000	エラーなし
I2C_ERROR_BERR	0x00000001	バスエラー
I2C_ERROR_ARLO	0x00000002	アービテーション・ロス・エラー
I2C_ERROR_AF	0x00000004	ACK フォルトエラー
I2C_ERROR_OVR	0x00000008	オーバーランまたはアンダーラン・エラー
I2C_ERROR_DMA	0x00000010	DMA 転送エラー（未実装）
I2C_ERROR_TIMEOUT	0x00000020	転送タイムアウト（未実装）
I2C_ERROR_SIZE	0x00000040	転送サイズエラー（未実装）

表 4.2.2.3.1 ErrorCode の内容

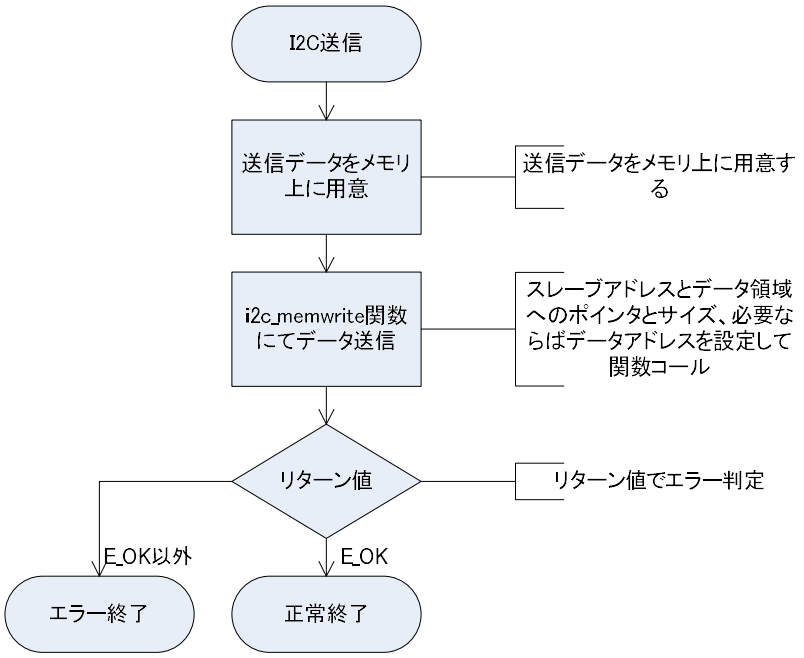


図 4.2.2.3.2 I2C 送信フローチャート

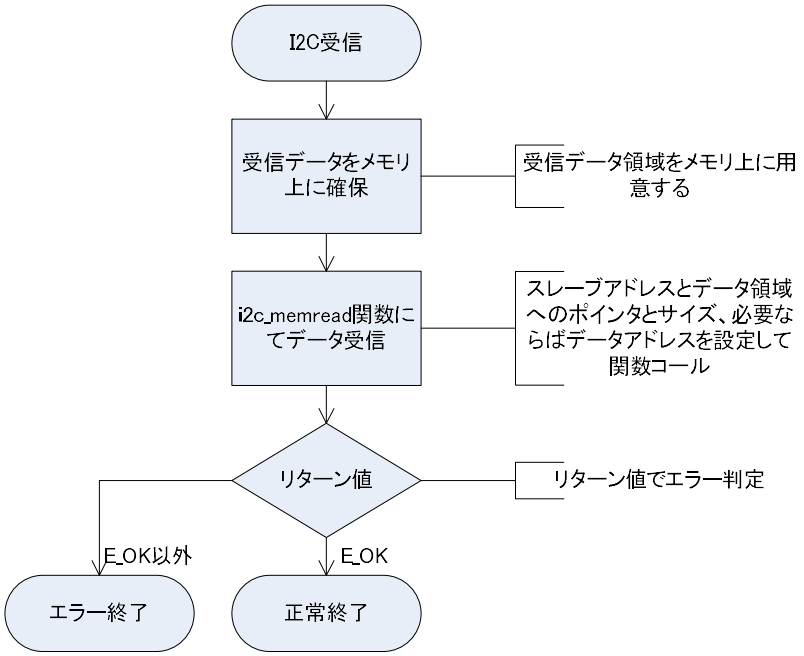


図 4.2.2.3.3 I2C 受信フローチャート

4.2.3 SPI

SPI はシリアル・ペリフェラル・インターフェイスの略で、I2C と同様にペリフェラル間の通信規格である。I2C が高速でも 400kbps であるのに比べ SPI は 1Mbps から 20Mbps まで高速転送が可能である。特に受信で使用する場合、ポーリングや割込みではオーバーラン・エラーとなってしまう場合が多い。SPI は 4 本の信号で通信を行う。スレーブが複数ある場合は、SS(Slave Select)を LOW にしたスレーブに対して通信を行う。そのため、SS 信号はスレーブの数だけ必要となる。また、双方向通信時は MISO と MOSI は同時にデータ通信するので、同時にデータ交換が行われる形となる。

① SCLK　　クロック信号
② MISO　　スレーブからのデータ信号
③ MOSI　　マスタからのデータ信号

④ SS　　スレーブのセレクト信号

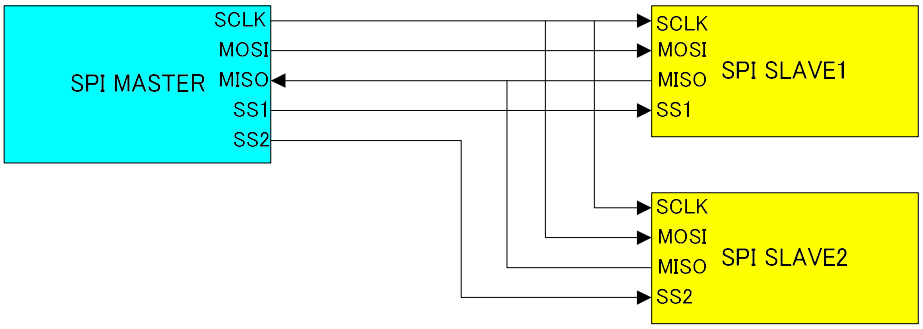


図 4.2.3.1 SPI 接続図

SPI 通信は、クロックの論理（正と負）、クロックに対するデータ設定タイミングにより 4 つのモードのデータ・タイミングが定義されている。

① モード 0 ：正パルス、前縁ラッチ、後端シフト
② モード 1 ：正パルス、前縁シフト、後端ラッチ
③ モード 2 ：負パルス、前縁ラッチ、後端シフト
④ モード 3 ：負パルス、前縁シフト、後端ラッチ

図 4.2.3.2 はもっとも一般的なモード 0 の動作タイミングを示す。

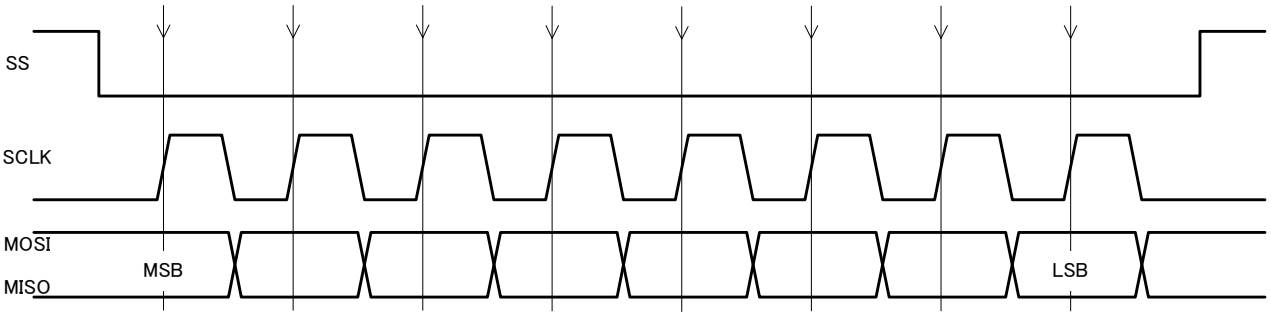


図 4.2.3.2 モード 0 、正パルス、前縁ラッチ、後端シフトのタイミング図

4.2.3.1 データ仕様

SPI ドライバは初期化用の型として、表 4.2.3.1.1 と表 4.2.3.1.2 の SPI コンフィギュレーション型と、ハンドラとして表 4.2.3.1.3 の SPI ハンドラ型を持つ。

番号	項目	型	機能
1	Mode	uint32_t	SPI マスタースレーブ設定
2	Direction	uint32_t	SPI 転送方向
3	DataSize	uint32_t	SPI 転送データサイズ
4	CLKPolarity	uint32_t	SPI 転送クロックの極性
5	CLKPhase	uint32_t	SPI クロック位相
6	NSS	uint32_t	SPI NSS
7	Prescaler	uint32_t	SPI クロック分周設定
8	SignBit	uint32_t	SPI MSB/LSB 設定
9	TIMode	uint32_t	SPI TI モード
10	CRC	uint32_t	SPI CRC 演算設定
11	CRCPolynoial	uint32_t	SPI CRC 多項式設定
12	semid	int	通信用セマフォ ID（0 でセマフォなし）
13	smlock	int	排他制御用セマフォ ID(0 で排他制御なし)

表 4.2.3.1.1 STM32F4xx/STM32L0xx SPI コンフィギュレーション型

番号	項目	型	機能
1	Mode	uint32_t	SPI マスタースレーブ設定
2	Direction	uint32_t	SPI 転送方向
3	DataSize	uint32_t	SPI 転送データサイズ
4	CLKPolarity	uint32_t	SPI 転送クロックの極性
5	CLKPhase	uint32_t	SPI クロック位相
6	NSS	uint32_t	SPI NSS
7	Prescaler	uint32_t	SPI クロック分周設定
8	SignBit	uint32_t	SPI MSB/LSB 設定
9	TIMode	uint32_t	SPI TI モード
10	CRC	uint32_t	SPI CRC 演算設定
11	CRCPolynoial	uint32_t	SPI CRC 多項式設定
12	CRCLength	uint32_t	SPI CRC 長
13	GpioCLKPull	int32_t	SPI クロックの Pull 属性を指定(*)
14	GpioDataPull	int32_t	SPI データの Pull 属性を指定(*)
15	TxDMAChannel(Stream)	int32_t	SPI の TX-DMA のチャネルを指定(*)
16	RxDMAChannel(Stream)	int32_t	SPI の RX-DMA のチャネルを指定(*)
17	semid	int	通信用セマフォ ID（0 でセマフォなし）
18	smlock	int	排他制御用セマフォ ID(0 で排他制御なし)

表 4.2.3.1.2 STM32F7xx SPI コンフィギュレーション型(*STM32H7xx/STM32G4xx のみ)

番号	項目	型	機能
1	Mode	uint32_t	SPI マスタースレーブ設定
2	Direction	uint32_t	SPI 転送方向
3	DataSize	uint32_t	SPI 転送データサイズ
4	CLKPolarity	uint32_t	SPI 転送クロックの極性
5	CLKPhase	uint32_t	SPI クロック位相
6	NSS	uint32_t	SPI NSS
7	Prescaler	uint32_t	SPI クロック分周設定
8	SignBit	uint32_t	SPI MSB/LSB 設定
9	TIMode	uint32_t	SPI TI モード
10	CRC	uint32_t	SPI CRC 演算設定
11	CRCPolynoial	uint32_t	SPI CRC 多項式設定
12	CRCLength	uint32_t	SPI CRC 長
13	NSSPMode	uint32_t	SPI NSSP モード
13	semid	int	通信用セマフォ ID（0 でセマフォなし）
14	smlock	int	排他制御用セマフォ ID(0 で排他制御なし)

表 4.2.3.1.3 STM32F0xx/STM32L4xx SPI コンフィギュレーション型

コンフュギュレーション型は、STM32F4xx と STM32F746 で若干異なる。STM32F746 の方が SPI ペリフェラルに若干の拡張がある。

semid はセマフォ通信用のセマフォ番号、ゼロで設定なし。このセマフォは割込みとドライバ間の伝達用に使用するため、設定なしの場合、通信遅延が発生する。smlock は、ドライバの排他制御に使用するセマフォ番号を指定する。ゼロの設定で排他制御なしとなる。

番号	項目	型	機能
1	base	uint32_t	SPI ベースアドレス

2	Init	SPI_Init_t	SPI コンフィギュレーション型
3	pTxBuffPtr	uint8_t *	送信データ領域へのポインタ
4	TxXferSize	uint16_t	送信バイト数
5	TxXferCount	uint16_t	送信済みバイト数
6	pRxBuffPtr	uint8_t *	受信データ領域へのポインタ
7	RxXferSize	uint16_t	受信バイト数
8	RxXferCount	uint16_t	受信済みバイト数
9	hdmatx	DMA_Handle_t*	送信用 DMA ハンドラ
10	hdmarx	DMA_Handle_t*	受信用 DMA ハンドラ
11	xmode		SPI 転送モード
12	status	volatile uint32_t	SPI ドライバの状態
13	ErrorCode	volatile uint32_t	SPI エラーコード

表 4.2.3.1.4 SPI ハンドラ型

① Mode

SPI のモード設定。

定義	値	内容
SPI_MODE_SLAVE	0x00000000	スレーブモード
SPI_MODE_MASTER	SPI_CR1_MSTR|SPI_CR1_SSI	マスターモード

表 4.2.3.1.5 Mode 設定値

② Direction

SPI 通信方向設定のモード設定。

定義	値	内容
SPI_DIRECTION_2LINES	0x00000000	２ラインモード
SPI_DIRECTION_2LINES_RXONLY	SPI_CR1_RXONLY	２ライン受信のみ
SPI_DIRECTION_1LINE	SPI_CR1_BIDIMODE	片方向モード

表 4.2.3.1.6 Direction 設定値

③ DataSize

SPI データ転送ビットサイズ、STM32F4xx と STM32F746 で設定値が異なる。

定義	値(STM32F4xx)	内容
SPI_DATASIZE_8BIT	0x00000000	データサイズ８ビット
SPI_DATASIZE_16BIT	SPI_CR1_DFF	データサイズ１６ビット

表 4.2.3.1.7 Direction 設定値

④ CLKPolarity

SPI 転送クロック極性定義。

定義	値	内容
SPI_POLARITY_LOW	0x00000000	極性 LOW
SPI_POLARITY_HIGH	SPI_CR1_CPOL	極性 HIGH

表 4.2.3.1.8 CLKPolarity 設定値

⑤ CLKPhase

SPI 転送クロック位相定義。

定義	値	内容
SPI_PHASE_1EDGE	0x00000000	前縁ラッチ
SPI_PHASE_2EDGE	SPI_CR1_CPHA	後縁ラッチ

表 4.2.3.1.9 CLKPhase 設定値

⑥ NSS

SPI NSS 定義。

定義	値	内容
SPI_NSS_SOFT	SPI_CR1_SSM	
SPI_NSS_HARD_INPUT	0x00000000	
SPI_NSS_HARD_OUTPUT	0x00040000	

表 4.2.3.1.10 NSS 設定値

⑦ Prescaler
SPI クロック分周比設定値。

定義	値	内容
SPI_BAUDRATEPRESCALER_2	0x00000000	2 分周
SPI_BAUDRATEPRESCALER_4	0x00000008	4 分周
SPI_BAUDRATEPRESCALER_8	0x00000010	8 分周
SPI_BAUDRATEPRESCALER_16	0x00000018	16 分周
SPI_BAUDRATEPRESCALER_32	0x00000020	32 分周
SPI_BAUDRATEPRESCALER_64	0x00000028	64 分周
SPI_BAUDRATEPRESCALER_128	0x00000030	128 分周
SPI_BAUDRATEPRESCALER_256	0x00000038	256 分周

表 4.2.3.1.11 Prescaler 設定値

⑧ SignBit
SPI データ MSB/LSB 設定値。

定義	値	内容
SPI_DATA_MSB	0x00000000	転送データ MSB
SPI_DATA_LSB	SPI_CR1_LSBFIRST	転送データ LSB

表 4.2.3.1.12 SignBit 設定値

⑨ TIMode
SPI NSS 定義。

定義	値	内容
SPI_TIMODE_DISABLE	0x00000000	TI モード無効
SPI_TIMODE_ENABLE	SPI_CR2_FRF	TI モード有効

表 4.2.3.1.13 TIMode 設定値

⑩ CRC
SPI 自動 CRC 設定。

定義	値	内容
SPI_CRC_DISABLE	0x00000000	自動 CRC 無効
SPI_CRC_ENABLE	SPI_CR1_CRCEN	自動 CRC 有効

表 4.2.3.1.14 CRC 設定値

⑪ CRCLength
SPI CRC 長設定、STM32F746 のみ。

定義	値	内容
SPI_CRC_LENGTH_DATASIZE	0x00000000	データサイズと同様
SPI_CRC_LENGTH_8BIT	0x00000001	8 ビット
SPI_CRC_LENGTH_16BIT	0x00000002	16 ビット

表 4.2.3.1.15 CRCLength 設定値

4.2.3.2 インターフェイス仕様

SPI を制御するドライバ関数は以下の通りである。SS の設定は、GPIO を使って別途制御しなければ

ならない。

関数名	型	引数	機能	備考
spi_init	SPI_Handler*	ID portid SPI_Init_t *spii	指定ポート ID の SPI ペリフェラルを初期化し、ハンドラへのポインタを返す	
spi_deinit	ER	SPI_Handler* hspi	SPI を未使用状態に戻す	
spi_transmit	ER	SPI_Handler* hspi uint8_t *pdata uint16_t length	SPI 送信を行う	
spi_receive	ER	SPI_Handler* hspi uint8_t *pdata uint16_t length	SPI 受信を行う	
spi_transrecv	ER	SPI_Handler* hspi uint8_t *ptxData uint8_t *prxData uint16_t length	SPI 送受信を行う	
spi_wait	ER	SPI_Handler* hspi	SPI 転送の終了待ちを行う	
spi_handler	void	SPI_Handler* hspi	SPI 割込みハンドラ関数	
spi_isr	void	intptr_t exinf	SPI 割込みサービスルーチン	

表 4.2.3.2.1 SPI ドライバ関数

4.2.3.3 フローチャート

SPI ドライバは、必ず送受信とも DMA を使用するように設計しています。スレーブの機能はありません。SPI の初期設定は、spi_init 関数を指定して対象ポート番号と初期設定したコンフィギュレーション構造体のポインタを指定します。ペリフェラルには受信のみ、送信のみ、送受信があり、それぞれの転送関数を用意しています。実際はほとんどの場合、spi_transsrev ですべての転送を行えます。V1.1.0 では転送の終了待ち用に spi_wait 関数を呼び出し関数内で終了待ちを必要があります。V1.2.0 以降ではコンパイルスイッチ SPI_WAIT_TIME に 0 を超える数値を設定すれば、spi_transrev 内で転送 length × SPI_WAIT_TIME(ms) の待ちを実行し、spi_wait 関数を使用する必要はありません。SPI_WAIT_TIME を定義しない場合、または 0 を設定した場合は V1.1.0 と同等の仕様となります。

シリアル通信を行う場合、対象のペリフェラルの SS 信号を LOW に制御する必要があります。SS 信号の操作は GPIO を直接制御します。

図 4.2.3.3.1 に SPI の初期化フローチャートを記載します。SPI 割込み、DMA 割込み、通信と排他制御用セマフォの静的 API を用いて登録します。SS ポートの管理は SPI ドライバでは行わないため、複数のスレーブがある場合、また、スレーブ自体で SS 信号を要求している場合、SS ポートの GPIO 初期化処理を行わなければならない。

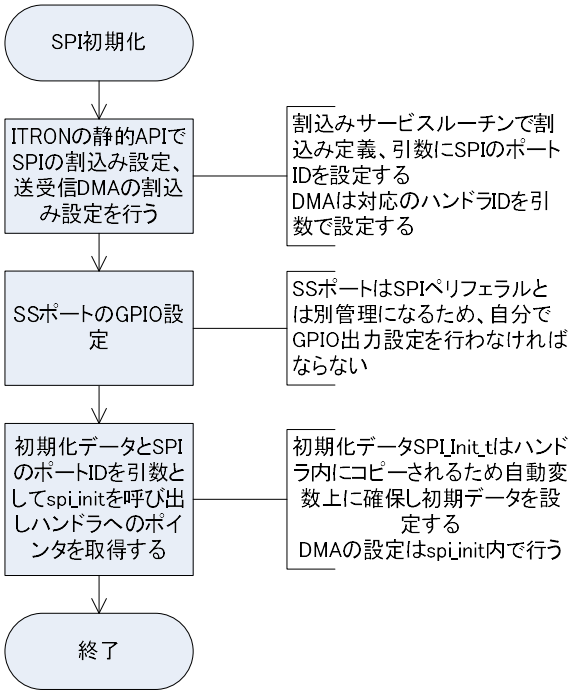


図 4.2.3.3.1 SPI 初期化フローチャート

図 4.2.3.3.2 に SPI の送受信のフローチャートを記載します。複数のスレーブの対応を行う場合は、SS の設定を行わなければならない。送受信の処理は、送信のみのスレーブや受信のみのスレーブであっても、spi_tranrecv 関数で処理を代用できる。送信のみのスレーブでこの関数を使用した場合、受信データとして 0xFF が受信され、受信のみのスレーブで、この関数を代用した場合、設定値が送信されるが、スレーブ側では受信しない。

送受信の場合、送信と同期して受信データが受信領域にセットされる。転送待ちは spi_wait 関数にて行う。戻り値が E_OK 以外はエラーが発生している。エラーの詳細は SPI ハンドラの ErrorCode にセットされる。

定義	値	内容
SPI_ERROR_NONE	0x00000000	エラーなし
SPI_ERROR_MODF	0x00000001	モードフォルト判定
SPI_ERROR_CRC	0x00000002	CRC エラー
SPI_ERROR_OVR	0x00000004	オーバーラン・エラー
SPI_ERROR_FRE	0x00000008	フレーム・エラー
SPI_ERROR_DMA	0x00000010	DMA 転送エラー
SPI_ERROR_TIMEOUT	0x00000020	ステート変更タイムアウト

表 4.2.3.3.1 ErrorCode の内容

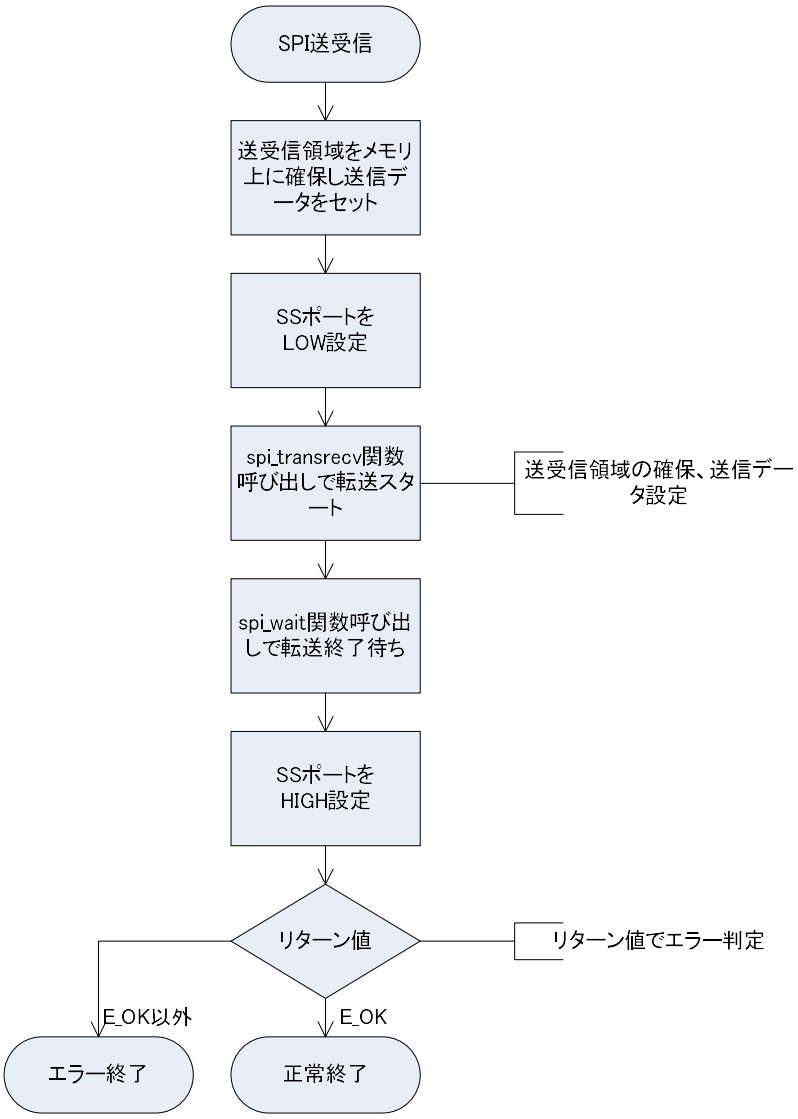


図 4.2.3.3.2 SPI 送受信フローチャート

4.2.4 ADC

ADC(アナログ・データ・コントローラ)は、アナログ入力データをデジタル・データに変換するドライバである。

4.2.4.1 データ仕様

ADC ドライバは初期化用の型として、表 4.2.4.1.1 の ADC コンフィギュレーション型と、ハンドラとして表 4.2.4.1.4 の ADC ハンドラ型を持つ。ADC の設定後、チャネルの設定用に表 4.2.4.1.2 の ADC チャネル設定型と、必ずしも設定の必要のない ADC ウォッチドック設定のための表 4.2.4.1.3 の ADC ウォッチドック型がある。

番号	項目	型	機能
1	ClockPrescaler	uint32_t	ADC クロックプリスケーラ値(各 ADC にて共通値)
2	Resolution	uint32_t	ADC のリゾリューション
3	DataAlign	uint32_t	ADC 結果データアライン設定
4	ScanConvMode	uint32_t	ADC スキャンコンバージョンモード
5	EOCSelection	uint32_t	ADC EOC 設定
6	ContinuousConvMode	uint32_t	ADC 継続モード

7	DMAContinuousRequests	uint32_t	ADC-DMA モード
8	NumConversion	uint32_t	ADC コンバージョン数
9	DiscontinuousConvMode	uint32_t	ADC 非継続モード
10	NumDiscConversion	uint32_t	ADC 非継続変換数
11	ExternalTrigConv	uint32_t	ADC 外部トリガ設定
12	ExternalTrigConvEdge	uint32_t	ADC 外部トリガエッジ設定
13	DMAChannel(Stream)	int32_t	STM32H7xx/STMG4xx に対して DMA を指定するアイテムを追加する

表 4.2.4.1.1 ADC コンフィギュレーション型

番号	項目	型	機能
1	Channel	uint32_t	ADC チャンネル番号
2	Rank	uint32_t	ADC ランク番号(1-16)
3	SamplingTime	uint32_t	ADC サンプリング時間
4	GpioBase	uint32_t	ADC-GPIO ベースアドレス
5	GpioPin	uint32_t	ADC-GPIO ピン番号

表 4.2.4.1.2 ADC チャネル設定型

番号	項目	型	機能
1	WatchdogMode	uint32_t	ADC アナログウォッチドッグモード
2	HighThreshold	uint32_t	ADC スレッシュホールド上限値
3	LowThreshold	uint32_t	ADC スレッシュホールド下限値
4	Channel	uint32_t	ADC 対応チャネル番号
5	ITMode	uint32_t	ADC アナログウォッチドック割込み設定

表 4.2.4.1.2 ADC ウォッチドック設定型

番号	項目	型	機能
1	base	uint32_t	ADC ペリフェラルのベースアドレス
2	Init	ADC_Init_t	ADC 初期化設定パラメータ
3	NumCurrentConversionRank	volatile uint32_t	コンバージョンカウンタ
4	hdmarx	DMA_Handle_t*	SPI 受信 DMA のハンドラ
5	xfercallback	void (*)()	転送完了時コールバック関数
6	xferhalfcallback	void (*)()	ハーフ転送発生時のコールバック関数
7	outofwincallback	void (*)()	ADC ウォッチドックのコールバック関数
8	errorcallback	void (*)()	ADC エラー発生時のコールバック関数
9	status	volatile uint32_t	ADC の状態
10	ErrorCode	volatile uint32_t	ADC エラーコード

表 4.2.4.1.4 ADC ハンドラ型

① ClockPrescaler
ADC クロック分周設定。

定義	値	内容
ADC_CLOCK_SYNC_PCLK_DIV2	0x00000000	2 分周
ADC_CLOCK_SYNC_PCLK_DIV4	ADC_CCR_ADCPRE_0	4 分周
ADC_CLOCK_SYNC_PCLK_DIV6	ADC_CCR_ADCPRE_1	6 分周
ADC_CLOCK_SYNC_PCLK_DIV8	ADC_CCR_ADCPRE	8 分周

表 4.2.4.1.5 ClockPrescaler 設定値

② Resolution
ADC リゾリューション（変換データのビット数）設定。

定義	値	内容
ADC_RESOLUTION_12B	0x00000000	12bit

ADC_RESOLUTION_10B	ADC_CR1_RES_0	10bit
ADC_RESOLUTION_8B	ADC_CR1_RES_1	8bit
ADC_RESOLUTION_6B	ADC_CR1_RES	6bit

表 4.2.4.1.6 Resolution 設定値

③ DataAlign
ADC 変換データの右詰、左詰めを設定。

定義	値	内容
ADC_DATAALIGN_RIGHT	0x00000000	
ADC_DATAALIGN_LEFT	ADC_CR2_ALIGN	

表 4.2.4.1.7 DataAlign 設定値

④ ScanConvMode
ADC スキャンモード設定。

定義	値	内容
ADC_SCANMODE_DISABLE	0x00000000	無効
ADC_SCANMODE_ENABLE	ADC_CR1_SCAN	有効

表 4.2.4.1.8 ScanConvMode 設定値

⑤ EOCSelection
ADC EOC シーケンスモード設定。

定義	値	内容
ADC_EOC_SEQ_DISABLE	0x00000000	無効
ADC_EOC_SEQ_ENABLE	ADC_CR2_EOCS	有効

表 4.2.4.1.9 EOCSelection 設定値

⑥ ContinuousConvMode
ADC 継続モード設定。

定義	値	内容
ADC_CONTINUOUS_DISABLE	0x00000000	無効
ADC_CONTINUOUS_ENABLE	ADC_CR2_CONT	有効

表 4.2.4.1.10 ContinuousConvMode 設定値

⑦ DMAContinuousRequests
ADC DMA 継続モード設定。

定義	値	内容
ADC_DMACONTINUOUS_DISABLE	0x00000000	無効
ADC_DMACONTINUOUS_ENABLE	ADC_CR2_DDS	有効(DMA CIRCULAR)
ADC_DMACONTINUOUS_ENABLE2	ADC_CR2_DDS+1<<31	有効(DMA NORMAL)

