本ドキュメントは Cypress (サイプレス) 製品に関する情報が記載されております。本ドキュメントには、「MB」から始まるシリーズ名，品名およびオーダ型格が記載されておりますが、これらはすべて「CY」から始まるシリーズ名，品名およびオーダ型格として、新規および既存のお客様に引き続き提供してまいります。

オーダ型格の調べ方について

1. www.cypress.com/pcnにアクセスしてください。
2. SEARCH PCNS フィールドに、オーダ型格などのキーワードを入力し、「Apply」をクリックしてください。
3. 該当するタイトル(Title)をクリックしてください。
4. 「Affected Parts List」ファイルを開いてください。
   当該ファイルに記載されている各種変更情報をご利用ください。

詳しいお問い合わせ先

Cypress 製品およびそのソリューションの詳細につきましては、お近くの営業所へお問い合わせください。

サイプレスについて

サイプレスは、世界で最も革新的な車載や産業機器、スマート家電、民生機器および医療機器製品向けに、最先端の組み込みシステム ソリューションを提供するリーディング カンパニーです。サイプレスのマイクロコントローラーや、アナログ IC、ワイヤレスおよび USB ベースのコネクティビティ ソリューション、高い信頼性と高性能を提供するメモリ製品は、各種機器メーカーの差異化製品の開発と早期市場参入を支援します。サイプレスは、ベストクラスのサポートと開発リソースをグローバルに提供することで、彼らが従来市場を破壊しまったく新しい製品カテゴリを歴史的なスピードで市場投入できるよう支援します。詳細はサイプレスのウェブサイト (japan.cypress.com) をご覧ください。

AN205409

Fm3 Family Mb9af310 Series 感熱式プリンタ開発キット用ファームウェア移植

関連ファミリ: MB9AF310 シリーズ

本書では MB9AF312K に基づく感熱式プリンタ開発キットソリューションを他の FM3 ファミリへ移植する手順について説明します。

Contents

1 はじめに

1.1 本書の目的

本書では MB9AF312K に基づく感熱式プリンタ開発キットソリューションを他の FM3 ファミリへ移植する手順について説明します。

1.2 略語の定義

API	:	Application Programming Interface
TPDK	:	Thermal Printer Development Kit
UART	:	Universal Asynchronous Receiver/Transmitter
FIFO	:	First In First Out
OUV	:	Over/Under Voltage
USB	:	Universal Serial BUS

1.3 本書の概要

次節以降の構成は以下のとおりです。

2. MCU ペリフェラル要件
3. システムハードウェア環境
4. 開発環境
5. システムファームウェアデザイン
6. MCU タイプの設定
7. MCU クロックの設定
8. アプリケーション層の移植
9. ユーザコードの移植
10. 実行とデバッグ

1.4 参考文献

(SCH)TPDK-Main

MCU-UM-510127-E-10-ThermalPrinterDevelopmentKit-SolutionFw

MCU-UM-510126-E-10-ThermalPrinterDevelopmentKit-SolutionHw

MB9A310K シリーズ データシート DS706-00029

FM3 ファミリ ペリフェラルマニュアル MN706-00002

2 MCU ペリフェラル要件

2.1 用語の定義

表 1 は TPDK ソリューションに必要なペリフェラルとメモリを示します。コードを移植する前に適切な FM3 ファミリまたは FM4 ファミリの MCU を選択してください。

表 1. MCU ペリフェラル要件リスト

ペリフェラルリソース	数
I/O	34
外部割込み	5
NMI	1
フリーランタイマチャネル	3
デュアルタイマチャネル	2
ソフトウェアウォッチドッグタイマ	1
12-bit A/D コンバータユニット	2
USB2.0 機能チャネル	1
マルチファンクションシリアルインタフェース	2
RAM	>=11K バイト
ROM	>= 40K バイト

3 システムハードウェア環境

3.1 MCU 情報

感熱式プリンタ開発ボードに使用する MB9AF312K のペリフェラルとメモリは以下のとおりです。

- CPU チップ: MB9AF312K
- 最大 CPU 周波数: 40 MHz
- MCU 端子数: 48 pin
- RAM: 16 K バイト
- ROM: 128 K バイト

4 開発環境

表 2. MCU 開発環境

名称	説明	品名	製造	備考
IAR Embedded Workbench 6.60 以降	ファームウェアの開発とデバッグ	N/A	N/A	N/A
J-Link / Mini_USB	ファームウェアのデバッグとロード	N/A	N/A	N/A
ハードウェア情報	TPDK-メイン V1.1.0 ボード	N/A	N/A	N/A

5 システムファームウェアデザイン

本章では感熱式プリンタプロジェクトのファームウェア構造を紹介します。

5.1 ファームウェア構造

IAR のファームウェア構造には図 1 と表 3 で示す 5 つのフォルダがあります。

図 1. ファームウェアの構成

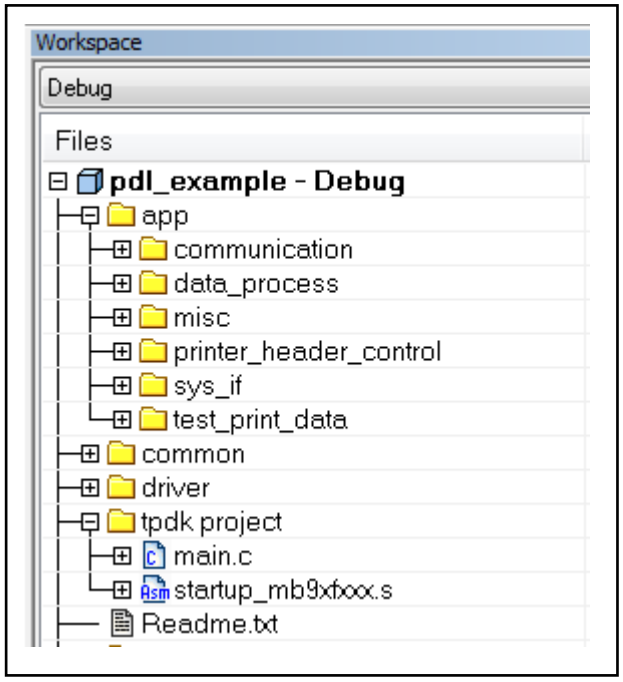


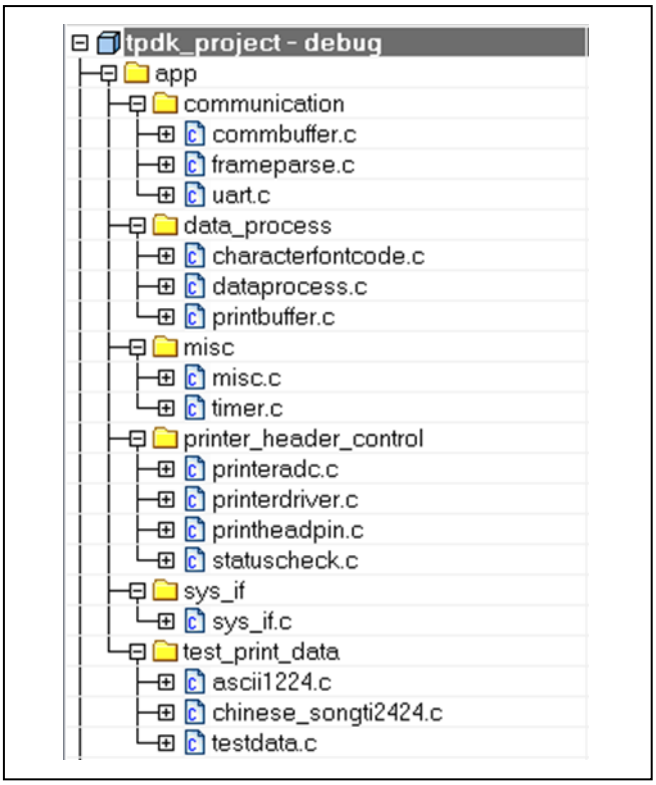
表 3. プロジェクトフォルダの説明

フォルダ	内容
App	communication, data_proces, misc, printer_header_control, sys_if, test_print_data のモジュールを含みます。
Common	システムファイルやグローバルマクロ定義, レジスタ定義, システム初期化関数を含みます。
Driver	MCU ペリフェラルドライバライブラリを含みます。
tpdk_project	IAR プロジェクトとシステムメイン機能を含みます。
Readme.txt	バージョンログやファームウェアの更新情報を記録します。

