The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205409

FM3 Family MB9AF310 Series Thermal Printer Development Kit Migration

Associated Part Family: MB9AF310 Series

This application note describes the Thermal Printer Development Kit solution based on MB9AF312K, and describes how to port code to other type chip of FM3 series.

Contents

1 Introduction....1
1.1 Purpose....1
1.2 Definitions, Acronyms and Abbreviations....1
1.3 Document Overview....2
1.4 Reference Documents....2
2 MCU Peripheral Requirement....3
3 System Hardware Environment....3
3.1 MCU Information....3
4 Development Environment....4
5 System Firmware Design....4
5.1 FW Structure....4
6 Configure FM3 Chip....7
6.1 Select MCU Device....7
6.2 Configure IAR....10
7 Configure MCU Clock....13
8 Port App Layer....14
8.1 Configuration UART....14
8.2 Configuration ADC....17
8.3 Configure Print Head Pin....18
8.4 Configure Key Pin....21
8.5 Configure LED1 and LED2 Pin....23
8.6 Configure Buzzer Pin....24
8.7 Configure Toggle Switch Pin....25
8.8 Configure FRT Timer....26
8.9 Configure Software Watchdog....28
9 Port User Code....29
10 Run and Debug....30
11 Additional Information....31
Document History....32

1 Introduction

1.1 Purpose

This application note describes the Thermal Printer Development Kit solution based on MB9AF312K, and describes how to port code to other type chip of FM3 series.

1.2 Definitions, Acronyms and Abbreviations

For example:

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 Document Overview

The rest of document is organized as the following:

Section 2 explains MCU Peripheral Requirement.

Section 3 explains System Hardware Environment.

Section 4 explains Development Environment.

Section 5 explains System Firmware Design.

Section 6 explains Configure FM3 Chip.

Section 7 explains Configure MCU Clock.

Section 8 explains Port App Layer.

Section 9 explains Port User Code

Section 10 explains Run and Debug

1.4 Reference Documents

(SCH)TPDK-Main.pdf

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

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

MB9AF311K-DS706-00029-0v01-E.pdf

MB9Bxxx-MN706-00002-4v0-E.pdf

2 MCU Peripheral Requirement

Table 1 shows that TPDK solution requires peripheral device and memory. Please select appropriate FM3 MCU before port code.

Table 1. MCU Peripheral Requirement List

Peripheral resource	Number
I/O	34
External Interruption	5
NMI	1
Free-run Timer Channel	3
Dual Timer Channel	2
Software Watchdog Timer	1
12-bit A/D Converter Unit	2
USB2.0 Function Channel	1
Multi-function Serial Interface	2
RAM	>=11KB
ROM	>= 40KB

3 System Hardware Environment

This section introduces the Hardware Environment.

3.1 MCU Information

Below shows the brief information of MCU which is used in thermal printer development board.

- CPU Chip: MB9AF312K;
- CPU Frequency: 40 MHz;
- MCU Pin Number: 48 pin;
- RAM: 16 KB;
- ROM: 128 KB;

4 Development Environment

Development Environment

Table 2. MCU Development Environment

Name	Description	Part Number	Manufacture	Remark
IAR Embedded Workbench6.60	FW development and debug	N/A	N/A	N/A
J-Link / Mini_USB	Debug and Load FW	N/A	N/A	N/A
Hardware Information	TPDK-MainV1.1.0 Board	N/A	N/A	N/A

5 System Firmware Design

This section introduces the FW structure of thermal printer project

5.1 FW Structure

There are 5 folders in the FW structure of IAR, which is shown in Figure 1 and Table 3.

Figure 1. Structure of FW

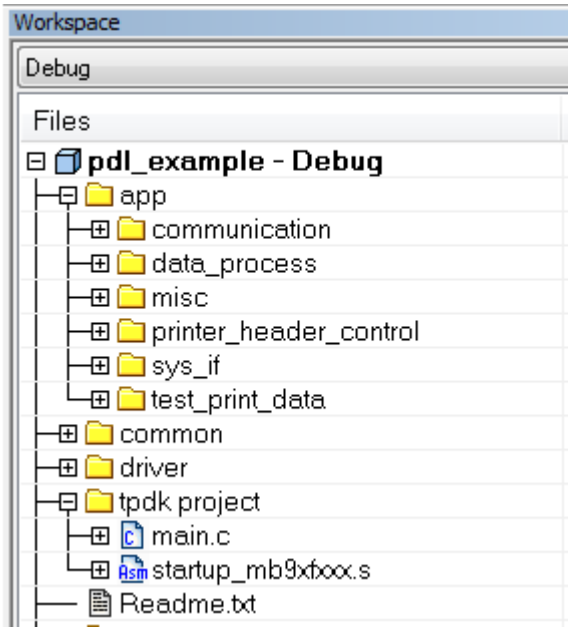


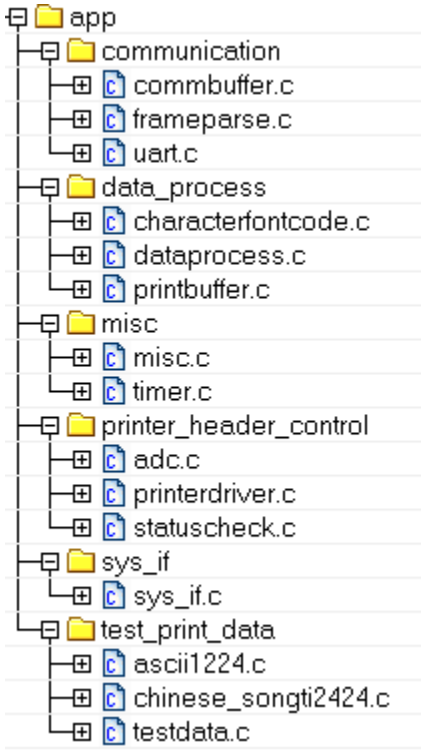
Table 3. Directory Description of Project

Folder	Description
app	It includes communication, data_proces, misc, printer_header_control, sys_if, and test_print_data module.
common	It includes system file and global macro define, register define and FM3 system initialization functions.
driver	It includes FM3 MCU peripheral driver.
tpdk project	It includes IAR project file and system main function.
readme.txt	It records the version log and upgrade information of firmware.