表 4.2.4.1.11 DMAContinuousRequests 設定値

⑧ DiscontinuousConvMode
ADC 非継続モード設定。

定義	値	内容
ADC_DISCONTINUOUS_DISABLE	0x00000000	無効
ADC_DISCONTINUOUS_ENABLE	ADC_CR1_DISCEN	有効

表 4.2.4.1.12 DiscontinuousConvMode 設定値

⑨ ExternalTrigConv
ADC 外部トリガソース設定。

定義	値	内容
ADC_EXTERNALTRIGCONV_T1_CC1	0x00000000	T1 CC1
ADC_EXTERNALTRIGCONV_T1_CC2	ADC_CR2_EXTSEL_0	T1 CC2

ADC_EXTERNALTRIGCONV_T1_CC3	ADC_CR2_EXTSEL_1	T1 CC3
ADC_EXTERNALTRIGCONV_T2_CC2	ADC_CR2_EXTSEL_1 | ADC_CR2_EXTSEL_0	T2 CC2
ADC_EXTERNALTRIGCONV_T2_CC3	ADC_CR2_EXTSEL_2	T2 CC3
ADC_EXTERNALTRIGCONV_T2_CC4	ADC_CR2_EXTSEL_2 | ADC_CR2_EXTSEL_0	T2 CC4
ADC_EXTERNALTRIGCONV_T2_TRGO	ADC_CR2_EXTSEL_2 | ADC_CR2_EXTSEL_1	T2 TRGO
ADC_EXTERNALTRIGCONV_T3_CC1	ADC_CR2_EXTSEL_2 | ADC_CR2_EXTSEL_1 | ADC_CR2_EXTSEL_0	T3 CC1
ADC_EXTERNALTRIGCONV_T3_TRGO	ADC_CR2_EXTSEL_3	T3 TRGO
ADC_EXTERNALTRIGCONV_T4_CC4	ADC_CR2_EXTSEL_3 | ADC_CR2_EXTSEL_0	T4 CC4
ADC_EXTERNALTRIGCONV_T5_CC1	ADC_CR2_EXTSEL_3 | ADC_CR2_EXTSEL_1	T5 CC1
ADC_EXTERNALTRIGCONV_T5_CC2	ADC_CR2_EXTSEL_3 | ADC_CR2_EXTSEL_1 | ADC_CR2_EXTSEL_0	T5 CC2
ADC_EXTERNALTRIGCONV_T5_CC3	ADC_CR2_EXTSEL_3 | ADC_CR2_EXTSEL_2	T5 CC3
ADC_EXTERNALTRIGCONV_T8_CC1	ADC_CR2_EXTSEL_3 | ADC_CR2_EXTSEL_2 | ADC_CR2_EXTSEL_0	T8 CC1
ADC_EXTERNALTRIGCONV_T8_TRGO	ADC_CR2_EXTSEL_3 | ADC_CR2_EXTSEL_2 | ADC_CR2_EXTSEL_1	T8 TRGO
ADC_EXTERNALTRIGCONV_Ext_IT11	ADC_CR2_EXTSEL	外部 IT11
ADC_SOFTWARE_START	ADC_CR2_EXTSEL+1	ソフトトリガ

表 4.2.4.1.13 ExternalTrigConv 設定値

⑩ ExternalTrigConvEdge

ADC 外部トリガエッジ設定。

定義	値	内容
ADC_EXTERNALTRIGCONVEDGE_NONE	0x00000000	なし
ADC_EXTERNALTRIGCONVEDGE_RISING	ADC_CR2_EXTEN_0	ライジング
ADC_EXTERNALTRIGCONVEDGE_FALLING	ADC_CR2_EXTEN_1	フォーリング
ADC_EXTERNALTRIGCONVEDGE_RISINGFALLING	ADC_CR2_EXTEN	両エッジ

表 4.2.4.1.14 ExternalTrigConvEdge 設定値

⑪ Channel

ADC チャネル番号設定。

定義	値	内容
ADC_CHANNEL_0	0x00000000	チャネル 0
ADC_CHANNEL_1	ADC_CR1_AWDCH_0	チャネル 1
ADC_CHANNEL_2	ADC_CR1_AWDCH_1	チャネル 2
ADC_CHANNEL_3	ADC_CR1_AWDCH_1 | ADC_CR1_AWDCH_0	チャネル 3
ADC_CHANNEL_4	ADC_CR1_AWDCH_2	チャネル 4
ADC_CHANNEL_5	ADC_CR1_AWDCH_2 | ADC_CR1_AWDCH_0	チャネル 5
ADC_CHANNEL_6	ADC_CR1_AWDCH_2 | ADC_CR1_AWDCH_1	チャネル 6
ADC_CHANNEL_7	ADC_CR1_AWDCH_2 | ADC_CR1_AWDCH_1 | ADC_CR1_AWDCH_0	チャネル 7
ADC_CHANNEL_8	ADC_CR1_AWDCH_3	チャネル 8
ADC_CHANNEL_9	ADC_CR1_AWDCH_3 | ADC_CR1_AWDCH_0	チャネル 9

ADC_CHANNEL_10	ADC_CR1_AWDCH_3 | ADC_CR1_AWDCH_1	チャネル 10
ADC_CHANNEL_11	ADC_CR1_AWDCH_3 | ADC_CR1_AWDCH_1 | ADC_CR1_AWDCH_0	チャネル 11
ADC_CHANNEL_12	ADC_CR1_AWDCH_3 | ADC_CR1_AWDCH_2	チャネル 12
ADC_CHANNEL_13	ADC_CR1_AWDCH_3 | ADC_CR1_AWDCH_2 | ADC_CR1_AWDCH_0	チャネル 13
ADC_CHANNEL_14	ADC_CR1_AWDCH_3 | ADC_CR1_AWDCH_2 | ADC_CR1_AWDCH_1	チャネル 14
ADC_CHANNEL_15	ADC_CR1_AWDCH_3 | ADC_CR1_AWDCH_2 | ADC_CR1_AWDCH_1 | ADC_CR1_AWDCH_0	チャネル 15
ADC_CHANNEL_16	ADC_CR1_AWDCH_4	チャネル 16
ADC_CHANNEL_17	ADC_CR1_AWDCH_4 | ADC_CR1_AWDCH_0	チャネル VREFINT
ADC_CHANNEL_18	ADC_CR1_AWDCH_4 | ADC_CR1_AWDCH_1	チャネル VBAT

表 4.2.4.1.15 Channel 設定値

⑫ SamplingTime

ADC チャネルサンプリングタイム設定。

定義	値	内容
ADC_SAMPLETIME_3CYCLES	0x00000000	3 サイクル
ADC_SAMPLETIME_15CYCLES	ADC_SMPR1_SMP10_0	15 サイクル
ADC_SAMPLETIME_28CYCLES	ADC_SMPR1_SMP10_1	28 サイクル
ADC_SAMPLETIME_56CYCLES	ADC_SMPR1_SMP10_0 | ADC_SMPR1_SMP10_1	56 サイクル
ADC_SAMPLETIME_84CYCLES	ADC_SMPR1_SMP10_2	84 サイクル
ADC_SAMPLETIME_112CYCLES	ADC_SMPR1_SMP10_2 | ADC_SMPR1_SMP10_0	112 サイクル
ADC_SAMPLETIME_144CYCLES	ADC_SMPR1_SMP10_2 | ADC_SMPR1_SMP10_1	144 サイクル
ADC_SAMPLETIME_480CYCLES	ADC_SMPR1_SMP10	480 サイクル

表 4.2.4.1.16 SamplingTime 設定値

⑬ WatchdogMode

ADC ウォッチドッグモード設定。

定義	値	内容
ADC_ANALOGWATCHDOG_NONE	0x00000000	
ADC_ANALOGWATCHDOG_SINGLE_REG	ADC_CR1_AWDSGL | ADC_CR1_AWDEN	
ADC_ANALOGWATCHDOG_SINGLE_INJEC	ADC_CR1_AWDSGL | ADC_CR1_JAWDEN	
ADC_ANALOGWATCHDOG_SINGLE_REGINJEC	ADC_CR1_AWDSGL | ADC_CR1_AWDEN | ADC_CR1_JAWDEN	
ADC_ANALOGWATCHDOG_ALL_REG	ADC_CR1_AWDEN	
ADC_ANALOGWATCHDOG_ALL_INJEC	ADC_CR1_JAWDEN	
ADC_ANALOGWATCHDOG_ALL_REGINJEC	ADC_CR1_AWDEN | ADC_CR1_JAWDEN	

表 4.2.4.1.17 WatchdogMode 設定値

⑭ ITMode

ADC アナログウォッチドック割込みモード設定。

定義	値	内容
ADC_ANALOGWATCHDOG_ITMODE_DISABLE	0x00000000	無効
ADC_ANALOGWATCHDOG_ITMODE_ENABLE	ADC_CR1_AWDIE	有効

表 4.2.4.1.18 ITMode 設定値

4.2.4.2 インターフェイス仕様

ADC を制御するドライバ関数は以下の通りである。入力ピンの設定は、GPIO を使って別途制御しなければならない。

関数名	型	引数	機能	備考
adc_init	ADC_Handler*	ID portid ADC_Init_t *pini	指定ポート ID の ADC ペリフェラルを初期化し、ハンドラへのポインタを返す	
adc_deinit	ER	ADC_Handler* hadc	ADC を未使用状態に戻す	
adc_start_int	ER	ADC_Handler* hadc	ADC 割込みモード開始	
adc_end_int	ER	ADC_Handler* hadc	ADC 割込みモード終了	
adc_start_dma	ER	ADC_Handler* hadc uint32_t *pdata uint32_t length	ADC DMA モード開始	
adc_end_dma	ER	ADC_Handler* hadc	ADC DMA モード終了	
adc_setupchannel	ER	ADC_Handler* hadc ADC_ChannelConf_t* sConfig	ADC チャネル設定	
adc_setupwatchdoc	ER	ADC_Handler* hadc ADC_AnalogWDGConf_t* AnalogWDGConfig	ADC ウォッチドック設定	
adc_handler	void	ADC_Handler* hadc	ADC 割込みハンドラ関数	
adc_int_handler	void	void	ADC 割込みハンドラ	

表 4.2.4.2.1 ADC ドライバ関数

4.2.4.3 フローチャート

実行手順に影響を与えるモードは、以下の４つである。

(1) ScanConvMode

複数のチャネルが設定された場合、自動的にチャネルを変えながら AD 変換を行う設定

(2) ContinuousConvMode

AD 変換回数を指定する。回数は NumConversion にセットする。このモードを無効にした場合は single mode になり、一度の AD 変換で終了。

Singe mode で NumConversion を設定した場合、割込みの停止回数設定となる。

(3) DiscontinuousConvMode

外部トリガの後、AD 変換を停止するチャネルの数を設定する。

(4) ExternalTrigConv

スキャンのタイミングを外部トリガで設定する。

ADC の複数機能を設定した場合、複雑なフローチャートのなるため、DMA を使用して１チャネルを１回変換するフローチャートを示す。

ADC の初期化(図 4.2.4.3.1)は、adc_init 関数にて ADC コントローラの初期化を行い。そのあと、ADC チャネルの設定を行う。ADC 変換の実行(図 4.2.4.3.2)は adc_start_dma で起動、ハンドラ内の status の値が ADC_STATUS_BUSY の間は実行中、終了すると ADC_STATUS_BUSY 以外の値に変わる。ADC_STATUS_READY ならば正常終了であり、それ以外の値はエラーを示す。Adc_end_dma 関数で処理の停止を指定する。

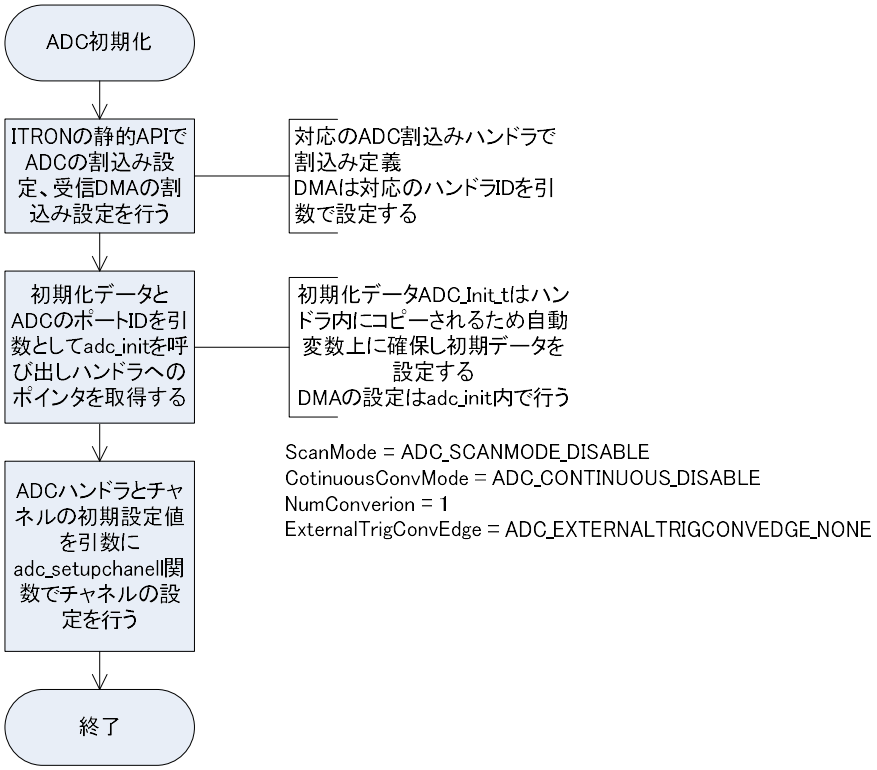


図 4.2.4.3.1 ADC の初期設定

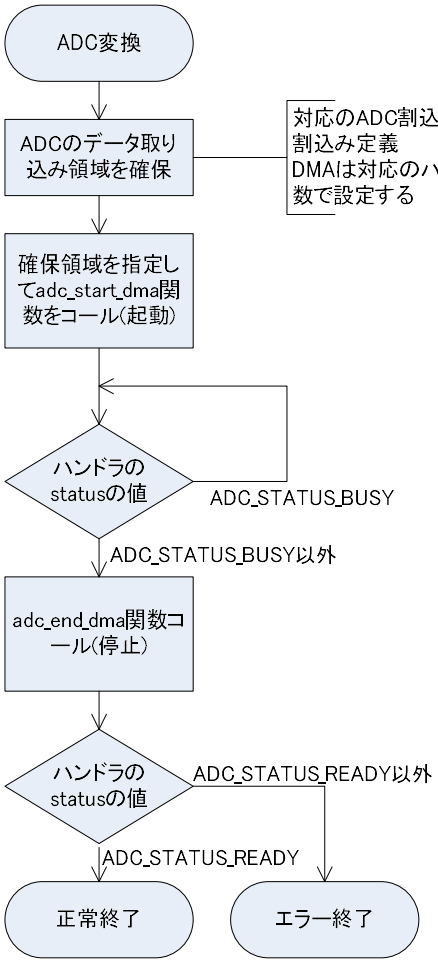


図 4.2.4.3.2 ADC 変換の実行

4.2.5 QSPI

QSPI(クアッド・エスピーアイ)は、Quad SPI フラッシュメモリ等のシリアルメモリと通信を行うためのシリアル通信ドライバである。

4.2.5.1 データ仕様

QSPI ドライバは Quad SPI フラッシュメモリの通信定義用の型として、表 4.2.5.1.1 の QSPI コンフィギュレーション型と、ハンドラとして表 4.2.4.1.4 の QSPI ハンドラ型を持つ。QSPI コンフィギュレーション型は、QUAD SPI フラッシュメモリ種別毎に設定を行う必要がある。

番号	項目	型	機能
1	manuf_id	uint32_t	Quad SPI フラッシュメモリのマニュファクチャ ID
2	type_capacity	uint32_t	Quad SPI フラッシュメモリのキャパシティタイプ
3	clockprescaler	uint32_t	AHB に対するプリスケーラ
4	fifothreshold	uint32_t	FIFO のスレッシュホールド値
5	sampleshift	uint32_t	
6	chipselecthightime	uint32_t	CHIP Select High Time
7	clockmode	uint32_t	クロックモード
8	flashid	uint32_t	FLASH ID
9	dualflash	uint32_t	DUAL FLASH モード
10	ddrmode	uint32_t	DDR モード
11	ddrholdhalfcycle	uint32_t	DDR ホールドのハーフサイクル設定
12	sioomode	uint32_t	スードモード
13	addr_size	uint32_t	アドレスサイズ設定
14	inst_type	uint32_t	通常インストラクションのタイプ
15	inst_data_xfer_type	uint32_t	通常インストラクションの転送タイプ
16	read_op_code	uint32_t	READ 処理のオペコード
17	read_addr_xfer_type	uint32_t	READ 処理の転送アドレスタイプ
18	read_data_xfer_type	uint32_t	READ 処理の転送データタイプ
19	read_dummy_cycle	uint32_t	READ ダミーサイクル値
20	write_op_code	uint32_t	WRITE 処理のオペコード
21	write_addr_xfer_type	uint32_t	WRITE 処理の転送アドレスタイプ
22	write_data_xfer_type	uint32_t	WRITE 処理の転送データタイプ
23	write_dummy_cycle	uint32_t	WRITE ダミーサイクル
24	erase_count	uint8_t	ERASE 処理の数
25	erase_size	uint32_t x 2	ERASE の領域サイズ
26	erase_cmds	uint8_t x 2	ERASE オペコード
27	erase_secor_idx	uint8_t	セクタ消去の番号
28	device_size	uint32_t	デバイスのバイトサイズ
29	init_func	ER (*)()	初期化関数
30	enable_write_func	ER (*)()	ENABLE WRITE 関数
31	wait_func	ER (*)()	ERASE/WRITE 待ち関数
32	friendly_name	const char	フラッシュメモリ名

表 4.2.5.1.1 QSPI コンフィギュレーション型

番号	項目	型	機能
1	InstructionMode	uint32_t	インストラクションモード
2	Instruction	uint32_t	インストラクション・オペコード
3	AddressMode	uint32_t	アドレスモード
4	AddressSize	uint32_t	アドレスサイズ
5	Address	uint32_t	アドレス

6	AlternateByteMode	uint32_t	アルティネートバイトモード
7	AlternateBytesSize	uint32_t	アルティネートバイトサイズ
8	AlternateBytes	uint32_t	アルティネートバイト
9	DummyCycles	uint32_t	ダミーサイクル
10	DataMode	uint32_t	データモード
11	NbData	uint32_t	データ長

表 4.2.5.1.2 QSPI コマンド生成型

番号	項目	型	機能
1	Match	uint32_t	マッチデータ
2	Mask	uint32_t	マッチデータのマスク値
3	Interval	uint32_t	READ とオートマチックポーリング間のインターバル
4	StatusBytesSize	uint32_t	ステータスのバイト長
5	MatchMode	uint32_t	マッチモード
6	AutomatchStop	uint32_t	マッチ後の自動停止設定

表 4.2.4.1.3 QSPI オートポーリング型

番号	項目	型	機能
1	base	uint32_t	QSPI ペリフェラルのベースアドレス
2	Init	QSPI_Init_t	QSPI コンフィギュレーション設定パラメータ
3	pBuffPtr	uint32_t *	データ転送領域のポインタ
4	XferSize	uint32_t	指定転送サイズ
5	XferCount	uint32_t	現在の転送データカウント
6	timeoutcallback	ER (*)()	タイムアウト時のコールバック関数
7	errorcallback	ER (*)()	エラーのコールバック関数
8	istatus	uint32_t	最後に発生した割込みステータスの値
9	timeout	uint32_t	デフォルトタイムアウト値(ms)
10	semid	int	通信セマフォ ID
11	status	uint32_t	QSPI の状態
12	errorcode	volatile uint32_t	QSPI エラーコード

表 4.2.5.1.4 QSPI ハンドラ型

① sampleshift

sample shift 設定。

定義	値	内容
QSPI_SAMPLE_SHIFTING_NONE	0x00000000	シフトなし
QSPI_SAMPLE_SHIFTING_HALFCYCLE	QUADSPI_CR_SHIFT	1/2 クロックシフト

表 4.2.5.1.5 sampleshift 設定値

② chipselecthightime

コマンド間の CS STAY HIGH 設定のクロックサイクルを設定。

定義	値	内容
QSPI_CS_HIGH_TIME_1_CYCLE	0x00000000	1cycle
QSPI_CS_HIGH_TIME_2_CYCLE	QUADSPI_DCR_CSHT_0	2cycle
QSPI_CS_HIGH_TIME_3_CYCLE	QUADSPI_DCR_CSHT_1	3cycle
QSPI_CS_HIGH_TIME_4_CYCLE	QUADSPI_DCR_CSHT_0|QUADSPI_DCR_CSHT_1	4cycle
QSPI_CS_HIGH_TIME_5_CYCLE	QUADSPI_DCR_CSHT_2	5cycle
QSPI_CS_HIGH_TIME_6_CYCLE	QUADSPI_DCR_CSHT_2|QUADSPI_DCR_CSHT_0	6cycle
QSPI_CS_HIGH_TIME_7_CYCLE	QUADSPI_DCR_CSHT_2|QUADSPI_DCR_CSHT_1	7cycle
QSPI_CS_HIGH_TIME_8_CYCLE	QUADSPI_DCR_CSHT	8cycle

表 4.2.5.1.6 chipselecthightime 設定値

③ clockmode
クロックモードを設定。

定義	値	内容
QSPI_CLOCK_MODE_0	0x00000000	CS 設定時 LOW
QSPI_CLOCK_MODE_3	QUADSPI_DCR_CKMODE	CS 設定時 HIGH に遷移

表 4.2.5.1.7 clockmode 設定値

④ flashselect
FLASH 選択設定。

定義	値	内容
QSPI_FLASH_ID_1	0x00000000	FLASH 1
QSPI_FLASH_ID_2	QUADSPI_CR_FSEL	FLASH 2

表 4.2.5.1.8 flahselect 設定値

⑤ dualflashmode
Dual Flash モード設定。

定義	値	内容
QSPI_DUALFLASH_DISABLE	0x00000000	無効
QSPI_DUALFLASH_ENABLE	QUADSPI_CR_DFM	有効

表 4.2.5.1.9 dualflashmode 設定値

⑥ ddrmode
Double data rate mode 設定。

定義	値	内容
QSPI_DDR_MODE_DISABLE	0x00000000	無効
QSPI_DDR_MODE_ENABLE	QUADSPI_CCR_DDRM	有効

表 4.2.5.1.10 ddrmode 設定値

⑦ ddrholdhalfcycle
DDR モード時、データ出力デレィ設定。

定義	値	内容
QSPI_DDR_HHC_ANALOG_DELAY	0x00000000	アナログデレィ
QSPI_DDR_HHC_HALF_CLK_DELAY	QUADSPI_CCR_DHHC	1/2 clock cycle

表 4.2.5.1.11 ddrholdhalfcycle 設定値

⑧ sioomode
送信インストラクションモード設定。

定義	値	内容
QSPI_SIOO_INST_EVERY_CMD	0x00000000	すべての転送にインストラクション送信
QSPI_SIOO_INST_ONLY_FIRST_CMD	QUADSPI_CCR_SIOO	転送の最初のみ

表 4.2.5.1.12 sioomode 設定値

⑨ AddressMode
コマンド生成、アドレスモード設定。

定義	値	内容
QSPI_ADDRESS_NONE	0x00000000	なし
QSPI_ADDRESS_1_LINE	QUADSPI_CCR_IMODE_0	1 lne
QSPI_ADDRESS_2_LINES	QUADSPI_CCR_IMODE_1	2 lines
QSPI_ADDRESS_4_LINES	QUADSPI_CCR_IMODE	4 lines

表 4.2.5.1.13 AddressMode 設定値

⑩ AddressSize

コマンド生成、アドレスサイズ設定。

定義	値	内容
QSPI_ADDRESS_8_BITS	0x00000000	8 ビットアドレス
QSPI_ADDRESS_16_BITS	QUADSPI_CCR_ADSIZE_0	16 ビットアドレス
QSPI_ADDRESS_24_BITS	QUADSPI_CCR_ADSIZE_1	24 ビットアドレス
QSPI_ADDRESS_32_BITS	QUADSPI_CCR_ADSIZE	32 ビットアドレス

表 4.2.5.1.14 AddressSize 設定値

⑪ InstructionMode

コマンド生成、インストラクションモード設定。

定義	値	内容
QSPI_INSTRUCTION_NONE	0x00000000	なし
QSPI_INSTRUCTION_1_LINE	QUADSPI_CCR_IMODE_0	1line
QSPI_INSTRUCTION_2_LINES	QUADSPI_CCR_IMODE_1	2lines
QSPI_INSTRUCTION_4_LINES	QUADSPI_CCR_IMODE	4lines

表 4.2.5.1.15 InstructionMode 設定値

⑫ AlternateBytesMode

コマンド生成、アルタネートバイトモード設定。

定義	値	内容
QSPI_ALTERNATE_BYTES_NONE	0x00000000	なし
QSPI_ALTERNATE_BYTES_1_LINE	QUADSPI_CCR_ABMODE_0	1line
QSPI_ALTERNATE_BYTES_2_LINES	QUADSPI_CCR_ABMODE_1	2lines
QSPI_ALTERNATE_BYTES_4_LINES	QUADSPI_CCR_ABMODE	4lines

表 4.2.5.1.16 AlternaleBytesMode 設定値

⑬ DataMode

コマンド生成、データモード設定。

定義	値	内容
QSPI_DATA_NONE	0x00000000	なし
QSPI_DATA_1_LINE	QUADSPI_CCR_DMODE_0	1line
QSPI_DATA_2_LINES	QUADSPI_CCR_DMODE_1	2lines
QSPI_DATA_4_LINES	QUADSPI_CCR_DMODE	4lines

表 4.2.5.1.17 DataMode 設定値

4.2.5.2 インターフェイス仕様

QSPI を制御するドライバ関数は以下の通りである。データの読み出しは関数を使って読み出す INDIRECT READ とメモリ参照で読み出す DIRECT READ がある。

関数名	型	引数	機能	備考
qspi_init	QSPI_Handler*	ID portid QSPI_Init_t *pini	指定ポート ID の QSPI ペリフェラルを初期化し、ハンドラへのポインタを返す	
qspi_deinit	ER	ADC_Handler* hqspi	QSPI ドライバを未使用状態に戻す	
qspi_read	ER	QSPI_Handler* hqspi void *dest uint32_t src uint32_t size	QSPI INDIRECT READ	
qspi_write	ER	QSPI_Handler* hspi uint32_t dest void *src	QSPI INDIRECT WRITE	

		uint32_t size		
qspi_erase	ER	QSPI_Handler* hqspi uint32_t address uint32_t size	アドレス指定消去	
qspi_erase_sector	ER	QSPI_Handler* hqspi uint32_t address	セクタ単位消去	
qspi_direct_disable	ER	QSPI_Handler* hqspi	DIRECT READ 無効	
qspi_direct_enable	ER	QSPI_Handler* hspi	DIRECT READ 有効	
qspi_handler	void	QSPI_Handler* hspi	QSPI 割込みハンドラ関数	
qspi_isr	void	intptr_t exinf	QSPI 割込みサービスルーチン	

表 4.2.5.2.1 QSPI ドライバ関数

4.2.5.3 フローチャート

QSPI 制御は初期化を行い。ERASE/WRITE/READ の関数を使ってデータの消去、書き込み、読み出しを行えばよい。注意点はデータ書き込む領域には、ERASE 関数で領域を消去する必要がある。

QSPI コンフィギュレーションデータは、以下の対応となるこれ以外の QSPI フラッシュメモリを使用する場合は、コンフィギュレーションデータを自作する必要がある。

① STM32F769 Discovery：MICRON N25Q512A13GSF40E
② STM32L476 Discovery：MICRON N25Q128A13EF840E
③ STM32F723 Discovery：MACRONIX MX25L51245G

4.2.6 RTC

RTC(Real-time clock)は時刻を管理するハードウェアである。ST マイクロエレクトロニクス社のボードでは RTC 用のクロックは実装されているが、バックアップ用バッテリは実装されていないため電源を落とすと時刻はリセットされる。RTC は時刻管理以外にアラート機能があり、アラート時刻を設定すると割込みにより、アラート通知を受け取ることができる。

RTC は（書き込みを行う）ファイル操作には必須の機能であるためサポートを行う。また、他のドライバのように複数のロジックを持つことはないためポート管理は行わない。

4.2.6.1 データ仕様

RTC ではUNIX で使用されている時刻管理構造体 tm と共有して時刻データのやり取りを行えるように表 4.2.6.1.1 として構造体名を変えた tm2 構造体を定義する。この構造体は tm 構造体と混在して使用されても定しくデータの受け渡しを行える。

RTC のアラーム動作設定用に表 4.2.6.1.2 として RTC アラーム型を用意する。割込みとして割込み番号(16+41)の IRQ_VECTOR_RTC_ALARM を使用する。

番号	項目	型	機能
1	tm_sec	int	秒(0～59)
2	tm_min	int	分(1～60)
3	tm_hour	int	時(0～23)
4	tm_mday	int	月の中の日
5	tm_mon	int	月
6	tm_year	int	年(1970 を 0 とした年数)
7	tm_wday	int	曜日
8	tm_yday	int	年の中の日
9	tm_isdst	int	夏時間： 0 以外の値

表 4.2.6.1.1 tm2 構造体

番号	項目	型	機能
1	alarm	uint32_t	アラーム選択

2	alarmmask	uint32_t	アラームマスク設定
3	subsecondmask	uint32_t	アラームサブセコンドマスク
4	dayselect	uint32_t	アラーム日付設定
5	subsecnd	uint32_t	アラームサブセコンド
6	callback	(*)(void)	アラームコールバック

表 4.2.6.1.2 RTC_Alarm 型

① alarm

アラームレジスタを指定する

定義	値	内容
RTC_ALARM_A	RTC_CR_ALRAE	アラームＡレジスタを指定
RTC_ALARM_B	RTC_CR_ALRBE	アラームＢレジスタを指定

表 4.2.6.1.3 alarm 設定値

② dayselect

垂直同期極性設定をする。

定義	値	内容
ALARMDAYSEL_DATE	0x00000000	月中の日を指定
ALARMDAYSEL_WEEKDAY	0x40000000	ウィークディを指定