TPDK プロジェクト用に "app" と "tpdk_project" フォルダはコアのコードを含みます。

"app" サブフォルダは図 2 で示す "communication", "misc", "printer_header_control", "sys_if", "data_proces", "test_print_data" フォルダを含みます。

図 2. App サブフォルダ



各フォルダの詳細な説明を表 4 に示します。

表 4. プロジェクトのファイル説明

フォルダ	ファイル	説明	マーク
communication	commbuffer.c	バッファの生成/読出し/書込み または他の操作	N/A
	commbuffer.h	バッファ機能およびグローバル変数を含む	N/A
	frameparse.c	Parse フレーム	N/A
	frameparse.h	Parse フレーム機能およびグローバル変数を含む	N/A
	uart.c	UART の初期化とデータの送信/受信	N/A
	Uart.h	UART 機能およびグローバル変数を含む	N/A
data_process	characterfontcode.c	ASCII/中国語フォントライブラリマトリックスドットの取得	N/A
	characterfontcode.h	ASCII/中国語フォントハンドル機能およびグローバル変数を含む	N/A
	dataprocess.c	ハンドルプリントデータ	N/A
	dataprocess.h	プリントデータ ハンドル機能およびグローバル変数を含む	N/A
	printbuffer.c	プリントバッファ状態の取得/設定	N/A
	printbuffer.h	プリントバッファ動作機能およびグローバル変数を含む	N/A
misc	misc.c	LED, ビープ音, キーの動作および設定を含む	N/A
	misc.h	動作および設定機能を含む	N/A
	timer.c	FRT, DT, WT タイマの設定および動作を含む	N/A
	timer.h	FRT, DT, WT タイマの設定および動作の機能およびグローバル変数を含む	N/A
printer_header_control	adc.c	ADC の初期化や操作	N/A
	adc.h	初期化, 動作の機能およびグローバル変数を含む	N/A
	printerdriver.c	プリントヘッドドライバの提供	N/A
	printerdriver.h	プリントヘッドドライバの機能およびグローバル変数を含む	N/A
	statuscheck.c	プリンタ状態確認	N/A
	statuscheck.h	ステータスチェックの機能およびグローバル変数を含む	N/A
sys_if	sys_if.c	システム割込みハンドル	N/A
	sys_if.h	システム割込みハンドル機能	N/A
test_print_data	ascii1224.c	ASCII フォントライブラリを含む	N/A
	ascii1224.h	ASCII フォントライブラリのグローバル変数を含む	N/A
	chinese_songti2424.c	中国語フォントライブラリを含む	N/A
	chinese_songti2424.h	中国語フォントライブラリのグローバル変数を含む	N/A
	testdata.c	テストプリント文字列および bitmap を含む	N/A
	testdata.h	テストプリント文字列および bitmap のグローバル変数を含む	N/A

6 MCU タイプの設定

TPDK プロジェクトは FM PDL プロジェクトを基に発展させたものです。FM PDL は MCU シリーズとパッケージを基に MCU システムパラメータを設定できます。

6.1 MCU デバイスの選択

手順:

1. FM3 ライブラリプロジェクト内のパッケージとデバイスシリーズについて、MCU チップタイプを確認します。
2. MCU タイプに応じて、MCU システムとペリフェラル機器を設定します。

例:

1. MCU タイプに従ってデバイスシリーズとパッケージを探します。図 3 は、FM3 ライブラリがサポートするすべてのシリーズとパッケージを示します。(ファイルパス: driver ¥ pdl.h)

   MB9AF312K のデバイスシリーズは "DEVICE_SERIES_MB9AF31X" です。
   MB9AF312K のデバイスパッケージは "DEVICE_PACKAGE_K" です。

図 3. FM3 デバイスマクロ定義

```
/**
 ******************************************************************************
 ** Global User Package Choice List
******************************************************************************/
#define DEVICE_PACKAGE_K      10
#define DEVICE_PACKAGE_L      20
#define DEVICE_PACKAGE_M      30
#define DEVICE_PACKAGE_N      40
#define DEVICE_PACKAGE_R      50
#define DEVICE_PACKAGE_S      60
#define DEVICE_PACKAGE_T      70
/**
 ******************************************************************************
 ** Device series definitions
******************************************************************************/
#define DEVICE_SERIES_MB9AF10X      0
#define DEVICE_SERIES_MB9AF11X      10
#define DEVICE_SERIES_MB9AF13X      20
#define DEVICE_SERIES_MB9AF14X      25
#define DEVICE_SERIES_MB9AF15X      26
#define DEVICE_SERIES_MB9AF31X      30
#define DEVICE_SERIES_MB9AF34X      33
#define DEVICE_SERIES_MB9AFA3X      34
#define DEVICE_SERIES_MB9AFA4X      36
#define DEVICE_SERIES_MB9AFB4X      37
#define DEVICE_SERIES_MB9BF10X      40
#define DEVICE_SERIES_MB9BF11X      45
#define DEVICE_SERIES_MB9BF12X      46
#define DEVICE_SERIES_MB9BF21X      55
#define DEVICE_SERIES_MB9BF30X      60
#define DEVICE_SERIES_MB9BF31X      65
#define DEVICE_SERIES_MB9BF32X      66
#define DEVICE_SERIES_MB9BF40X      70
#define DEVICE_SERIES_MB9BF41X      75
#define DEVICE_SERIES_MB9BF50X      80
#define DEVICE_SERIES_MB9BF51X      85
#define DEVICE_SERIES_MB9BF52X      86
#define DEVICE_SERIES_MB9BF61X      100
#define DEVICE_SERIES_MB9BFD1X      110
```

2. デバイスシリーズとデバイスパッケージ用マクロ定義を修正します。(ファイルパス: driver ¥ pdl_device.h)

図 4. MB9AF312K デバイスマクロ定義

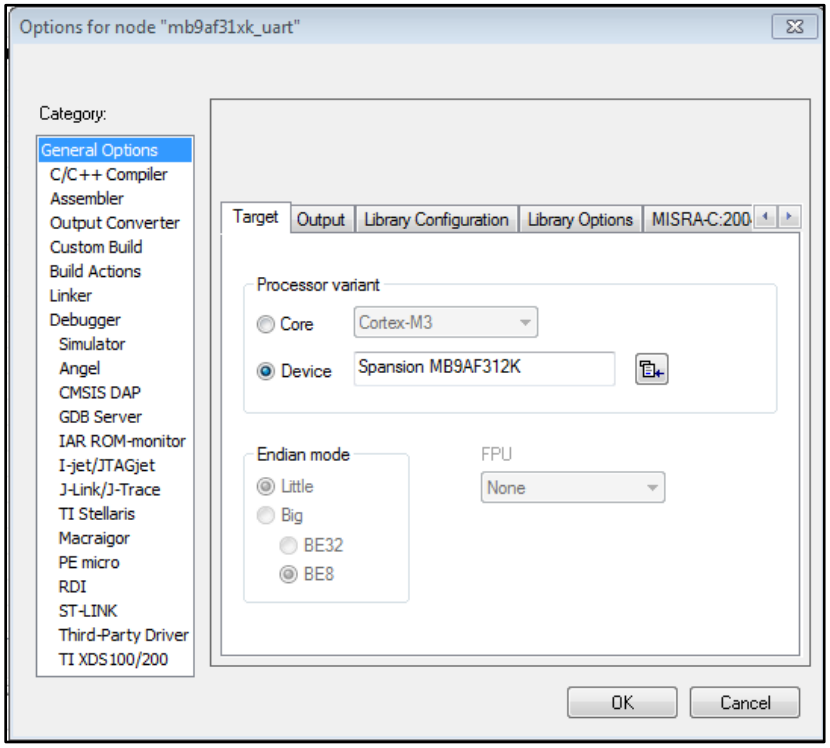
```
/*!
 ******************************************************************************
 ** \brief Choose device series [User configuration]
 ******************************************************************************
 */
#define MCU_SERIES        DEVICE_SERIES_MB9AF31X
/*!
 ******************************************************************************
 ** \brief Choose device package [User configuration]
 ******************************************************************************
 */
#define MCU_PACKAGE       DEVICE_PACKAGE_K
```