For the TPDK project, the “app” and “tpdk project” folders is the core code.

The app subdirectory file includes “communication”, “misc”, “printer_header_control”, “sys_if”, “data_proces”, and “test_print_data” folders which are shown in Figure 2.

Figure 2. App Subdirectory



The detail description is shown in Table 4.

Table 4. File Description of Project

Folder	File	Description	Mark
communication	commbuffer.c	Create/read/write buffer and others operation	N/A
	commbuffer.h	Include buffer functions and global variables	N/A
	frameparse.c	Parse frame	N/A
	frameparse.h	Include parsing frame functions and global variables	N/A
	uart.c	Initial UART and send/receive data	N/A
	Uart.h	Include UART functions and global variables	N/A
data_process	characterfontcode.c	Get ASCII/Chinese font lib matrix dot	N/A
	characterfontcode.h	Includes ASCII/Chinese font handling functions and global variables	N/A
	dataprocess.c	Handle print data	N/A
	dataprocess.h	Include print data handling functions and global variables	N/A
	printbuffer.c	Get/set print buffer status	N/A
	printbuffer.h	Include print buffer operation functions and global variables	N/A
misc	misc.c	Include LED, beep, key operations and configuration.	N/A
	misc.h	Include operations and configuration functions.	N/A
	timer.c	Include FRT, DT, and WT timer configuration and operation.	N/A
	timer.h	Include FRT, DT, and WT timer configuration and operation functions and global variables.	N/A
printer_header_control	adc.c	Initial and operate ADC	N/A
	adc.h	Include initialization and operation functions, and global variable	N/A
	printerdriver.c	Provide print head driver	N/A
	printerdriver.h	Include print head driver functions and global variable	N/A
	statuscheck.c	Printer status check	N/A
	statuscheck.h	Include status check functions and global variable	N/A
sys_if	sys_if.c	System interrupt handle	N/A
	sys_if.h	System interrupt handle functions	N/A
test_print_data	ascii1224.c	Include ASCII font library	N/A
	ascii1224.h	Include ASCII font library global variable	N/A
	chinese_songti2424.c	Include Chinese font library	N/A
	chinese_songti2424.h	Include Chinese font library global variables	N/A
	testdata.c	Include test print characters and bitmap	N/A
	testdata.h	Include test print characters and bitmap global variables	N/A

6 Configure FM3 Chip

The TPDK project is developed base on FM3 library project. The FM3 library configures MCU system parameters by MCU series and package.

6.1 Select MCU Device

Steps:

1. Confirm MCU chip type by package and device series in FM3 library project.
2. Configure MCU system and peripheral equipment according with MCU type.

For example:

1. Find the device series and package according with MCU typ. Figure 3 shows that FM3 library support all series and package (file path: driver / pdl.h).

   Device series of MB9AF312K is ‘DEVICE_SERIES_MB9AF31X’.
   Device package of MB9AF312K is ‘DEVICE_PACKAGE_K’.

Figure 3. FM3 Device Macro Define

```
/**
 ******************************************************************************
 ** 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. Modify device series and device package macro defines (file path: driver / pdl_device.h).

Figure 4. MB9AF312K Device Macro Define

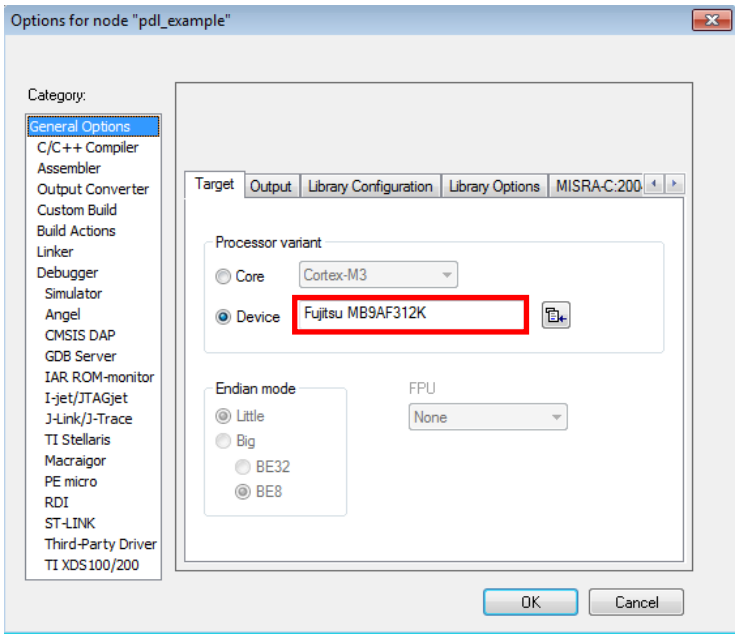
```
/*!
 ******************************************************************************
 ** \brief Choose device series [User configuration]
 ******************************************************************************
 */
#define MCU_SERIES       DEVICE_SERIES_MB9AF31X
/*!
 ******************************************************************************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

6.2 Configure IAR

Open TPDK IAR project and configure IAR options.