表 4.2.6.1.4 dayselect 設定値

③ alarmmask

アラームのマスク設定を行う。

定義	値	内容
ALARMMASK_NONE	0x00000000	マスクなし
ALARMMASK_DATESEL	RTC_ALRMAR_MSK4	日のアラームをマスク
ALARMMASK_HOURS	RTC_ALRMAR_MSK3	時のアラームをマスク
ALARMMASK_MINUTES	RTC_ALRMAR_MSK2	分のアラームをマスク
ALARMMASK_SECONDS	RTC_ALRMAR_MSK1	秒のアラームをマスク

表 4.2.6.1.5 alarmmask 設定値

④ subsecondmask

サブセコンドマスク設定をする。

定義	値	内容 s
ALARMSSMASK_ALL	0x00000000	サブセコンドとのマッチをしない
ALARMSSMASK_SS14_1	0x01000000	SS[0]がマッチでアラーム
ALARMSSMASK_SS14_2	0x02000000	SS[1:0]がマッチでアラーム
ALARMSSMASK_SS14_3	0x03000000	SS[2-0]がマッチでアラーム
ALARMSSMASK_SS14_4	0x04000000	SS[3-0]がマッチでアラーム
ALARMSSMASK_SS14_5	0x05000000	SS[4-0]がマッチでアラーム
ALARMSSMASK_SS14_6	0x06000000	SS[5-0]がマッチでアラーム
ALARMSSMASK_SS14_7	0x07000000	SS[6-0]がマッチでアラーム
ALARMSSMASK_SS14_8	0x08000000	SS[7-0]がマッチでアラーム
ALARMSSMASK_SS14_9	0x09000000	SS[8-0]がマッチでアラーム
ALARMSSMASK_SS14_10	0x0A000000	SS[9-0]がマッチでアラーム
ALARMSSMASK_SS14_11	0x0B000000	SS[10-0]がマッチでアラーム
ALARMSSMASK_SS14_12	0x0C000000	SS[11-0]がマッチでアラーム
ALARMSSMASK_SS14_13	0x0D000000	SS[12-0]がマッチでアラーム
ALARMSSMASK_SS14	0x0E000000	SS[13-0]がマッチでアラーム
ALARMSSMASK_NONE	0x0F000000	SS[14-0]がマッチでアラーム

表 4.2.6.1.6 subsecondmask 設定値

4.2.6.2 インターフェイス仕様

RTC を設定するドライバ関数を以下に示す。

関数名	型	引数	機能	備考
rtc_init	void	intptr_exinf	RTC の初期化を行う。引数に意味なし	
rtc_set_time	ER	struct tm2 *pt	時刻をセットする	
rtc_get_time	ER	struct tm2 *pt	時刻を取り出す	
rtc_setalarm	ER	RTC_alarm_t *parm struct tm2 *ptm	アラートをセットする	
rtc_stopalarm	ER	uint32_t Alarm	アラートを停止する	
rtc_getalarm	ER	RTC_alarm_t *parm struct tm2 *ptm uint32_t Alarm	アラート情報を取り出す	
rtc_handler	void	void	割込みハンドラ	

表 4.2.6.2.1 RTC 設定関数

4.2.6.3 設定手順

初期化は rtc_init 関数を用いて行う。ATT_INI を使用して設定が行えるように引数を用意したが、この引数に意味はない。使用は以下の手順に従う。

① rtc_set_time
時刻の設定を行う。tm2 構造体中に設定に使用するのは以下の６つの項目で他の項目は意味を持たない。
(1) tm_year
(2) tm_mon
(3) tm_mday
(4) tm_hour
(5) tm_min
(6) tm_sec

② rtc_get_time
時刻を取り出す。tm2 構造体中に実際に設定される項目は以下の７つの項目である。
他の設定も設定したい場合は mktime 関数を用いて設定を行う必要がある。
(1) tm_year
(2) tm_mon
(3) tm_mday
(4) tm_wday
(5) tm_hour
(6) tm_min
(7) tm_sec

③ rtc_setalarm
アラームの設定を行う。アラームレジスタはＡとＢの二つがあり別個に設定ができる。
アラームは時刻との比較と、サブセコンドとの比較の二種類がある。
時刻との比較の場合、tm2 構造体に比較の時刻設定を行い。マスク設定でマスクのない項目との比較で一致した場合割込みが発生する。コールバック関数をセットすれば割込み時コールバック関数が呼び出される。
サブセコンドとの比較の場合、一致のセコンド値と比較しないマスク設定を行い、一致すれば割込みが発生する。

④ rtc_stopalarm
アラームを停止する。停止するアラームレジスタを引数として渡す。

⑤ rtc_getalarm
引数の Alarm にアラームレジスタを設定する。現在のレジスタ内容から RTC_Alarm 型を生成する。

4.2.7 USB OTG

USB OTG は、ホスト、デバイスの USB 管理を行うドライバである。USB ホストとして動作させるためには USB ホストクラスドライバ、USB デバイスとして動作させるためには USB デバイスライブラリを上位のモジュールとして用意する必要がある。

4.2.7.1 データ仕様

USB OTG ドライバは初期化用の型として、表 4.2.7.1.1 の USB OTG 初期設定定義型と、ハンドラとして表 4.2.7.1.4 の USB OTG ハンドラ型を持つ。内部の型としてエンドポイントを管理するための表 4.2.7.1.2 のエンドポイント定義型と、ホストチャネルを管理するための表 4.2.7.1.3 のホストクラス定義型をもつ。エンドポイントとホストクラス定義は、USB の上位層のモジュールで設定を行う必要がある。この実装では、Stm32Cube 中の USB ミドルウェアが設定を行っている。

初期設定定義型の有効無効設定は、1 が有効、0 が無効の設定となる。

番号	項目	型	機能
1	usb_otg_mode	uint32_t	USB OTG の実行モード指定
2	dev_endpoints	uint32_t	デバイス用のエンドポイントの数(1-15)
3	host_channels	uint32_t	ホストチャネルの数(1-15)
4	speed	uint32_t	USB コアスピード
5	dma_enable	uint32_t	USB DMA 機能の有効無効設定
6	phy_itface	uint32_t	PHY の指定
7	sof_enable	uint32_t	デバイスモードでの SOF 割込みの有効無効設定
8	low_power_enable	uint32_t	LOW POWER モードの有効無効設定
9	lpm_enable	uint32_t	LPM 機能の有効無効設定
10	vbus_sessing_enable	uint32_t	VBUS のセンシング機能の有効無効設定
11	use_dedicated_ep1	uint32_t	エンドポイント 1 専用割込みの有効無効設定
12	use_external_vbus	uint32_t	外部 VBUS の有効無効設定

表 4.2.7.1.1 USB OTG 初期設定定義型

番号	項目	型	機能
1	num	uint8_t	エンドポイント番号(0-15)
2	is_in	uint8_t：1	エンドポイントのディレクション(1:IN 0:OUT)
3	is_stall	uint8_t：1	スティール状態(1:スティール状態)
4	type	uint8_t：2	通信タイプ
5	data_pid_start	uint8_t：1	スタートの PID(0 または 1)
6	even_off_frame	uint8_t：1	フレームのパリティ(0:EVEN 1:ODD)
7	tx_fifo_num	uint16_t	送信 FIFO 番号
8	maxpacket	uint16_t	最大パケットサイズ(64KB 以内)
9	xfer_buff	uint8_t *	送受信バッファへのポインタ
10	dma_addr	uint32_t	DMA アドレス(4 バイトのアライン値)
11	xfer_len	uint32_t	現在転送サイズ
12	xfer_count	uint32_t	指定の転送サイズ

表 4.2.7.1.2 エンドポイント定義型

番号	項目	型	機能
1	dev_addr	uint8_t	デバイスアドレス(1-255)
2	ch_num	uint8_t	ホストチャネル番号(1-15)
3	ep_num	uint8_t	エンドポイント番号(1-15)
4	ep_is_in	uint8_t	エンドポイントのディレクション(1:IN 0:OUT)
5	speed	uint8_t	USB ホストのスピード
6	do_ping	uint8_t	HS モード時の PING プロトコルの有効無効設定
7	process_ping	uint8_t	PING プロトコル実行状態
8	ep_type	uint8_t	エンドポイントの型
9	max_paket	uint16_t	最大パケットサイズ(64KB 以内)
10	data_pid	uint8_t	初期設定データ PID
11	xfer_buff	uint8_t *	通信バッファ領域へのポインタ
12	xfer_len	uint32_t	現在の通信サイズ
13	xfer_count	uint32_t	指定通信サイズ
14	toggle_in	uint8_t	IN 通信のトグルフラグ
15	toggle_out	uint8_t	OUT 通信のトグルフラグ
16	dma_addr	uint32_t	DMA モードでのデータアドレス(4 バイトのアライン)
17	err_count	uint32_t	エラー発生数
18	urb_state	uint8_t	URB ステータス
19	state	uint8_t	ホストステータス

表 4.2.7.1.2 ホストクラス定義型

番号	項目	型	機能
1	base	uint32_t	USB-OTG ペリフェラルのベースアドレス
2	Init	USB_OTG_Init_t	USB-OTG 初期化設定パラメータ
3	hc[15]	USB_OTG_HCTypedef	ホストチャネル領域
4	IN_ep[15]	USB_OTG_EPTypedef	IN エンドポイント領域
5	OUT_ep[15]	USB_OTG_EPTypedef	OUT エンドポイント領域
6	Setup[12]	uint32_t	セットアップパケット保存領域
7	BESL	uint32_t	BESL データの保存領域
8	lpm_state	uint8_t	LPM の状態
9	lpm_active	uint8_t	LPM 機能有効無効設定
10	hostsofcallback	void (*)()	ホスト用 SOF 割込みコールバック関数
11	hostconnectcallback	void (*)()	ホストコネクト時コールバック関数
12	hostdisconnectcallback	void (*)()	ホストディスコネクト時コールバック関数
13	hostchangeurbcallback	void (*)()	ホスト URB 変更時コールバック関数
14	devsetupstagecallback	void (*)()	デバイスセットアップステージコールバック関数
15	devdataoutstagecallback	void (*)()	デバイスデータアウトステージコールバック関数
16	devdatainstagecallback	void (*)()	デバイスデータインステージコールバック関数
17	devsofcallback	void (*)()	デバイス SOF 割込みコールバック関数
18	devresetcallback	void (*)()	デバイスリセットコールバック関数
19	devsuspendcallback	void (*)()	デバイスサスペンドコールバック関数
20	devresumecallback	void (*)()	デバイスレジュームコールバック関数
21	devisooutcallback	void (*)()	デバイス ISOOUT コールバック関数
22	devisoincallback	void (*)()	デバイス ISOIN コールバック関数
23	devconnectcallback	void (*)()	デバイスコネクトコールバック関数
24	devdisconnectcallback	void (*)()	デバイスディスコネクトコールバック関数
25	devlpmcallback	void (*)()	デバイス LPM コールバック関数

26	pHost	void *	上位ホストハンドラ格納領域
27	pDev	void *	上位デバイスハンドラ格納領域

表 4.2.5.1.4 USB OTG ハンドラ型

② usb_otg_mode

USB OTG の設定モードを指定する。

定義	値	内容
USB_OTG_MODE_DEVICE	0	USB DEVICE 固定
USB_OTG_MODE_HOST	1	USB HOST 固定
USB_OTG_MODE_DRD	2	OTG モード

表 4.2.7.1.5 usb_otg_mode 設定値

③ speed

USB の転送スピードを設定する。

定義	値	内容
USB_SPEED_HIGH	0	HIGH スピード指定
USB_SPEED_FULL	3	FULL スピード指定

表 4.2.7.1.6 speed 設定値

④ phy_inface

PHY 種別設定を行う。

定義	値	内容
USB_PHY_ULPI	1	ULPI-PHY
USB_PHY_EMBEDDED	2	EMBEDDED-PHY

表 4.2.7.1.7 phy_inface 設定値

4.2.7.2 インターフェイス仕様

USB OTG を制御するドライバ関数は以下の通りである。

関数名	型	引数	機能	備考
usbo_init	USB_OTG_Handler*	ID portid USB_OTG_Init_t *pini	指定ポート ID の USB-OTG ペリフェラルを初期化	
usbo_deinit	ER	USB_OTG_Handler* husb	USB モジュールの無効化	
usbo_getcurrentframe	uint32_t	USB_OTG_Handler* husb	現在実行中のフレーム番号を取りだず	
usbo_setcurrentmode	ER	USB_OTG_Handler* husb	USB-OTG のモード設定を行う	
usbo_coreinit	ER	USB_OTG_Handler* husb	USB-OTG コアの初期化	
usbo_enableglobalint	ER	USB_OTG_Handler* husb	USB-OTG グローバル割込みを有効に設定	
usbo_disableglobalint	ER	USB_OTG_Handler* husb	USB-OTG グローバル割込みを無効に設定	
usbo_flushTxFifo	ER	USB_OTG_Handler* husb uint32_t num	送信 FIFO をフラッシュする	
usbo_flushRxFifo	ER	USB_OTG_Handler* husb	受信 FIFO をフラッシュする	
usbo_iniFifo	ER	USB_OTG_Handler* husb	送信 FIFO 初期化設定	
usbo_hc_init	ER	USB_OTG_Handler* husb uint8_t ch_num uint8_t epnum	ホストチャネルを初期化する	

usbo_hc_startxfer	ER	USB_OTG_Handler* husb uint8_t ch_num	ホストチャネル送信要求	
usbo_hc_halt	ER	USB_OTG_Handler* husb uint8_t ch_num	ホストチャネルを HALT 状態にする	
usbo_resetport	ER	USB_OTG_Handler* husb	ホストポートをリセットする	
usbo_drivevbus	ER	USB_OTG_Handler* husb uint8_t state	ホスト用 VBUS のオンオフ設定を行う	
usbo_hostinit	ER	USB_OTG_Handler* husb	ホストの初期化を行う	
usbo_starthost	ER	USB_OTG_Handler* husb	ホストを開始する	
usbo_stophost	ER	USB_OTG_Handler* husb	ホストを停止する	
usbo_gethostspeed	uint32_t	USB_OTG_Handler* husb	ホストのコアスピードを取り出す	
usbo_hcd_irqhandler	void	USB_OTG_Handler* husb	USB-OTG ホスト割込み処理	
usbo_devconnect	ER	USB_OTG_Handler* husb	USB デバイス接続要求	
usbo_devdisconnect	ER	USB_OTG_Handler* husb	USB デバイス切断要求	
usbo_activateEndpoint	ER	USB_OTG_Handler* husb USB_OTG_EPTypedef *ep	エンドポイントアクティベイト要求	
usbo_disactivateEndpoint	ER	USB_OTG_Handler* husb USB_OTG_EPTypedef *ep	エンドポイントディスアクティベイト要求	
usbo_epsetStall	ER	USB_OTG_Handler* husb USB_OTG_EPTypedef *ep	エンドポイントをスティール状態に設定	
usbo_epclearStall	ER	USB_OTG_Handler* husb USB_OTG_EPTypedef *ep	エンドポイントのスティール状態解除	
usbo_setDevAddress	ER	USB_OTG_Handler* husb uint8_t address	デバイスアドレスを設定する	
usbo_getDevSpeed	uint8_t	USB_OTG_Handler* husb	USB デバイススピードを取り出す	
usbo_ep0_outstart	ER	USB_OTG_Handler* husb uint8_t *psetup	エンドポイント 0-OUT 開始要求	
usbo_ep0startxfer	ER	USB_OTG_Handler* husb USB_OTG_EPTypedef* ep	エンドポイント0の転送開始要求	
usbo_epstartxfer	ER	USB_OTG_Handler* husb USB_OTG_EPTypedef* ep	エンドポイントの転送開始要求	
usbo_devinit	ER	USB_OTG_Handler* husb	USB デバイス初期化	
usbo_stopdevice	ER	USB_OTG_Handler* husb	USB デバイス停止	
usbo_init_lpm	ER	USB_OTG_Handler* husb	LPM の初期化	
usbo_pcd_irqhandler	void	USB_OTG_Handler* husb	USB デバイスの割込み処理	
usb_otg_isr	void	intptr_t exinf	USB-OTG 割込みサービスルーチン	
usb_otg_wkup_isr	void	intptr_t exinf	USB-OTG WKUP 割込みサービスルーチン	

表 4.2.7.2.1 USB-OTG ドライバ関数

4.2.7.3 フローチャート

USB-OTG はホスト機能とデバイス機能をもつペリフェラルである。ホスト機能をデバイスの接続を待って以下の機能を実行する。

(1) ホスト初期化

USB-OTG ハードウェア、上位モジュールの初期化を行い、USB ホストをスタートする。USB の

状態遷移は割込み関数からのコールバックで実行される。

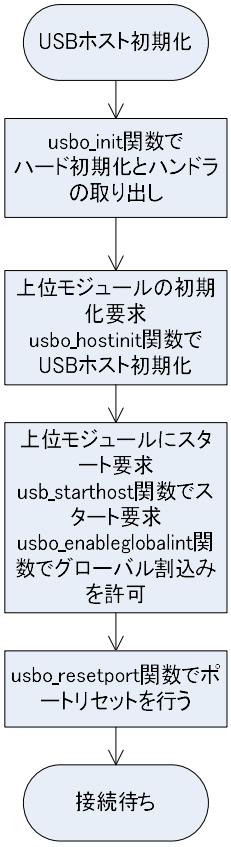


図 4.2.7.3.1 USB ホスト初期化

(2) ホスト接続とエナミュネーション

USB ホストの初期化が終了すると、デバイスの接続待ち状態に移行する。状態遷移は割込み関数中のコールバック関数で上位モジュールに伝達される。USB ホストで使用されるコールバック関数は以下の 4 つである

① hostsofcallback
SOF のタイミングで割込みを発生する。USB は 1ms 単位に SOF を発行するため 1ms 間隔にコールバックされる。

② hostconnectcallback
ホストのポート変化割込み(USB_OTG_GINTSTS_HPRTINT)が発生し、ポート接続チェックを行いコールバックする。

③ hostdisconnectcallback
ホストのディスコネクト割込み(USB_OTG_GINTSTS_DISCINT)が発生時、コールバックする。

④ hostchangeurbcallback
ホストチャネル割込み(USB_OTG_GINTSTS_HCINT)が発生時、コールバックする。

接続の検知は hostconnectcallback 関数呼び出しから開始する。コールバック関数はイベント通知のみで、処理はタスクレベルで実行される。

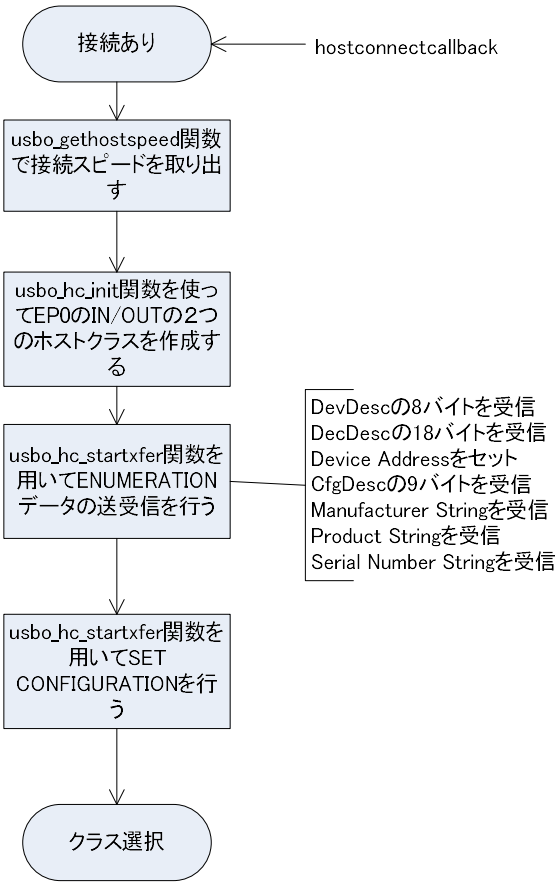


図 4.2.7.3.2 接続処理

(3) ホストクラスの選択と確認

上位モジュールでエナミュネーションにより取得した bInterfaceClass からクラスモジュールを選択する。対応するクラスモジュールがない場合は処理中止となる。クラスモジュールが決定された場合は、クラスモジュールで定めらえた手順で Init（通信用のエンドポイントの設定等）→Requests（クラスで使用するパラメータの取得等）を実行し、正常終了すればクラスモジュールの通常プロセスを実行する。これらの処理はクラスモジュール毎に異なるので、統一したフローチャートで記載はできない。ここで状態遷移に用いるコールバック関数は hostchangeurbcallback 関数である。

(4)ホスト切断

切断が行われた場合、USB ホストの割込みから hostdisconnectcallback が行われる。このコールバックにより、上位モジュールが呼ばれ上位モジュールの切断手順が行われる。手順は以下のフローチャートの通りである。クラスモジュールの Deinit 移行はタスクレベルで行われ、終了後接続待ちとなる。

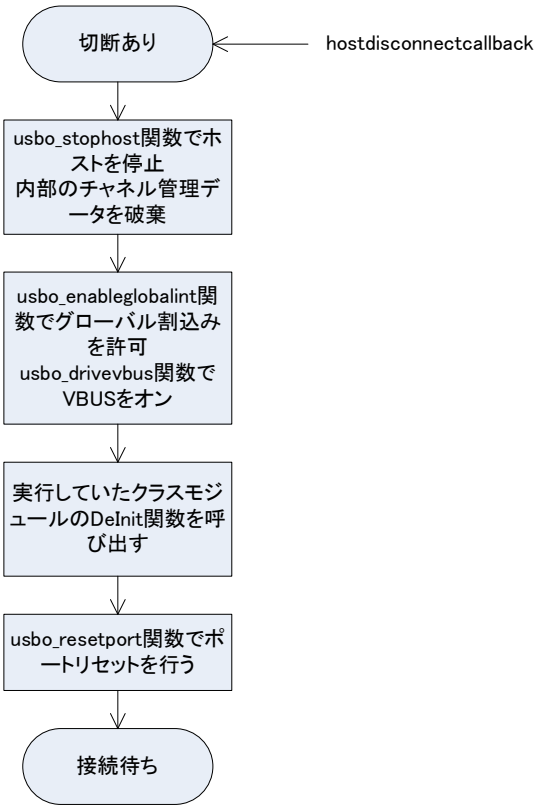


図 4.2.7.3.3 切断処理

デバイス側はホスト機能に対応する動作を行う。また、設定によっては、USB 割込み内で処理を完結可能である。状態の遷移は USB デバイスからのコールバックによって行われる。

① devsetupstagecallback
OUT エンドポイントの SETUP パケット通知割込み(USB_OTG_DOEPINT_STUP)からのコールバック
② devdataoutstagecallback
OUT エンドポイントの転送割込み(USB_OTG_DOEPINT_XFRC)からのコールバック
③ devdatainstagecallback
IN エンドポイントの送信終了割込み(USB_OTG_DIEPINT_XFRC)からのコールバック
④ devsofcallback
SOF 割込み(USB_OTG_GINTSTS_SOF)からのコールバック
⑤ devresetcallback
ENUMERATION 終了割込み(USB_OTG_GINTSTS_ENUMDNE)からのコールバック
⑥ devsuspendcallback
SUSPEND 割込み(USB_OTG_GINTSTS_USBSUSP)からのコールバック
⑦ devresumecallback
RESUME 割込み(USB_OTG_GINTSTS_WKUINT)からのコールバック
⑧ devisooutcallback
ISO-OUT 未完結割込み(USB_OTG_GINTSTS_PXFR_INCOMPISOOUT)からのコールバック
⑨ devisoincallback
ISO-IN 未完結割込み(USB_OTG_GINTSTS_IISOIXFR)からのコールバック
⑩ devconnectcallback
コネクション割込み(USB_OTG_GINTSTS_SRQINT)からのコールバック
⑪ devdisconnectcallback
ディスコネクション割込み(USB_OTG_GINTSTS_OTGINT)からのコールバック
⑫ devlpmcallback
LPM をサポートする場合の状態遷移コールバック

(5)デバイス初期化
USB デバイスの初期化手順を以下の表に示す。

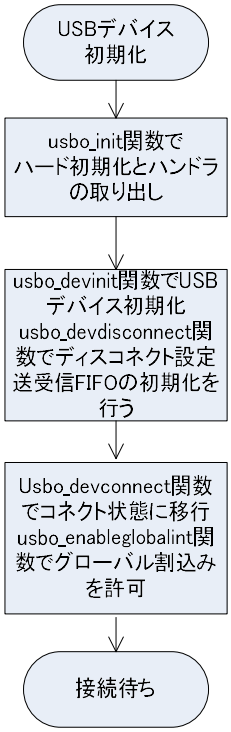


図 4.2.7.3.4 USB デバイスの初期化

(6) デバイス接続とエナミュネーション
USB の接続は、SUSPEND、RESUME のコールバックが何度か続いた後、リセットコールバック(devresetcallback)により、接続を開始する。これも、接続の状態で何度と発生する可能性がある。まず、エナミュレーションを行うためにエンドポイント 0 の IN/OUT を作成する。作成後、以下のコールバックが発生し、内容に従って、デバイス情報の通信を行う。

① devsetupstagecallback　　SETUP ステージの要求
② devdataoutstagecallback　　データ受信要求
③ devdatainstagecallback　　データ送信要求

最後に、SET CONFIGURATION を受信したら、USB クラスの通信に移行する。

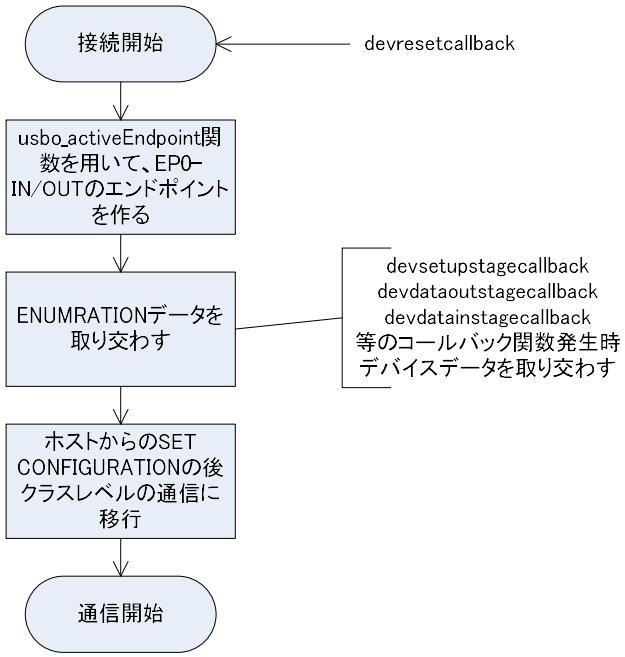


図 4.2.7.3.5 USB デバイスの接続

(7)SUSPEND と RESUME

USB デバイスの場合、通信確立後も SUSPEND と RESUME の要求により、必要な電源操作を行わなければならない。これらの移行は以下のコールバック関数によって要求される。切断、接続時も、これらのコールバック関数が呼ばれるので注意が必要である。

① devsuspendcallback　　省エネモード要求
② devresumecallback　　省エネ復帰要求

(8)デバイス切断

USB デバイスの切断は切断コールバック(devdisconnectcallback)の呼び出しにより行う。このコールバックの前に SUSPEND もコールバックも発生する。

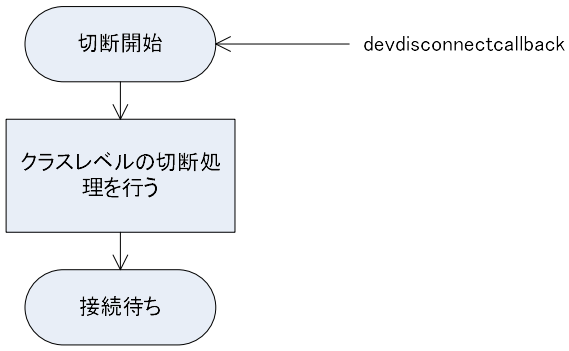


図 4.2.7.3.6 USB デバイスの切断

4.3 F7 Depend Driver

4.3.1 概要

F7 Depend Driver は、STM32F746 ハードウェア専用のデバイスドライバである。F7 専用ドライバとして以下のドライバをサポートする。

(1) ltdc　　ROCKTECH 4.3-inch 480x272 LCD-TFT 用デバイスドライバ
(2) tp　　FT5336 を用いた touch panel インターフェイスドライバ
(3) sdmmc　　SoC 内の SDMMC1 を用いた MicroSD card 用デバイスドライバ
(4) audio　　SoC 内の SAI と WM8994 を用いたオーディオ用デバイスドライバ

4.3.2 LTDC

LTDC(LCD-TFT Controller)は LCD-TFT 用のパラレル RGB で制御するコントローラである。
入力は以下のデータに対応する。
(1) ARGB8888
(2) RGB888
(3) RGB565
(4) ARGB1555
(5) ARGB4444
(6) L8(8-bit Luminance or CLUT)
(7) AL44 (4-bit alpha + 4-bit luminance)
(8) AL88 (8-bit alpha + 8-bit luminance)
出力は 24-bit RGB Parallel Pixel Output, 8 bits-per-pixel(RGB888)

4.3.2.1 データ仕様

LTDC の初期化に用いるデータと構造体について記載する。色の設定を表 4.3.2.1.1 の Color_t で定義する。LTDC の初期化には表 4.3.2.1.2 の LTDC_Init_t 型を使用する。レイア設定を表 4.3.2.1.3 の LTDC_LayerCfg_t 型を使用する。表 4.3.2.1.4 に LTDC ハンドラを定義する。

番号	項目	型	機能
1	Blue	uint8_t	青色
2	Green	uint8_t	緑色
3	Red	uint8_t	赤色
4	Reserved	uint8_t	リザーブ

表 4.3.2.1.1 Color_t 型

番号	項目	型	機能
1	HSPolaiity	uint32_t	水平同期極性設定
2	VSPolarity	uint32_t	垂直同期極性設定
3	DEPolarity	uint32_t	データ許可極性設定
4	PCPolarity	uint32_t	ピクセルクロック極性設定
5	HorizontalSync	uint32_t	水平同期設定(-1 の値を設定)
6	VerticalSync	uint32_t	垂直同期設定(-1 の値を設定)
7	AccumulatedHBP	uint32_t	計算上の水平バックポーチ値(-1 の値を設定)
8	AccumulatedVBP	uint32_t	計算上の垂直バックポーチ値(-1 の値を設定)
9	AccumulatedActiveW	uint32_t	計算上のアクティブ幅値(-1 の値を設定)
10	AccumulatedActiveH	uint32_t	計算上のアクティブ高さ値(-1 の値を設定)
11	TotalWidth	uint32_t	トータルの幅値(-1 の値を設定)
12	TotalHeight	uint32_t	トータルの高さ値(-1 の値を設定)
13	Backcolor	Color_t	バックグラウンド色