6.2 IAR の設定

TPDK 用 IAR プロジェクトを開き、IAR オプションを設定します。

6.2.1 Processor Variant の設定

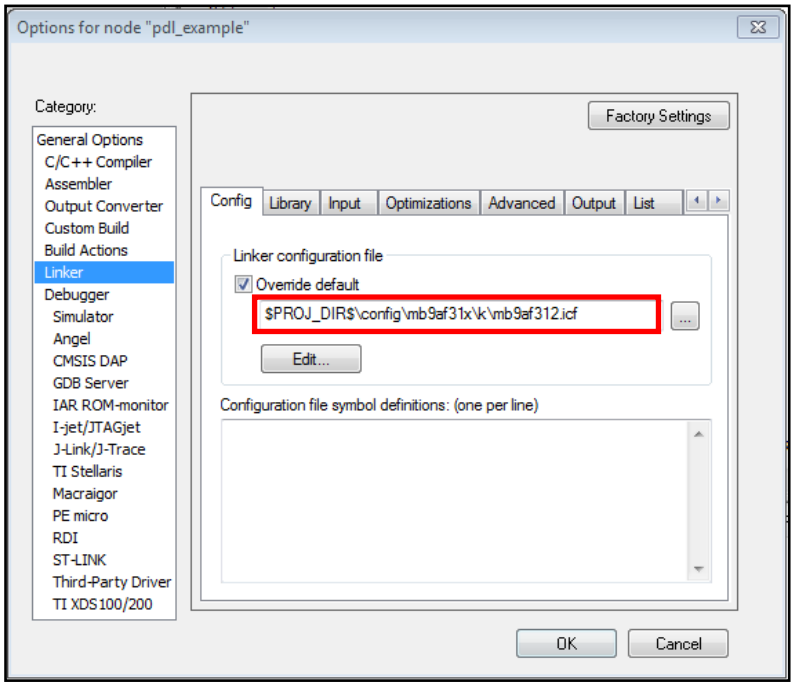
図 5. Processor Variant の設定



6.2.2 Linker Configuration File の設定

リンカオプション内の mb9af312.icf ファイルを選択します。(ファイルパス: ..¥ tpdk project¥ IAR¥ config)

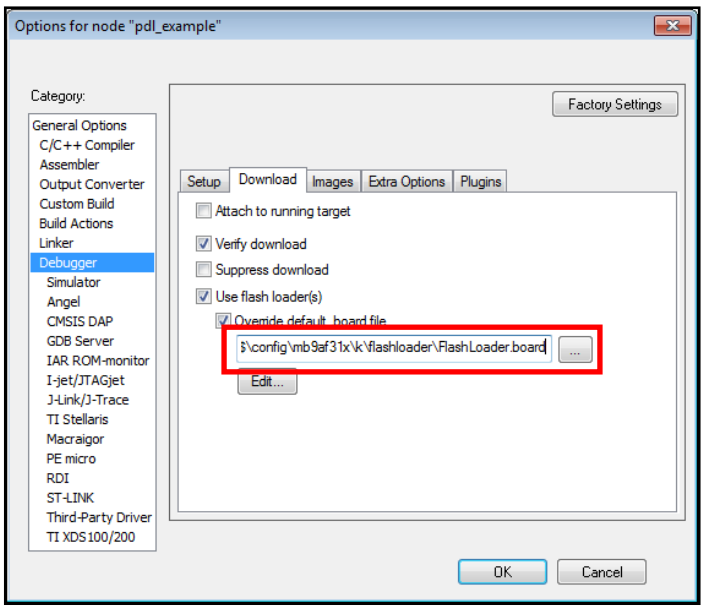
図 6. Linker Configuration File の設定



6.2.3 Flash Loader の設定

デバッグオプション内で該当する flash loader ファイルを選択します。(ファイルパス: ..¥ tpdk project¥ IAR¥ config)

図 7. Flash loader File の設定

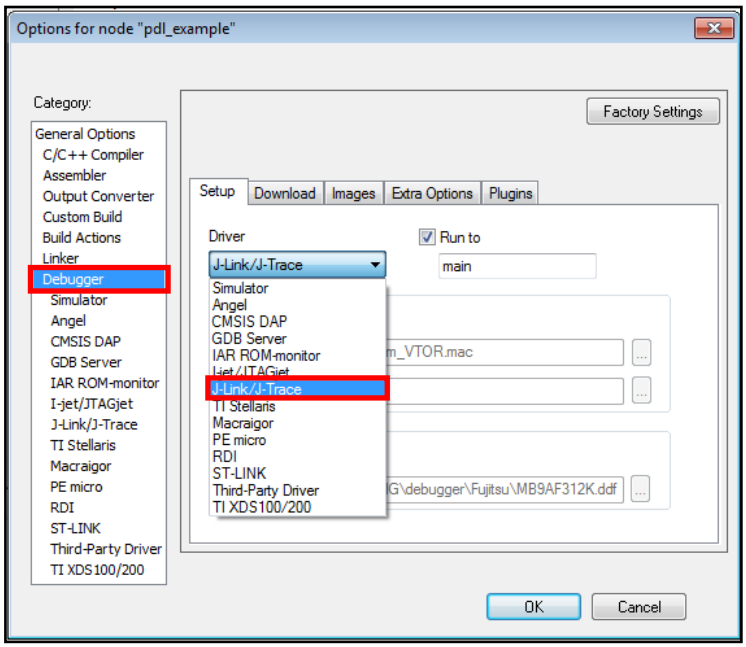


6.2.4 IAR デバッガ用ドライバの選択

TPDK ボードは、2 つのデバッグモードをサポートします。

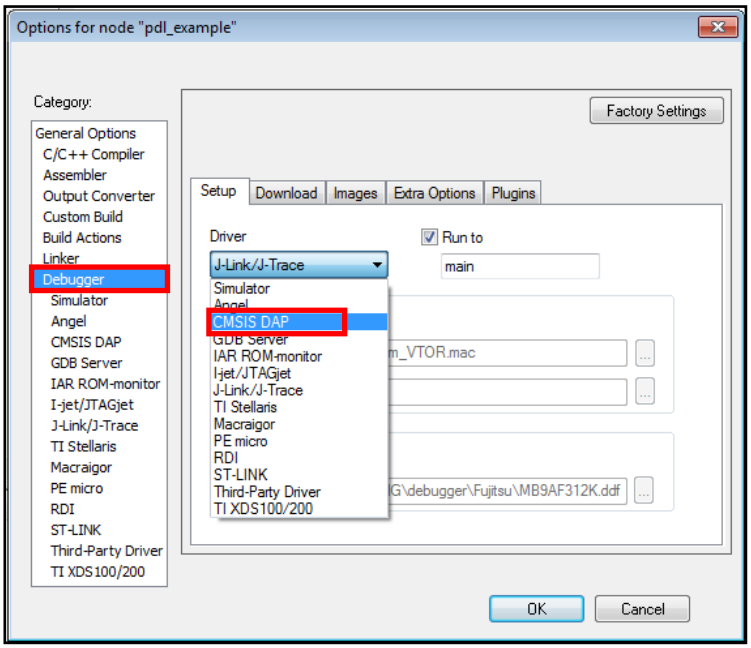
1. SWD デバッガドライバ

図 8. J-Link/J-Trace の設定



2. DAP デバッガドライバ

図 9. CMSIS_DAP の設定



7 MCU クロックの設定

FM PDL プロジェクトは MCU タイプ設定後にシステムクロックを設定できます。

クロック設定の手順:

1. 6.1 に従って MCU タイプを設定します。
2. マスタクロックソースを確認します。
   （ベースクロック，ベースクロック，APB0 クロック，APB1 クロック，APB2 クロック周波数）
3. 手順 2 の結果に従ってクロックパラメータとマクロ定義を修正します。

＜注意事項＞

- MB9AF312K クロックの詳細情報は FM3 ファミリ ペリフェラルマニュアル (MN706-00002) の CHAPTER 2-1: Clock を参照してください。
- プリンタ性能は MCU クロックに応じて変わります。

例

手順 1: 6.1 に従って MCU TYPE 5 を確認します。

手順 2: メイン PLL クロックはマスタクロックとして使用します。

ベースクロック周波数:	40 MHz.
APB0, APB1 クロック周波数:	20 MHz.
APB2 クロック周波数:	40 MHz.

