



Ameba-Z SDK change



Outline

- Ameba-Z Platform
- Memory Layout
- Pin Name
- MBED API
- Raw API
- Pinmap configuration
- Sleep configuration



Platform (UM0110)



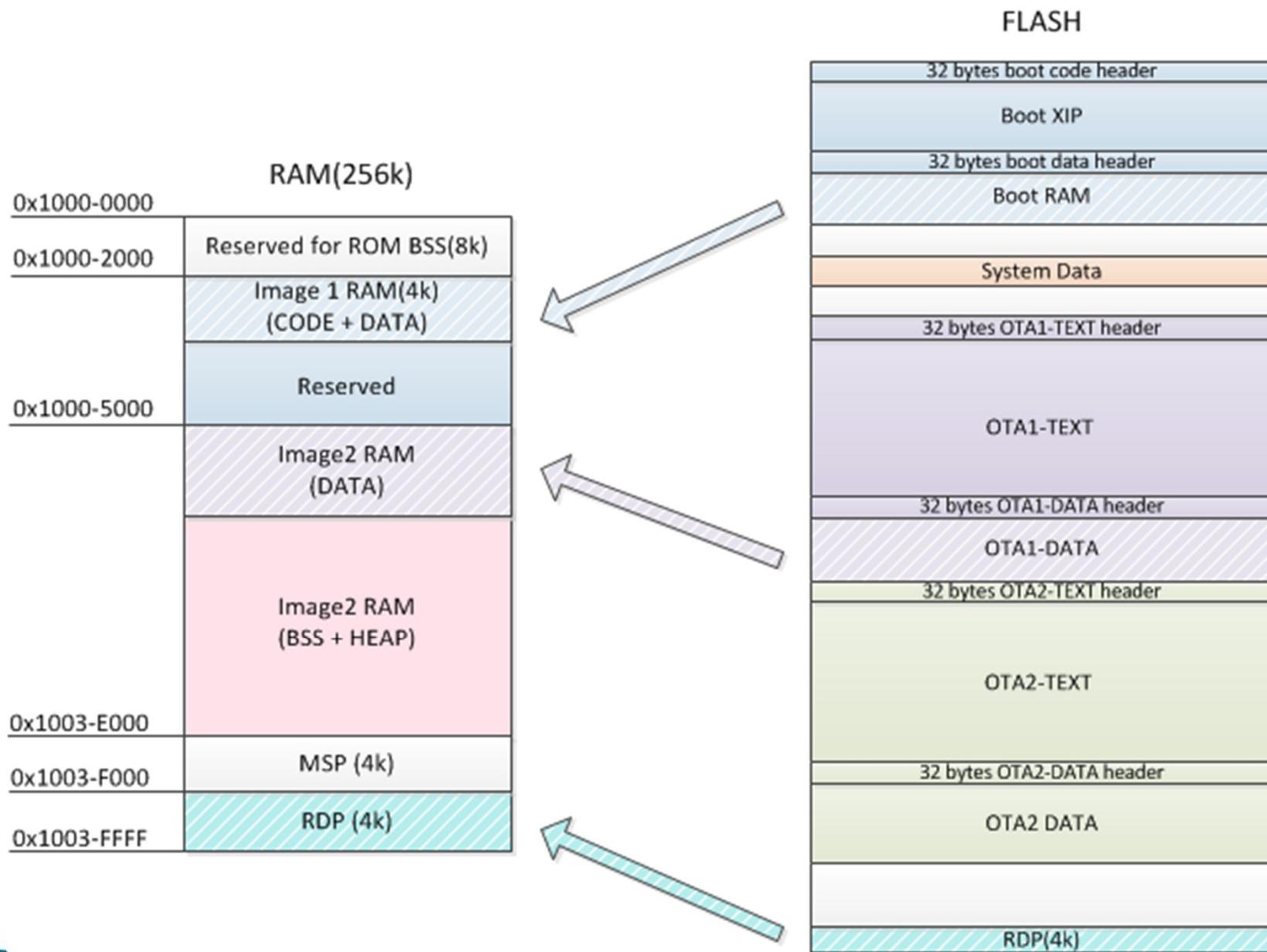
Platform

Feature list		QFN68	QFN48	QFN32
Integrated core	Core type	ARM CM4F		
	Core clock maximum freq.	125MHz		
Memory	Internal ROM	512KB		
	Internal SRAM	256KB		
	External FLASH	128MB (XIP 16MB)		