6.2.1 Configure Processor Variant

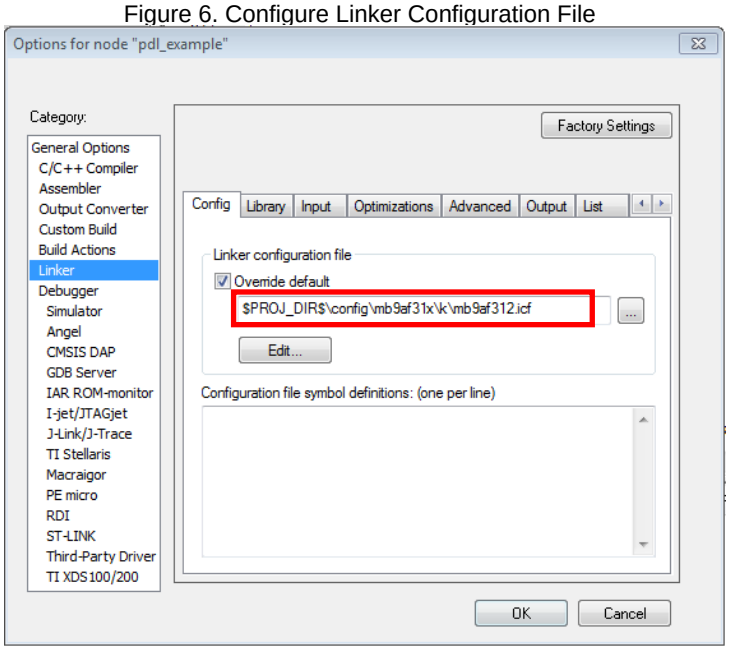
Figure 5. Configure Processor Variant



6.2.2 Configure Linker Configuration File

Select corresponding icf file in the linker option (file path: ..\ tpdk project\ IAR\ config).

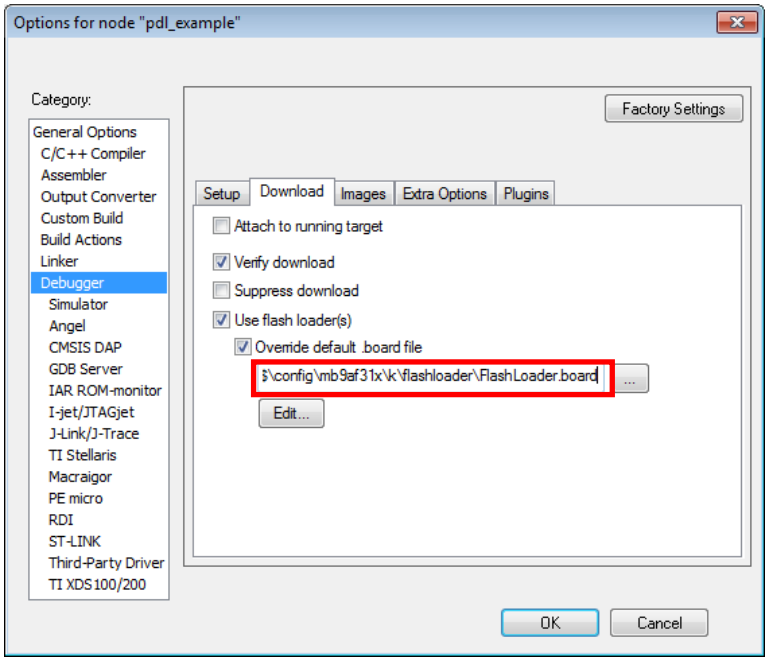
Figure 6. Configure Linker Configuration File



6.2.3 Configure Flash Loader

Select corresponding flash loader file in the debug options (file path: ..\ tpdk project\ IAR\ config).

Figure 7. Configure Flash loader File

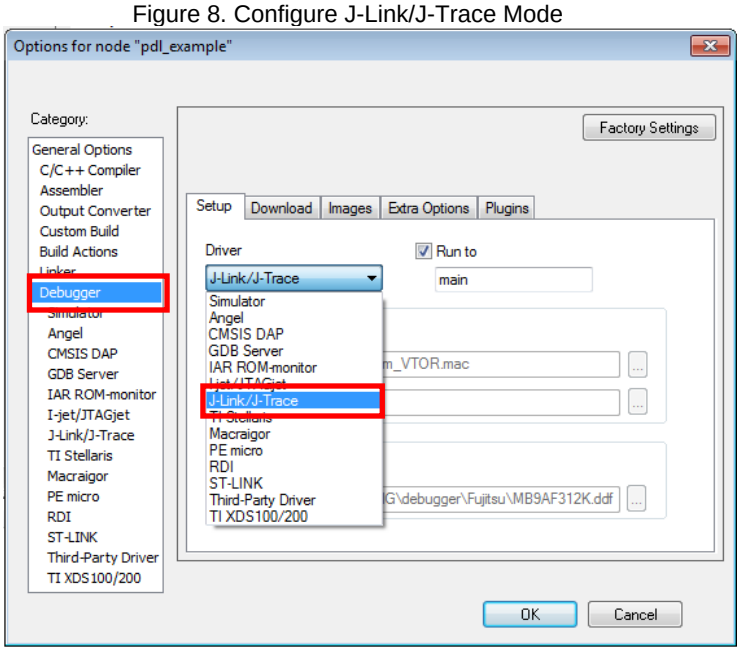


6.2.4 Select IAR Debug Mode

TPDK board supports two debug modes.