手順 3: マクロ定義を修正します。(ファイルパス: com ¥ system_mb9xfxxx.h)

図 10. クロックの設定

```
#define __CLKMO          ( 4000000UL)          // External   4MHz Crystal
#define __CLKSO          (   32768UL)          // External  32KHz Crystal
#define __CLKHC          ( 4000000UL)          // Internal   4MHz CR Oscillator
#define __CLKLC          (  100000UL)          // Internal 100KHz CR Oscillator
#define SCM_CTL_Val             0x00000052     // SCM_CTL
#define BSC_PSR_Val             0x00000000     // BSC_PSR
#define APBC0_PSR_Val           0x00000001     // APBC0_PSR
#define APBC1_PSR_Val           0x00000081     // APBC1_PSR
#define APBC2_PSR_Val           0x00000080     // APBC2_PSR
#define SWC_PSR_Val             0x00000003     // SWC_PSR
#define TTC_PSR_Val             0x00000000     // TTC_PSR
#define CSW_TMR_Val             0x0000005C     // CSW_TMR
#define PSW_TMR_Val             0x00000000     // PSW_TMR
#define PLL_CTL1_Val            0x00000004     // K=1, M=5
#define PLL_CTL2_Val            0x00000009     // N=10
#define __PLLK          (((PLL_CTL1_Val >> 4) & 0x0F) + 1)
#define __PLLN          (((PLL_CTL2_Val     ) & 0x3F) + 1)
```

8 アプリケーション層の移植

新しいボードにプロジェクトを移植する場合、パラメータやマクロ定義はハードウェアやペリフェラルデバイスに合わせて修正します。例えば、UART, キー端子, FRT, デュアルタイマ, プリントヘッド端子などです。

8.1 UART の設定

他の MCU や UART チャネルを選択した場合、MFS チャネルと UART 端子パラメータとマクロ定義を修正します。

UART 設定手順:

1. MFS チャネルと UART 端子を確認します。
2. UART FIFO サイズを確認します。
3. MFS チャネルと UART 端子パラメータとマクロ定義を修正します。
4. UART モードと FIFO パラメータとマクロ定義を修正します。

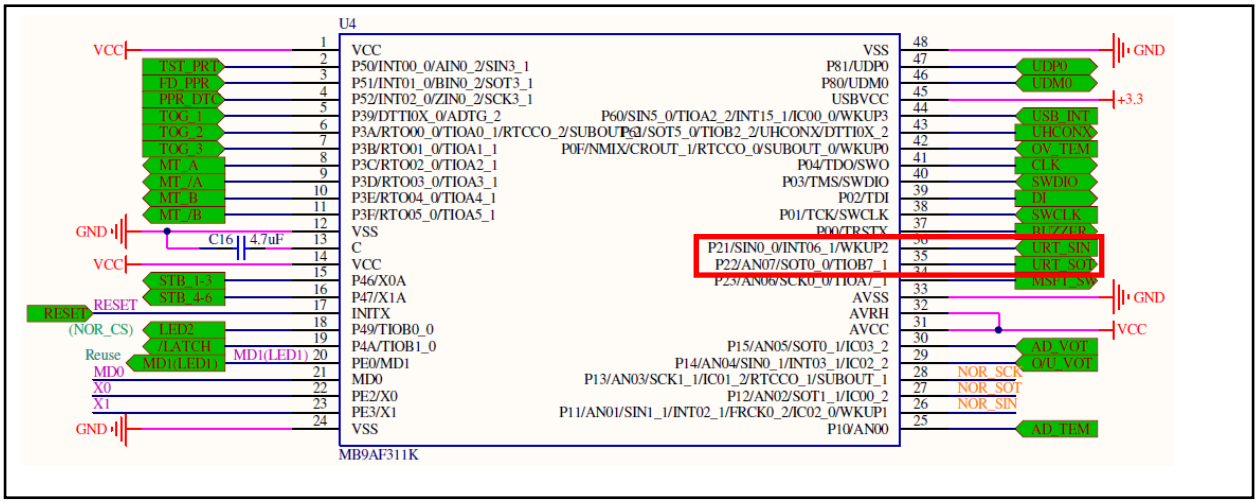
注意 1: UART 端子が特殊 I/O ポートの場合、その端子を設定する前に特殊機能をディゼーブルにします。

注意 2: FIFO 無しの MFS チャネルまたは、FIFO を使用しない場合、マクロ定義 "#define UART_FIFO" を削除します。

例

手順 1: 図 11 は UART モジュールが MFS channel 0 に該当する UART 端子として P21, P22 を選択していることを示します。

図 11. UART 回路



手順 2: MB9AF312K の MFS チャネル 0 の FIFO サイズは "MB9A310K シリーズ データシート (DS706-00029)" 内に記載のとおり 16 バイトです。

手順 3: 手順 1 の結果として図 12 に示される MFS チャネルと UART 端子パラメータを修正します。(ファイルパス: app ¥ communication ¥ uart. h)

図 12. UART 端子とチャネルの設定

```
/*****************************************************************************
*/
/* Global pre-processor symbols/macros('#define')
*/
/*****************************************************************************
*/
#define   MFS_UART_CH           (MFS_Ch0)

/*! \brief IO MFS channel */
#define   IO_MFS_CH             (MFS_UART_CH)

/*! \brief IO MFS port */
#define   IO_MFS_PORT           (IO_PORT2)

/*! \brief IO MFS SOT pin */
#define   IO_MFS_SOT_PIN        (IO_PINx2)

/*! \brief IO MFS SIN pin */
#define   IO_MFS_SIN_PIN        (IO_PINx1)

/*! \brief IO MFS SOT pin location */
#define   IO_MFS_SOT_PIN_LOC    (IO_MFS_SOTx_SOTx_0)

/*! \brief IO MFS SIN pin location */
#define   IO_MFS_SIN_PIN_LOC    (IO_MFS_SINx_SINx_0)
```

手順 4: P21 は、AD 機能を持つ特殊な I/O ポートです。図 13 のように、特殊機能をディゼーブルにします。(ファイルパス: app ¥ communication ¥ uart.c).

図 13. UART 端子機能の修正

```
static void UartPortInitial(void)
{
    IO_DisableAnalogInput(IO_AN07);
    /* Enable SOT and SIN */
    IO_EnableFunc(IO_MFS_PORT, IO_MFS_SOT_PIN);
    IO_EnableFunc(IO_MFS_PORT, IO_MFS_SIN_PIN);
    IO_ConfigFuncMFSSOTxPin(IO_MFS_CH, IO_MFS_SOT_PIN_LOC);
    IO_ConfigFuncMFSSINxPin(IO_MFS_CH, IO_MFS_SIN_PIN_LOC);

    return;
}
```

手順 5: 手順 2 の結果に従って、図 14 のようにモードと FIFO を設定します。(ファイルパス: app ¥ communication ¥ uart.c).

図 14. UART モードと FIFO の設定

```
/*****************************************************************************/
/*Local pre-processor symbols/macros('#define')                              */
/*****************************************************************************/
#define UART_FIFO

#ifdef UART_RX_FIFO
#define UART_RX_FIFO_SIZE    12
#define UART_TX_FIFO_SIZE    1
#endif

static MFS_UARTModeConfigT stcUARTModeConfig =
{
    115200,              ///< Baudrate 19200
    UART_DATABITS_8,     ///< Data bit length
    UART_STOPBITS_1,     ///< Stop bit length
    UART_PARITY_NONE,    ///< Parity
    UART_BITORDER_LSB,   ///< Bit order
    UART_NRZ,            ///< Signal NRZ
};

#ifdef UART_RX_FIFO
static MFS_UARTFIFOConfigT stcUARTFIFOConfig =
{
    MFS_FIFO1TX_FIFO2RX,  ///< Transmit FIFO:FIFO1; Receive FIFO:FIFO2
    UART_TX_FIFO_SIZE,    ///< Transmit FIFO size
    UART_RX_FIFO_SIZE,    ///< Receive FIFO size
};
#endif
```

8.2 ADC の設定

他の MCU や ADC チャネルを選択した場合、ADC ユニット, チャネル, 端子パラメータを修正します。

ADC 設定の手順:

1. ADC ユニット, チャネル, 端子を確認します。
2. ADC パラメータとマクロ定義を修正します。

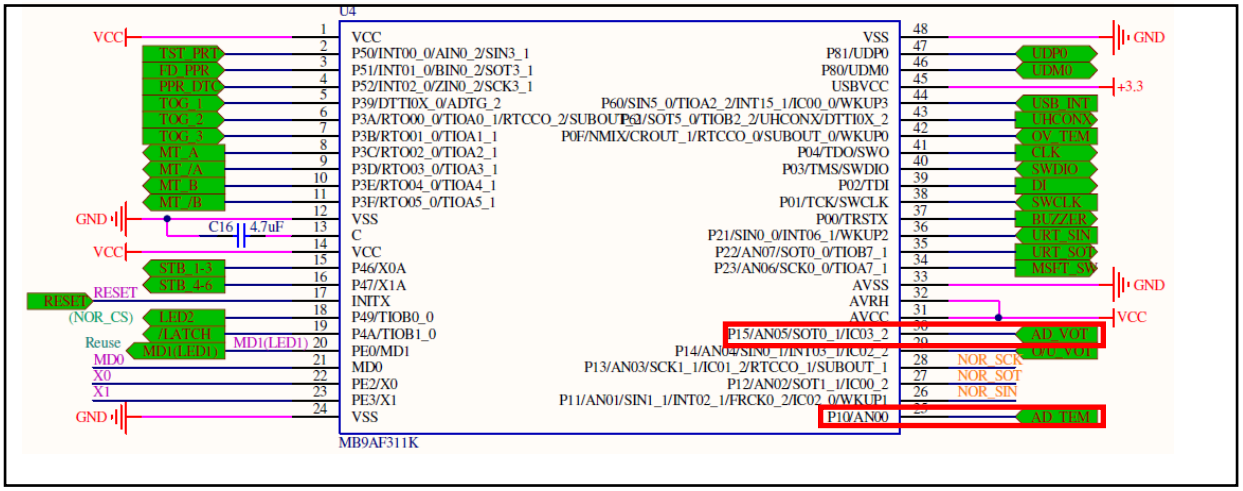
例

手順 1: 図 15 は ADC モジュールが MFS チャネル 5 とチャネル 0 に該当する ADC 端子として P10/AN05 と P15/AN00 を選択していることを示します。

"MB9A310K シリーズ データシート (DS706-00029)" 内に記載のとおり 2 つの ADC ユニットがあります。

ファームウェアデザインに従って ADC ユニット 0 とチャネル 0 をサンプル温度に選択し、ADC ユニット 1 とチャネル 5 をサンプル電圧に選択します。

図 15. 電圧と温度サンプル回路



手順 2: 手順 1 の結果に従って図 16 と図 17 のように ADC ユニットとチャネルパラメータを修正します。(ファイルパス: app¥ printer_header_control¥ adc.h)

図 16. ADC ユニットとチャネルの設定

```
/*! \brief ADC unit for temperature */
#define TEMP_ADC_UNIT                   ADC12_UNIT0
#define TEMP_ADC_SCAN_CH                ADC12_CH0
/*! \brief ADC unit for voltage */
#define VOLT_ADC_UNIT                   ADC12_UNIT1
#define VOLT_ADC_SCAN_CH                ADC12_CH5
```

図 17. ADC 端子の設定

```
/*-------------------- AD VOT  I/O -------------- */
#define AD_VOLT_IO_AN        IO_AN05
```

8.3 プリントヘッド端子の設定

他の MCU や GPIO を選択した場合、プリントヘッド端子のパラメータを修正します。

1. プリントヘッドを制御するために使用するすべての端子を確認します。
2. 端子パラメータとマクロ定義を修正します。
3. 初期化関数を修正します。

注意事項：端子が特殊 I/O ポートの場合、それらの端子を設定する前に、特殊機能をディゼーブルにします。

例

手順 1: 図 18 はプリンタヘッド端子を示し、表 5 は端子関数を示します。

図 18. プリントヘッド端子

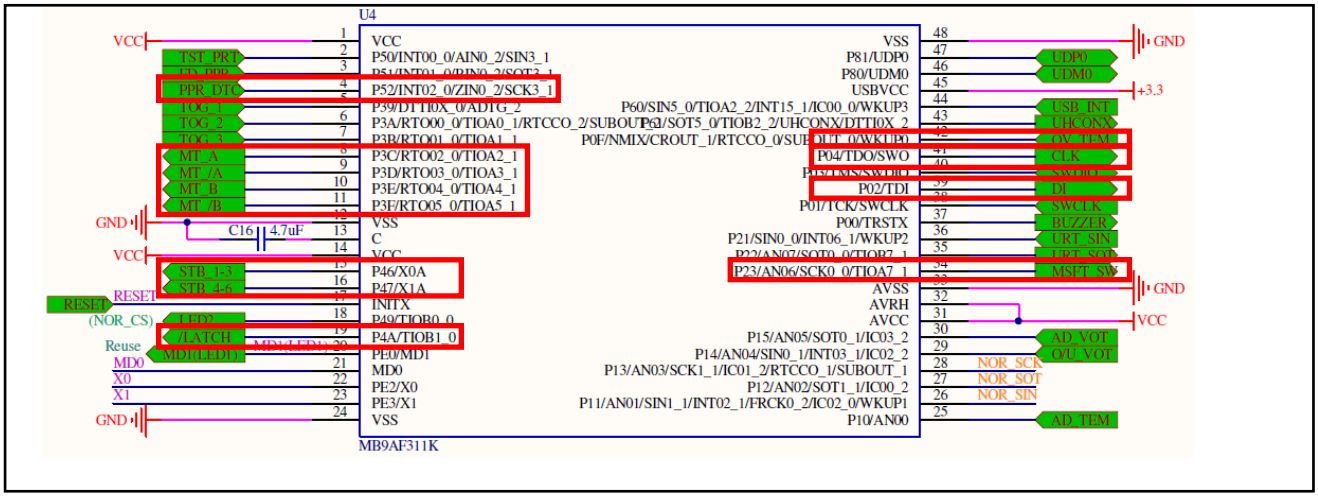


表 5. プリントヘッド端子機能の制御

端子	機能
P52	紙の検出
P3C	モーターA
P3D	モーターNA
P3E	モーターB
P3F	モーターNB
P46	stb1, stb2, stb3 の制御
P47	stb4, stb5, stb6 の制御
P0F	オーバーヒートの検出
P04	CLK
P02	データ入力
P23	MOSFET 回路の制御

手順 2: 手順 1 の結果に従って図 19 と同様に端子パラメータを修正します。(ファイルパス: app ¥ printer_header_control ¥ printheadpin.h)

図 19. プリントヘッド端子の設定

```
#define MSFT_IO_PORT    IO_PORT2
#define MSFT_IO_PIN     IO_PINx3
/*-------------------- STB  I/O -------------- */
#define STB_IO_PORT     IO_PORT4
#define STB13_IO_PIN    IO_PINx6
#define STB46_IO_PIN    IO_PINx7
/*-------------------- CLK  I/O -------------- */
#define CLK_IO_PORT     IO_PORT0
#define CLK_IO_PIN      IO_PINx4
/*-------------------- DATA  I/O -------------- */
#define DATA_IO_PORT    IO_PORT0
#define DATA_IO_PIN     IO_PINx2
/*-------------------- LATCH  I/O -------------- */
#define NLATCH_IO_PORT   IO_PORT4
#define NLATCH_IO_PIN    IO_PINxA
/*-------------------- MOTOR  I/O -------------- */
#define MOTOR_IO_PORT       IO_PORT3
#define MOTOR_IO_PIN_A      IO_PINxC
#define MOTOR_IO_PIN_NA     IO_PINxD
#define MOTOR_IO_PIN_B      IO_PINxE
#define MOTOR_IO_PIN_NB     IO_PINxF
/*-------------------- PAPER INTERRUPT -------------- */
#define  PAPER_IO_PORT   IO_PORT5
#define  PAPER_IO_PIN    IO_PINx2
#define  PAPER_EXT_INT_IO_CH            IO_EXT_INT_CH2
#define  PAPER_EXT_INT_IO_CH_LOC        IO_INTx_INTx_0
#define  PAPER_EXT_INT_CH               EXTI_CH2
#define  PAPER_EXT_INT_DETECT_MODE      EXTI_LEVEL_HIGH_DETECT
/*-------------------- OUV INTERRUPT -------------- */
#define  OUV_IO_PORT   IO_PORT1
#define  OUV_IO_PIN    IO_PINx4
#define  OUV_EXT_INT_IO_CH          IO_EXT_INT_CH3
#define  OUV_EXT_INT_IO_CH_LOC      IO_INTx_INTx_1
#define  OUV_EXT_INT_CH             EXTI_CH3
#define  OUV_EXT_INT_DETECT_MODE    EXTI_LEVEL_LOW_DETECT
/*-------------------- OVERHAET INTERRUPT -------------- */
#define  OVERHAET_IO_PORT   IO_PORT0
#define  OVERHAET_IO_PIN    IO_PINxF
```

手順 3: 機能"void PHC_PrintDriverInit(void) "を修正します。(ファイルパス:app ¥ printer_header_control ¥ printheadpin.c)

加熱制御のために P46 と P47 を選択します。MB9AF312K の P46 および P47 にはサブクロック機能があります。GPIO 機能の設定の前に、サブクロック機能をディゼーブルにします。