JTAG/SWD		SWD		
FPU	Float process unit	Yes		
XIP	Execute in place	Yes		
FPB	Flash patch breakpoint	Yes		
Backup register	Backup register for power save	16B		
Boot Reason	Reset reason	Yes		
F/W protection		Yes		
Read protection	RAM read protection	4KB		
WIFI	802.11 B/G/N	Yes		
External 32K	External 32K	1		
Dsleep wakepin	Deep sleep wake pin	4		

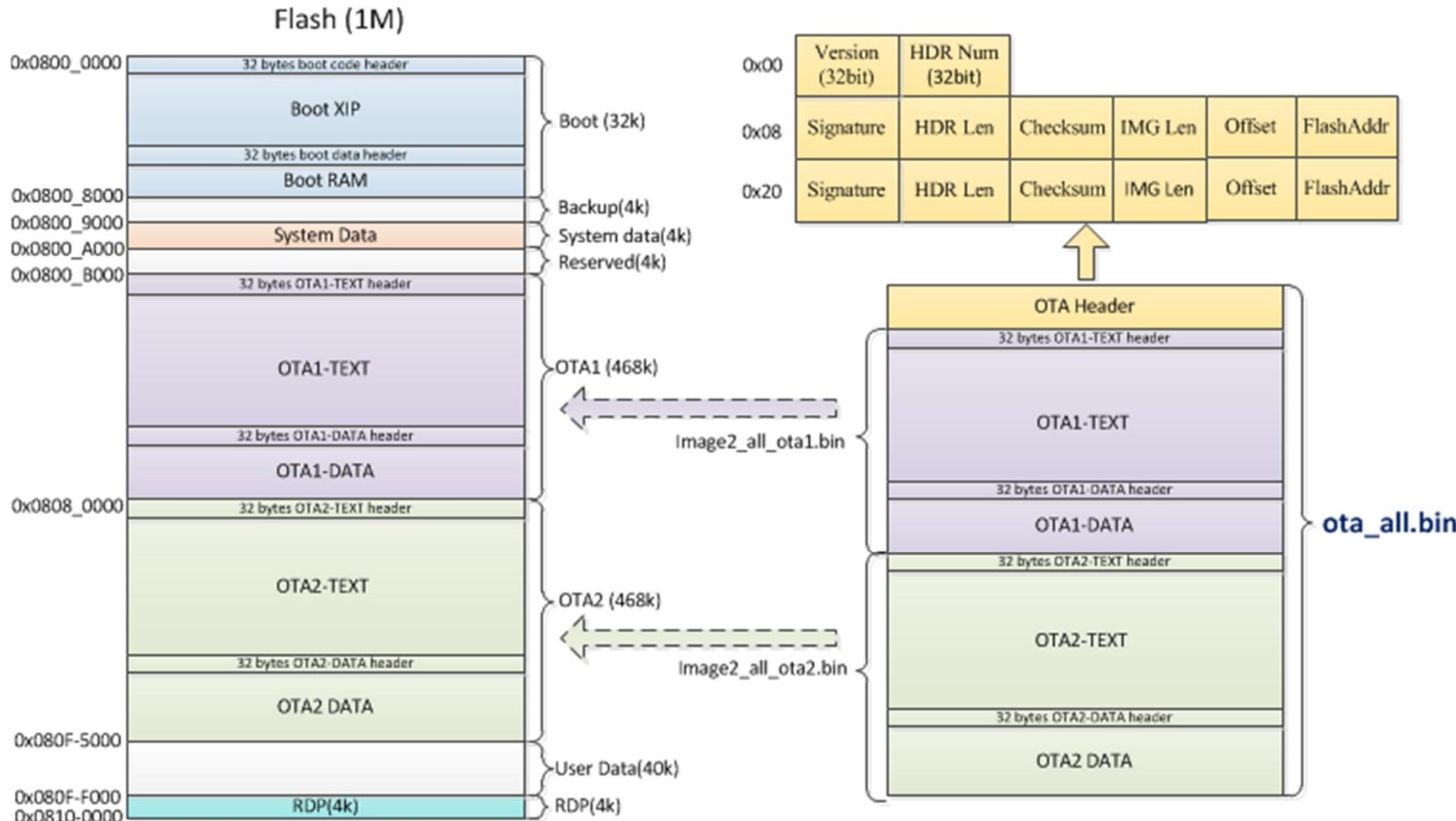


Memory Layout (UM0111)