表 4.3.2.1.2 LTDC_Init_t 型

番号	項目	型	機能
1	WindowX0	uint32_t	Window の水平スタート位置
2	WindowX1	uint32_t	Window の水平ストップ位置
3	WindowY0	uint32_t	Window の垂直スタート位置
4	WindowY1	uint32_t	Window の垂直ストップ位置
5	PixelFormat	uint32_t	ピクセルのファーマット
6	Alpha	uint32_t	アルファ定数値
7	Alpa0	uint32_t	デフォルトアルファ値
8	BlendingFactor1	uint32_t	ブランディングファクター１
9	BlendingFactor2	uint32_t	ブランディングファクター２

10	FBStartAddress	uint32_t	ビットマップメモリのスタートアドレス
11	ImageWidth	uint32_t	イメージのピクセル数
12	ImageHeight	uint32_t	イメージのピクセル高さ
13	Backcolor	Color_t	バックグラウンド色

表 4.3.2.1.3 LTDC_LayerCfg_t 型

番号	項目	型	機能
1	Init	LTDC_Init_t	初期化型
2	LayerCfg[MAX_LAYER]	LTDC_LayerCfg_t	レイヤ型

表 4.3.2.1.4 LTDC_Handle_t 型

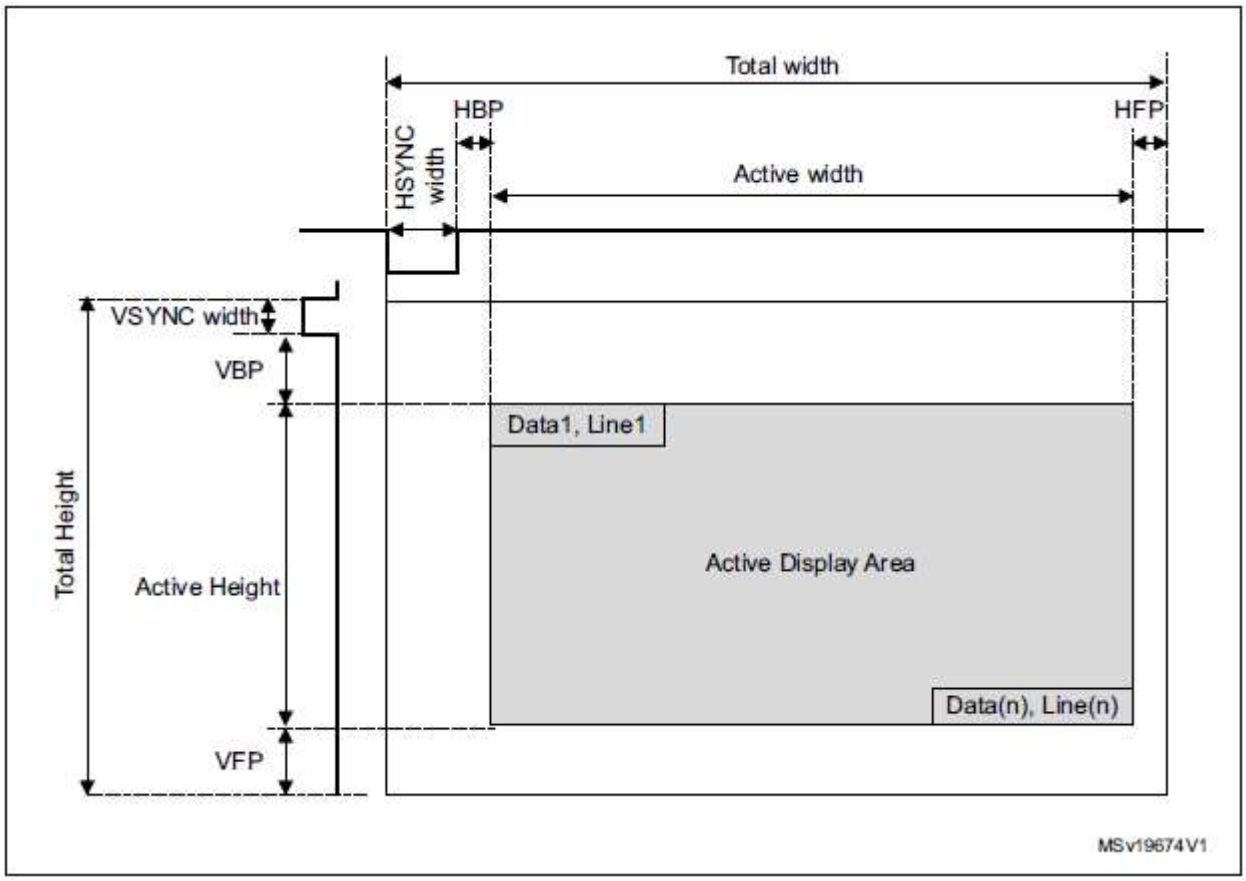


図 4.3.2.1.1 LTDC 設定値

① HSPolarity

水平同期極性設定を設定する

定義	値	内容
LTDC_HSPOLARITY_AL	0x00000000	アクティブロー
LTDC_HSPOLARITY_AH	LTDC_GCR_HSPOL	アクティブハイ

表 4.3.2.1.5 HSPolerty 設定値

② VSPolarity

垂直同期極性設定をする。

定義	値	内容
LTDC_VSPOLARITY_AL	0x00000000	アクティブロー
LTDC_VSPOLARITY_AH	LTDC_GCR_VSPOL	アクティブハイ

表 4.3.2.1.6 VSPolarity 設定値

③ DEPolarity
データ許可極性設定を行う。

定義	値	内容
LTDC_DEPOLARITY_AL	0x00000000	アクティブロー
LTDC_DEPOLARITY_AH	LTDC_GCR_DEPOL	アクティブハイ

表 4.3.2.1.7 DEPolarity 設定値

④ PCPolarity
ピクセルクロック極性設定を設定する。

定義	値	内容
LTDC_PCPOLARITY_IPC	0x00000000	Input pixel clock
LTDC_PCPOLARITY_IIPC	LTDC_GCR_PCPOL	Inverted input pixel clock

表 4.3.2.1.8 PCPolarity 設定値

⑤ BlendingFactor1
ブランディングファクター１を設定する。

定義	値	内容
LTDC_BLENDING_FACTOR1_CA	0x00000400	Cte Alpha
LTDC_BLENDING_FACTOR1_PAxCA	0x00000600	Cte Alpha x Pixel Alpha

表 4.3.2.1.9 BlendingFactor1 設定値

⑥ BlendingFactor2
ブランディングファクター１を設定する。

定義	値	内容
LTDC_BLENDING_FACTOR2_CA	0x00000005	Input pixel clock
LTDC_BLENDING_FACTOR2_PAxCA	0x00000007	Cte Alpha x Pixel Alpha

表 4.3.2.1.10 BlendingFactor2 設定値

4.3.2.2 インターフェイス仕様

LTDC を設定するドライバ関数を以下に示す。

関数名	型	引数	機能	備考
ltdc_init	ER	LTDC_handle_t *phandle	LTDC の初期化を行う	必須
ltdc_configlayer	ER	LTDC_handle_t *phandle LTDC_LayerCfg_t *pconfg uint32_t index	index に対応したレイヤ設定を設定する	必須
ltdc_setalpha	ER	LTDC_handle_t *phandle uint32_t Alpha uint32_t index	index に対応したレイヤのアルファ値を変更する	オプション
ltdc_setaddress	ER	LTDC_handle_t *phandle uint32_t Adress uint32_t index	index に対応したレイヤのスタートアドレスを変更する	オプション
ltdc_setwindowsize	ER	LTDC_handle_t *phandle uint32_t XSize uint32_t YSize uint32_t index	index に対応したレイヤの Window サイズとイメージ幅、高さを変更する	オプション
ltdc_setwindowposition	ER	LTDC_handle_t *phandle uint32_t X0 uint32_t Y0 uint32_t index	index に対応したレイヤの Window の起点を(X0,Y0)に変更する	オプション
ltdc_configcolorkeying	ER	LTDC_handle_t *phandle uint32_t RGBValue	index に対応したレイヤのカラーキーイング値	オプション

		uint32_t index	を設定する	
ltdc_enablecolorkeying	ER	LTDC_handle_t *phandle uint32_t index	index に対応したレイヤのカラーキーイング設定を有効にする	オプション
ltdc_disablecolorkeying	ER	LTDC_handle_t *phandle uint32_t index	index に対応したレイヤのカラーキーイング設定を無効にする	オプション

表 4.3.2.2.1 LTDC 設定関数

4.3.2.3 設定手順

本開発環境では、LTDC の設定は Stm32Cube_FW_F7 中の stm32746_discovery_lcd.c の BSP 機能を用いて設定している。これは GUI として BSP の機能をそのまま流用するためである。ボードで使用している 4.3-inch 480x272 のカラーLCD を用に LTDC の設定を行うためには、LCD に対応した設定パラメータを LTDC_Init_t 型を介してハンドラに設定し、ltdc_init 関数を用いて行い。レイヤの設定を、ltdc_configlayer 関数を用いて行えばよい。設定後は表示データに対応した RAM 領域を描画内容に従って書き換えれば、その内容が表示データとして LCD に表示される。

番号	項目	設定値
1	HSPolaiity	LTDC_HSPOLARITY_AL
2	VSPolarity	LTDC_VSPOLARITY_AL
3	DEPolarity	LTDC_DEPOLARITY_AL
4	PCPolarity	LTDC_PCPOLARITY_IPC
5	HorizontalSync	RK043FN48H_HSYNC(41) - 1
6	VerticalSync	RK043FN48H_VSYNC(10) - 1
7	AccumulatedHBP	RK043FN48H_HSYNC - RK043FN48H_HBP(13) - 1
8	AccumulatedVBP	RK043FN48H_VSYNC - RK043FN48H_VBP(2) - 1
9	AccumulatedActiveW	RK043FN48H_WIDTH(480) + RK043FN48H_HSYNC + RK043FN48H_HBP - 1
10	AccumulatedActiveH	RK043FN48H_HEIGHT(272) + RK043FN48H_VSYNC + RK043FN48H_VBP - 1
11	TotalWidth	RK043FN48H_WIDTH + RK043FN48H_HSYNC + RK043FN48H_HBP + RK043FN48H_HFP(32) - 1
12	TotalHeight	RK043FN48H_HEIGHT + RK043FN48H_VSYNC + RK043FN48H_VBP + RK043FN48H_VFP(2) - 1
13	Backcolor	オールゼロ

表 4.3.2.3.1 480x272LCD の LTDC_Init_t 型の設定値

番号	項目	機能
1	WindowX0	0
2	WindowX1	RK043FN48H_WIDTH(480)
3	WindowY0	0
4	WindowY1	RK043FN48H_HEIGHT(272)
5	PixelFormat	LTDC_PIXEL_FORMAT_ARGB8888
6	Alpha	255
7	Alpa0	0
8	BlendingFactor1	可変
9	BlendingFactor2	可変
10	FBStartAddress	メモリアドレス（可変）
11	ImageWidth	RK043FN48H_WIDTH
12	ImageHeight	RK043FN48H_HEIGHT
13	Backcolor	オールゼロ

表 4.3.2.3.2 480x272LCD の LTDC_LayerCfg_t 型の設定値

4.3.3 TP

TP(touch panel Controller)は FT5336 を使用しており、LCD をタッチした場合、その座標を読み取ることができる。ドライバ自体は StmCube で供給されており、I2C ポート 3 に接続されている。I2C の制御に関しては、BSP 中の stm32746g_discovery.c が一括して管理し、TP 部の制御は BSP 中の stm32746g_discovery_ts.c 中のインターフェイス関数を、そのまま使用している。

4.3.4 SDMMC

SDMMC(SD/SDIO/MMC card host interface)は SD カードメモリや SDIO とのインターフェイスを行うペリフェラルである。STM32F746 Discovery ボードではマイクロ SD カードソケットに実装されているため SD カード、SDHC カードとのインターフェイスを行うように設計している。SD カードとの通信には送受信とも DMA を使用している。Cortex-F7 ではデータキャッシュ機能を持つため、キャッシュ処理も同時に行っている。

4.3.4.1 データ仕様

SDMMC ドライバは SD カード用に設計している。表 4.3.4.1.1 に SDMMC ドライバ用のハンドラ型 SDMMC_Handle_t を示す。表 4.3.4.1.2 に SD カード情報を設定する SDMMC_CardInfo_t 型を定義する。

番号	項目	型	機能
1	base	uint32_t	SDMMC レジスタベースアドレス
2	ClockMode	uint32_t	指定クロックモード
3	BusWide	uint32_t	指定バス幅
4	RetryCount	uint32_t	リトライ回数
5	cardtype	uint32_t	カード種類
6	RCA	uint32_t	RCA 値
7	CSD[4]	uint32_t	CSD 値
8	CID[4]	uint32_t	CID 値
9	status	volatile uint32_t	転送状態
10	SdCmd	volatile uint32_t	転送指定コマンド
11	hdmarx	DMA_Handle_t *	受信用 DMA ハンドラ
12	hdmatx	DMA_Handle_t *	送信用 DMA ハンドラ

表 4.3.4.1.1 SDMMC_Handle_t 型

番号	項目	型	機能
1	capacity	uint64_t	容量(バイト)
2	blocksize	uint32_t	ブロックサイズ
3	maxsector	uint32_t	ブロックの数
4	RCA	uint16_t	SD RCA 値
5	cardtype	uint8_t	カード種類
6	status	uint8_t	ステータス

表 4.3.4.1.2 SDMMC_CardInfo_t 型

① ClockMode

クロックモードを設定する。

定義	値	内容
SDMMC_TRANSFER_CLK_DIV	0x0	DIV なし

表 4.3.4.1.3 ClockMode 設定値

② BusWidth

SDMMC の使用データバス幅を設定する。

定義	値	内容

SDMMC_BUS_WIDE_1B	0x00000000	1 ビット
SDMMC_BUS_WIDE_4B	SDMMC_CLKCR_WIDBUS_0	4 ビット
SDMMC_BUS_WIDE_8B	SDMMC_CLKCR_WIDBUS_1	8 ビット

表 4.3.4.1.4 BusWidth 設定値

③ cardtype

カード情報取得後判定したカード種類

定義	値	内容
SD_CARD_V11	0	SD CARD V1.1
SD_CARD_V20	1	SD CARD V2.0
SD_CARD_HC	2	SDHC CARD
MMC_CARD	3	MMC CARD(未対応)
SD_IO_CARD	4	SD IO CARD(未対応)
HS_MM_CARD	5	HS MMC CARD(未対応)
SD_IO_COMBO_CARD	6	SD IO CARD(未対応)
MMC_CARD_HC	7	MM HC CARD(未対応)

表 4.3.4.1.5 cardtype 設定値

ドライバの返り値を ER コードで設定しているが、SDMMC 独自エラーが発生した場合の拡張定義を表 4.3.4.1.6 に示す。

定義	値	内容
E_SDCOM	-80	コマンドエラー
E_SDCRC	-81	CRC エラー
E_SDECMD	-82	コマンドインデックスエラー
E_SDVOL	-83	電圧エラー
E_SDTRS	-84	通信エラー

表 4.3.4.1.6 エラーコード拡張

4.3.4.2 インターフェイス仕様

SDMMC を設定するドライバ関数を以下に示す。ドライバは表 4.3.5.2.1 に示す基本動作を行うドライバと、表 4.3.5.2.2 の SD カードの初期設定を行うドライバがある。初期設定ドライバは後述の Storage Device Manager からカード検知時、一定の手順で呼び出しを行う。

関数名	型	引数	機能	備考
sdmmc_init	void	intptr_t exinf	SDMMC ハード初期化	
sdmmc_sense	bool_t	int id	指定 ID の SD card をセンスする。ある場合 true を返す	
sdmmc_open	SDMMC_Handle_t *	int id	指定 ID の SDMMC ドライバをオープンする。戻り値はハンドラへのポインタ	
sdmmc_close	ER	SDMMC_handle_t *hsd	SDMMC ドライバをクローズする	
sdmmc_erase	ER	SDMMC_handle_t *hsd uint64_t startaddr uint64_t endaddr	SD card のイレーズを行う	
sdmmc_blockread	ER	SDMMC_handle_t *hsd uint32_t *pbuff uint32_t ReadAddr uint32_t blocksize uint32_t num	SD card からブロック単位で READ する	
sdmmc_blockwrite	ER	SDMMC_handle_t *hsd uint32_t *pbuff	SD card へブロック単位で WRITE する	

		uint32_t WriteAddr uint32_t blocksize uint32_t num		
sdmmc_wait_transfar	ER	SDMMC_handle_t *hsd uint32_t Timeout	READ/WRITE 転送待ちを行う	
sdmmc_checkint	void	SDMMC_handle_t *hsd	割込み関数	

表 4.3.4.2.1 SDMMC 基本ドライバ関数

関数名	型	引数	機能	備考
sdmmc_getcardinfo	ER	SDMMC_handle_t *hsd SDMMC_CardInfo_t *pCinfo	コネクトした SD card の情報を取り出す	
sdmmc_checkCID	ER	SDMMC_handle_t *hsd	CID を取得する	
sdmmc_setaddress	ER	SDMMC_handle_t *hsd	相対アドレス(RCA)を取得する。	
sdmmc_sendCSD	ER	SDMMC_handle_t *hsd	CSD を取得する。	
sdmmc_select_card	ER	SDMMC_handle_t *hsd uint32_t addr	カードセレクトコマンドを発行する	
sdmmc_configuration	ER	SDMMC_handle_t *hsd	コンフィギュレーション設定を行う	
sdmmc_set_widebus	ER	SDMMC_handle_t *hsd	バス幅を設定する	
sdmmc_getstatus	ER	SDMMC_handle_t *hsd	SD カードステートを取り込む。	

表 4.3.4.2.2 SDMMC 初期化ドライバ関数

4.3.4.3 設定手順

SDMMC のハードウェアの初期化は電源投入時、リセット後に sdmmc_init 関数を用いて行う。SD カードの場合、スロットにメディアがあるかないかで初期化手順が異なる。メディアの管理は後述の Storage Device Manager が行う。挿抜処理が可能なメディアの場合、Storage Device Manager は 500ms 周期に mci_ses_por 関数（仮想関数で SDMMC の場合、sdmmc_init が呼び出される）を呼び出してメディアの有無をチェックする。これ以降は、図 4.3.5.3.1 の初期化フローに従い、メディアが挿入された場合は、初期化処理を行う。初期化処理中にエラーが発生した場合は、メディア使用不可の設定が行われ、すべて正常に終了した場合、メディアが使用可能になる。使用可能な状態の場合、ファイルシステムを用いて、ファイル処理が可能になる。メディアが抜かれた場合は、mci_cls_por 関数（仮想関数で SDMMC の場合、sdmmc_close が呼び出される）を呼び出し、メディアを使用不可にする。

SD カードのドライバは STM32F4xx の場合、SPI となる。SPI の場合は SPI 用にドライバを用いて SD カードの初期化やアクセスを行う。上位層が仮想関数を用いるのは、ドライバ関数を入れ替えれば上位層のプログラムを書き換えなくてもファイルシステムを処理させるためである。

メディアの初期化が終了したと、FATFs のドライバ層を経由して、メディアに対するブロック単位の READ/WRITE と IO アクセスが発生する。図 4.3.5.3.2 に FATFs のドライバからのブロックリードのフローを示す。ブロックライトは実行関数を mci_wri_blk(sdmmc_block_write)に置き換えればよい。

なお、ブロックリード、ライトのバッファ領域のポインタが 4 バイトアラインとなっているが、FATFs のブロックリード、ライトドライバは 1 バイトアラインの領域にデータの読み書きを指定する。そのため、SDMMC ドライバのブロックリード、ライトは 1 バイトアラインの読み書きを有効にしている。

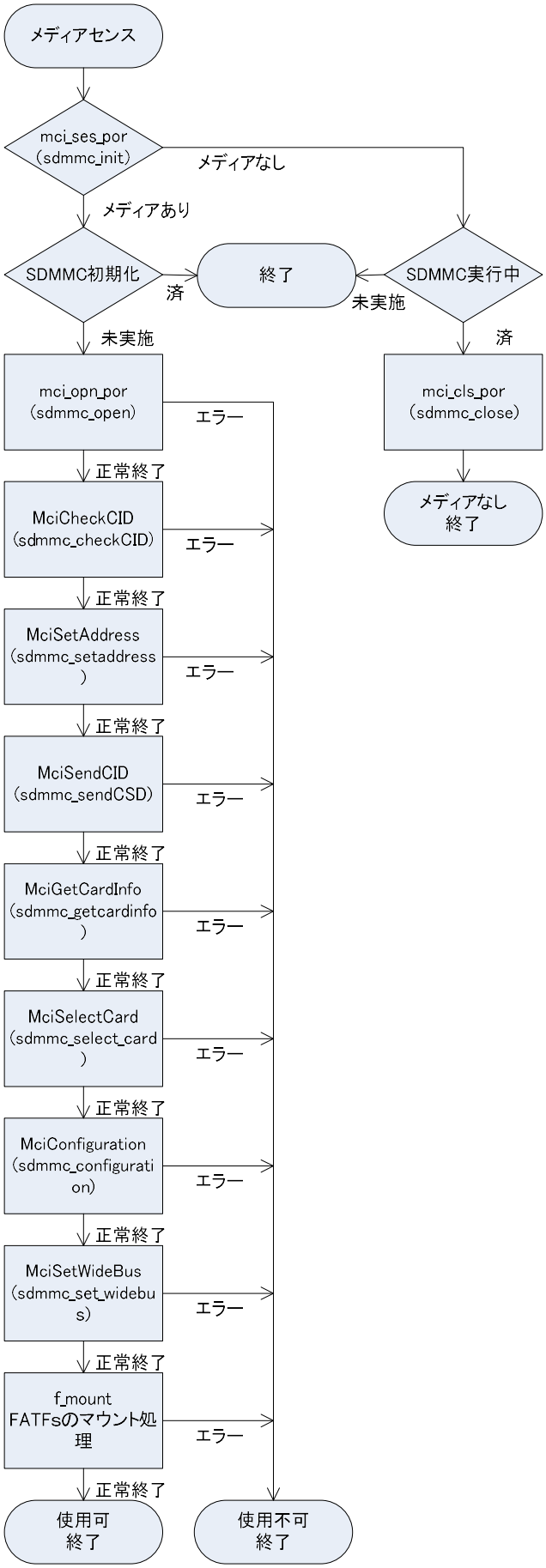


図 4.3.4.3.1 初期化フロー

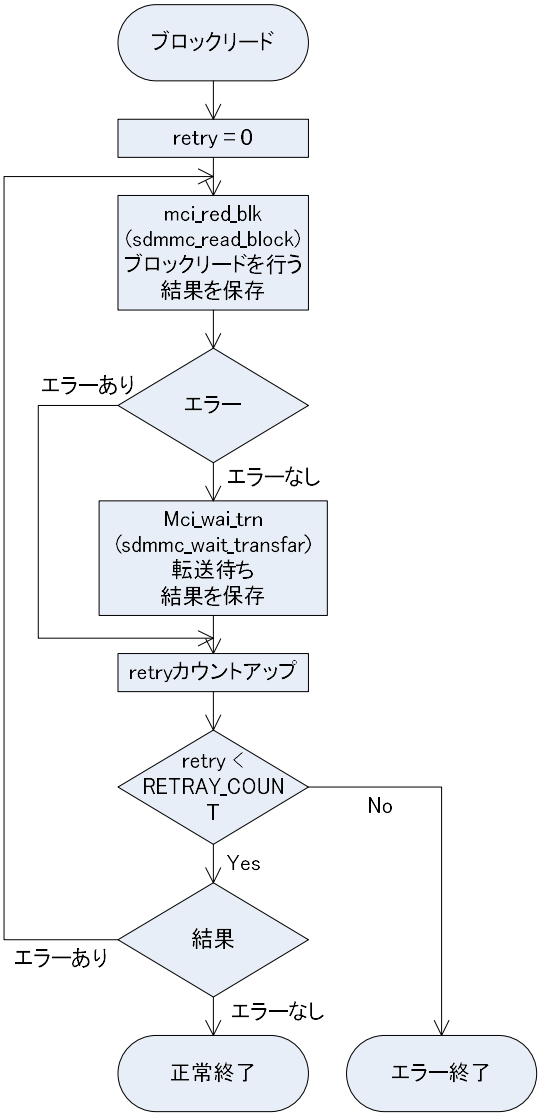


図 4.3.4.3.2 ドライバーブロックリード

4.3.5 AUDIO

AUDIO ドライバは、入力としてアナログ音楽信号をデジタルの PCM データに変換を行い、出力としてデジタル PCM データをアナログ音楽信号に変換する。STM32F746-Discovery では、アナログ出力用に wm8994 を使用している。wm8994 は TP 用のドライバと同様に I2C ポート 3 に接続されている。I2C の制御に関しては、BSP 中の stm32746g_discovery.c が一括して管理し、これを含めたオーディオ制御は BSP 中の stm32746g_discovery_audio.c 中のインターフェイス関数を、そのまま使用しても制御を行うことができる。

4.3.5.1 データ仕様

AUDIO ドライバは入力部と出力部の共通に初期化を行う。ハンドラを用いて入力と出力の制御を行う。入力処理と出力処理は使用する関数がことなるが、ハンドラは共通に使用する。表 4.3.5.1.1 に SDMMC ドライバ用の初期化型 SAI_Init_t を示す。表 4.3.5.1.2 にハンドラ型 AUDIO_Handle_t 型を定義する。ハードウェア定義構造体は内部処理に使用するため構成は示さない。

番号	項目	型	機能
1	AudioMode	uint32_t	AUDIO モード
2	Synchro	uint32_t	SAI ブロックシンクロ値

3	SynchroExt	uint32_t	SAI ブロックシンクロ拡張値
4	OutputDrive	uint32_t	出力デバイス
5	NoDivider	uint32_t	DIVIDER 有効/無効
6	FIFOThreshold	uint32_t	FIFO スレッシュホールド設定
7	AudioFrequency	uint32_t	AUDIO サンプリング周波数
8	Mckdiv	uint32_t	マスタークロック DIVIDER 値
9	MonoStereoMode	uint32_t	モノクロ/ステレオモード
10	CompandingMode	uint32_t	コンパウンディングモード
11	TriSize	uint32_t	TRIState マネージメント設定
12	Protocol	uint32_t	SAI BLOCK プロトコル
13	DataSize	uint32_t	SAI BLOCK データサイズ
14	FirstBit	uint32_t	MSB/LSB 設定
15	ClockStrobing	uint32_t	クロック・ストロービング設定
16	FrameLength	uint32_t	フレーム長
17	ActiveFrameLength	uint32_t	アクティブ・フレーム長
18	FSDefinition	uint32_t	フレーム・シンクロナス定義
19	FSPolarity	uint32_t	FS POLARITY 設定
20	FSOffset	uint32_t	FS OFFSET 設定
21	FirstBitOffset	uint32_t	スロット中の最初のデータビット位置
22	SlotSize	uint32_t	スロットサイズ
23	SlotNumber	uint32_t	スロット番号
24	SlotActive	uint32_t	SLOT アクティブ設定

表 4.3.5.1.1 SAI_Init_t 型

番号	項目	型	機能
1	base	uint32_t	SAI デバイスのベースアドレス
2	pcb	AUDIO_PortControlBlock*	ハードウェア定義構造体へのポインタ
3	OutInit	SAI_Init_t	出力ブロック初期設定
4	InInit	SAI_Init_t	入力ブロック初期設定
5	pBuffPtr	uint8_t*	データ領域へのポインタ
6	XferSize	uint16_t	データサイズ
7	XferCount	uint16_t	送受信カウント
8	hdmatx	DMA_Handle_t *	送信 DMA ハンドラへのポインタ
9	hdmarx	DMA_Handle_t *	受信 DMA ハンドラへのポインタ
10	audiomutecallback	void(*)(Audio_Handle_t*)	MUTE コールバック関数
11	audioerrorcallback	void(*)(Audio_Handle_t*)	割込みエラーコールバック関数
12	transcallback	void(*)(Audio_Handle_t*)	送信終了コールバック関数
13	transhalfcallback	void(*)(Audio_Handle_t*)	ハーフ送信終了コールバック関数
14	recevcallback	void(*)(Audio_Handle_t*)	受信終了コールバック関数
15	recvhalfcallbak	void(*)(Audio_Handle_t*)	ハーフ受信終了コールバック関数
16	errorcallback	void(*)(Audio_Handle_t*)	エラーコールバック関数
17	status[2]	volatile uint32_t	AUDIO ステータス
18	ErrorCode	volatile uint32_t	エラーコード

表 4.3.5.1.2 AUDIO_Handle_t 型

① AudioMode

オーディオモードを設定する。

定義	値	内容
SAI_MODEMASTER_TX	0x0000000	マスタ送信モード
SAI_MODEMASTER_RX	SAI_xCR1_MODE_0	マスタ受信モード
SAI_MODESLAVE_TX	SAI_xCR1_MODE_1	スレーブ送信モード
SAI_MODESLAVE_RX	(SAI_xCR1_MODE_0 |	スレーブ受信モード

	SAI_xCR1_MODE_1)	

表 4.3.5.1.3 AudioMode 設定値

② Syncro

シンクロモードを設定する。

定義	値	内容
SAI_ASYNCHRONOUS	0x00000000	アシンクロナス
SAI_SYNCHRONOUS	SAI_xCR1_SYNCEN_0	シンクロナス
SAI_SYNCHRONOUS_EXT	SAI_xCR1_SYNCEN_1	拡張シンクロナス

表 4.3.5.1.4 Syncro 設定値

③ SynchroExt

SAI ブロックシンクロナス拡張値

定義	値	内容
SAI_SYNCEXT_DISABLE	0x00000000	無効
SAI_SYNCEXT_IN_ENABLE	0x00000001	IN 有効
SAI_SYNCEXT_OUTBLOCKA_ENABLE	0x00000002	OUTBLOCKA 有効
SAI_SYNCEXT_OUTBLOCKB_ENABLE	0x00000004	OUTBLOCKB 有効

表 4.3.5.1.5 SynchroExt 設定値

④ OutputDrive

出力ドライブ設定

定義	値	内容
SAI_OUTPUTDRIVE_DISABLE	0x00000000	無効
SAI_OUTPUTDRIVE_ENABLE	SAI_xCR1_OUTDRIV	有効

表 4.3.5.1.6 OutputDrive 設定値

⑤ NoDivider

DIVIER 有効、無効設定

定義	値	内容
SAI_MASTERDIVIDER_ENABLE	0x00000000	有効
SAI_MASTERDIVIDER_DISABLE	SAI_xCR1_NODIV	無効

表 4.3.5.1.7 NoDivider 設定値

⑥ FIFOThreshold

FIFO スレシュホールド設定

定義	値	内容
SAI_FIFOTHRESHOLD_EMPTY	0x00000000	0
SAI_FIFOTHRESHOLD_1QF	SAI_xCR2_FTH_0	1/4
SAI_FIFOTHRESHOLD_HF	SAI_xCR2_FTH_1	1/2
SAI_FIFOTHRESHOLD_3QF	(SAI_xCR2_FTH_0| SAI_xCR2_FTH_1)	3/4
SAI_FIFOTHRESHOLD_FULL	SAI_xCR2_FTH_2	1