図 20. STB 端子の設定

```
/*-------------------- STB I/O  Initial-------------- */
    IO_ConfigFuncSClkPin(IO_SCLK_X0AX1A_GPIO);
    IO_DisableFunc(STB_IO_PORT, STB13_IO_PIN);
    IO_DisableFunc(STB_IO_PORT, STB46_IO_PIN);
    IO_ConfigGPIOPin(STB_IO_PORT, STB13_IO_PIN, IO_DIR_OUTPUT, IO_PULLUP_OFF);
    IO_ConfigGPIOPin(STB_IO_PORT, STB46_IO_PIN, IO_DIR_OUTPUT, IO_PULLUP_OFF);
    PHC_HeatStbAllOff();/*-------------------- AD TMP  I/O -------------- */
#define AD_TMP_IO_AN        IO_AN00
```

MOSFET 回路の制御のために、P23 を設定します。MB9AF312K の P23 には、ADC 端子機能があります。GPIO 機能の設定の前に、ADC 端子機能をディゼーブルにします。

図 21. MOSFET 端子の設定

```
/*-------------------- MSFT  I/O  Initial-------------- */
    IO_DisableAnalogInput(IO_AN06);
    IO_DisableFunc(MSFT_IO_PORT, MSFT_IO_PIN);
    IO_ConfigGPIOPin(MSFT_IO_PORT, MSFT_IO_PIN, IO_DIR_OUTPUT, IO_PULLUP_OFF);
```

プリントヘッド電圧検知の制御のために、P14 を設定します。MB9AF312K の P14 には、ADC 端子機能があります。割込み機能の設定の前に、ADC 端子機能をディゼーブルにします。

図 22. オーバ/アンダ電圧割込み端子の設定

```
    /*-------------------- OUV  INTERRUPT -------------- */
    IO_DisableAnalogInput(IO_AN04);
    IO_EnableFunc(OUV_IO_PORT, OUV_IO_PIN);
    IO_ConfigFuncINTxPin(OUV_EXT_INT_IO_CH, OUV_EXT_INT_IO_CH_LOC);
    IO_ConfigGPIOPin(OUV_IO_PORT, OUV_IO_PIN, IO_DIR_INPUT, IO_PULLUP_ON);
    EXTI_SetIntDetectMode(OUV_EXT_INT_CH, OUV_EXT_INT_DETECT_MODE);
    EXTI_EnableInt(OUV_EXT_INT_CH, PHC_OuvIsr);
    EXTI_ClrIntFlag(OUV_EXT_INT_IO_CH);
```

8.4 キー端子の設定

他の MCU を選択した場合、キー端子パラメータを修正します。

キー端子の設定手順:

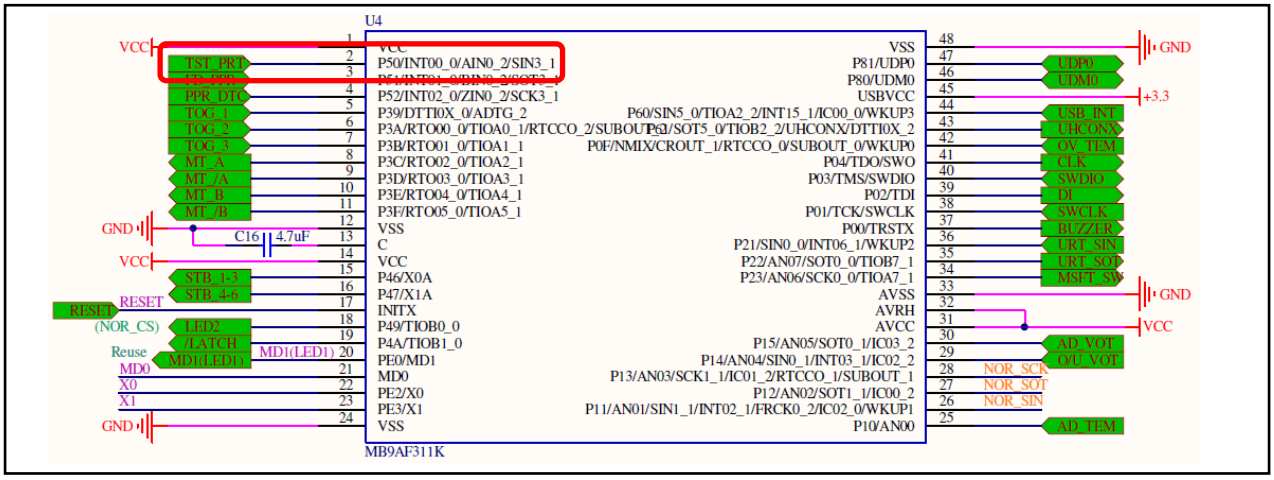
1. キー端子を確認します。
2. 端子パラメータとマクロ定義を修正します。

注意事項: 端子が特殊 I/O ポートの場合、それらの端子を設定する前に、特殊機能をディゼーブルにします。

例

手順 1: 図 23 は P50 と P51 端子がテストキーや給紙キーとして選択されていることを示します。実装用の外部割込み関数を選択します。

図 23. テストキー，給紙キーの回路



手順 2: INT00_0 と INT01_0 端子は外部割込みチャネル 0 とチャネル 1 に該当します。手順 1 の結果に従って、図 24 と同様に端子パラメータを修正します。(ファイルパス: app \ misc \ misc.h)

図 24. テストキー、給紙キー端子の設定

```
/*-------------------- TEST KEY INTERRUPT -------------- */
#define  TEST_KEY_INT_IO_PORT    IO_PORT5
#define  TEST_KEY_INT_IO_PIN     IO_PINx0
#define  TEST_KEY_EXT_INT_IO_CH            IO_EXT_INT_CH0
#define  TEST_KEY_EXT_INT_IO_CH_LOC        IO_INTx_INTx_0
#define  TEST_KEY_EXT_INT_CH               EXTI_CH0
#define  TEST_KEY_EXT_INT_DETECT_MODE      EXTI_EDGE_FALLING

/*-------------------- FEED PAPER KEY INTERRUPT -------------- */
#define  FEED_PAPER_KEY_INT_IO_PORT    IO_PORT5
#define  FEED_PAPER_KEY_INT_IO_PIN     IO_PINx1
#define  FEED_PAPER_KEY_EXT_INT_IO_CH            IO_EXT_INT_CH1
#define  FEED_PAPER_KEY_EXT_INT_IO_CH_LOC        IO_INTx_INTx_0
#define  FEED_PAPER_KEY_EXT_INT_CH               EXTI_CH1
#define  FEED_PAPER_KEY_EXT_INT_DETECT_MODE      EXTI_EDGE_FALLING
```

8.5 LED1 と LED2 端子の設定

他の MCU を選択した場合、LED 端子パラメータを修正します。

LED 端子の設定手順:

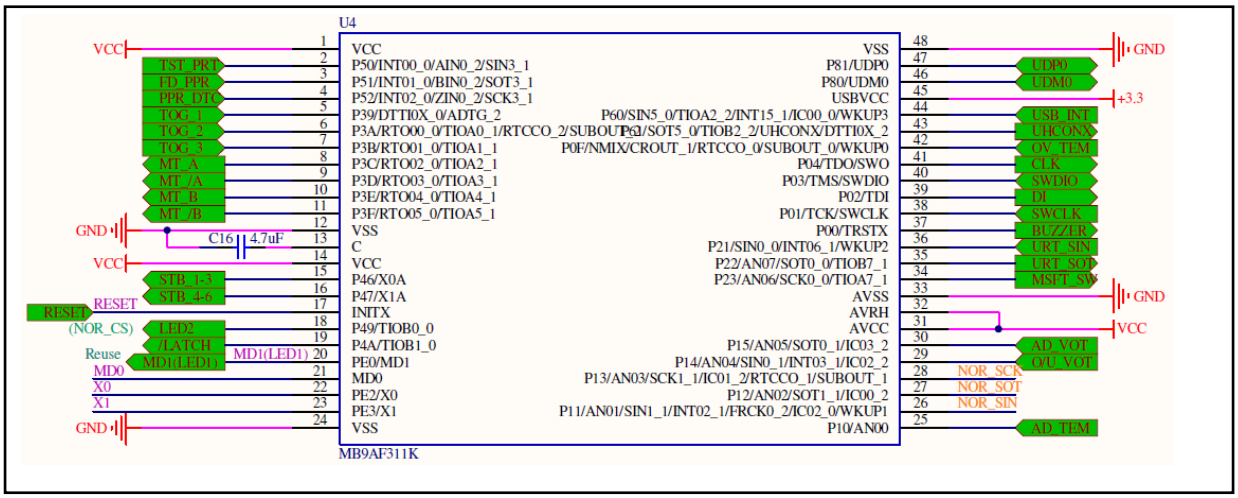
1. LED 端子を確認します。
2. 端子パラメータとマクロ定義を修正します。