Memory Layout



OTA Mechanism





Pin Name



Pin Name

```

typedef enum {
    PA_0 = (PORT_A<<4| 0),
    PA_1 = (PORT_A<<4| 1),
    PA_2 = (PORT_A<<4| 2),
    PA_3 = (PORT_A<<4| 3),
    PA_4 = (PORT_A<<4| 4),
    PA_5 = (PORT_A<<4| 5),
    PA_6 = (PORT_A<<4| 6),
    PA_7 = (PORT_A<<4| 7),

    PB_0 = (PORT_B<<4| 0),
    PB_1 = (PORT_B<<4| 1),
    PB_2 = (PORT_B<<4| 2),
    PB_3 = (PORT_B<<4| 3),
    PB_4 = (PORT_B<<4| 4),
    PB_5 = (PORT_B<<4| 5),
    PB_6 = (PORT_B<<4| 6),
    PB_7 = (PORT_B<<4| 7),

    PC_0 = (PORT_C<<4| 0),
    PC_1 = (PORT_C<<4| 1),
    PC_2 = (PORT_C<<4| 2),
    PC_3 = (PORT_C<<4| 3),
    PC_4 = (PORT_C<<4| 4),
    PC_5 = (PORT_C<<4| 5),
    PC_6 = (PORT_C<<4| 6),
    PC_7 = (PORT_C<<4| 7),
    PC_8 = (PORT_C<<4| 8),
    PC_9 = (PORT_C<<4| 9),

    PD_0 = (PORT_D<<4| 0),
    PD_1 = (PORT_D<<4| 1),
    PD_2 = (PORT_D<<4| 2),
    PD_3 = (PORT_D<<4| 3),
    PD_4 = (PORT_D<<4| 4),
    PD_5 = (PORT_D<<4| 5),
    PD_6 = (PORT_D<<4| 6),
    PD_7 = (PORT_D<<4| 7),
    PD_8 = (PORT_D<<4| 8),
    PD_9 = (PORT_D<<4| 9),

    PE_0 = (PORT_E<<4| 0),
    PE_1 = (PORT_E<<4| 1),
    PE_2 = (PORT_E<<4| 2),
    PE_3 = (PORT_E<<4| 3),
    PE_4 = (PORT_E<<4| 4),
    PE_5 = (PORT_E<<4| 5),
    PE_6 = (PORT_E<<4| 6),
    PE_7 = (PORT_E<<4| 7),
    PE_8 = (PORT_E<<4| 8),
    PE_9 = (PORT_E<<4| 9),
    PE_A = (PORT_E<<4| 10),

```



```

/* (((port)<<5)| (pin)) */
typedef enum {
    PA_0 = (PORT_A<<5| 0),
    PA_1 = (PORT_A<<5| 1),
    PA_2 = (PORT_A<<5| 2),
    PA_3 = (PORT_A<<5| 3),
    PA_4 = (PORT_A<<5| 4),
    PA_5 = (PORT_A<<5| 5),
    PA_6 = (PORT_A<<5| 6),
    PA_7 = (PORT_A<<5| 7),
    PA_8 = (PORT_A<<5| 8),
    PA_9 = (PORT_A<<5| 9),
    PA_10 = (PORT_A<<5| 10),
    PA_11 = (PORT_A<<5| 11),
    PA_12 = (PORT_A<<5| 12),
    PA_13 = (PORT_A<<5| 13),
    PA_14 = (PORT_A<<5| 14),
    PA_15 = (PORT_A<<5| 15),
    PA_16 = (PORT_A<<5| 16),
    PA_17 = (PORT_A<<5| 17),
    PA_18 = (PORT_A<<5| 18),
    PA_19 = (PORT_A<<5| 19),
    PA_20 = (PORT_A<<5| 20),
    PA_21 = (PORT_A<<5| 21),
    PA_22 = (PORT_A<<5| 22),
    PA_23 = (PORT_A<<5| 23),
    PA_24 = (PORT_A<<5| 24),
    PA_25 = (PORT_A<<5| 25),
    PA_26 = (PORT_A<<5| 26),
    PA_27 = (PORT_A<<5| 27),
    PA_28 = (PORT_A<<5| 28),
    PA_29 = (PORT_A<<5| 29),
    PA_30 = (PORT_A<<5| 30),
    PA_31 = (PORT_A<<5| 31),

    PB_0 = (PORT_B<<5| 0),
    PB_1 = (PORT_B<<5| 1),
    PB_2 = (PORT_B<<5| 2),
    PB_3 = (PORT_B<<5| 3),
    PB_4 = (PORT_B<<5| 4),
    PB_5 = (PORT_B<<5| 5),
    PB_6 = (PORT_B<<5| 6),
    PB_7 = (PORT_B<<5| 7),
    PB_8 = (PORT_B<<5| 8),

    // Not connected
    NC = (uint32_t)0xFFFFFFFF
} ? end PinName ? PinName;