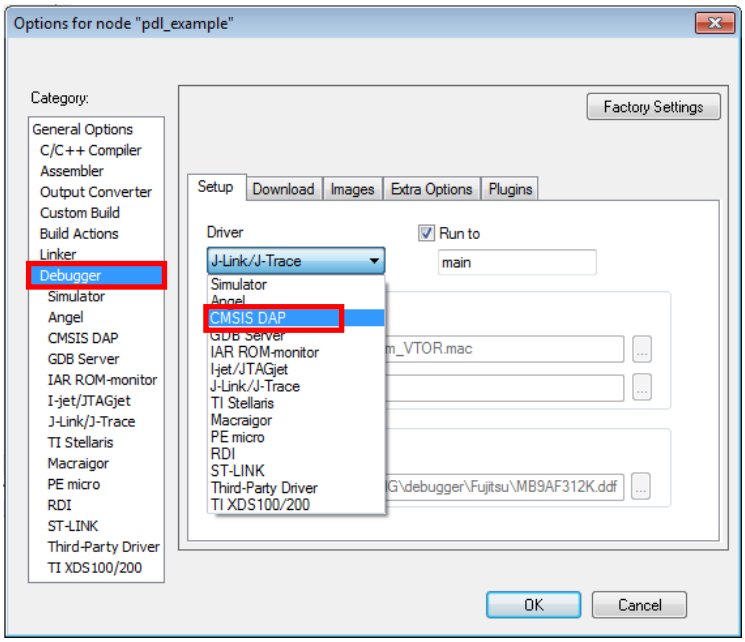
1. SWD Debug

Figure 8. Configure J-Link/J-Trace Mode



2. DAP Debug

Figure 9. Configure CMSIS_DAP Mode



7 Configure MCU Clock

FM3 library project can set system clock according with chip type after configure FM3 chip.

Steps of configuring clock:

1. Confirm MCU type according with section 6.1.
2. Confirm master clock source; confirm base clock, APB0 clock, APB1 clock and APB2 clock frequency.
3. Modify clock parameters and macro defines according with result of step 2.

NOTE 1: detail information about clock refers Chapter 2-1: Clock of MB9Bxxx-MN706-00002-4v0-E.pdf

NOTE 2: printer performance can be different when MCU clock is different.

For example,

Step1: Confirm MCU type5 according with section 6.1.

Step2: The main PLL clock is used as a master clock.

Base clock frequency is 40 MHz.

APB0 and APB1 clock frequency is 20 MHz.

APB2 clock frequency is 40 MHz.

Step3: Modify macro defines (file path: com / system_mb9xfxxx.h).

Figure 10. Configure Clock

```
#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 Port App Layer

When port project to new board, some parameters and macro defines which is relevant to hardware and peripheral device shall been modified. For example, UART, key pin, FRT, dual timer, print head pin and so on.

8.1 Configuration UART

MFS channel and UART pin parameters and macro defines shall been modified, when select different FM3 MCU and UART channel.

Steps of configuring UART:

1. Confirm MFS channel and UART pin.
2. Confirm UART FIFO size.
3. Modify MFS channel and UART pin parameters and macro defines.
4. Modify UART mode and FIFO parameters and macro defines.

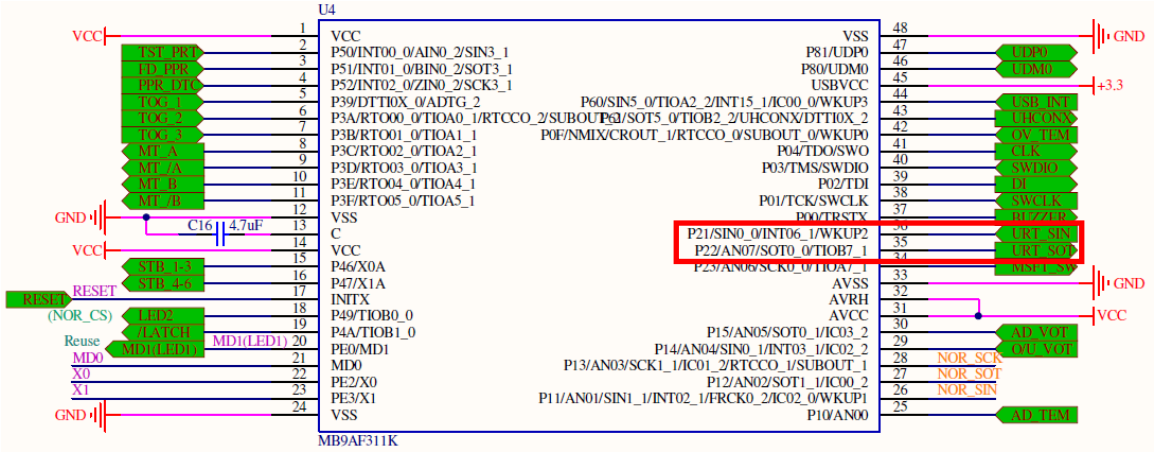
NOTE 1: If UART pins are special I/O ports, disable the special function before configure this pin.

NOTE 2: If MFS channel has no FIFO or doesn't use FIFO, delete macro define "#define UART_FIFO".

For example,

Step1: Figure 11 shows that UART module selects P21 and P22 as UART pins which correspond with MFS channel 0.

Figure 11. UART Circuit



Step2: MFS channel 0 FIFO size of MB9AF312K is 16 bytes which refers 'MB9AF311K-DS706-00029-0v01-E.pdf'.

Step3: Modify MFS channel and UART pin parameters as Figure 12 according with result of step 1 (file path: app / communication / uart.h).

Figure 12. Configure UART Pin and Channel

```
/*****************************************************************************/
/* 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)
```

Step4: P21 is special I/O port which has AD function. Disable special function as Figure 13 (file path: app / communication / uart.c).

Figure 13. Modify UART Pin Function

```
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;
}
```

Step5: Configure mode and FIFO as Figure 14 according with result of step 2 (file path: app / communication / uart.c).

Figure 14. Configure UART Mode and FIFO

```
/*****************************************************************************/
/*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 Configuration ADC

ADC unit, channel and pin parameters shall been modified when select different FM3 MCU and ADC channel.