表 4.3.5.1.8 FIFOThreshold 設定値

⑦ MonoStereoMode

モノクロ/ステレオモード設定

定義	値	内容
SAI_STEREOMODE	0x00000000	ステレオ
SAI_MONOMODE	SAI_xCR1_MONO	モノクロ

表 4.3.5.1.9 MonoStrereoMode 設定値

⑧ CompandMode

コンパウンディングモード設定

定義	値	内容
SAI_NOCOMPANDING	0x00000000	なし
SAI_ULAW_1CPL_COMPANDING	SAI_xCR2_COMP_1	ULAW 1CPL
SAI_ALAW_1CPL_COMPANDING	(SAI_xCR2_COMP_1| SAI_xCR2_COMP_0)	ALAW 1CPL
SAI_ULAW_2CPL_COMPANDING	(SAI_xCR2_COMP_1| SAI_xCR2_CPL)	ULAW 2CPL
SAI_ALAW_2CPL_COMPANDING	(SAI_xCR2_COMP_1| SAI_xCR2_COMP_0| SAI_xCR2_CPL)	ALAW 2CPL

表 4.3.5.1.10 CompandingMode 設定値

⑨ TriState

TriState マネージメント設定

定義	値	内容
SAI_OUTPUT_NOTRELEASED	0x00000000	無効
SAI_OUTPUT_RELEASED	SAI_xCR2_TRIS	有効

表 4.3.5.1.11 TriState 設定値

⑩ Protocol

BLOCK プロトコル設定

定義	値	内容
SAI_FREE_PROTOCOL	0x00000000	フリー
SAI_SPDIF_PROTOCOL	SAI_xCR1_PRTCFG_0	SPDIF
SAI_AC97_PROTOCOL	SAI_xCR1_PRTCFG_1	AC97

表 4.3.5.1.12 Protocol 設定値

⑪ DataSize

BLOCK データサイズ設定

定義	値	内容
SAI_DATASIZE_8	SAI_xCR1_DS_1	8
SAI_DATASIZE_10	(SAI_xCR1_DS_1| SAI_xCR1_DS_0)	10
SAI_DATASIZE_16	SAI_xCR1_DS_2	16
SAI_DATASIZE_20	(SAI_xCR1_DS_2| SAI_xCR1_DS_0)	20
SAI_DATASIZE_24	(SAI_xCR1_DS_2| SAI_xCR1_DS_1)	24
SAI_DATASIZE_32	(SAI_xCR1_DS_2| SAI_xCR1_DS_1| SAI_xCR1_DS_0)	32

表 4.3.5.1.13 DataSize 設定値

⑫ FirstBit

MSB/LSB 設定

定義	値	内容
SAI_FIRSTBIT_MSB	0x00000000	MSB
SAI_FIRSTBIT_LSB	SAI_xCR1_LSBFIRST	LSB

表 4.3.5.1.14 FirstBit 設定値

⑬ ClockStrobing

クロック・ストロービング設定

定義	値	内容

SAI_CLOCKSTROBING_FALLINGEDGE	0x00000000	立下りエッジ
SAI_CLOCKSTROBING_RISINGEDGE	SAI_xCR1_CKSTR	立ち上がりエッジ

表 4.3.5.1.15 ClockStrobing 設定値

⑭ FSDefinition

フレーム・シンクロナス設定

定義	値	内容
SAI_FS_STARTFRAME	0x00000000	START FRAME
SAI_FS_CHANNEL_IDENTIFICATION	SAI_xFRCR_FSDEF	CHANNEL IDENTIFICATION

表 4.3.5.1.16 FSDefintion 設定値

⑮ FSPolarity

フレーム・シンクロナス POLARITY 設定

定義	値	内容
SAI_FS_ACTIVE_LOW	0x00000000	アクティブロー
SAI_FS_ACTIVE_HIGH	SAI_xFRCR_FSPO	アクティブハイ

表 4.3.5.1.17 FSPolarity 設定値

⑯ FSOffset

フレーム・シンクロナス　オフセット設定

定義	値	内容
SAI_FS_FIRSTBIT	0x00000000	ファーストビット
SAI_FS_BEFOREFIRSTBIT	SAI_xFRCR_FSDEF	ファーストビットの前

表 4.3.5.1.18 FSDefintion 設定値

⑰ SlotSize

スロットサイズ設定

定義	値	内容
SAI_SLOTSIZE_DATASIZE	0x00000000	データサイズ
SAI_SLOTSIZE_16B	SAI_xSLOTR_SLOTSZ_0	16 ビット
SAI_SLOTSIZE_32B	SAI_xSLOTR_SLOTSZ_1	32 ビット

表 4.3.5.1.19 SlotSize 設定値

⑱ SlotActive

スロットアクティブ設定

定義	値	内容
SAI_SLOT_NOTACTIVE	0x00000000	なし
SAI_SLOTACTIVE_0	0x00010000	0
SAI_SLOTACTIVE_1	0x00020000	1
SAI_SLOTACTIVE_2	0x00040000	2
SAI_SLOTACTIVE_3	0x00080000	3
SAI_SLOTACTIVE_4	0x00100000	4
SAI_SLOTACTIVE_5	0x00200000	5
SAI_SLOTACTIVE_6	0x00400000	6
SAI_SLOTACTIVE_7	0x00800000	7
SAI_SLOTACTIVE_8	0x01000000	8
SAI_SLOTACTIVE_9	0x02000000	9
SAI_SLOTACTIVE_10	0x04000000	10
SAI_SLOTACTIVE_11	0x08000000	11
SAI_SLOTACTIVE_12	0x10000000	12
SAI_SLOTACTIVE_13	0x20000000	13
SAI_SLOTACTIVE_14	0x40000000	14
SAI_SLOTACTIVE_15	0x80000000	15

SAI_SLOTACTIVE_FULL	0xFFFF0000	フル

表 4.3.5.1.20 SlotActive 設定値

AUDIO ハンドラ内のエラー定義(ErrorCode)を以下の表に示す。

定義	値	内容
AUDIO_ERROR_NONE	0x00000000	エラーなし
AUDIO_ERROR_OVRUDR	0x00000001	オーバーランまたはアンダーラン
AUDIO_ERROR_AFSDET	0x00000004	Anticipated Frame synchronisation detection
AUDIO_ERROR_LFSDET	0x00000008	Late Frame synchronisation detection
AUDIO_ERROR_WCKCFG	0x00000010	Wrong clock configuration
AUDIO_ERROR_TIMEOUT	0x00000040	タイムアウト

表 4.3.5.1.21 エラーコード値

4.3.5.2 インターフェイス仕様

AUDIO ドライバは表 4.3.5.2.1 の設定ドライバと表 4.3.6.2.2 の入出力制御を行うドライバの二種類がある。引数で設定する mode は入力用と主力用に関数を設定する。mode の設定値は以下の２つがある。

① AUDIO_OUT_BLOCK：出力制御を行う
② AUDIO_IN_BLOCK：入力制御を行う

関数名	型	引数	機能	備考
audio_init	AUDIO_Handle_t*	ID id	AUDIO ハード初期化	
audio_deinit	void	AUDIO_Handle_t* haudio uint32_t mode	AUDIO 設定解除	
audio_clockconfig	void	AUDIO_Handle_t* haudio uint32_t AudioFreq void *Param	AUDIO クロック設定	
audio_irqhandler	void	SDMMC_handle_t *hsd	AUDIO 割込み関数	

表 4.3.5.2.1 AUDIO 設定ドライバ関数

関数名	型	引数	機能	備考
audio_start	ER	AUDIO_Handle_t* haudio uint32_t mode	入出力機能開始	
audio_end	ER	AUDIO_Handle_t* haudio uint32_t mode	入出力機能終了	
audio_transmit	ER	AUDIO_Handle_t* haudio uint8_t *pData uint16_t Size	出力データ転送開始	出力
audio_receive	ER	AUDIO_Handle_t* haudio uint8_t *pData uint16_t Size	入力データ転送開始	入力
audio_dmapause	ER	AUDIO_Handle_t* haudio uint32_t mode	入出力停止	
audio_dmaresume	ER	AUDIO_Handle_t* haudio uint32_t mode	入出力再開	
audio_dmastop	ER	AUDIO_Handle_t* haudio uint32_t mode	入出力終了	
audio_enable	void	SDMMC_handle_t *hsd	SD カードステートを取り込む。	
audio_disable	ER	AUDIO_Handle_t* haudio uint32_t mode	SAI 開始	
audio_gethandle	AUDIO_Ha	ID id	SAI 停止	

	ndle_t*			
audio_status	void	AUDIO_Handle_t* haudio uint32_t mode	実行状態取出し	

表 4.3.5.2.2 AUDIO 制御ドライバ関数

4.3.5.3 設定手順

音楽演奏の手順について記載する。STM32F746 Discovery ではオーディオデータの管理は SAI モジュールで行う、AUDIO ドライバは SAI モジュールの制御が主な作業である。オーディオデータをコネクタに入手力を行うのは wm8994 が行う、wm8994 は I2C3 に接続されており、I2C を通して制御を行う。ここではオーディオ出力の手順について説明を行う。MP3 用の PCM データは数１０MB から数１００MB に及ぶため、すべてをメモリに蓄えて演奏を行うことはできない。演奏スピードに合わせてデータ領域の演奏済領域に次の演奏データを絶え間なく書き込み、DMA を循環モードにして連続演奏を行う。STM32F のストリーム DMA には、半分転送終了時とデータ領域全体の転送終了時に完了割込みを発生させる機能があり、コールバック関数でイベントを取り込み、メインで音楽演奏を行っているタスクにデータ書き込みを要求していくことで大容量の演奏を可能にしている。

図 4.3.5.3.1 の初期化フローでは、SAI ハードウェアの初期化を行う。この時点で SAI モジュールを入力用に使用するか、出力用に使用するかは決定されていない。

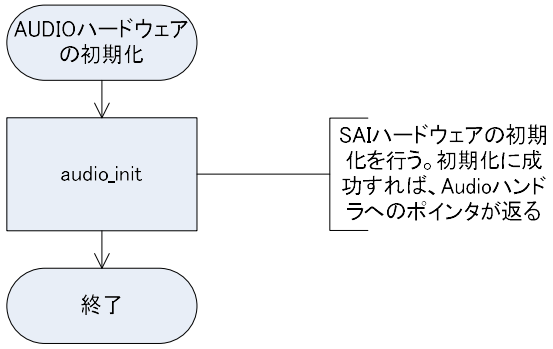


図 4.3.5.3.1 AUDIO 初期化

図 4.3.5.3.2 の出力設定フローチャートで、SAI を出力用にデータ設定を行い、wm8994 の初期化を行う。

図 4.3.5.3.2 の演奏開始フローチャートで、演奏用のデータをデータ領域にセットし、コールバック関数を設定後、演奏を開始する。

演奏中は、図 4.3.5.3.3 の二つのコールバック関数のフローチャートのように、半分のデータが wm8994 に転送終了時、DMA ハーフ転送コールバック関数が読みだされる。これにより、データ領域の前半分が空となったことがわり、演奏用のメインタスクに前半分の次のデータをセットするように要求を行う。DMA 全転送終了コールバック関数が読みだされた場合、データ領域の後ろ半分が空になったことがわかり、演奏用のメインタスクに後ろ半分の次のデータをセットするように要求を行う。両方のコールバック関数で、演奏データがないとき、演奏用のメインタスクに演奏の終了を通知する。

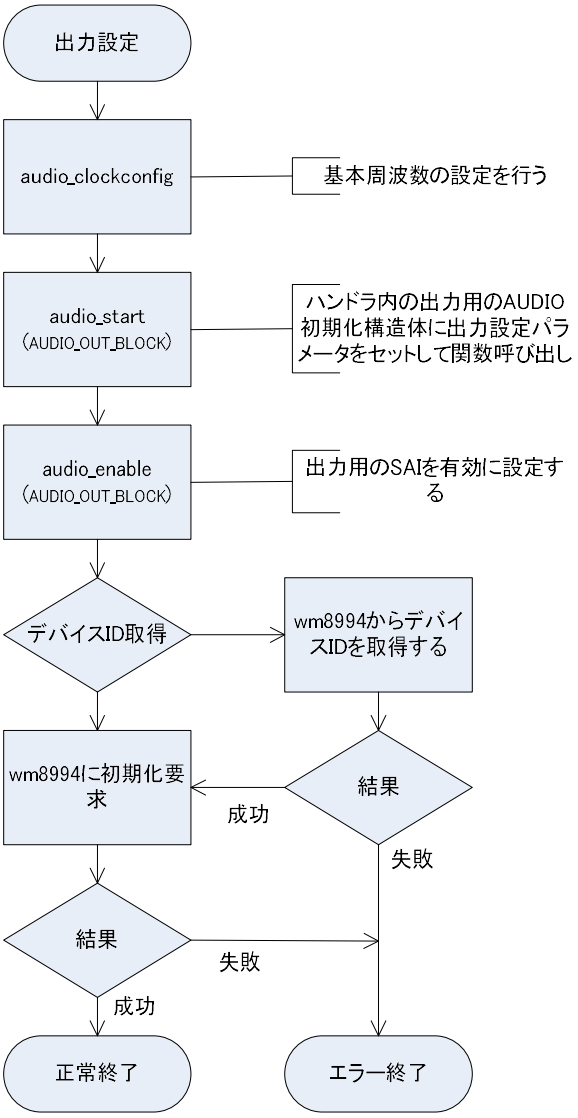


図 4.3.5.3.2 出力設定フローチャート

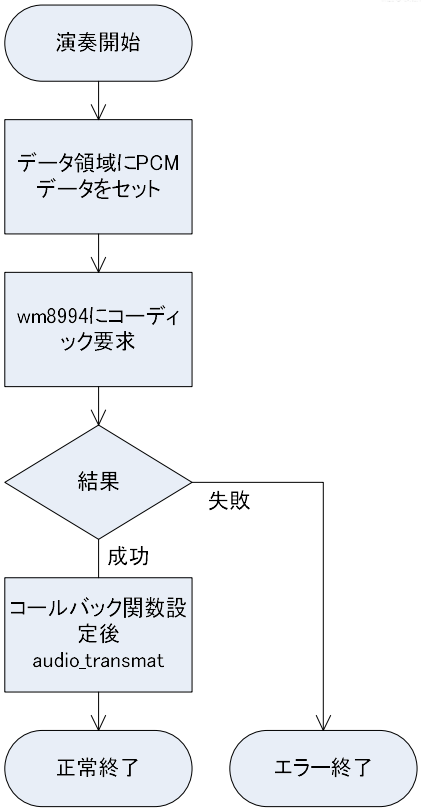


図 4.3.6.3.3 演奏開始フローチャート

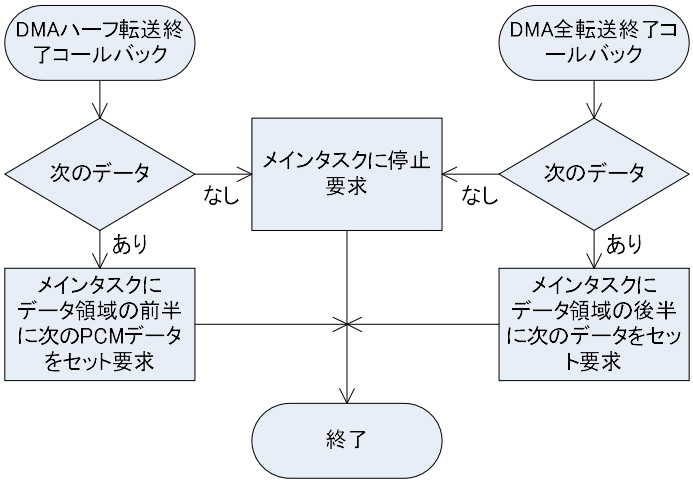


図 4.3.5.3.4 コールバック関数呼び出しのフローチャート

音楽演奏が終了した時点で、図 4.3.6.3.5 演奏終了フローチャートの手順でデバイスを停止する。

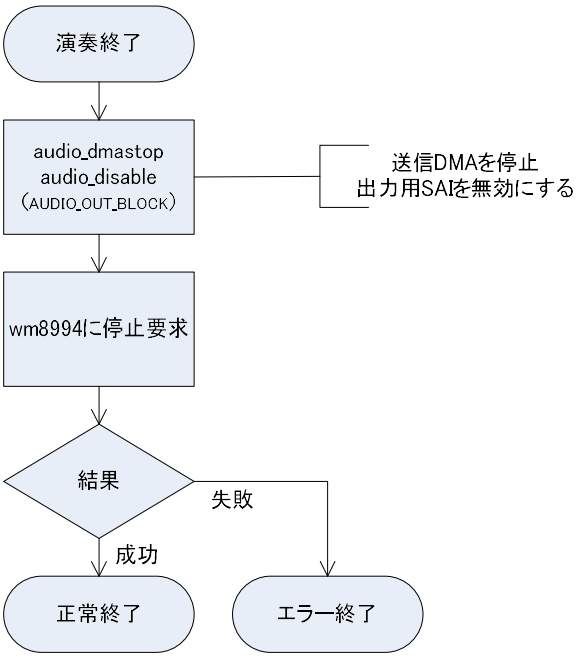


図 4.3.5.3.5 演奏終了フローチャート

4.3.6 DSI

DSI ドライバは、MIPI 仕様に準じた Display serial Interface をサポートする。LTDC で生成された LCD 画像を Display serial Interface へ変換を行う。

4.3.6.1 データ仕様

表 4.3.6.1.1 に DSI ドライバの初期化型 DSI_Init_t を示す。表 4.3.6.1.2 にハンドラ型 DSI_Handle_t 型を定義する。ハードウェア定義構造体は内部処理に使用するため構成は示さない。

番号	項目	型	機能
1	AutomaticClockLaneControl	uint32_t	Lane クロック自動制御
2	TXEscapeCkdiv	uint32_t	TX Escape のクロック分周値
3	NumberOfLanes	uint32_t	Lane 数
4	pllndiv	uint32_t	PLL loop 分周値
5	pllidf	uint32_t	PLL 入力分周値
6	pllodf	uint32_t	PLL 出力分周値

表 4.3.6.1.1 DSI_Init_t 型

番号	項目	型	機能
1	base	uint32_t	DSI デバイスのベースアドレス
2	Init	DSI_Init_t	DSI 初期化データ
3	teffectcallback	void(*)(DSI_Handle_t*)	TearEffct コールバック関数
4	refreshcallback	void(*)(DSI_Handle_t*)	リフレッシュコールバック関数
5	errorcallback	void(*)(DSI_Handle_t*)	エラーコールバック関数
6	status	volatile uint32_t	DSI ステータス
7	errorCode	volatile uint32_t	エラーコード

表 4.3.6.1.2 DSI_Handle_t 型

① AutomaticClockLaneControl
自動 Lane クロック制御モードを設定する。

定義	値	内容
DSI_AUTO_CLK_LANE_CTRL_DISABLE	0x0000000	無効

DSI_AUTO_CLK_LANE_CTRL_ENABLE	DSI_CLCR_ACR	有効

表 4.3.6.1.3 AutomaticClockLaneControl 設定値

DSI ハンドラ内のエラー定義(ErrorCode)を以下の表に示す。

定義	値	内容
DSI_ERROR_NONE	0x00000000	エラーなし
DSI_ERROR_ACK	0x00000001	ACK エラー
DSI_ERROR_PHY	0x00000002	PHY エラー
DSI_ERROR_TX	0x00000004	送信エラー
DSI_ERROR_RX	0x00000008	受信エラー
DSI_ERROR_ECC	0x00000010	ECC エラー
DSI_ERROR_CRC	0x00000020	CRC エラー
DSI_ERROR_PSE	0x00000040	パケットサイズエラー
DSI_ERROR_EOT	0x00000080	EOT エラー
DSI_ERROR_OVF	0x00000100	FIFO オーバーフローエラー
DSI_ERROR_GEN	0x00000200	ジェネリック FIFO エラー

表 4.3.6.1.4 エラーコード値

4.3.6.2 インターフェイス仕様

DSI ドライバの設計関数を表 4.3.7.2.1 に示す。

関数名	型	引数	機能	備考
dsi_init	ER	DSI_Handle_t *hdsi	DSI ハード初期化	
dsi_deinit	ER	DSI_Handle_t* hdsi	DSI 制御終了	
dsi_configvideo	ER	DSI_Handle_t* hdsi DSI_VideoConfig_t *pini	VIDEO 設定を行う	
dsi_configltdc	ER	LTDC_handle_t *hltdc DSI_VideoConfig_t *pini	DSI の設定を LTDC の初期設定に反映する	
dsi_configadaptedCmdMode	ER	DSI_Handle_t *hdsi DSI_CommandConfig_t *pcfg uint32_t ackactive	適合コマンドモード設定	
dsi_configcommand	ER	DSI_Handle_t *hdsi uint32_t lpcmode uint32_t ackactive	コマンド転送モード設定	
dsi_configPhyTimer	ER	DSI_Handle_t *hdsi DSI_PHY_Time_t *ptime	DSI-PHY のタイミング設定	
dsi_configHostTimeout	ER	DSI_Handle_t *hdsi DSI_HostTimeout_t *ptimeout	DSI-HOST タイムアウト設定	
dsi_start	ER	DSI_Handle_t *hdsi	DSI モジュール実行開始	
dsi_stop	ER	DSI_Handle_t *hdsi	DSI モジュール停止	
dsi_swrite	ER	DSI_Handle_t *hdsi uint32_t Channel uint32_t Mode uint32_t Param1 uint32_t Param2	ショートコマンド発行	
dsi_lwrite	ER	DSI_Handle_t *hdsi uint32_t Channel uint32_t Mode uint32_t NbParams uint32_t Param1 uint8_t *buf	ロングコマンド発行	
dsi_read	ER	DSI_Handle_t *hdsi	DSI コマンド受信	

		uint32_t Channel uint8_t *Array uint32_t Size uint32_t Mode uint32_t DCSmd uint8_t *buf		
dsi_enterULPMData	ER	DSI_Handle_t *hdsi	UPLM 設定(D-PHY PLL 実行)	
dsi_exitULPMData	ER	DSI_Handle_t *hdsi	UPLM 終了(D-PHY PLL 実行)	
dsi_enterULPM	ER	DSI_Handle_t *hdsi	UPLM 設定(D-PHY PLL オフ)	
dsi_exitULPM	ER	DSI_Handle_t *hdsi	UPLM 設定(D-PHY PLL オフ)	
dsi_startPatternGenerator	void	DSI_Handle_t *hdsi uint32_t Mode uint32_t Orientation	テストパターン生成開始	
dsi_stopPatternGenerator	void	DSI_Handle_t *hdsi	テストパターン生成終了	
dsi_setLanePinsConfiguration	ER	DSI_Handle_t *hdsi uint32_t CustmLane uint32_t Lane uint32_t active	カスタムレーン設定	
dsi_setPHYTiming	ER	DSI_Handle_t *hdsi uint32_t timing uint32_t active uint32_t Value	PHY タイミング設定	
dsi_forceTXStopMode	ER	DSI_Handle_t *hdsi uint32_t Lane uint32_t active	クロック、データレーン強制停止	
dsi_irqhandler	void	DSI_Handle_t *hdsi	DSI 割込みハンドラ	

表 4.3.6.2.1 DSI ドライバ関数

4.3.6.3 設定手順

LCD の初期化手順は以下のフローチャートに従う。

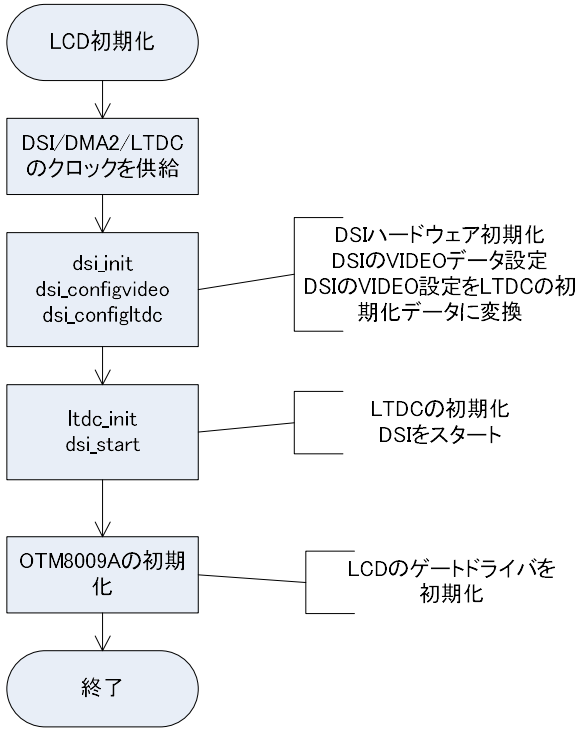


図 4.3.6.3.1 LCD 初期化

4.3.7 EMAC

EMAC は、有線インサーネット用ドライバである。EMAC ドライバはパケットのインターフェイスとして LWIP 用 PBUF を想定して、設計している。パケット・インターフェイスを修正すれば、他の TCP/IP プロトコル・スタックに対応可能である。

4.3.7.1 データ仕様

EMAC ドライバは初期化用の型として、表 4.3.7.1.1 の EMAC コンフィギュレーション構造体と、ハンドラとして表 4.3.7.1.2 の EMAC ハンドラ型を持つ。EMAC コンフィグレーション構造体は DESCRIPTOR キューと送受信バッファ等の設定を定義する。EMAC ドライバは、この設定に従って DESCRIPTOR を設定する。

番号	項目	型	機能
1	packetbufsize	uint32_t	パケットバッファ長
2	rxquecount	uint32_t	受信キューの数
3	txquecount	uint32_t	送信キューの数
4	quebuffer	void *	キューバッファ領域へのポインタ
5	databuffer	void *	データ領域へのポインタ
6	semid	ID	mdio 用セマフォ ID(LOCK 用)

表 4.3.7.1.1 EMAC コンフィギュレーション構造体

番号	項目	型	機能
1	base	uint32_t	EMAC モジュールベースアドレス
2	Init	EMAC_Init_t	EMAC 初期設定値保存領域
3	tx_wait	uint8_t	送信終了待ちフラグ
4	intno	uint8_t *	EMAC 割り込み番号
5	macid	uint8_t	PORTID 番号（保存用）
6	phy_addr	uint8_t	PHY アドレス番号
7	phy_features	uint32_t	PHY の機能ステータス
8	linkvalue	uint32_t	要求 LINK 値
9	tx_top_desc	volatile uint32_t *	送信キュー(DESCRIPTOR)の先頭アドレス
10	tx_cur_desc	volatile uint32_t *	現在の送信キューのアドレス
11	rx_top_desc	volatile uint32_t *	受信キュー(DESCRIPTOR)の先頭アドレス
12	rx_cur_desc	volatile uint32_t *	現在の受信キューのアドレス
13	emacevent	void (*) ()	EMAC 状態のコールバック関数
14	phy_opt_init	void (*) ()	PHY のオプション設定関数
15	ifShortPkts	uint32_t	ショートパケットの受信回数
16	mmc_tx_irq_n	uint32_t	MMC 割り込み MMCTIS の発生回数
17	mmc_rx_irq_n	uint32_t	MMC 割り込み MMCRIS の発生回数
18	mmc_rx_csum_offload_irq_n	uint32_t	MMC 割り込み MMCCSUM の発生回数
19	irq_receive_pmt_irq_n	uint32_t	PMT 割り込みの発生回数
20	irq_tx_path_in_lpi_mode_n	uint32_t	LPI 割り込み TLPEN の発生回数
21	irq_tx_path_exit_lpi_mode_n	uint32_t	LPI 割り込み TLPEX の発生回数
22	irq_rx_path_in_lpi_mode_n	uint32_t	LPI 割り込み RLPIEN の発生回数
23	irq_rx_path_exit_lpi_mode_n	uint32_t	LPI 割り込み RLPIEX の発生回数
24	normal_irq_n	uint32_t	TX/RX NORMAL 割り込みの発生回数
25	fatal_bus_error_irq	uint32_t	ABNORMAL/FBI の発生回数
26	tx_undeflow_irq	uint32_t	ABNORMAL/UNF の発生回数
27	tx_jabber_irq	uint32_t	ABNORMAL/TJT の発生回数
28	tx_early_irq	uint32_t	ABNORMAL/ETI の発生回数
29	tx_process_stopped_irq	uint32_t	ABNORMAL/TPS の発生回数

30	rx_normal_irq_n	uint32_t	TX/RX NORMAL/RIE の発生回数
31	rx_early_irq	uint32_t	TX/RX NORMAL/ERI の発生回数
32	rx_overflow_irq	uint32_t	ABNORMAL/OVF の発生回数
33	rx_buf_unav_irq	uint32_t	ABNORMAL/RU の発生回数
34	rx_process_stopped_irq	uint32_t	ABNORMAL/RPS の発生回数
35	rx_watchdog_irq	uint32_t	ABNORMAL/RWT の発生回数

表 4.3.7.1.2 EMAC ハンドラ型

① packetbufsize
パケットバッファサイズを指定する。キャッシュアラインサイズ出なければならない。

② rxquecount
受信用のキュー(DESCRIPTOR)の数を指定する。

③ txquesize
送信用のキュー(DESCRIPTOR)の数を指定する。

④ quebuffer
EMAC で使用するキュー(DESCRIPTOR)領域のポインタを指定する。この領域はキャッシュアライン・アドレスでなければならない。quebuffer のバイトサイズは以下の計算による。
que_size = (rxquecount + txreqcount) × 4

⑤ databuffer
EMAC で使用する通信用のバッファ領域のポインタを指定する。この領域はキャッシュアライン・アドレスでなければならない。databuffer のバイトサイズは以下の計算による。
data_size = (rxquecount + txreqcount) × packetbufsize