注意事項: 端子が特殊 I/O ポートの場合、それらの端子を設定する前に、特殊機能をディゼーブルにします。

例

手順 1: 図 25 は P49 と PE0 端子が LED2 と LED1 を制御するために選択されていることを示します。

図 25. LED1 または LED2 回路



手順 2: 手順 1 の結果に従って図 26 と同様に端子パラメータを修正します。(ファイルパス: app ¥ misc ¥ misc.h)

図 26. LED1 または LED2 端子の設定

```
/*-------------------- LED -------------- */
#define  LED1_IO_PORT IO_PORTE
#define  LED1_IO_PIN  IO_PINx0

#define  LED2_IO_PORT IO_PORT4
#define  LED2_IO_PIN  IO_PINx9
```

8.6 ブザー端子の設定

他の MCU を選択した場合、ブザー端子パラメータを修正します。

ブザー端子の設定手順:

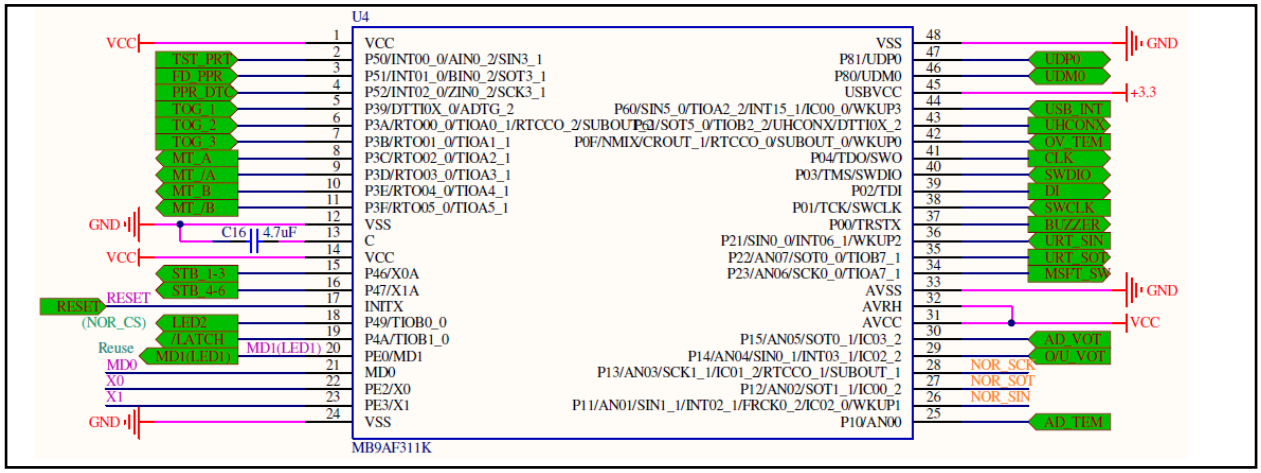
1. ブザー端子を確認します。
2. 端子パラメータとマクロ定義を修正します。

注意事項: 端子が特殊 I/O ポートの場合、それらの端子を設定する前に、特殊機能をディゼーブルにします。

例

手順 1: 図 27 は P00 端子がブザーを制御するために選択されていることを示します。

図 27. ブザー回路



手順 2: 手順 1 の結果に従って図 28 と同様に端子パラメータを修正します。(ファイルパス: app ¥ misc ¥ misc.h)

図 28. ブザー端子の設定

```
/*-------------------- BEEP -------------- */
#define  BEEP_IO_PORT  IO_PORT0
#define  BEEP_IO_PIN   IO_PINx0
```

8.7 トグルスイッチ端子の設定

他の MCU を選択した場合、トグルスイッチ端子パラメータを修正します。

トグルスイッチ端子の設定手順:

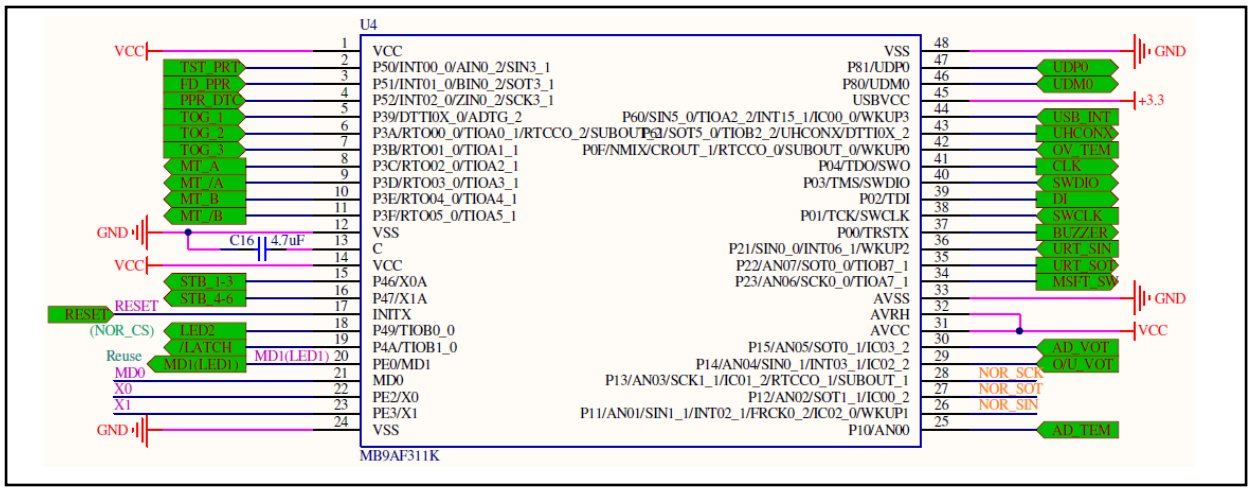
1. トグルスイッチ端子を確認します。
2. 端子パラメータとマクロ定義を修正します。

注意事項: 端子が特殊 I/O ポートの場合、それらの端子を設定する前に、特殊機能をディゼーブルにします。

例

手順 1: 図 29 は P39, P3A, P3B 端子がトグルスイッチ端子として選択されていることを示します。

図 29. トグルスイッチ回路



手順 2: 手順 1 の結果に従って図 30 と同様に端子パラメータを修正します。(ファイルパス: app \ misc \ misc.h)

図 30. トグルスイッチ端子の設定

```
/*-------------------- TOG SWITCH -------------- */
#define  TOG_1_IO_PORT IO_PORT3
#define  TOG_1_IO_PIN  IO_PINx9

#define  TOG_2_IO_PORT IO_PORT3
#define  TOG_2_IO_PIN  IO_PINxA

#define  TOG_3_IO_PORT IO_PORT3
#define  TOG_3_IO_PIN  IO_PINxB
```

8.8 FRT タイマの設定

他の MCU や FRT タイマを選択した場合、FRT タイマのパラメータを修正します。

FRT の設定手順:

1. FRT バスクロックと周波数を確認します。
2. FRT タイマユニットとチャネルを確認します。
3. カウントサイクルを確認します。
4. 上記結果に従って FRT マクロ定義とパラメータを修正します。

注意: 他の MCU を使用したりシステムクロック設定を修正した場合、APB0, APB1, APB2 クロックが異なります。カウント値を再計算し、FRT パラメータを再設定してください。

例

手順 1: MB9AF312K の MFT を "MB9A310K シリーズ データシート (DS706-00029)" にある APB1 に接続します。

システムが起動すると "void SystemInit (void)" 関数により、ベースクロック，APB0 バスクロック，APB1 バスクロック，APB2 バスクロックを初期化します。この関数は APB1 クロック周波数を 20 MHz に設定します。(ファイルパス: common \ system_mb9xfxxx.c)

手順 2: MB9AF312K は "MB9A310K シリーズ データシート(DS706-00029)" にある 16-bit フリーランタイマ x 3ch を持っています。タイムアウトのカウント用に FRT channel0 を選択し、テストキー用に channel1 を選択し、給紙キー用に channel2 を選択します。

手順 3: アップカウントモード内で MFT のソースクロックとして APB1 を設定します。

FRT カウントクロックサイクル = APB0 サイクル × 倍数

FRT カウントサイクル = カウント値 × FRT カウントクロックサイクル

従って、タイムアウト FRT 倍数値は "32"