```



Mbed API (UM0118)



Mbed API

function	change	add	delete	comment
GPIO	N	N	N	
GPIO IRQ	N	N	N	
GPIO PORT	Y	N	N	
UART	N	N	N	
LOGUART	N/A	N/A	Y	LOGUART=NOMAL UART
I2C	N	Y	N	Add API
I2S	Y	N	N	Add input parameters
SPI	Y	N	Y	Add obj parameter
PWM	N	N	N	
Gtimer	N	N	N	
GDMA	N	N	Y	Not support aggregation copy
Flash	N	N	N	
ADC	N	N	N	
RTC	N	Y	N	Add alarm

GPIO

■ API change

- Ameba-Z pin name is mapping to GPIO directly
- **pin_def** is not needed

```

1 struct port_s {
2     PortName port;
3     uint32_t mask;
4     uint8_t *pin_def; // Line 4 is highlighted in red
5 };

```

■ Example:

```

1 void main(void)
2 {
3     int i;
4     unsigned int pin_mask;
5
6     port0.pin_def = My_Port_Def; // Line 6 is highlighted in red
7     pin_mask = 0xFF; // each bit map to 1 pin: 0: pin
8     port_init(&port0, PortA, pin_mask, PIN_OUTPUT);
9     port_mode(&port0, PullNone);
10
11    while(1){
12        for (i=0;i<LED_PATTERN_NUM;i++) {
13            port_write(&port0, led_pattern[i]);
14            wait_ms(200);
15        }
16    }
17 }

```

```

1 void main(void)
2 {
3     int i;
4     unsigned int pin_mask;
5
6     pin_mask = 0xFF; // each bit map to 1 pin: 0: pin
7     port_init(&port0, PortA, pin_mask, PIN_OUTPUT);
8     port_mode(&port0, PullNone);
9
10    while(1){
11        for (i=0;i<LED_PATTERN_NUM;i++) {
12            port_write(&port0, led_pattern[i]);
13            wait_ms(200);
14        }
15    }
16 }

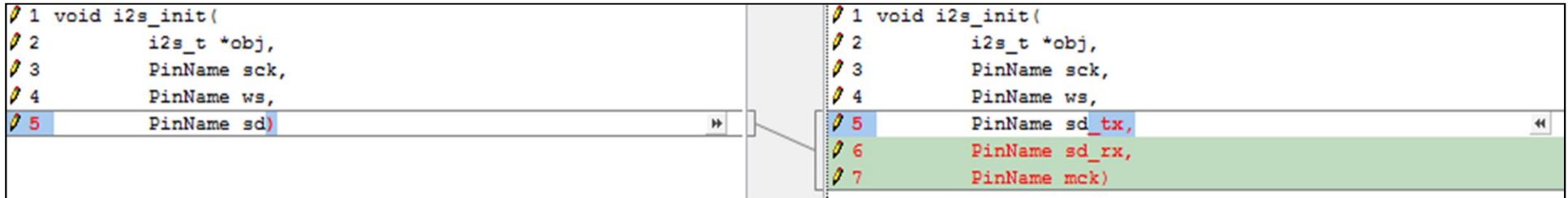
```