Steps of configuring ADC:

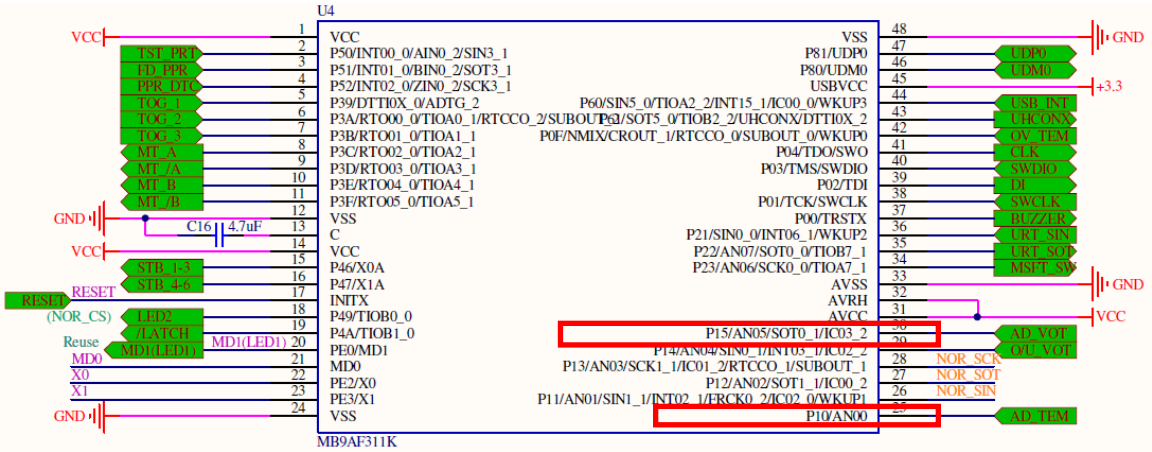
1. Confirm ADC unit, channel and pin.
2. Modify ADC parameters and macro defines.

For example,

Step1: Figure 15 shows that ADC module selects P10/AN05 and P15/AN00 as ADC pins which correspond with MFS channel 5 and channel 0.

ADC module has two ADC units which refers 'MB9AF311K-DS706-00029-0v01-E.pdf'. Select ADC unit 0 and channel 0 to sample temperature and select ADC unit 1 and channel 5 to sample voltage according with FW design.

Figure 15. Voltage and Temperature Sample Circuit



Step2: Modify ADC unit and channel parameters as Figure 16 and Figure 17 according with result of step 1 (file path: app / printer_header_control/ adc.h).

Figure 16. Configure ADC Unit and Channel

```
/*! \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
```

Figure 17. Configure ADC Pin

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

8.3 Configure Print Head Pin

Print head pin parameters shall been modified when select different FM3 MCU and GPIO.

Steps of configuring print head pin:

1. Confirm all pins which are used to control print head.
2. Modify pin parameters and macro defines.
3. Modify initial functions.

NOTE: If pins are special I/O ports, disable the special function before configure those pins.

For example,

Step1: Figure 18 shows print head pins and Table 5 shows pins function.

Figure 18. Print Head Pin

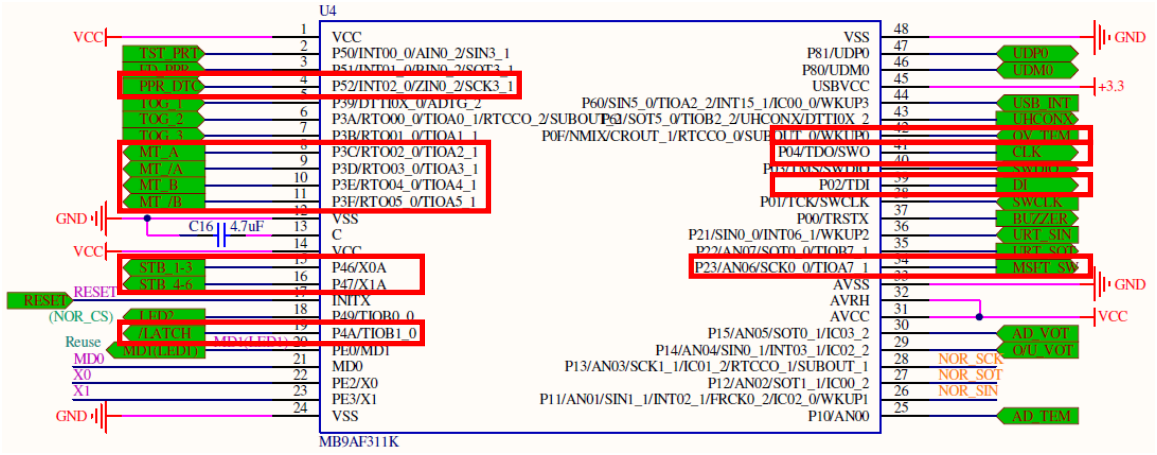


Table 5. Control Print Head Pin Functions

Pin	Function
P52	Detect paper
P3C	Motor A
P3D	Motor NA
P3E	Motor B
P3F	Motor NB
P46	Control stb1 stb2 and stb3
P47	Control stb4 stb5 and stb6
P0F	Detect overheat
P04	CLK
P02	Data input
P23	Control mosfet circuit

Step2: Modify pin parameters as Figure 19 according with result of step 1 (file path: app / printer_header_control / printerdriver.h).

Figure 19. Configure Print Head Interrupt Pin

```
#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
```

Step3: Modify function ‘void PHC_PrintDriverInit(void)’ (file path:app / printer_header_control / printerdriver.c).