⑥ semid
PHY との通信時、排他制御を行うためのセマフォ ID。ゼロなら排他制御を行わない。

4.3.7.2 インターフェイス仕様

EMAC を用いてイーサネット通信を行うためのドライバ関数は以下の通りである。

関数名	型	引数	機能	備考
emac_init	EMAC_Hand_t*	ID portid EMAC_Init_t *ini	指定ポート ID の EMAC ペリフェラルを初期化し、ハンドラへのポインタを返す	
emac_deinit	ER	EMAC_Handle_t* hmac	EMAC を未使用状態に戻す	
emac_send	ER	EMAC_Handle_t* hmac struct emac_pkt *pktp	パケットのデータを送信する	
emac_recv_length	int	EMAC_Handle_t* hmac	受信パケット長を取り出す	
emac_recv	ER	EMAC_Handle_t*hmac struct emac_pkt *pktp	受信パケットを取り出す	
emac_start	ER	EMAC_Handle_t* hmac uint32_t link	EMAC と PHY をスタートする	
emac_stop	ER	EMAC_Handle_t* hmac int disable	EMAC を停止する	
emac_link_detect	uint16_t	EMAC_Handle_t* hmac	LINK 情報を取り出す	
emac_link_mode_set	ER	EMAC_Handle_t* hmac uint32_t link	LINK 情報を EMAC に設定する	
emac_reset	ER	EMAC_Handle_t* hmac	EMAC をリセットする	

		uint8_t *mac_addr		
emac_hps_isr	void	intptr_t exinf	EMAC 割込みサービスルーチン	

表 4.3.7.2.1 EMAC ドライバ関数

4.3.7.3 フローチャート

EMAC ドライバは、TCP/IP プロトコル・スタックから呼び出されることを前提に設計している。TCP/IP プロトコル・スタックとの依存性が高い部分は、送受信用のパケットの受け渡し処理である。このドライバでは、LWIP 用の pbuf を使用してパケットを受け渡すことを前提に設計している。

EMAC の初期化フローを図 4.2.7.3.1 に示す。ここで特に注意が必要なのは、EMAC モジュールはリセット後、割り込みマスクが許可に以降するため、必ず、EMAC の割り込みコントローラは割り込み禁止の状態でシステム起動しなければならない。emac_init にて EMAC の初期化を行う。このとき、EMAC_Init_t 構造体にて、キューや送受信バッファ領域の設定を行う。ST-ETH では、hardware_init_hook 関数の MPU 設定で EMAC のデータ領域をノンキャッシュメモリに変更するためキャッシュ制御は不要となる。初期化後、emac_reset 関数にて EMAC のリセットを行う。この関数は通信中に EMAC のハードエラー等が発生した場合、EMAC のリセットに使用かのうである。リセット後、emac_start 関数にて LINK 情報を設定して、PHY のスタート、EMAC のスタートを設定する。emac_start を実行しても、LINK していない場合、通信動作が開始できないばかりでなく、PHY の設定が AUTO NEGOTIATION の場合、EMAC の通信設定も確定しない。emac_link_detect 関数にて接続情報を取得できる。AUTO NEGOTIATION の場合、接続情報を EMAC に反映するために、emac_link_mode_set 関数を使用する。

送信処理のフローを図 4.3.7.3.2 に示す。pbuf に送信データをつめて、emac_send 関数を呼び出せばよい。

受信処理のフローを図 4.3.7.3.3 に示す。受信データの有無は、emac_recv_length 関数で受信データサイズを取り出せる。受信データがある場合は、サイズ分の pbuf を用意して emac_recv 関数にて、受信データの取得を行う。受信の有無はコールバック関数のイベントでも通知されるので、常に emac_recv_length 関数でポーリングを行う必要はない。

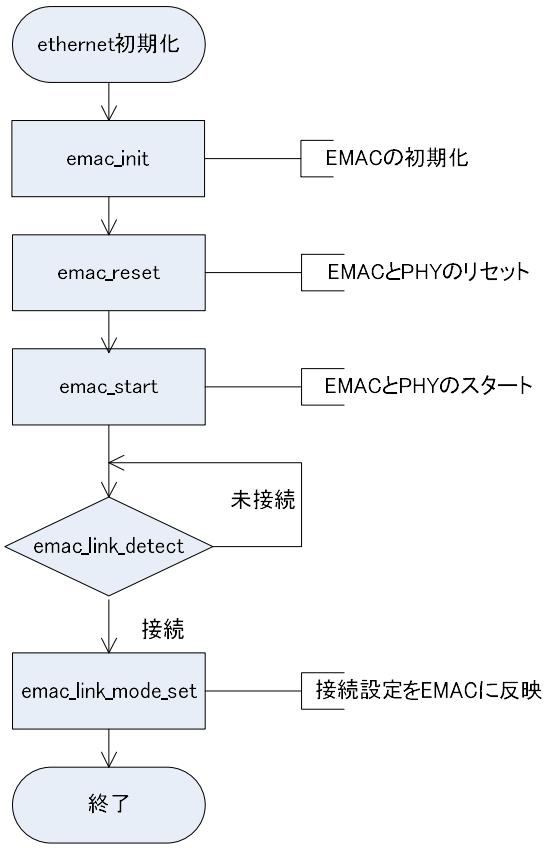


図 4.3.7.3.1 QSPI 初期化

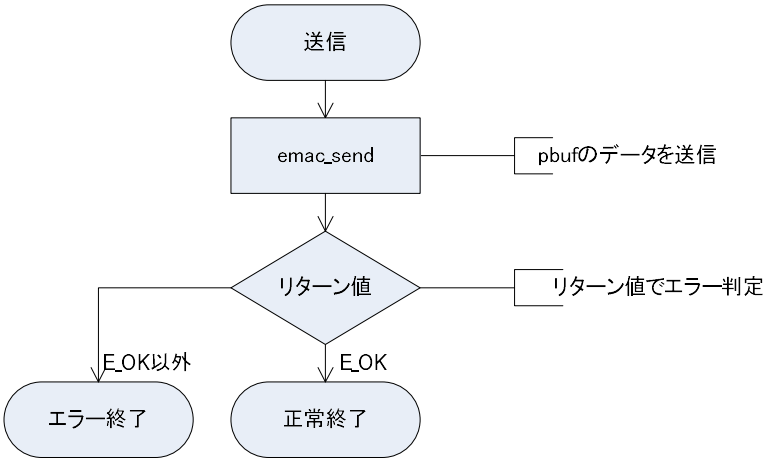


図 4.3.7.3.2 EAC 送信

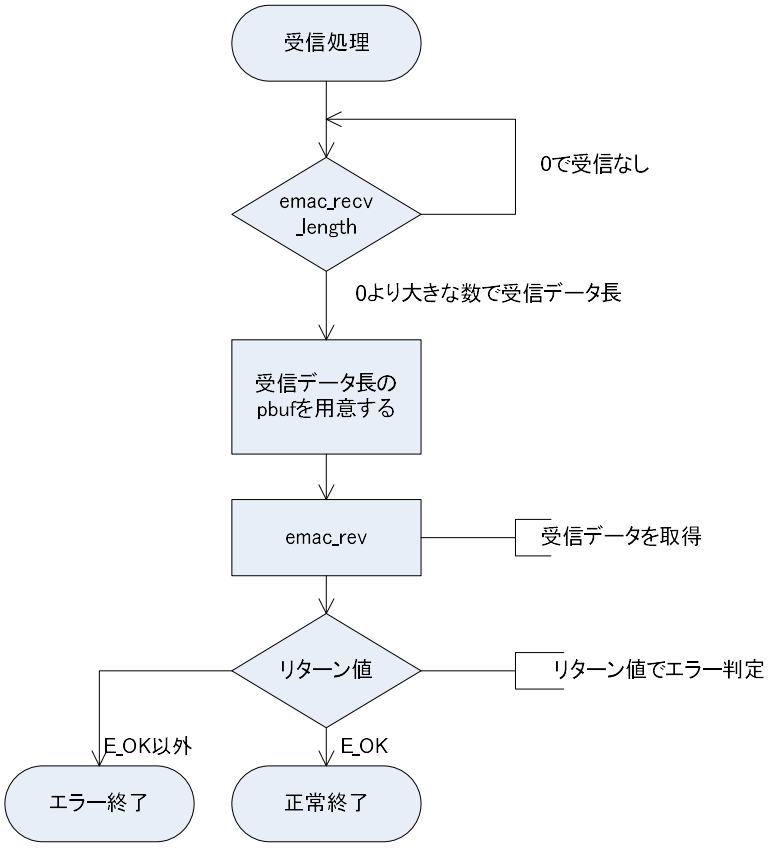


図 4.3.7.3.3 EMAC 受信

図 4.3.7.3.4 に EMAC の終了処理を示す。

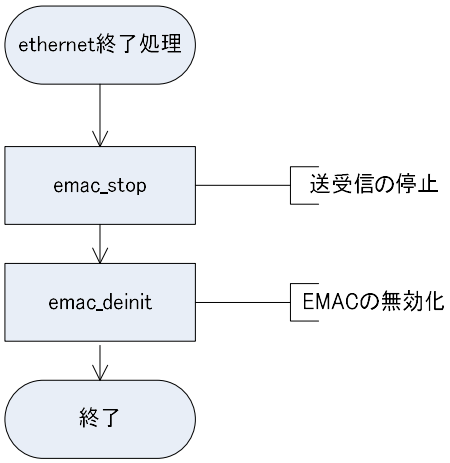


図 4.3.7.3.4 EMAC 終了処理

5 タスクモニタ

本章では、標準入出力機能付きのタスクモニタの仕様に関して記載する。

5.1 概要

タスクモニタはタスク上で動作し、デバッグ用のコマンドを用いてプラットフォーム部の機能確認やテストを行う。デバックコマンドは設定によりコマンド追加が可能である。これによりアプリケーションでも、アプリケーション用デバッグコマンドを追加することができる。タスクモニタの入出力は標準入出力に対して行う、デフォルトの標準入出力は ASP カーネルにて実装されているシリアルデバイスであるが、入出力の切り替えにより、telnet の端末等に切り替えが可能である。

5.2 標準入出力

タスクモニタの入出力は標準入出力に対して行う。標準入出力は FILE 型を定義し、入力、出力、エラーの 3 つの FILE へのポインタを以下の名称で定義することで実現する。

⑤ stdin
⑥ stdout
⑦ stderr

FILE 型は表 5.2.1 の構成となる。FILE 型は fread や fwrite でファイルにアクセスする場合のハンドラとして使用される。

番号	項目	型	機能
1	_flags	int	ファイル用フラグ
2	_file	int	ファイル番号
3	_func_in	int	1byte 入力コールバック関数
4	_func_ins	int	n bytes 入力コールバック関数
5	_func_out	void	1byte 出力コールバック関数
6	_func_outs	int	n bytes 出力コールバック関数
7	_func_flush	int	データフラッシュコールバック関数
8	_dev	void *	デバイス構造体へのポインタ

表 5.2.1 FILE 型

標準入出力では、以下の関数をサポートする。

関数名	型	引数	機能	備考
fgetc	int	FILE *fp	ファイルから 1byte 読み込み	
fgets	int	char *c FILE *fp	ファイルから文字列読み込み	
fputc	int	int c	ファイルに 1byte 書き込み	

		FILE *fp		
fputs	int	const char *str FILE *	ファイルに文字列書き込み	
putchar	int	int c	1byte 書き込み	
puts	int	const char *str	文字列を標準出力に書き込み	
printf	int	const char const …	標準出力へのプリント	結果は項目数
sprintf	int	char *c const char const …	バッファへプリント	結果は項目数
scanf	int	const char const …	標準入力からスキャン	結果は項目数
sscanf	int	char *c const char const …	スキャンしバッファにセット	結果は項目数
fflush	int	FILE *fp	ファイルのフラッシュ	
fread	size_t	void *buf size_t len size_t num FILE *fp	ファイルからデータ読み込み	結果は num 数
fwrite	size_t	const void *buf size_t len size_t num FILE *fp	ファイルへデータ書き込み	結果は num 数
fprintf	int	FILE *fp const char *const …	ファイルへプリント	結果は項目数
putc	int	int c FILE *fp	fputc と同様	
getchar	int		標準入力から 1byte 読み込み	
getc	int	FILE *fp	fgetc と同様	

表 5.2.2 サポートしている標準入出力関数

5.3 標準デバッグコマンド

タスクモニタは、標準のデバッグコマンドとして以下のコマンドをサポートする。タスクモニタのデバッグコマンドは第１（カテゴリ）、第２の２つのコマンドで機能を指定する形をとる。また、コマンドを設定する場合、最初の１文字以降を省略可能である。省略名で同一のコマンドがある場合、はじめにディスパッチするコマンドが選択される。

第１コマンド	第２コマンド	引数	機能
DISPLAY	BYTE	start address[hex]	バイト単位でメモリ DUMP する
	HALF	start address[hex]	２バイト単位でメモリ DUMP する
	WORD	start address[hex]	４バイト単位でメモリ DUMP する
	TASK	-	タスクの状態を表示する
	REGISTER	-	CPU レジスタの内容を表示する
SET	BYTE	set address[hex]	バイト単位で、メモリ内容を変更する
	HALF	set address[hex]	２バイト単位で、メモリ内容を変更する
	WORD	set address[hex]	４バイト単位で、メモリ内容を変更する
	COMMAND	mode[1 or 2]	デフォルト 2、1 の場合最初の 1 文字のみ比較
	SERIAL	portno	標準入出力のシリアルポート番号を変更
	TASK	taskid	TASK コマンドの対象タスクを指定する
TASK	ACTIVATE	-	タスクの起動要求(act_tsk)
	TERMINATE	-	タスクを終了する(ter_tsk)
	SUSPEND	-	タスクの待ち要求(sus_tsk)
	RESUME	-	タスクの待ち再開(rsm_tsk)

	RELEASE	-	タスクの待ち解除(rel_wai)
	WAKEUP	-	タスクの起床(wup_tsk)
	PRIORITY	priority	タスクの優先度を変更する
LOG	MODE	[logmask][lowmask]	syslog の表示モードを変更する
	TASK	[time]	タスクの実行状態表示(指定が必要)
	PORT	[no][logno][portaddress]	ポートアクセスログ
HELP	Arg1		コマンドヘルプ

表 5.3.1 標準デバッグコマンド

ファイルライブラリが追加された場合、以下のコマンドを追加でサポートする。

第１コマンド	第２コマンド	引数	機能
VOLUME	FORMAT	drive	ドライブのフォーマット(未サポート)
	DIR	path	ディレクトリの表示
	MKDIR	path	ディレクトリの作成
	RMDIR	path	ディレクトリの消去
	ERASE	path	ファイルの消去

表 5.3.2 ファイルデバッグコマンド

RTC ドライバをサポートした場合、以下のコマンドを追加でサポートする。

第１コマンド	第２コマンド	引数	機能
RTC	DATE	year month day	日にちを設定する
	TIME	hour min sec	時間を設定する
	CLOCK	-	現在の日にちと時間を表示する

表 5.3.3 RTC デバッグコマンド

5.4 デバッグコマンド拡張

タスクモニタは、コマンドを拡張する機能を持つ。コマンドの拡張は第１（カテゴリ）コマンド単位で追加される。

5.4.1 データ仕様

コマンド追加には２つの型を使用する。COMMAND_INFO 型は第２コマンドの設定を行い、COMMAND_LINK 型は、複数の COMMAND_INFO 型をまとめて登録カテゴリを指定する。COMMAND_LINK 型の pcnext はデバッグコマンドのリンクに使用する、そのため、COMMAND_LINK は値付きの変数で作成しなければならない。

番号	項目	型	機能
1	command	const char *	第２コマンド名
2	func	int_t (*)()	第２コマンド関数へのポインタ

表 5.4.1.1 COMMAND_INFO 型

番号	項目	型	機能
1	pcnext	COMMAND_LINK *	COMMAND_LINK のチェーン用
2	num_command	int	第２コマンドの数
3	command	const char *	第１（カテゴリ）コマンド名
4	func	int_t (*)()	カテゴリコマンドの実行関数（通常は NULL）
5	help	const char *	カテゴリの HELP メッセージ
6	pcinfo	COMMAND_INFO *	COMMAND_INFO の配列へのポインタ

表 5.4.1.2 COMMAND_LINK 型

5.4.2 インターフェイス仕様

デバッグコマンドの追加は、COMMAND_LINK のインスタンスへのポインタを引数に以下の関数コールにて追加される。

関数名	型	引数	機能	備考

setup_command	int	COMMAND_LINK *	コマンドカテゴリを追加する	

表 5.4.2.1 デバッグコマンド追加定関数

6 API 層

API 層はアプリケーションに対して標準的なインターフェイスを提供する層である。この層はハードウェアや RTOS の仕様に影響されず、一意のインターフェイスを提供することにより、アプリケーションを汎用的に作成することができる。また、C 言語の規約や POSIX のように汎用的な API に準拠すれば、LINUX 等のオープンソースのライブラリを未修整で使用することができる。

6.1 概要

ストレージ機能を提供するファイルシステムと時刻の管理を行う時間管理の 2 つの API について説明を行う。

6.2 ファイルシステム

ファイルシステムは TOPPERS BASE PLATFORM で提供するストレージ機能である。ファイルシステムは以下の 3 つのモジュールで構成される。

① ファイルライブラリ
　C 言語標準のファイル関数をサポートするライブラリ
② ストレージデバイスマネージャー
　ストレージデバイスの管理モジュール
③ FATFs
　赤松武史氏が開発し、フリーソフトウェアとして公開されている、FAT 仕様準拠のローカルファイルシステム

TOPPERS BASE PLATFORM アプリケーション、ハードウェアを除く部分を供給している。

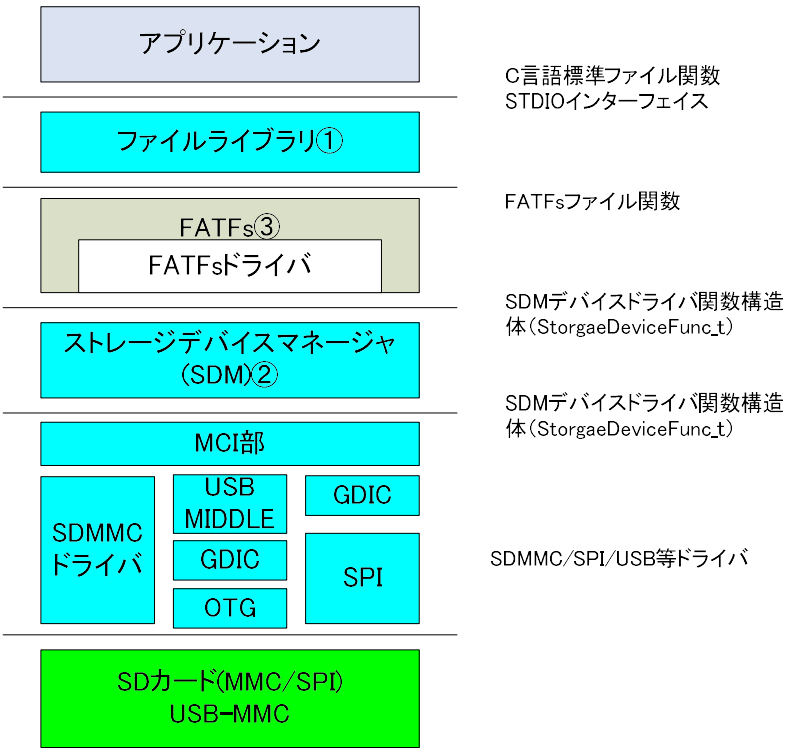


図 6.2.1 ファイルシステムのレイア構造

6.2.1 ファイルライブラリ

ファイルライブラリは標準入出力関数に合わせて、表 6.2.1.1 関数を提供する。これらはC言語上のファイル関数とファイル関係の POSIX 仕様であるが、以下の規定に従って関数の選択を行った。

・C言語標準のもの（C89,C99）
・POSIX.1-2001 でよく使われるもの
・LINUX 固有であるが、通常使用に必要なもの

関数名	型	引数	機能	備考
open	int	const char *pathname int flags	ファイルのオープン、ファイルディスクリプタを返す	エラー時-1 を返す
close	int	int fd	ファイルのクローズ	成功０、失敗-1
fstat	int	int fd struct stat *buf	ファイルの状態を取り出す。読み出しサイズを返す	-1 でエラー
lseek	off_t	int fd off_t offset int whence	ファイルの読み書き位置を設定する。	-1 でエラー
read	long	int fd void *buf long count	ファイルからデータを読み出す。読み出しサイズを返す。	0 で終端、-1 でエラー
write	long	int fd const void *buf long count	ファイルにデータを書き込む。書き込みサイズを返す。	-1 でエラー
mmap	void *	void *start size_t length int prot int flags int fd off_t offset	ファイルをメモリ上に配置する	一部ファイルシステムのみサポート
mumap	int	void *start size_t length	mmap の解除	
fopen	FILE*	const char *name const char *attr	ファイルオープン	
fclose	int	FILE *fp	ファイルクローズ	
fseek	int	FILE *fp long offset int whence	ファイルの読み書き位置を設定する	
stat	int	const char *name struct stat *buf	ファイルの状態の取出し	成功０、失敗-1
lstat	int	const char *name struct stat *buf	ファイルの状態の取出し	成功０、失敗-1
access	int	const char *name int mode	ファイルの状態の取出し	成功時 0、その他は-1
mkdir	int	const char *name mode_t mode	ディレクトリの作成	成功時 0、その他は-1
rmdir	int	const char *name	ディレクトリの削除	成功時 0、その他は-1
chmod	int	const char *name mode_t mode	ファイルモードの変更	成功０、失敗-1
remove	int	const char *name	ファイル削除	成功０、失敗-1
unlink	int	const char *name	ファイル削除	
rename	int	const char *oldpath const char *newpath	ファイル名の変更	成功０、失敗-1
opendir	void *	const char *path	path に対応したディレクトリストリームをオープンし、ストリームのポインタを返す。	エラー時 NULL

closedir	int	void *dir	ディレクトリストリームのクローズ	成功０、失敗-1
readdir	struct dirent*	void *dir	順番にディレクトリから dirent 構造体を読み、dirp で指されたバッファに格納する。	POSIX とは I/F 仕様が異なる
statfs	int	const char *path struct statfs2*stat	マウントされたファイルシステムについての情報を返す。	成功０、失敗-1

表 6.2.1.1 ファイルライブラリ関数

6.2.2 Storage Device Manager

Storage Device Manager は、ストレージデバイスとその下で実行される複数のローカルファイルシステムを管理するモジュールである。

6.2.2.1 データ仕様

Storage Device Manager は以下の４つの型で構成される。最初の２つは関数テーブルであり、表 6.2.2.1.1 の StorageDeviceFunc_t はローカルファイルシステムのデバイスを複数のストレージに対応させるために、関数テーブル化に用いるデバイスファンクションテーブルの型である。実際使用しているローカルファイルシステムが FATFs であるため、FATFs のデバイス I/F の関数テーブルとなっている。表 6.2.2.1.2 の StorageDeviceFileFunc_t 型は、ファイルライブラリを作成するために、ローカルファイルシステムのファイル関数をテーブル化するための型である。

番号	項目	型	機能
1	_sdev_sens	int (*)()	デバイスセンス関数
2	_sdev_diskinit	int (*)()	デバイスの初期化関数
3	_sdev_diskstatus	int(*)()	デバイスの状態取出し関数
4	_sdev_diskread	int(*)()	デバイスブロックリード関数
5	_sdev_diskwrite	int(*)()	デバイスブロックライト関数
6	_sdev_diskioctl	int(*)()	デバイス IOCTL 関数

表 6.2.2.1.1 StroageDeviceFunc_t 型

番号	項目	型	機能
1	_sdevff_opendir	void *(*)()	opendir 関数
2	_sdevff_closedir	int (*)()	closedir 関数
3	_sdevff_readdir	int (*)()	readdir 関数
4	_sdevff_mkdir	int (*)()	mkdir 関数
5	_sdevff_rmdir	int (*)()	rmdir 関数
6	_sdevff_unlink	int (*)()	unlink 関数
7	_sdevff_rename	int (*)()	rename 関数
8	_sdevff_chmod	int (*)()	chmod 関数
9	_sdevff_stat	int (*)()	stat 関数
10	_sdevff_statfs	int (*)()	statfs 関数
11	_sdevff_open	int (*)()	open 関数
12	_sdevff_close	int (*)()	close 関数
13	_sdevff_fstat	int (*)()	fstat 関数
14	_sdevff_lseek	off_t (*)()	lseek 関数
15	_sdevff_read	long (*)()	read 関数
16	_sdevff_write	long (*)()	write 関数
17	_sdevff_mmap	void *(*)()	mmap 関数

表 6.2.2.1.2 StorageDeviceFunc_t 型

あとの２つはストレージを管理する型で、表 6.2.2.1.3 StorageDevice_t 型はストレージデバイス自体

の情報管理用の型で、表 6.2.2.1.4 StorageDeviceHead_t 型は StorageDevice_t 型のインスタンスを管理するヘッダ部である。表 6.2.2.1.5 は StorageDevice_t 型の_sdev_attribute のビット指定値を示す。SDEV_INSERTCHK がオンになっていないデバイスでは、メディアの挿抜チェックを行わない。この場合、初期化時のメディアの状態でメディアの有無を決定する。

番号	項目	型	機能
1	_sdev_attribute	uint16_t	デバイスの状態
2	_sdev_devno	uint8_t	デバイス番号
3	_sdev_port	uint8_t	デバイス種別
4	_sdev_maxsec	uint32_t	デバイスの最大セクタ数
5	_sdev_secsize	uint32_t	デバイスのセクタサイズ
6	_sdev_instimer	uint16_t	挿入タイマー
7	_sdev_inswait	uint16_t	挿入時待ち時間
8	_sdev_notice	void (*)()	検知コールバック関数
9	_sdev_local[4]	void *	ローカルエリア
10	pdevf	StorageDeviceFunc_t	デバイス関数テーブル
11	pdevff	StorageDeviceFileFunc_t*	ファイル関数テーブル

表 6.2.2.1.3 StroageDevice_t 型

番号	項目	型	機能
1	_num_activedev	uint16_t	登録済ストレージデバイスの数
2	_sdev_active	uint8_t	デバイスマネージャの有効無効
3	_default_device	uint8_t	デフォルトデバイス番号
4	_get_datetime	uint32_t (*)()	時刻取出し関数
5	_psd	StorageDevice_t *	ストレージデバイス配列

表 6.2.2.1.4 StroageDeviceHead_t 型

定義	値	内容
SDEV_ACTIVE	(1<<15)	デバイスがアクティブの状態
SDEV_INSERTCHK	(1<<14)	挿入・排出の設定あり
SDEV_CHKREMOVE	(1<<13)	排出検査を行う
SDEV_ONEEXIT	(1<<12)	一度以上排出があった
SDEV_EMPLOY	(1<<8)	デバイス動作中
SDEV_ERROR	(1<<7)	デバイスエラー
SDEV_DEVNOTUSE	(1<<0)	デバイス使用不可
SDEV_NOTUSE	255	使用不可のビット定義

表 6.2.2.1.5 _sdev_attribute 設定値

6.2.2.2 インターフェイス仕様

表 6.2.2.2.1 はストレージデバイスマネージャーで使用する関数である。sdev_init 関数にてストレージデバイスマネージャーは有効となり、sdev_terminate 関数で無効となる。この間で、ストレージデバイスの管理を行う。まず、SDMSetupDevice 関数でストレージデバイスの登録を行う。デバイス番号を指定して、この関数を呼び出すと、ppsdev にストレージデバイス(StoregeDevice_t)がセットされて戻る。使用者は、戻されたストレージデバイスに以下の属性等をセットする。

① デバイス関数テーブル：デバイスドライバテーブル
② ファイル管理テーブル：ローカルファイルシステムに対応したファイル関数テーブル
③ 属性：挿抜処理の有無
④ ローカルデータ：下位のデバイス等の情報を_sdev_local に設定する

ストレージデバイスに挿抜検知は SDMSense_task で行うが、デバイスに挿抜検知がない場合は、タスクレベルで検知を行う必要はない。この場合、SDEV_SENSE_ONETIME をコンパイルスイッチで定義してビルドすれば関数として設定され、デバイスの登録後、SDMSense_task(0)を関数コールすれば、メ

ディアの有り無し処理を行う。SDMSense_task では、センス関数として psdev->pdevf->_sdevf_sense にてメディアのセンスを行うため、この関数を設定しておく必要がある。

SDMSense_task を使用しない場合は、SDMEmploy 関数を使って直接メディアの有無を設定することができる。

関数名	型	引数	機能	備考
sdev_init	void	intptr_t exinf	ストレージデバイスマネージャーを初期化する	
sdev_terminate	void		ストレージデバイスマネージャーを終了する	
SDMSetupDevice	ER	int16_t devno StorageDevice_t **ppsdev	指定のデバイスを追加する	
SDMDeviceNo	ER_ID	const char **ppathname	パス名からデバイス番号を取り出す。パス名のデバイス名はスキップする	
SDMGetStorageDevice	Storage Device_t *	int devno	デバイス番号からストレージデバイスへのポインタを取り出す	
SDMEmploy	ER	StorageDevice_t *psdev bool_t sw	デバイスの動作中(true)、停止中(false)を設定する	
SDMSence_task	void	intptr_t exinf	デバイスセンスタスク（関数の場合あり）	

表 6.2.2.2.1 ストレージデバイスマネージャー関数

定義	内容
SDEV_SENSE_ONETIME	SDMSense_task を関数として使用
NUM_STORAGEDEVICE	ストレージデバイスの数(デフォルトは 4)
DEAFULT_DEVNO	デフォルトのデバイス番号(通常は 0)

表 6.2.2.2.2 コンパイルスイッチ設定値

6.2.3 FatFs

PLATFORM V1.X では、FAT12,16,32 用のローカルファイルシステムとして FatFs R0.07a を RTOS 対応のため一部修正して使用している。また、FatFs のデバイスドライバ(diskio.c)に関しては、複数のストレージに対応可能なようにデバイスファンクションテーブルを呼び出す形で改造を行っている。

6.3 時間管理

RTC デバイスが有効な場合、時間管理関数が有効となる。時間管理では、２つの型を用いて時刻の管理を行う。

6.3.1 データ仕様

表 6.3.1.1 の tm(tm2)構造体は時刻の管理を行う構造体である。tm 構造体はライブラリの time.h 中に定義されている時刻用構造体と同一のものである。tm2 は tm と同一の構造体で、standard device でローカルに定義しているものである。RTC デバイスの時刻処理は tm2 構造体を使用する。もうひとつが types.h 等で定義されている time_t 型で 1970 年 1 月 1 日からの経過時間を秒で表す。

番号	項目	型	機能
1	tm_sec	int	秒(0～59)
2	tm_min	int	分(0～59)
3	tm_hour	int	時(0～23)
4	tm_mday	int	月中の日(1～31)
5	tm_mon	int	月(1～12)

6	tm_year	int	年：西暦、但し 0 は 1970 年
7	tm_wday	int	週中の日(0～6)
8	tm_yday	int	年中の日(1～366)
9	tm_isdst	int	