タイムアウトインターバルサイクルは 1 ms

タイムアウト FRT カウントクロックサイクル = 32 / 20 MHz

1ms = カウント値 × (32 / 20 MHz)

カウント値 = 625

テストキーFRT 倍数は "32"

タイムアウトインターバルサイクルは 10 ms

タイムアウト FRT カウントクロックサイクル = 32 / 20 MHz

10ms = Count value * (32 / 20 MHz)

Count value = 6250

給紙キーFRT 倍数は "32"

タイムアウトインターバルサイクルは 10 ms

タイムアウト FRT カウントクロックサイクル = 32 / 20 MHz

10ms = Count value * (32 / 20 MHz)

Count value = 6250

手順 4: 手順 1 の結果に従って図 31 と同様に FRT パラメータを修正します。(ファイルパス: app ¥ misc ¥ timer.h)

図 31. FRT の設定

```
/*Configure FRT timer for timeout count */
#define TIME_OUT_FRT_UNIT    0      // FRT unit
#define TIME_OUT_FRT_CH      0      // FRT channel
#define TIME_OUT_CNT_CYCLE   625    //count value: 1ms
#define TIME_OUT_SRC_CLK     FRT_SRC_CLK_PCLK
#define TIME_OUT_CLK_DIV     FRT_DIV32
#define TIME_OUT_CNT_MODE    CNT_MODE_UP

/*Configure FRT timer for test key */
#define TEST_FRT_UNIT    0      // FRT unit
#define TEST_FRT_CH      1      // FRT channel
#define TEST_CNT_CYCLE   6250   //count value: 10ms
#define TEST_SRC_CLK     FRT_SRC_CLK_PCLK
#define TEST_CLK_DIV     FRT_DIV32
#define TEST_CNT_MODE    CNT_MODE_UP

/*Configure FRT timer for feed paper key */
#define FEED_FRT_UNIT    0      // FRT unit
#define FEED_FRT_CH      2      // FRT channel
#define FEED_CNT_CYCLE   6250   //count value: 10ms
#define FEED_SRC_CLK     FRT_SRC_CLK_PCLK
#define FEED_CLK_DIV     FRT_DIV32
#define FEED_CNT_MODE    CNT_MODE_UP
```

8.9 ソフトウェアウォッチドッグの設定

他の MCU を選択した場合、ソフトウェアウォッチドッグタイマのパラメータを修正します。

ソフトウェアウォッチドッグの設定手順:

1. ソフトウェアウォッチドッグのバスクロックと周波数を確認します。
2. ソフトウェアウォッチドッグのカウントサイクル値を計算します。
3. 上記の結果に従ってソフトウェアウォッチドッグのマクロ定義とパラメータを修正します。

例

手順 1: MB9AF312K のソフトウェアウォッチドッグを "MB9A310K シリーズ データシート (DS706-00029)" にある APB0 に接続します。

システムが起動すると "void SystemInit (void)" により、ベースクロック, APB0 クロック, APB1 クロック, APB2 クロック, ソフトウェアウォッチドッグクロックを初期化します。この関数は APB0 クロック周波数を 20 MHz に設定し、ウォッチドッグクロックを 2.5 MHz に設定します。(ファイルパス: common \ system_mb9xfxxx.c)

手順 2: 最後のサービスから 2 秒未満にウォッチドッグタイマがサービスされない場合、システムをリセットしてください。なぜならリセットは第 2 のアンダーフローを生成し、ウォッチドッグインターバルタイマは 1 s になります。

ウォッチドッグインターバル時間 = ウォッチドッグカウント × SW_CLKFREQ

1 s = ウォッチドッグカウント×2.5 MHz

ウォッチドッグカウント = 2500000

手順 3: 手順 2 の結果に従って図 32 と同様にソフトウェアウォッチドッグのカウント値を修正します。(ファイルパス: app ¥ misc ¥ timer.h)

図 32. ソフトウェアウォッチドッグカウント値の設定

```
/*Software watchdog interval time*/
#define SOFT_WATCH_DOG_INTERVAL 2500000   //2500000 1s
```

9 ユーザコードの移植

他の MCU を選択した場合、TPDK ソリューションを満たすためにバッファサイズを修正してください。

例

手順 1: ヒープサイズを "0x0" に設定し、コンパイルしてください。

データサイズをリードライトするために Thermal Printer.map ファイルを解析します。(ファイルパス: (FWSC)Thermal_Printer_Development_Kit¥tpdk project¥IAR¥output¥debug¥list¥ Thermal Printer .map)

リードライトデータメモリサイズ = 2 677 バイト

これはグローバルデータやブロック CSTACK サイズと等しいです。

図 33. Thermal Printer.map

```
30 884 bytes of readonly  code memory
 8 116 bytes of readonly  data memory
 2 677 bytes of readwrite data memory
```

ブロック HEAP はプリントバッファ, フレームバッファ, 通信バッファ, ESC/POS バッファにメモリを配分します。プリントバッファ, フレームバッファ, 通信バッファサイズは表 6 のように固定されます。

表 6. バッファサイズ

バッファ	サイズ
プリントバッファ	6144
フレームバッファ	518 + 1 + 12
通信バッファ	512 + 1 + 12
計	7200

以下のとおりヒープサイズと ESC/POS バッファサイズを計算してください:

RAM サイズ – 2677 > ヒープサイズ > ESC/POS バッファサイズ + 7200 + 1 +12

16 × 1024 - 2677 > ヒープサイズ > ESC/POS バッファサイズ + 7213

13707 > ヒープサイズ > ESC/POS バッファサイズ + 7213

ヒープサイズを "12362" に設定してください

12362 > ESC/POS バッファサイズ + 7213

ESC/POS バッファサイズ < 5149;

TPDK プロジェクト内で ESC/POS バッファサイズを "5000" に設定してください。

注意 1: ROM/flash が 90K バイト未満の場合、マクロ定義 "#define TEST_PRINT_BITMAP" を削除してください。

注意 2: ESCPOS_BUFFER_SIZE の値は 500 以上でなければならず、異なる ESCPOS_BUFFER_SIZE の値はプリンタパフォーマンスに影響します。

手順 2: 手順 1 の結果に従って図 34 や図 35 と同様に IAR リンカオプションのヒープサイズや ESC/POS バッファサイズを修正してください。(ファイルパス: tpdk ¥ source ¥ main.c)

図 34. IAR リンカオプションのヒープサイズ設定

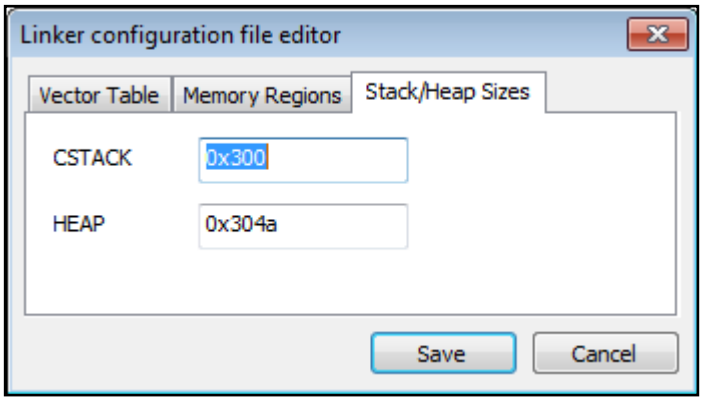


図 35. ESC/POS バッファの設定

```
#define ESCPOS_BUFFER_SIZE 5000
```

10 実行とデバッグ

移植後、TPDK IAR プロジェクトを開き、コンパイルして、実行します。

11 その他の情報

詳細な情報については、当社 Web サイトをご利用ください。

http://www.cypress.com/applications/consumer-electronics/microcontrollers-thermal-printers

技術的な質問に関しては、サポート窓口にお問い合わせください。

改訂履歴

文書名: AN205409 - FM3 Family MB9AF310 Series 感熱式プリンタ開発キット用ファームウェア移植

文書番号: 002-05410

版	ECN 番号	変更者	発行日	変更内容
**	-	NNAK	09/11/2015	サイプレスとしてスパンションアプリケーションノート MCU-AN-510125-J-11 をドキュメントコード 002-05410 に登録しました。 本版の内容およびフォーマットに変更はありません。
*A	5705463	NNAK	04/21/2017	これは英語版の 002-05409 Rev. *A を翻訳した日本語版です。

セールス，ソリューションおよび法律情報

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

製品

ARM® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス／RF	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

サイプレス開発者コミュニティ

フォーラム | WICED IOT Forums | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709