Select P46 and P47 to control heating. P46 and P47 of MB9AF312K have sub clock function. Disable sub clock function, before configure GPIO function.

Figure 20. Configure STB Pin

```
/*-------------------- 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
```

Select P23 to control mosfet circuit. P23 of MB9AF312K has ADC pin function. Disable ADC pin function, before configure GPIO function.

Figure 21. Configure Mosfet Pin

```
/*-------------------- 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);
```

Select P14 to control detect print head voltage. P14 of MB9AF312K has ADC pin function. Disable ADC pin function, before configure interrupt function.

Figure 22. Configure Over/Under Voltage Interrupt Pin

```
    /*-------------------- 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 Configure Key Pin

Key pin parameters shall been modified when select different FM3 MCU.

Steps of configuring key pin:

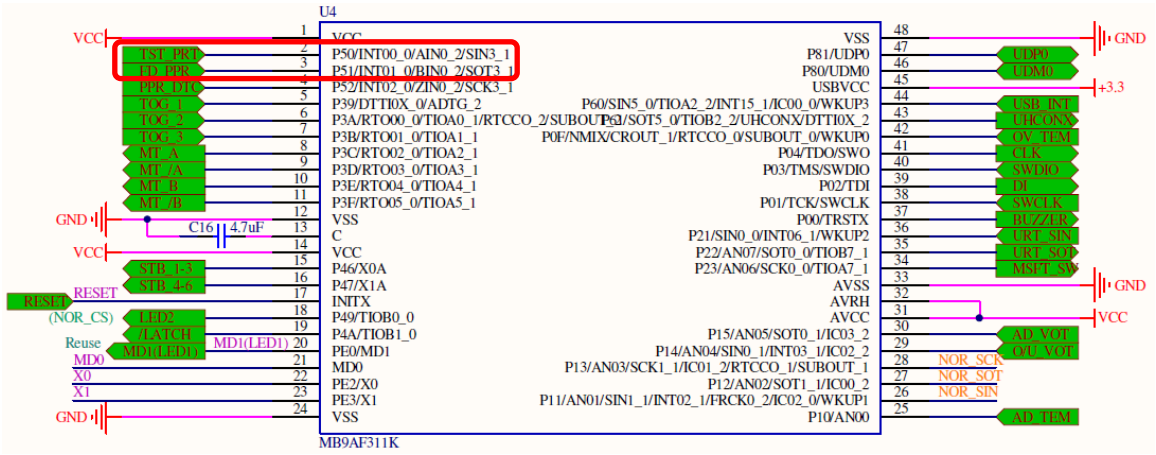
1. Confirm key pins.
2. Modify pin parameters and macro defines.

NOTE: If key pins are special I/O ports, please disable the special function before configure those pins.

For example,

Step1: Figure 23 shows that P50 and P51 pins are selected as test key and feed paper key. And select external interrupt function to implement.

Figure 23. Test Key and Feed Paper Key Circuit



Step2: INT00_0 and INT01_0 pins correspond with external interrupt channel 0 and channel 1. Modify pin parameters as Figure 24 according with result of step 1

(file path: app / misc / misc.h).

Figure 24. Configure Test Key and Feed Paper Key Pin

```
/*-------------------- 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 Configure LED1 and LED2 Pin

Led pins parameters shall been modified when select different FM3 MCU.

Steps of configuring LED pin:

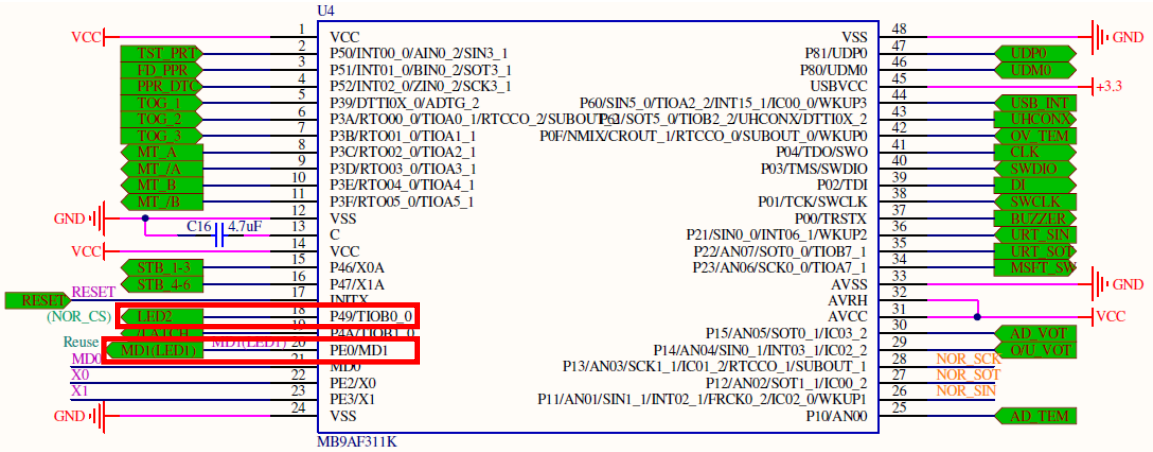
1. Confirm led pin.
2. Modify pin parameters and macro defines.

NOTE: If led pins are special I/O ports, disable the special function before configure those pins.

For example,

Step1: Figure 25 shows that P49 and PE0 pins are selected to control LED2 and LED1.

Figure 25. LED1 and LED2 Circuit



Step2: Modify pin parameters as Figure 26 according with result of step 1(file path: app / misc / misc.h).