表 6.2.1.1 tm 構造体

6.3.2 インターフェイス仕様

tm 構造体と time_t 型との変換する関数として表 6.3.2.1 に関数を用意する。

関数名	型	引数	機能	備考
mktime	time_t	struct tm *ptm	tm 構造体を time_t に変換する	
gmtime_r	struct tm *	const time_t *pt struct tm *ptm	time_t を tm 構造体に変換する	

表 6.3.2.1 時刻管理関数

6.4 USB ミドルウェア

USB ミドルウェアとして、TOPPERS USB MIDDLEWARE を対応する。これは USB OTG デバイス用の USB ホスト、デバイス機能、USB デバイスに対して USB デバイス機能を提供する。

6.4.1 USB ホスト機能

USB ホスト機能は、マルチデバイスを対応して HUB クラスに対して複数のクラスを同時管理可能である USB ホストは以下の５つのクラスをサポートする。

クラス名	ID	機能	アプリとの通信機能	備考
HUB	0x09	HUB のサポート		
HID	0x03	MOUSE/KEYBOAED	コールバック関数設定	
MSC	0x08	USB メモリ	API による関数コール	
PRT	0x07	プリンタ	API による関数コール/コールバック関数	
CDC/SERIAL	0x02	USB シリアル	API による関数コール/コールバック関数	
BLUETOOTH	0xE0	BLUETOOTH	API による関数コール/コールバック関数	1.3.1

表 6.4.1.1 USB ホストクラス

6.4.2 USB デバイス機能

USB デバイス機能は、指定のクラスに対してのデバイス機能を提供する。

クラス名	ID	機能	備考
HID	0x03	MOUSE/KEYBOAED	
SERIAL	0x02	USB シリアル	
MSC	0x08	USB メモリ	1.4.5

表 6.4.1.1 USB デバイスクラス

6.5 UI ミドルウェア

UI ミドルウェアとして、グラフィック LCD に対して、図形描画、文字描画を行うインターフェイスを追加する。図形描画の API は GDIC/adafuit_st7735 の描画関数を使用する。拡張として STM32Cube の BSP で使用している描画関数の rename を用意し、BSP 用のアプリケーションがそのまま使用できるよう考慮する。文字描画は、STM32Cube で用意されたフォントと TOPPERS ECHONET WG で作成した東雲フォントの２つの文字環境に対応できるように設計した。

6.5.1 UI ディレクトリ

UI ディレクトリ以下に２つのフォント環境を用意する。UI 機能を取り込むには Makefile のミドルウェアの設定に以下のインクルードを行えばよい。

include $(SRCDIR)/ui/$(UI ディレクトリ)/Makefile.config

フォント	１バイトフォント高さ	漢字フォント高さ	備考
STM32Cube フォント	8,12,16,20,24	なし	
東雲フォント	9,12,16	12,16	漢字フォントはファイルで提供

表 6.5.1.1 UI ディレクトリ

東雲フォントの漢字ビットマップデータは以下のヘッダを持つ、ファイルといて参照する。TOPPERS BASE PLATFORM(ST)で使用するには、QSPI フラッシュ ROM 上に flash_file の形でデータを置くことを推奨する。東雲フォント漢字ビットマップファイルの flash_file 形式の書き込みデータを ui/snfone_disp/shinonome_font.bin で提供する。このデータを SD カードまたは USB メモリにコピーし、QSPI ファイル書き込みコマンドを使って QSPI フラッシュに書き込み flash_file でファイルアクセスする。

なお、QSPI ファイル書き込み、漢字表示のサンプルプログラムを以下に用意している。

OBJ/STM32F723DISCOVERY_GCC/MON3

漢字フォントをプログラムのデータ領域に置きたい場合は、USE_ROMFONT コンパイルスイッチを有効にすれば、romfont というグローバル変数として定義される。ROM ファイル機能を使って ROM ファイルに変換すればファイルとしてアクセスができる。

以下に漢字フォントビットマップファイルのヘッダを示す。数値はリトルエンディアンとなる。

番号	項目		型	機能
1	magic	0	char [4]	“FONT”
2	name	4	char [32]	フォント名
3	fsize	36	unsigned int	ファイルサイズ
4	csize	40	unsigned int	ビットマップデータのコード部のバイト数
5	isize	44	unsigned int	ビットマップデータのイメージ部のバイト数
6	off_cnv_table1	48	unsigned int	UTF-8N 2 バイトコード検索テーブルへのオフセット
7	siz_cnv_table1	52	unsigned int	UTF-8N 2 バイトコード検索テーブルへのコード数
8	off_cnv_table2	56	unsigned int	UTF-8N 3-1 バイトコード検索テーブルへのオフセット
9	siz_cnv_table2	60	unsigned int	UTF-8N 3-1 バイトコード検索テーブルへのコード数
10	off_cnv_table3	64	unsigned int	UTF-8N 3-2 バイトコード検索テーブルへのオフセット
11	siz_cnv_table3	68	unsigned int	UTF-8N 3-2 バイトコード検索テーブルへのコード数
12	off_2byte_font	72	unsigned int	UTF-8N 2 バイトコード＋イメージへのオフセット
13	siz_2byte_font	76	unsigned int	UTF-8N 2 バイトコード＋イメージのアイテム数
14	off_3byte_font	80	unsigned int	UTF-8N 3 バイトコード＋イメージへのオフセット
15	siz_3byte_font	84	unsigned int	UTF-8N 3 バイトコード＋イメージのアイテム数

表 6.5.1.2 東雲フォントファイルのヘッダ部構造体

6.6 LWIP ミドルウェア

TCP/IP スタック lwip の拡張手順について記載する。現状 lwip は以下のバージョンに対応している。RTOS 上で実行するには問題があるソースに関しては PATCH を用意している。

・lwip-2.1.3

3 つのパッチ

・contrib-2.1.0

3 つのパッチ

ASP カーネル用のドライバを contrib-2.1.0/ports/toppers に追加している。ETHERNET ドライバは lwip の ETHERNET を pdic の emac ドライバに接続を行う。

6.6.1 ポーティング方法

lwip のプログラム設定手順を以下に記載します。

(1) lwip の Web から以下の ZIP ファイルをダウンロードします。

・lwip-2.1.3.zip

・contrib-2.1.0.zip

(2) asp/lwip の contrib-2.1.0(ASP カーネル用 ethernet 用ドライバ)を別のディレクトリに移動します。

(3) lwip-2.1.3.zip と contrib-2.1.0.zip を解凍して、asp/lwip にコピーする。

(4) 別のディレクトリに退避した contrib-2.1.0 を asp/lwip/contrib-2.1.0 にドラックする。（contrib-2.1.0 に ASP カーネル用 ethernet 用ドライバが追加される）

(5) asp/lwip/patch/contrib-2.1.0 を asp/lwip/contrib-2.1.0 にドラックする。(contrib-2.1.0 用のパッチを反映)

(6) asp/lwip/patch/lwip-2.1.3 を asp/lwip/lwip-2.1.3 にドラックする。(contrib-2.1.0 用のパッチを反映)

OBJ/SOCFPGA_CV_GCC/MAC のサンプルプログラムがビルド可能になります。

6.7 STM32Cube ミドルウェア

STM32Cube でサポートするミドルウェアを実行するために、stmcube/Middleware 上に追加となるミドルウェア用ディレクトリを配置する。メインの TOPPERS BASE PLATFORM(ST)でサポートする２つのミドルウェア以外にオプションボードを追加した場合のミドルウェアのプログラムもこのディレクトリに配置する。

6.7.1 STM32_WPAN ディレクトリ

STM32WB 用の BLE 関係のミドルウェアを stmcube/Middleware/ST/STM32_WPAN 上に配置する。現状の STM32_WPAN ミドルウェアの対応バージョンは STM32Cube_FW_WB_V1.11.0 である。このミドルウェアは STM32WB の STM32WBxx_HAL_Driver に依存性をもつため、いくつかのソースファイルに修正が発生している。TOPPERS BASE PLATFORM(ST)では、修正が発生したファイルのみを公開している。STM32Cube_FW_WB_V1.11.0/Middleware/ST/STM32_WPAN を stmcube/Middleware/ST/にコピー後、修正ファイルを上書きすれば、STM32WB55 で BLE が使用可能になる。

ファイル名	ディレクトリ	内容	区分
serial_if.c	STM32_WPAN/ble/mesh/Src	UART ドライバの違いを修正	修正
shci.c	STM32_WPAN/interface/ble_thread/shci	ハードウェアアクセス直接記載部修正	修正
tl_mbox.c	STM32_WPAN/interface/ble_thread/tl	サブコアとの通信処理の違いを修正	修正
cmsis_compiler.h	STM32_WPAN/	アプリから移動し一本化	追加
hw_if.h	STM32_WPAN/	アプリから移動し一本化	追加
	STM32_WPAN/	アプリから移動し一本化	追加
Makefile.config	STM32_WPAN/	STM32_WPAN 用 makefile	追加

表 6.7.1 STM32_WPAN ミドルウェア追加、修正ファイル

STM32_WPAN を使用する場合は、以下のユーティリティも必要となるため、STM32Cube_FW_WB_V1.11.0/Utilities からコピーして、asp/stmcube/Utilities/にコピーしなければならない。

ディレクトリ名	STMCube32 上のパス	内容
lpm	Utilities	LPM モジュール用プログラム
sequencer	Utilities	BLE のシーケンサ

表 6.7.2 STM32_WPM 用のユーティリティ

6.7.2 OpenAMP ディレクトリ

STM32MP1 用のコア間通信機能 OpenAMP をサポートするため stmcube/Middleware/Third_Paty/OpenAMP 上に配置する。OpenAMP はハードウェアに依存する記載はないため(メモリ等にハードウェア上の約束ごとがある)、STMCube_FW_MP1_V1.2.0/Middleware/Third_Party/OpenAMP をコピーし、TOPPERS BASE PLATFORM(ST)で配布された Makefile.config を OpenAMP の直下に置けば、MP1 用の OpenAMP サンプルプログラムが使用可能になる。

6.8 RTOS 隠蔽機能

pdic ディレクトリの直下に base_platform.h インクルードファイルを置く、このファイルは TOPPERS BASE PLATFORM 内の以下のモジュールから参照される。このファイル内に RTOS との API のマクロ化定義を行い。マクロ定義を書き換えることにより、種々の RTOS の API に対応することができる。このマクロを参照するディレクトリは 7.1 共通部に記載。

マクロの記載は一種と二種があり、以下の通り。

ASP/JSP での記載		ASP3 での記載	
DELAY_TASK	dly_tsk	DELAY_TASK(t)	dly_tsk((t)*1000U)
SLEEP_TASK	slp_tsk	ASP に同じ	ASP に同じ
TIME_SLEEP_TASK	tslp_tsk	tslp_tsk(t)	tslp_tsk((t)+1000U)
WAKEUP_TASK	wup_tsk	ASP に同じ	ASP に同じ
WAIT_SEMAPHORE	wai_sem	ASP に同じ	ASP に同じ
TIME_WAIT_SEMAPHORE	twai_sem		
SIGNAL_SEMAPHORE	sig_sem	ASP に同じ	ASP に同じ
INTERRUPT_SIGNAL_SEMAPHORE	isig_sem	ASP に同じ	ASP に同じ
LOCK_CPU	loc_sem	ASP に同じ	ASP に同じ
UNLOCK_CPU	unl_sem	ASP に同じ	ASP に同じ
ENABLE_INTERRUPT	ena_int	ASP に同じ	ASP に同じ
DISABLE_INTERRUPT	dis_int	ASP に同じ	ASP に同じ

表 6.8.1 一種 RTOS マクロ定義

ASP/JSP/ASP3/FMP での記載	
GET_TASK_ID	get_tid
ACTIVATE_TASK	act_tsk
START_CYCLIC_HANDLER	sta_cyc
STOP_CYCCLIC_HANDLER	stp_cyc
RECEIVE_DATA_QUEUE	rcv_dtq
TIME_RECEIVE_DATA_QUEUE	trcv_dtq
POLLING_SEND_DATA_QUEUE	psnd_dtq
INTERRUPT_POLLING_SEND_DATA_QUEUE	ipsnd_dtq

表 6.8.2 二種 RTOS マクロ定義

7 ファイルの構成

TOPPERS BASE PLATFORM のソースファイル構造について記載する。共通部はファイルシステムやタスクモニタ等共通となる部分について記載する。BASE PLATFORM のソフトウェア部品は asp のベースディレクトリ上に配置する。

7.1 共通部

共通部のディレクトリ構成を表 7.1.1 に示す。

ディレクトリ	内容	備考

files	ファイルシステムのソースとインクルードファイル	RTOS 隠蔽化対応
monitor	タスクモニタと標準入手力のソースとインクルードファイル	
gdic	GDIC ドライバ	RTOS 隠蔽化対応
pdic	PDIC ドライバ、STM32xxx にボード依存ドライバを持つ	RTOS 隠蔽化対応
syssvc	malloc,calloc,free 関数	
ui	GLCD に文字、グラフィックデータ表示	V1.3.1 以降 RTOS 隠蔽化対応
usb	USB ホスト、デバイスのミドルウェア	RTOS 隠蔽化対応
stmcube	V1.3.1 より Font のみを使用、BSP 部は gdic に移管した (gdic ドライバには、この領域を拡張して使用するものがある)	
jpeg-9b	JPEG ライブラリ､Web より jpeg-9b をダウンロードセットし、ディレクトリ中の Makefile でライブラリをビルドしてください	ソースなし
libmad-0.15.1b	MAP3 でコードライブラリ、Web より libmad-0.15.1b をダウンロードセットし、ディレクトリ中の Makefile でライブラリをビルドしてください	ソースなし
lwip	lwip プロトコル･スタック､Web より lwip-2.1.2 と contrib-2.1.0 をダウンロードセットし、patch ディレクトリ中のソースパッチを行ってください	ソースなし RTOS 隠蔽化対応

表 7.1.1 共通部ディレクトリ

pdic の直下に RTOS 隠蔽用のインクルードファイル base_platform.h を置く。

7.2 STM32F4xx ドライバ

STM32F4xx 用ドライバ部、pdic/stm32f4xx にソースファイルがある。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	
adc.h	ADC ドライバ・インクルードファイル	
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	Base
device.cfg	LED,SW の RTOS リソースファイル	Base
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	Base
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・ソースファイル	
pinmode.h	Arduino の GPIO ピン設定・インクルードファイル	
pwm.c	TIM1/TIM2 用の PWM ドライバ・ソースファイル	V1.4.3
pwm.h	TIM1/TIM2 用の PWM ドライバ・インクルードファイル	V1.4.3
rts.c	RTS ドライバ・ソースファイル	
rts.cfg	RTS の RTOS リソースファイル	
rts.h	RTS ドライバ・インクルードファイル	
spi.c	SPI ドライバ・ソースファイル	
spi.h	SPI ドライバ・インクルードファイル	
usb_otg.c	USB-OTG ドライバ・ソースファイル	144 のみ
usb_otg.h	USB-OTG ドライバ・インクルードファイル	144 のみ

表 7.2.1 STM32F4xx ドライバファイル

7.3 STM32L4xx ドライバ

STM32L4xx 用ドライバ部、pdic/stm32l4xx にソースファイルがある。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	
adc.h	ADC ドライバ・インクルードファイル	
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	Base

device.cfg	LED,SW の RTOS リソースファイル	Base
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	Base
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・ソースファイル	
pinmode.h	Arduino の GPIO ピン設定・インクルードファイル	
qspi.c	QSPI ドライバ・ソースファイル	
qspi.h	QSPI ドライバ・インクルードファイル	
rts.c	RTS ドライバ・ソースファイル	
rts.cfg	RTS の RTOS リソースファイル	
rts.h	RTS ドライバ・インクルードファイル	
spi.c	SPI ドライバ・ソースファイル	
spi.h	SPI ドライバ・インクルードファイル	

表 7.3.1 STM32L4xx ドライバファイル

7.4 STM32F7xx ドライバ

STM32F7xx 用ドライバ部、pdic/stm32f7xx にソースファイルがある。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	
adc.h	ADC ドライバ・インクルードファイル	
clock.c	RTC デバッグコマンド・ソースファイル	
device.c	GPIO,DMADMA2D,RTC,CHACHE,LED,SW ドライバ・ソースファイル	Base
device.cfg	RTS,LED,SW の RTOS リソースファイル	Base
device.h	GPIO,DMADMA2D,RTC,CHACHE,LED,SW ドライバ・インクルードファイル	Base
dfsdm.c	DFSDM 入力ドライバ・ソースファイル	769 のみ
dfsdm.h	DFSDM 入力ドライバ・インクルードファイル	769 のみ
dsi.c	DSI LCD インターフェイス・ソースファイル	769 のみ
dsi.h	DSI LCD インターフェイス・インクルードファイル	769 のみ
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
ltdc.c	GLCD ドライバ・ソースファイル	
ltdc.h	GLCD インクルード・ソースファイル	
mcicmd.h	ファイルシステム用インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・ソースファイル	
pinmode.h	Arduino の GPIO ピン設定・インクルードファイル	
qspi.c	QSPI ドライバ・ソースファイル	
qspi.h	QSPI ドライバ・インクルードファイル	
sai.c	オーディオドライバ・ソースファイル	
sai.h	オーディオドライバ・インクルードファイル	
sdmmc.c	SD-card ドライバ・ソースファイル	
sdmmc.cfg	SD-card ドライバ・RTOS リソースファイル	
sdmmc.h	SD-card ドライバ・インクルードファイル	
spi.c	SPI ドライバ・ソースファイル	
spi.h	SPI ドライバ・インクルードファイル	
~~stm32f7xx_hal.h~~	~~STMCube 連結用インクルードファイル~~	1.3.1 で削除
usb_otg.c	USB-OTG ドライバ・ソースファイル	
usb_otg.h	USB-OTG ドライバ・インクルードファイル	
emac.c	EMAC ドライバ・ソースファイル	1.4.1 で追加
emac.h	EMAC ドライバ・インクルードファイル	1.4.1 で追加

phyreg.h	ehernet-phy 設定・インクルードファイル	1.4.1 で追加
~~wifi.c~~	~~ESP32 用 WIFI ドライバ・ソースファイル~~	gdic に移動
~~wifi.h~~	~~ESP32 用 WIFI ドライバ・インクルードファイル~~	gdic に移動
wifiuart.c	WIFI 用 UART 通信ドライバ・ソースファイル	1.4.1 で追加
wifiuart.h	WIFI 用 UART 通信ドライバ・インクルードファイル	1.4.1 で追加
wifiuart.cfg	WIFI 用 UART 通信 RTOS リソースファイル	1.4.1 で追加

表 7.4.1 STM32F7xx ドライバファイル

7.5 STM32F0xx ドライバ

STM32F0xx 用ドライバ部、pdic/stm32f0xx にソースファイルがある。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	
adc.h	ADC ドライバ・インクルードファイル	
clock.c	RTC デバッグコマンド・ソースファイル	
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	Base
device.cfg	RTS,LED,SW の RTOS リソースファイル	Base
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	Base
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・ソースファイル	
pinmode.h	Arduino の GPIO ピン設定・インクルードファイル	
rtc.c	RTS ドライバ・ソースファイル	
rtc.cfg	RTS の RTOS リソースファイル	
rtc.h	RTS ドライバ・インクルードファイル	
spi.c	SPI ドライバ・ソースファイル	
spi.h	SPI ドライバ・インクルードファイル	
stm32f0xx.h	stm32f0xx コアデバイスアドレス定義ファイル	1.4.4 で追加

表 7.5.1 STM32F0xx ドライバファイル

7.6 STM32L0xx ドライバ

STM32L0xx 用ドライバ部、pdic/stm32l0xx にソースファイルがある。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	
adc.h	ADC ドライバ・インクルードファイル	
clock.c	RTC デバッグコマンド・ソースファイル	
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	Base
device.cfg	RTS,LED,SW の RTOS リソースファイル	Base
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	Base
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・ソースファイル	
pinmode.h	Arduino の GPIO ピン設定・インクルードファイル	
rtc.c	RTS ドライバ・ソースファイル	
rtc.cfg	RTS の RTOS リソースファイル	
rtc.h	RTS ドライバ・インクルードファイル	
spi.c	SPI ドライバ・ソースファイル	
spi.h	SPI ドライバ・インクルードファイル	
usb_device.c	USB デバイスドライバ・ソースファイル	
usb_device.h	USB デバイスドライバ・インクルードファイル	
stm32l0xx.h	stm32l0xx コアデバイスアドレス定義ファイル	1.4.4 で追加

表 7.6.1 STM32L0xx ドライバファイル

7.7 STM32G0xx ドライバ

STM32L0xx 用ドライバ部、pdic/stm32l0xx にソースファイルがある。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	
adc.h	ADC ドライバ・インクルードファイル	
clock.c	RTC デバッグコマンド・ソースファイル	
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	Base
device.cfg	RTS,LED,SW の RTOS リソースファイル	Base
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	Base
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・ソースファイル	
pinmode.h	Arduino の GPIO ピン設定・インクルードファイル	
rtc.c	RTS ドライバ・ソースファイル	
rtc.cfg	RTS の RTOS リソースファイル	
rtc.h	RTS ドライバ・インクルードファイル	
spi.c	SPI ドライバ・ソースファイル	
spi.h	SPI ドライバ・インクルードファイル	
usb_device.c	USB ドライバ・ソースファイル	1.4.3 で追加
usb_device.h	USB ドライバ・インクルードファイル	1.4.3 で追加
stm32g0xx.h	stm32g0xx コアデバイスアドレス定義ファイル	1.4.4 で追加

表 7.7.1 STM32G0xx ドライバファイル

7.8 STM32H7XX ドライバ

STM32H7xx 用ドライバ部、pdic/stm32h7xx にソースファイルがある。このドライバは V1.4.1 での追加である。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	
adc.h	ADC ドライバ・インクルードファイル	
clock.c	RTC デバッグコマンド・ソースファイル	
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	
device.cfg	RTS,LED,SW の RTOS リソースファイル	
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	
emac.c	EMAC ドライバ・ソースファイル	
emac.h	EMAC ドライバ・インクルードファイル	
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
phyreg.h	ehernet-phy 設定・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・ソースファイル	
pinmode.h	Arduino の GPIO ピン設定・インクルードファイル	
spi.c	SPI ドライバ・ソースファイル	
spi.h	SPI ドライバ・インクルードファイル	

表 7.8.1 STM32H7xx ドライバファイル

7.9 STM32G4XX ドライバ

STM32G4xx 用ドライバ部、pdic/stm32g4xx にソースファイルがある。このドライバは V1.4.1 での追加である。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	

adc.h	ADC ドライバ・インクルードファイル	
clock.c	RTC デバッグコマンド・ソースファイル	
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	
device.cfg	RTS,LED,SW の RTOS リソースファイル	
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・インクルードファイル	
pinmode.h	SPI ドライバ・ソースファイル	
rtc.c	RTS ドライバ・ソースファイル	
rtc.cfg	RTS の RTOS リソースファイル	
rtc.h	RTS ドライバ・インクルードファイル	
spi.c	SPI ドライバ・インクルードファイル	
spi.h	SPI ドライバ・インクルードファイル	
usb_device.c	USB デバイスドライバ・ソースファイル	
usb_device.h	USB デバイスドライバ・インクルードファイル	

表 7.9.1 STM32G4xx ドライバファイル

7.10 STM32WBXX ドライバ

STM32WBxx 用ドライバ部、pdic/stm32wbxx にソースファイルがある。このドライバは V1.4.1 での追加である。

ファイル	内容	備考
adc.c	ADC ドライバ・ソースファイル	
adc.h	ADC ドライバ・インクルードファイル	
clock.c	RTC デバッグコマンド・ソースファイル	
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	
device.cfg	RTS,LED,SW の RTOS リソースファイル	
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	
i2c.c	I2C ドライバ・ソースファイル	
i2c.h	I2C ドライバ・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・インクルードファイル	
pinmode.h	SPI ドライバ・ソースファイル	
rtc.c	RTS ドライバ・ソースファイル	
rtc.cfg	RTS の RTOS リソースファイル	
rtc.h	RTS ドライバ・インクルードファイル	
spi.c	SPI ドライバ・インクルードファイル	
spi.h	SPI ドライバ・インクルードファイル	
usb_device.c	USB デバイスドライバ・ソースファイル	
usb_device.h	USB デバイスドライバ・インクルードファイル	
wakeup_timer.c	WAKEUP タイマードライバー・ソースファイル	
wakeup_timer.h	WAKEUP タイマードライバ・インクルードファイル	
~~wifi.c~~	~~ESP32 用 WIFI ドライバ・ソースファイル~~	gdic に移動
~~wifi.h~~	~~ESP32 用 WIFI ドライバ・インクルードファイル~~	gdic に移動
wifiuart.c	WIFI 用 UART 通信ドライバ・ソースファイル	V1.4.2
wifiuart.cfg	WIFI 用 UART 通信ドライバ・リソースファイル	V1.4.2
wifiuart.h	WIFI 用 UART 通信ドライバ・インクルードファイル	V1.4.2
pwm.c	PWM ドライバ・ソースファイル（TIM1,TIM2 限定）	V1.4.2
pwm.h	PWM ドライバ・インクルードファイル	V1.4.2

表 7.10.1 STM32WBxx ドライバファイル

7.11 STM32MP1XX ドライバ

STM32MP1xx 用ドライバ部、pdic/stm32mp1xx にソースファイルがある。このドライバは V1.4.2 での追加である。

ファイル	内容	備考
device.c	GPIO,DMA,LED,SW ドライバ・ソースファイル	
device.cfg	RTS,LED,SW の RTOS リソースファイル	
device.h	GPIO,DMA,LED,SW ドライバ・インクルードファイル	
i2c.c	I2C ドライバ・ソースファイル	
I2c.h	I2C ドライバ・インクルードファイル	
pinmode.c	Arduino の GPIO ピン設定・インクルードファイル	
pinmode.h	SPI ドライバ・ソースファイル	
spi.c	SPI ドライバ・インクルードファイル	
spi.h	SPI ドライバ・インクルードファイル	

表 7.11.1 STM32MP1xx ドライバファイル

7.12 GDIC ドライバ

ディレクトリ gdic 以下に標準の TOPPERS BASE PLATFORM で提供する GDIC ドライバを示す。GDIC ドライバは PDIC に依存性し、デバイスに依存した機能を提供する。GDIC ドライバはオプション追加により拡張が可能である。

ディレクトリ	内容	備考
usb_otg	usb_otg(DWC2-OTG ドライバ)上に位置し、USB ミドルウェアに OTG 機能を提供する	
usb_device	usb_device(ST 社 USB デバイス IP)上に位置し、USB ミドルウェアに USB デバイス機能を提供する	
spi_driver	SPI インターフェイスの SD カード用ドライバ、ファイルシステムに SD カードドライバを提供する	
adafruit_st7735	SPI インターフェイスの Adafruit 1.8”LCD に対して、グラフィック API を提供する	
adafruit_st7789	SPI インターフェイスの Adafruit1.54”LCD に対して、グラフィック API を提供する	1.4.1
adafruit_ILI9341	SPI インターフェイスの Adafruit 2.4”LCD に対して、グラフィック API を提供する	1.4.1
aqm0802_st7032	I2C インターフェイスの AQM0802 キャラクタ LCD に対して、キャラクタ表示 API を提供する	
aqm1284_st7565	SPI インターフェイスの AQM1284 に対して、グラフィック API を提供する	1.3.1
bleshield2.1	Arduino BLE Shield2.1 に対して、BLE-API を提供する	1.3.1
esp32	ESP32-UART WIFI 機能を提供する	1.4.4
stlcd_otm8009a	ST 用 otm8009a に対して、LCD 用 API を提供する	1.3.1
stlcd_st7789h2	ST 用 st7789h2 に対して、LCD 用 API を提供する	1.3.1
stlcd_rk043fn48h	ST 用 rk043fn48h に対して、LCD 用 API を提供	1.3.1
stts_ft06x06	ST 用 ft06x06 に対して、タッチパネル用 API を提供する	1.3.1
stts_ft5336	ST 用 ft5336 に対して、タッチパネル用 API を提供する	1.3.1
adafruitst_stmp610	Adafruit 2.4”LCD 用 stmp610 に対して、タッチパネル用 API を提供する	1.4.1
studio_wm8994	ST 用 wm8994 に対して、オーディオ用 API を提供する	1.3.1
flash_file	QSPI フラッシュ上のデータを読み出し専用ファイルとしてアクセスする機能を提供する	1.3.1
rom_file	ROM 上のデータを読み出し専用ファイルとしてアクセスする機能を提供する	1.4.1

表 7.12.1 GDIC ディレクトリ

8　変更履歴詳細

ソース公開を行った v1.3.0 以降の変更履歴詳細を記載する。

version	date	functions
1.4.0	06/22/2018	spi-SD-HC カードの不具合対応(プログラムのみ)
	06/25/2018	リードオンリーの fopen で open エラーとならない不具合対応(プログラムのみ)
	07/05/2018	usb host SERIAL/PRT 送受信を並行処理できるよう修正 usb host BLUETOOTH クラス対応
	07/31/2018	gdic/stlcd_otm8009a:gdic/stts_ft07x07:gdic/staudio_wm8994 を追加
	08/07/2018	STM32F723 Discovery ボード対応、gdic/stlcd_st7789h2 を追加
	08/31/2018	SYM32L4R5 Nucleo-144 ボードの対応
	09/06/2018	図 3.1 を TEB001 に修正
	09/11/2018	gdic/stlcd_rk043fn48h:gdic/stts_ft5336 を追加、stmcube の BSP ドライバを削除
	10/17/2018	ui ディレクトリ、gidc/flash_file:gdic/aqm1284_st7565 を追加
	11/01/2018	gdic/bleshield2.1 を追加
	11/07/2018	WATCH DOC ドライバの対応
	01/23/2019	STM32G071 Nucleo-64 ボードの対応
1.4.1	03/23/2019	STM-BSP を削除
	04/05/2019	gdic/adafruit_ILI9341 を追加
	07/28/2019	STM32L4R5 の USB が動作可能に変更
	08/14/2019	STM32H743 Nucleo-144 ボードの対応
	11/07/2019	TLSF2.4.6 アロケート・フリーを繰り返すとアロケート領域の先頭８バイトが破壊される問題の暫定対策
	12/03/2019	ROM ファイル機能を GDIC に追加。ui/snfont_disp に漢字フォントをプログラムデータとして取り込み機能を追加
	12/23/2019	STM32G474/431 Nucleo-64 ボードの対応
	03/16/2020	STM32WB44 Nucleo ボードの対応
	03/31/2020	メモリヒープの libc/LOCAL_HEAP 対応
	04/18/2020	ESP32 用 WIFI ドライバを STM32F7xx 用に追加
	05/11/2020	IWHI 1.54" LCD シールド(TEB002)の記載を追加
1.4.2	06/22/2020	STM32WBXX に WIFI ドライバを拡張/BLE モジュール StmcubeV1.7.0 対応
	09/29/2020	STM32MP157C-DK2 用に OpenAMP の対応
	11/03/2020	ファイルシステムで long name を使用ない設定を可能する
	01/24/2021	STM32WBXX に PWM ドライバ追加
	04/03/2021	BLE モジュール StmcubeV1.11.0 対応(差分のみリリース)
	05/24/2021	lwip と STMCube ミドルウェアの設定方法を記載。
	06/01/2021	STM32_WPAN 用のユーティリティを追加。
1.4.3	08/21/2021	STM32F401 nucleo 用 TIM1/2 PWM ドライバを追加と USBD 対応
	10/30/2021	STM32G0B1 nucleo-64 ボードを対応
	12/08/2021	RTOS のサービス・コール隠蔽機能を追加
	01/05/2022	BLE モジュール StmcubeV1.12.0 対応(差分のみリリース)
	01/14/2022	STM32WB15 nucleo ボードを対応
1.4.4	10/16/2022	STM32F4XX 用 PWM ドライバに deinit 関数を追加