Figure 26. Configure LED1 and LED2 Pin

```
/*-------------------- 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 Configure Buzzer Pin

Buzzer pin parameters shall been modified when select different FM3 MCU.

Steps of configuring buzzer pin:

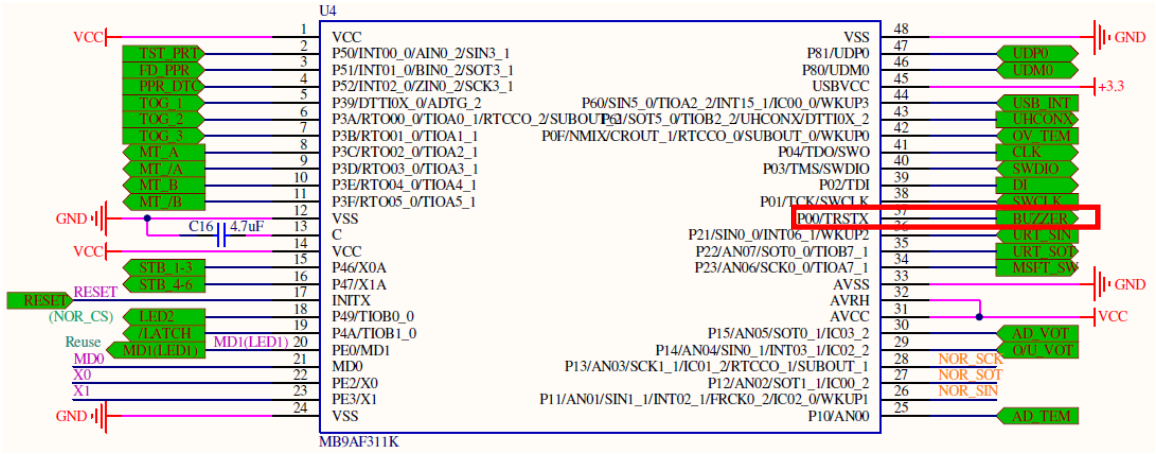
1. Confirm buzzer pin.
2. Modify pin parameters and macro defines.

NOTE: If buzzer pin is special I/O port, disable the special function before configure this pin.

For example,

Step1: Figure 27 shows that P00 pin is selected to control buzzer.

Figure 27. Buzzer Circuit



Step2: Modify pin parameters as Figure 28 according with result of step 1(file directory: app / misc / misc.h).

Figure 28. Configure Buzzer

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

8.7 Configure Toggle Switch Pin

Toggle switch pins parameters shall been modified when select different FM3 MCU.

Steps of configuring toggle switch pin:

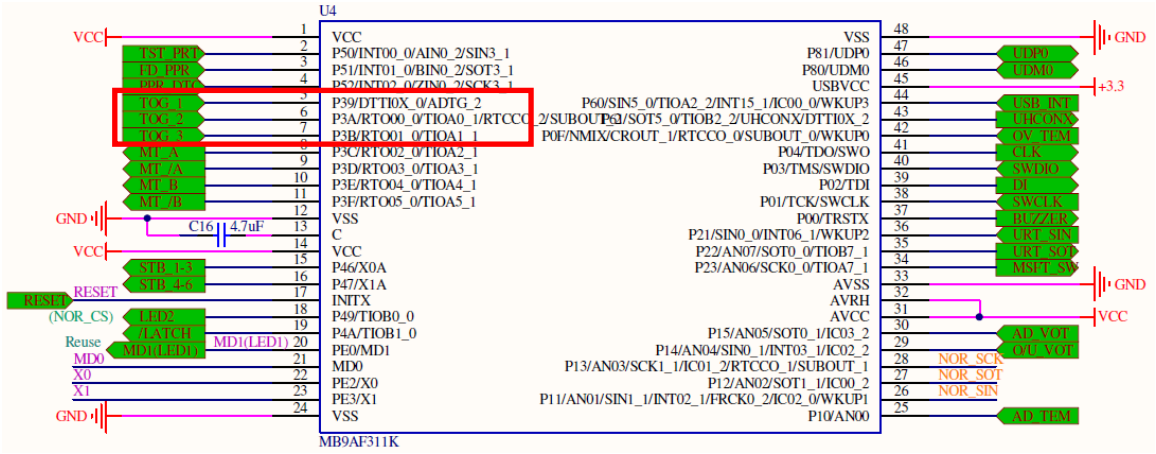
1. Confirm toggle switch pins.
2. Modify pin parameters and macro defines.

NOTE: If toggle switch pins are special I/O ports, disable the special function before configure those pins.

For example,

Step1: Figure 29 shows that P39, P3A and P3B pins are selected as toggle switch pins.

Figure 29. Toggle Switch Circuit



Step2: Modify pin parameters as Figure 30 according with result of step 1 (file directory: app / misc / misc.h).

Figure 30. Configure Toggle Switch Pin

```
/*-------------------- 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 Configure FRT Timer

FRT timer parameters shall been modified when select different FM3 MCU and FRT timer.

Steps of configuring FRT:

1. Confirm FRT the bus block and frequency.
2. Confirm FRT timer unit and channel.
3. Confirm the count cycle.
4. Modify FRT macro defines and parameters according with result of above.

NOTE: Different MCU and different system clock configuration have different APB0, APB1, and APB2 clock. Please re-calculate count value and re-configure FRT parameters when switch MCU and modify clock system clock configuration.

For example,

Step1: MFT of MB9AF312K connects to APB1 which refers 'MB9AF311K-DS706-00029-0v01-E.pdf'.

When system start up, initial base clock, APB0 bus clock, APB1 bus clock, APB2 bus clock by the function 'void SystemInit (void)'. This function configures APB1 clock frequency to 20MHz (file directory: common / system_mb9xfxxx.c).