	11/04/2022	STM-USB デバイスの CDC のエンドポイントの不具合を修正
	11/19/2022	USB Device HID 使用時に Win10 でデータを送れない問題の対応、USB 隠蔽化を再修正
	12/12/2022	Cortex-M0 と M0+のデバイスアドレス定義ファイルを pdic に追加
	03/14/2023	gdic に esp32 を追加
	04/13/2023	lwip を 2.1.3 にバージョンアップ
1.4.5	7/13/2023	lwIP の OPTTEST 用 TCPIP ロック機能を追加
	10/26/2023	ネットワークモニタ用に標準出力処理の変更
	12/12/2023	USB デバイスに MSC を追加
	01/12/2024	STM32WB のミドルウェア対応バージョンを 1.12 から 1.18 にアップ
1.4.6	02/27/2024	STM32WB55 の ble_remote アプリの Nucleo に対応可能に修正、コンパイル時のワーニング対応(stddev.c, sddiskio.c)

Appendix A　STM32F401RE Nucleo

STM32F401RE Nucleo のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_0
	2	A1	PA1	ADC1_1
	3	A2	PA4	ADC1_4
	4	A3	PB0	ADC1_8
	5	A4	PC1 or PB9	ADC1_11 or I2C1_SDA
	6	A5	PC0 or PB8	ADC1_10 or I2C1_SCL

表 A.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM1_CH1N or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 degital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

表 A.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	Arduino D6
		SDA	PB3	Arduino D3
3	I2C3	SCL	PA8	Arduino D7
		SDA	PB4	Arduino D6

表 A.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12

		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	
		MISO	PB14	
		MOSI	PB15	

表 A.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1

表 A.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART2	TX	PA2	Arduino D1/ ST-LINK TX
		RX	PA3	Arduino D0/ ST-LINK RX
2	USART1	TX	PA9	Arduino D8
		RX	PA10	Arduino D2

表 A.5.1 UART ポート割り当て

(6) その他

Adafruit 1.8" TFT Shield 動作

RedBear BLE Shield2.1 動作

Appendix B　STM32F4 Discovery

STM32F446RE Nucleo のボード依存仕様を記載する。

(1) Arduino connectors 定義

コネクタなし

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	
		SDA	PB9	
2	I2C2	SCL	PB10	
		SDA	PB11	
3	I2C3	SCL	PA8	
		SDA	PC9	

表 B.2.1 I2C 接続ピン

(3) SPI

STM32F401RE Nucleo に同じ

(4) ADC

STM32F401RE Nucleo に同じ

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART2	TX	PA2	
		RX	PA3	

表 B.5.1 UART ポート割り当て

(6) その他

USB-OTG FULL-HOST サポート

Appendix C　STM32F746 Discovery

STM32F746 Discovery のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC3_0
	2	A1	PF10	ADC3_8
	3	A2	PF9	ADC3_7
	4	A3	PF8	ADC3_6
	5	A4	PF7 or PB9	ADC3_5 or I2C1_SDA
	6	A5	PF6 or PB8	ADC3_4 or I2C1_SCL

表 C.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PI1	SPI2_SCK
	5	D12	PB14	SPI2_MISO
	4	D11	PB15	TIM12_CH2 or SPI2_MOSI
	3	D10	PI0	TIM5_CH4 or SPI2_NSS
	2	D9	PA15	TIM2_CH1
	1	D8	PI2	-
CN9 degital	8	D7	PI3	-
	7	D6	PH6	TIM12_CH1
	6	D5	PA8	TIM1_CH1
	5	D4	PG7	-
	4	D3	PB4	TIM3_CH1
	3	D2	PG6	-
	2	D1	PC6	USART6_TX
	1	D0	PC7	USART6_RX

表 C.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PH1	
		SDA	PH0	
3	I2C3	SCL	PH7	ボード内：Audio/Tp デバイス
		SDA	PH8	ボード内：Audio/Tp デバイス

表 C.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	

		MISO	PA6	
		MOSI	PA7	
2	SPI2	SCK	PI1	Arduino D13
		MISO	PB14	Arduino D12
		MOSI	PB15	Arduino D11

表 C.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2
3	ADC3

表 C.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART1	TX	PA9	ST-LINK TX
		RX	PB7	ST-LINK RX
2	USART6	TX	PC6	Arduino D1
		RX	PC7	Arduino D0

表 C.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

USB-OTG HS ポートサポート、但し、オーディオと並行動作させると動作不安定

Adafruit 1.8" TFT Shield：ADC 動作、LCD 不安定、SD カード通信できず

RedBear BLE Shield2.1：未評価

lwip 動作

Appendix D　STM32F446RE Nucleo-64

STM32F446RE Nucleo-64 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC123_0
	2	A1	PA1	ADC123_1
	3	A2	PA4	ADC12_4
	4	A3	PB0	ADC12_8
	5	A4	PC1 or PB9	ADC123_11 or I2C1_SDA
	6	A5	PC0 or PB8	ADC123_10 or I2C1_SCL

図 D.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground

	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM14_CH1N or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM8_CH2
	1	D8	PA9	-
CN9	8	D7	PA8	-
degital	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

図 D.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	Arduino D6
		SDA	PB3	Arduino D3
3	I2C3	SCL	PA8	Arduino D7
		SDA	PC9	

表 D.2.1 I2C 接続ピン

(3) SPI

STM32F401RE Nucleo に同じ

(4) ADC

STM32F401RE Nucleo に同じ

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART2	TX	PA2	Arduino D1/ ST-LINK TX
		RX	PA3	Arduino D0/ ST-LINK RX

表 D.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

Adafruit 1.8" TFT Sheild 動作

RedBear BLE Shield2.1 動作

Appendix E　STM32F446ZE Nucleo-144

STM32F446ZE Nucleo-144 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6	1	NC	-	-
power	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground

	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA3	ADC123_IN3
	2	A1	PC0	ADC123_IN10
	3	A2	PC3	ADC123_IN13
	4	A3	PF3	ADC3_IN9
	5	A4	PF5 or PB9	ADC3_IN15 or I2C1_SDA
	6	A5	PF10 or PB8	ADC3_IN8 or I2C1_SCL

図 E.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7 or PB5	TIM14_CH1N or SPI1_MOSI
	3	D10	PD14	TIM4_CH3 or SPI1_CS
	2	D9	PD15	TIM4_CH4
	1	D8	PF12	-
CN9 degital	8	D7	PF13	-
	7	D6	PE9	TIM1_CH1
	6	D5	PE11	TIM1_CH2
	5	D4	PF14	-
	4	D3	PE13	TIM1_CH3
	3	D2	PF15	-
	2	D1	PG14	USART6_TX
	1	D0	PG9	USART6_RX

図 E.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	
		SDA	PB3	
3	I2C3	SCL	PA8	
		SDA	PC9	

表 E.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	
		MISO	PB14	
		MOSI	PB15	

表 E.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2

3	ADC3

表 E.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART3	TX	PD8	ST-LINK TX
		RX	PD9	ST-LINK RX
2	USART6	TX	PG14	Arduino D1
		RX	PG9	Arduino D0

表 E.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能
USB-OTG FULL ポートサポート
Adafruit 1.8" TFT Sheild 動作
RedBear BLE Shield2.1 動作しない
Bluetooth ドングル正常動作しない

Appendix F　STM32F746ZG Nucleo-144

STM32F746ZGT6 Nucleo-144 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA3	ADC123_IN3
	2	A1	PC0	ADC123_IN10
	3	A2	PC3	ADC123_IN13
	4	A3	PF3	ADC3_IN9
	5	A4	PF5 or PB9	ADC3_IN15 or I2C1_SDA
	6	A5	PF10 or PB8	ADC3_IN8 or I2C1_SCL

図 E.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7 or PB5	TIM14_CH1N or SPI1_MOSI
	3	D10	PD14	TIM4_CH3 or SPI1_CS
	2	D9	PD15	TIM4_CH4
	1	D8	PF12	-
CN9 degital	8	D7	PF13	-
	7	D6	PE9	TIM1_CH1
	6	D5	PE11	TIM1_CH2
	5	D4	PF14	-

	4	D3	PE13	TIM1_CH3
	3	D2	PF15	-
	2	D1	PG14	USART6_TX
	1	D0	PG9	USART6_RX

図 F.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	
		SDA	PB3	
3	I2C3	SCL	PA8	
		SDA	PC9	

表 F.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	
		MISO	PB14	
		MOSI	PB15	

表 F.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2
3	ADC3

表 F.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART3	TX	PD8	ST-LINK TX
		RX	PD9	ST-LINK RX
2	USART6	TX	PG14	Arduino D1
		RX	PG9	Arduino D0

表 F.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能
USB-OTG FULL ポートサポート
Adafruit 1.8" TFT Shield：SD カードは SPIDRIVER がないので未確認
RedBear BLE Shield2.1 動作しない
lwip 動作

Appendix G　STM32F767ZIT6 Nucleo-144

STM32F767ZIT6T6 Nucleo-144 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6	1	NC	-	-

power	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA3	ADC123_IN3
	2	A1	PC0	ADC123_IN10
	3	A2	PC3	ADC123_IN13
	4	A3	PF3	ADC3_IN9
	5	A4	PF5 or PB9	ADC3_IN15 or I2C1_SDA
	6	A5	PF10 or PB8	ADC3_IN8 or I2C1_SCL

図 G.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7 or PB5	TIM14_CH1N or SPI1_MOSI
	3	D10	PD14	TIM4_CH3 or SPI1_CS
	2	D9	PD15	TIM4_CH4
	1	D8	PF12	-
CN9 degital	8	D7	PF13	-
	7	D6	PE9	TIM1_CH1
	6	D5	PE11	TIM1_CH2
	5	D4	PF14	-
	4	D3	PE13	TIM1_CH3
	3	D2	PF15	-
	2	D1	PG14	USART6_TX
	1	D0	PG9	USART6_RX

図 G.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PH1	
		SDA	PH0	
3	I2C3	SCL	PH7	
		SDA	PH8	
4	I2C4	SCL	PD12	
		SDA	PB13	

表 G.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	
		MISO	PB14	

		MOSI	PB15	

表 G.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2
3	ADC3

表 G.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART3	TX	PD8	ST-LINK TX
		RX	PD9	ST-LINK RX
2	USART6	TX	PG14	Arduino D1
		RX	PG9	Arduino D0

表 G.5.1 UART ポート割り当て

(6) その他
RTC ドライバを使用可能
USB-OTG FULL ポートサポート
Adafruit 1.8" TFT Shield：SD カードは SPIDRIVER がないので未確認
RedBear BLE Shield2.1 動作しない
lwip 動作

Appendix H　STM32F769NIH6 Discovery

STM32F769NHH6 Discovery のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA6	ADC1_IN6
	2	A1	PA4	ADC1_IN4
	3	A2	PC2	ADC1_IN12
	4	A3	PF10	ADC3_IN8
	5	A4	PF8 or PB9	ADC3_IN6 or I2C1_SDA
	6	A5	PF90 or PB8	ADC3_IN7 or I2C1_SCL

図 H.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA12	SPI2_SCK
	5	D12	PB14	SPI2_MISO
	4	D11	PB15	TIM12_CH2 or SPI2_MOSI

	3	D10	PA11	TIM1_CH4 or SPI2_CS
	2	D9	PH6	TIM12_CH1
	1	D8	PJ4	-
CN9 degital	8	D7	PI3	-
	7	D6	PF7	TIM11_CH1
	6	D5	PC8	TIM3_CH3
	5	D4	PJ0	-
	4	D3	PF6	TIM10_CH1
	3	D2	PJ1	-
	2	D1	PC6	USART6_TX
	1	D0	PC7	USART6_RX

図 H.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PH1	
		SDA	PH0	
3	I2C3	SCL	PH7	
		SDA	PH8	
4	I2C4	SCL	PD12	
		SDA	PB7	

表 H.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	
		MISO	PA6	
		MOSI	PA7	
2	SPI2	SCK	PA12	Arduino D13
		MISO	PB14	Arduino D12
		MOSI	PB15	Arduino D11

表 H.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2
3	ADC3

表 H.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART1	TX	PA9	ST-LINK TX
		RX	PA10	ST-LINK RX
2	USART6	TX	PC6	Arduino D1
		RX	PC7	Arduino D0

表 H.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

USB-OTG HIGH ポートサポート

Adafruit 1.8" TFT Shield：ADC 動作、LCD 不安定、SD カード通信できず

RedBear BLE Shield2.1 動作しない

ボード上の問題で lwip は動作しない

Appendix I　STM32F091 Nucleo-64

STM32F091RCT6 Nucleo-64 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9	ADC_IN11 or I2C1_SDA
	6	A5	PC0 or PB8	ADC_IN10 or I2C1_SCL

図 I.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM16_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	- or UART1_TX
CN9 degital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	- or UART1_RX
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

図 I.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	
		SDA	PB11	

表 I.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13

		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	
		MISO	PB14	
		MOSI	PB15	

表 I.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1

表 I.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART2	TX	PA2	ST-LINK TX
		RX	PA3	ST-LINK RX
2	USART1	TX	PA9	Arduino D8
		RX	PA10	Arduino D2

表 I.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

Adafruit 1.8" TFT Shield：ADC、LCD 動作、SD カード通信できず

RedBear BLE Shield2.1 動作

Appendix J　STM32L073 Nucleo-64

STM32L073RZT6 Nucleo-64 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9	ADC_IN11 or I2C1_SDA
	6	A5	PC0 or PB8	ADC_IN10 or I2C1_SCL

図 J.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM22_CH2 or SPI1_MOSI

	3	D10	PB6	SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	- or USART1_TX
CN9 degital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	- or USART1_RX
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

図 J.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	
		SDA	PB11	
3	I2C3	SCL	PA8	Arduino D7
		SDA	PB4	Arduino D6

表 J.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	
		MISO	PB14	
		MOSI	PB15	

表 J.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1

表 J.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART2	TX	PA2	ST-LINK TX
		RX	PA3	ST-LINK RX
2	USART1	TX	PA9	Arduino D8
		RX	PA10	Arduino D2

表 J.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

Adafruit 1.8" TFT Shield：ADC、LCD、SD カード動作

RedBear BLE Shield2.1 動作

Appendix K　STM32L476 Nucleo-64

STM32L476RGT6 Nucleo-64 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC12_IN5
	2	A1	PA1	ADC12_IN6
	3	A2	PA4	ADC12_IN9
	4	A3	PB0	ADC12_IN15
	5	A4	PC1 or PB9	ADC123_IN2 or I2C1_SDA
	6	A5	PC0 or PB8	ADC123_IN1 or I2C1_SCL

図 K.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	- or USART1_TX
CN9 degital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	- or USART1_RX
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

図 K.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	
		SDA	PB11	
3	I2C3	SCL	PC0	Arduino A5
		SDA	PC1	Arduino A4

表 K.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	

		MISO	PB14	
		MOSI	PB15	
3	SPI3	SCK	PC10	
		MISO	PC11	
		MOSI	PC12	

表 K.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2
3	ADC3

表 K.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART2	TX	PA2	ST-LINK TX
		RX	PA3	ST-LINK RX
2	USART1	TX	PA9	Arduino D8
		RX	PA10	Arduino D2

表 K.5.1 UART ポート割り当て

(6) その他
RTC ドライバを使用可能
Adafruit 1.8" TFT Shield：ADC、LCD は動作、SD カードは動作せず
RedBear BLE Shield2.1 動作

Appendix L　STM32L476 Discovery

STM32L473VGT6 Discovery のボード依存仕様を記載する。
(1) Arduino connectors 定義
コネクタなし

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	
		SDA	PB11	
3	I2C3	SCL	PC0	Arduino A5
		SDA	PC1	Arduino A4

表 L.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	
		MISO	PB14	
		MOSI	PB15	
3	SPI3	SCK	PC10	
		MISO	PC11	
		MOSI	PC12	

表 L.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2
3	ADC3

表 L.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART2	TX	PD5	ST-LINK TX
		RX	PD6	ST-LINK RX
2	USART1	TX	PB6	
		RX	PB7	

表 L.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

Appendix M　STM32F723 Discovery

STM32F723IEK6 Discovery のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA6	ADC1_IN6
	2	A1	PA4	ADC1_IN4
	3	A2	PC4	ADC1_IN14
	4	A3	PF10	ADC3_IN8
	5	A4	PC1	ADC1_IN10
	6	A5	PC0	ADC1_IN11

図 M.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PH4	I2C1_SCL
	9	D14	PH5	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PB4	SPI1_MISO
	4	D11	PB5	TIM17_CH1 or SPI1_MOSI
	3	D10	PA1	NSS1
	2	D9	PH6	TIM12_CH1
	1	D8	PE4	-
CN9 degital	8	D7	PE3	-
	7	D6	PE6	TIM9_CH2
	6	D5	PB0	TIM3_CH3

	5	D4	PH3	-
	4	D3	PE5	TIM9_CH1
	3	D2	PC5	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

図 M.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	wm8994
		SDA	PB9	wm8994
2	I2C2	SCL	PH4	Arduino D15
		SDA	PH5	Arduino D14
3	I2C3	SCL	PA8	ft06x06
		SDA	PH8	ft06x06

表 M2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PH4	Arduino D12
		MOSI	PH5	Arduino D11

表 M3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2
3	ADC3

表 M.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART6	TX	PC6	ST-LINK TX
		RX	PC7	ST-LINK RX
2	USART2	TX	PA2	TX/D1
		RX	PA2	RX/D0

表 M.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

USB-OTG FULL ポートサポート、HIGH ポートは動作しない

Appendix N　STM32F4R5 Nucleo-144

STM32L4R5ZIT6 Nucleo-144 のボード依存仕様を記載する。

(7) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground

	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA3	ADC12_IN8
	2	A1	PC0	ADC123_IN1
	3	A2	PC3	ADC123_IN14
	4	A3	PC1	ADC123_IN2
	5	A4	PC4	ADC12_IN13
	6	A5	PC5	ADC12_IN14

図 N.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM_E_PWM1 or SPI1_MOSI
	3	D10	PD14	SPI1_CS/TIM4_CH3
	2	D9	PD15	TIM4_CH4
	1	D8	PF12	-
CN9 degital	8	D7	PF13	-
	7	D6	PE9	TIM1_CH1
	6	D5	PE11	TIM1_CH2
	5	D4	PF14	-
	4	D3	PE13	TIM1_CH3
	3	D2	PF15	-
	2	D1	PD8	USART3_TX
	1	D0	PD9	USART3_RX

図 N.1.2 右 Arduino connector 設定

(8) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB10	-
		SDA	PB11	-
3	I2C3	SCL	PC0	-
		SDA	PC1	-

表 M2.1 I2C 接続ピン

(9) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	B13	-
		MISO	B14	-
		MOSI	B15	-
3	SPI3	SCK	C10	-
		MISO	C11	-
		MOSI	C12	-

表 M3.1 SPI 接続ピン

(10) ADC

Port No.	Device
1	ADC1

表 M.4.1 ADC ポート割り当て

(11) USART

Port No.	Device	pin	MCU pin	connection
1	LPUART1	TX	PG8	ST-LINK TX
		RX	PG9	ST-LINK RX
2	USART3	TX	PD8	TX/D1
		RX	PD9	RX/D0

表 M.5.1 UART ポート割り当て

(12) その他

RTC ドライバを使用可能

Adafruit 1.8" TFT Shield：ADC、LCD は動作、SD カードは動作せず

USB-OTG FULL は 1.4.0 で動作確認できませんでしたが、1.4.1 で動作しました。電源設定の問題でした。

Appendix O STM32G071/STM32G0B1 Nucleo-64

STM32G071RBT6/STM32G0B1R Nucleo-64 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ARD_A0_IN0
	2	A1	PA1	ARD_A1_IN1
	3	A2	PA2	ARD_A2_IN4
	4	A3	PB1	ARD_A3_IN9
	5	A4	PB11 or PB9	ARD_A4_IN15/I2C2_SCL or I2C1_SCL
	6	A5	PB12 or PB8	ARD_A5_IN16/I2C2_SDA or I2C1_SDA

図 O.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	SPI1_MOSI/TIM14_CH1
	3	D10	PB0	SPI1_CS/TIM3_CH3
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 degital	8	D7	PA8	-
	7	D6	PB14	TIM15_CH1
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-

	4	D3	PB3	TIM1_CH2
	3	D2	PA10	-
	2	D1	PC4	USART1_TX
	1	D0	PC5	USART1_RX

図 O.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PB11	Arduino A4
		SDA	PB12	Arduino A5

表 O2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	A0	-
		MISO	A3	-
		MOSI	A10	-

表 O3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1

表 O.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART2	TX	PA2	ST-LINK TX
		RX	PA3	ST-LINK RX
2	USART1	TX	PC4	TX/D1
		RX	PC5	RX/D0

表 O.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

Adafruit 1.8" TFT Shield：ADC、LCD は動作、SD カードは動作せず

RedBear BLE Shield2.1, TB001:uart/lcd シールドは動作

G0B1 のみ、USB シールドの USB デバイス使用可能

Appendix P　STM32H743 Nucleo-144

STM32H743 Nucleo-144(旧ボード)のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground

	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA3	ADC12_IN15
	2	A1	PC0	ADC123_IN10
	3	A2	PC3	ADC123_IN13
	4	A3	PF3	ADC3_IN5
	5	A4	PF5	ADC3_IN4 or I2C1_SDA(PB9)
	6	A5	PF10	ADC3_IN6 or I2C1_SCL(PB8)

図 P.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	SPI1_MOSI/TIM14_CH1
	3	D10	PD14	SPI1_CS/TIM4_CH3
	2	D9	PD15	TIM4_CH4
	1	D8	PF12	-
CN9 degital	8	D7	PF13	-
	7	D6	PE9	TIM1_CH1
	6	D5	PE11	TIM2_CH1
	5	D4	PF14	-
	4	D3	PE13	TIM1_CH3
	3	D2	PF15	-
	2	D1	PG14	USART6_TX/LPUART1_TX
	1	D0	PG9	USART6_RX/LPUART1_RX

図 P.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PH1	
		SDA	PH0	
3	I2C3	SCL	PH7	
		SDA	PH8	
4	I2C4	SCL	PD12	
		SDA	PB13	

表 P2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	-
		MISO	PB14	-
		MOSI	PB15	-
3	SPI3	SCK	PC10	
		MISO	PC11	
		MOSI	PC12	

表 P3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2
3	ADC3

表 P.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART3	TX	PD8	ST-LINK TX
		RX	PD9	ST-LINK RX
2	LPUART1	TX	PB6	TX/D1
		RX	PB7	RX/D0

表 P.5.1 UART ポート割り当て

(6) その他
RTC ドライバを使用可能
Adafruit 1.8" TFT Shield：ADC、LCD は動作、SD カードは動作せず
TB001:uart/lcd シールドは動作
USB-OTG はエナミュレーション中に通信エラーが発生して停止

Appendix Q　STM32G474/431 Nucleo-64

STM32G474 及び 431 Nucleo-64 のボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC12_IN1
	2	A1	PA1	ADC12_IN2
	3	A2	PA4	ADC12_IN17
	4	A3	PB0	ADC3_IN12 or ADC1_IN15
	5	A4	PC1	ADC12_IN7 or I2C1_SDA(PB9)
	6	A5	PC0	ADC12_IN6 or I2C1_SCL(PA15)

図 Q.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	SPI1_MOSI/TIM3_CH2
	3	D10	PB6	SPI1_CS/TIM4_CH1

	2	D9	PC7	TIM3_CH2 or TIM8_CH2
	1	D8	PA9	-
CN9 degital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PC4/PA2	USART1_TX/LPUART1_TX
	1	D0	PC5/PA3	USART1_RX/LPUART1_RX

図 Q.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	Arduino D15
		SDA	PB9	Arduino D14
2	I2C2	SCL	PA8	Arduino D7
		SDA	PA9	Arduino D8
3	I2C3	SCL	PF3	
		SDA	PF4	
4	I2C4	SCL		対応しない
		SDA		対応しない

表 Q2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	-
		MISO	PB14	-
		MOSI	PB15	-
3	SPI3	SCK	PC10	
		MISO	PC11	
		MOSI	PC12	

表 P3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1
2	ADC2

表 Q.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	LPUART1	TX	PA2	ST-LINK TX
		RX	PA3	ST-LINK RX
2	USART1	TX	PC4	TX/D1
		RX	PC5	RX/D0

表 Q.5.1 UART ポート割り当て

(6) その他

RTC ドライバを使用可能

Adafruit 1.8" TFT Shield：ADC、LCD は動作、SD カードは初期化中通信エラー

TB001:uart/lcd シールドは動作
USB シールドを用いて、USB-DEVICE は動作

Appendix R　STM32WB55 Nucleo

STM32WB55 ボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PC0	ADC1_IN1
	2	A1	PC1	ADC1_IN2
	3	A2	PA1	ADC1_IN5
	4	A3	PA0	ADC1_IN6
	5	A4	PC3	ADC1_IN4
	6	A5	PC2	ADC1_IN3

図 R.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	Ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	SPI1_MOSI/TIM1_CH1
	3	D10	PA4	SPI1_CS/TIM2_CH3(PB10)
	2	D9	PA9	TIM17_CH1
	1	D8	PC12	-
CN9 degital	8	D7	PC13	-
	7	D6	PA8	TIM1_CH1
	6	D5	PA15	TIM2_CH1
	5	D4	PC10	-
	4	D3	PA10	TIM1_CH3
	3	D2	PC6	-
	2	D1	PA2	LPUART1_TX
	1	D0	PA3	LPUART1_RX

図 R.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PA8	Arduino D15
		SDA	PA9	Arduino D14
2	I2C3	SCL	PC0	Arduino A0
		SDA	PC1	Arduino A1

表 R.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection

1	SPI1	SCK	PA5	Arduino D13
		MISO	PA6	Arduino D12
		MOSI	PA7	Arduino D11
2	SPI2	SCK	PB13	-
		MISO	PB14	-
		MOSI	PB15	-

表 R.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	ADC1

表 R.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection
1	USART1	TX	PA2	ST-LINK TX
		RX	PA3	ST-LINK RX
2	LPUART1	TX	PC4	TX/D1
		RX	PC5	RX/D0

表 R.5.1 UART ポート割り当て

(6) その他

RF 有効時は RTC ドライバを使用不可
Adafruit 1.8" TFT Shield：ADC、LCD は動作、SD カードは初期化中通信エラー
TB001:uart/lcd シールドは動作
USB シールドを用いて、USB-DEVICE は動作

Appendix S　STM32MP157C DK2

STM32MP157C DK2 ボード依存仕様を記載する。

(1) Arduino connectors 定義

CN No.	Pin No.	Pin 名	MCU pin	機能
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3V3	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	Ground
	7	GND	-	Ground
	8	VIN	-	Power input
CN8 analog	1	A0	PF14	(ADC1_IN0)
	2	A1	PF13	(ADC1_IN1)
	3	A2	ANA0	(ADC1_IN6)
	4	A3	ANA1	(ADC1_IN2)
	5	A4	PC3/PA12	(ADC1_IN13)
	6	A5	PF12/PA11	

図 S.1.1 左 Arduino connector 設定

CN No.	Pin No.	Pin 名	MCU pin	機能
CN5 digital	10	D15	PA11	I2C5_SCL
	9	D14	PA12	I2C5_SDA
	8	AREF	-	VREF+
	7	GND	-	Ground
	6	D13	PE12	SPI4_SCK
	5	D12	PE13	SPI4_MISO

	4	D11	PE14	SPI4_MOSI/TIM1_CH4
	3	D10	PE11	SPI4_CS/TIM1_CH2
	2	D9	PH6	TIM12_CH1
	1	D8	PG3	-
CN9 degital	8	D7	PD1	-
	7	D6	PE9	TIM1_CH1
	6	D5	PD15	TIM4_CH4
	5	D4	PE10	-
	4	D3	PD14	TIM4_CH3
	3	D2	PE1	-
	2	D1	PE8	UART7_TX
	1	D0	PE7	UART7_RX

図 S.1.2 右 Arduino connector 設定

(2) I2C

Port No.	Device	pin	MCU pin	connection
1	I2C1	SCL	PB8	
		SDA	PB9	
2	I2C2	SCL	PB10	
		SDA	PB11	
3	I2C3	SCL	PH7	
		SDA	PH8	
4	I2C4	SCL	PD12	
		SDA	PD13	
5	I2C5	SCL	PA11	Arduino D15
		SDA	PA12	Arduino D14

表 S.2.1 I2C 接続ピン

(3) SPI

Port No.	Device	pin	MCU pin	connection
1	SPI1	SCK	PA5	
		MISO	PA6	
		MOSI	PA7	
2	SPI2	SCK	PI1	-
		MISO	PI2	-
		MOSI	PI3	-
3	SPI3	SCK	PE0	
		MISO	PD10	
		MOSI	PD6	
4	SPI4	SCK	PE12	Arduino D13
		MISO	PE13	Arduino D12
		MOSI	PE14	Arduino D11
5	SPI5	SCK	PF7	
		MISO	PF8	
		MOSI	PF9	

表 S.3.1 SPI 接続ピン

(4) ADC

Port No.	Device
1	-

表 S.4.1 ADC ポート割り当て

(5) USART

Port No.	Device	pin	MCU pin	connection

1	UART7	TX	PE8	Arduino D1
		RX	PE7	Arduino D0

表 S.5.1 UART ポート割り当て

(6) その他

iwhi TFT Shield：LCD は実行中まれにタイムアウト発生、SD カードは未評価

I2C は SCL/SDA にプルアップ抵抗を付けると NACK エラーとなる