Step 2: MB9AF312K has 16-bit free-run timer × 3ch which refers 'MB9AF311K-DS706-00029-0v01-E.pdf'. Select FRT channel0 for count of timeout; select channel1 for test key; select channel2 for feed paper key.

Step 3: Set APB1 as the source clock of MFT in up-count mode;

FRT's count clock cycle = APB0 cycle * multiple

FRT's count cycle = Count value * FRT's count clock cycle

So, Timeout FRT multiple value is '32';

Timeout interval cycle is 1ms;

Timeout FRT's count clock cycle = 32 / 20MHz

1ms = Count value * (32 / 20MHz)

Count value = 625

Test key FRT multiple is '32';

Timeout interval cycle is 10ms;

Timeout FRT's count clock cycle = 32 / 20MHz

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

Count value = 6250

Feed paper key FRT multiple is '32';

Timeout interval cycle is 10ms;

Timeout FRT's count clock cycle = 32 / 20MHz

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

Count value = 6250

Step4: Modify FRT parameters as Figure 31 according with result of step 1 (file path: app / misc / timer.h).

Figure 31. Configure FRT

```
/*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 Configure Software Watchdog

Software watchdog timer parameters shall been modified when select different FM3 MCU.

Steps of configuring software watchdog:

1. Confirm software watchdog the bus block and frequency.
2. Calculate software watchdog count cycle value.
3. Modify software watchdog macro defines and parameters according with result of above.

For example,

Step1: Software watchdog of MB9AF312K connects to APB0 which refers 'MB9AF311K-DS706-00029-0v01-E.pdf'.

When system start up, initial Base clock, APB0 clock, APB1 clock, APB2 clock and software watchdog clock by the function 'void SystemInit (void)'. This function configures APB0 clock frequency to 20MHz, and set watchdog clock to 2.5MHz. (file directory: common / system_mb9xfxxx.c)

Step2: Reset system if don't feed dog between 2s. Because a reset is generated at the second underflow, watchdog interval time is 1s.

watchdog interval time = watchdog count * SW_CLKFREQ

1s = watchdog count *2.5MHz

watchdog count = 2500000

Step3: Modify software watchdog count value as Figure 32 according with result of step 2 (file path: app / misc / timer.h).

Figure 32. Configure Software Watchdog Count Value

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

9 Port User Code

Modify some buffer size to meet TPDK solution, when select different MCU

For example,

Step1: Configure heap size to “0x0”, and then make compile.

Analyse Thermal Printer.map file to rw data size (file path: (FWSC)Thermal_Printer_Development_Kit\tpdk project\IAR\output\debug\list\ Thermal Printer .map).

readwrite data memory size = 2 677 bytes,

It equals global data and block CSTACK size.

Figure 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
```

Block HEAP allocates memory to print buffer, frame buffer, communication buffer and ESC/POS buffer. Print buffer, frame buffer and communication buffer size is fixed as Table 6.

Table 6. Buffer Size

Buffer	Size
Print buffer	6144
Frame buffer	518 + 1 + 12
Communication buffer	512 + 1 + 12
Total	7200

So calculate heap size and ESC/POS buffer size as follows:

RAM size – 2677 > Heap size > ESC/POS buffer size + 7200 + 1 +12

16*1024 – 2677 > Heap size > ESC/POS buffer size + 7213

13707 > Heap size > ESC/POS buffer size + 7213

Set heap size to “12362”;

12362 > ESC/POS buffer size + 7213

ESC/POS buffer size < 5149;

So set ESC/POS buffer size to “5000” in TPDK project.

NOTE 1: delete macro define “#define TEST_PRINT_BITMAP” if ROM is less than 90 KB.

NOTE 2: ESCPOS_BUFFER_SIZE value must be more than 500 and different ESCPOS_BUFFER_SIZE value effort printer performance.

Step2: Edit heap size of IAR linker option and ESC/POS buffer size as Figure 34 and Figure 35 according with result of step 1 (file directory: tpdk / source / main.c).

Figure 34. Configure Heap Size of IAR Linker Option

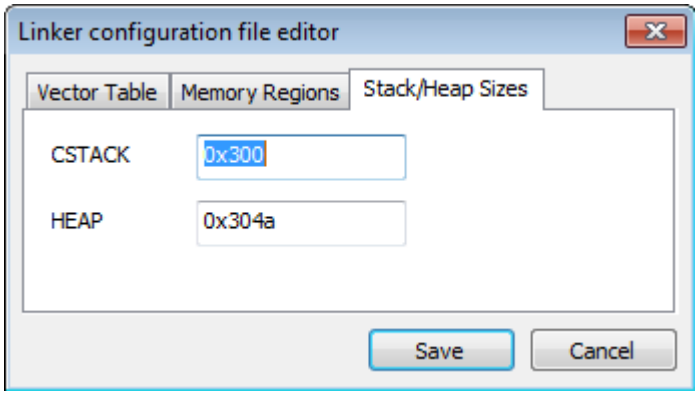


Figure 35. Configure ESC/POS Buffer

```
#define ESCPOS_BUFFER_SIZE 5000
```

10 Run and Debug

Open TPDK IAR project, compile and run, after port.

11 Additional Information

For further information, please visit our website:

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

Please contact your local support team for any technical question.

Document History

Document Title: AN205409 – FM3 Family MB9AF310 Series Thermal Printer Development Kit Migration

Document Number: 002-05409

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	HUAL	01/22/2014	Initial release
			02/14/2014	1. Modify section 3. 2. Add section 7
*A	5276092	HUAL	05/18/2016	Migrated Spansion Application Note MCU-AN-510125-E-10 to Cypress format.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

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

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2014-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.
TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.
Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.