I2C

■ API add

```
/**  
 * @brief I2C master send data and read data in poll mode.  
 * @param obj: i2c object define in application software.  
 * @param address: slave address which will be transmitted.  
 * @param pWriteBuf: point to the data to be sent.  
 * @param Writelen: the length of data that to be sent.  
 * @param pReadBuf: point to the buffer to hold the received data.  
 * @param Readlen: the length of data that to be received.  
 * @retval the length of data received.  
 */  
int i2c_repeatread(i2c_t *obj, int address, char *pWriteBuf, int Writelen, char *pReadBuf, int Readlen)  
{  
    if (i2c_target_addr[obj->i2c_idx] != address) {  
        /* Deinit I2C first */  
        i2c_reset(obj);  
  
        /* Load the user defined I2C target slave address */  
        i2c_target_addr[obj->i2c_idx] = address;  
        I2CInitDat[obj->i2c_idx].I2CAckAddr = address;  
  
        /* Init I2C now */  
        I2C_Init(obj->I2Cx, &I2CInitDat[obj->i2c_idx]);  
        I2C_Cmd(obj->I2Cx, ENABLE);  
    }  
    I2C_MasterRepeatRead(obj->I2Cx, pWriteBuf, Writelen, pReadBuf, Readlen);  
  
    return Readlen;  
}
```

I2S



- i2s_init include 5 I2S pins:
 - PinName sck
 - PinName ws
 - PinName sd_tx
 - PinName sd_rx
 - PinName mck

SPI

```
struct spi_s {  
    /* user variables */  
    uint32_t spi_idx;  
  
    /* internal variables */  
    uint32_t irq_handler;  
    uint32_t irq_id;  
    uint32_t state;  
    uint8_t sclk;  
    uint32_t bus_tx_done_handler;  
    uint32_t bus_tx_done_irq_id;  
};
```

```
spi_master.spi_idx=MBED_SPI1;  
spi_init(&spi_master, SPI1_MOSI, SPI1_MISO, SPI1_SCLK, SPI1_CS);  
spi_format(&spi_master, 8, 0, 0);  
spi_frequency(&spi_master, 200000);  
  
spi_slave.spi_idx=MBED_SPI0;  
spi_init(&spi_slave, SPI0_MOSI, SPI0_MISO, SPI0_SCLK, SPI0_CS);  
spi_format(&spi_slave, 8, 0, 1);
```

■ API change

- spi_idx should be set before spi_init, assert will happen if you forgot

■ API delete

- void spi_slave_select_bypin(spi_t *obj, PinName pinname)



GDMA

■ API delete

- dma_memcpy_aggr_init: not support
- dma_memory_aggr: not support



RTC

■ API Add

- u32 rtc_set_alarm(alarm_t *alrm, alarm_irq_handler alarmHandler);
- void rtc_disable_alarm(void);

```
typedef void (*alarm_irq_handler)(void);

struct alarm_s {
    uint32_t yday; //which day of the year
    uint32_t hour;
    uint32_t min;
    uint32_t sec;
};

typedef struct alarm_s alarm_t;
```



RAW API (UM0117)



Document

AmebaZ_Peripheral_API

隐藏 查找 上一步 前进 停止 刷新 主页 字体 打印 选项①

目录 | 索引 | 搜索 | 收藏夹 |

Modules

- AmebaZ_Outline
 - AmebaZ Address Map
 - AmebaZ_Peripheral_Registers_Structures
 - AmebaZ_Peripheral Declarations
- AmebaZ_Platform
 - BKUP_REG
 - CLOCK
 - DELAY
 - OTA
 - PIN
 - PMC
 - CACHE
 - DIAG
 - EFUSE
 - PROTECTION
 - RCC
 - SYSCFG
 - IRQ
 - Debug
- AmebaZ_Periph_Driver
 - CRYPTO
 - DONGLE
 - INIC
 - SDIO
 - USOC
 - ADC
 - FLASH
 - GDMA
 - GPIO
 - I2S
 - RTC
 - SPI
 - Timer
 - UART
 - WDG
 - I2C
- Classes
- Files

The 'UART' node in the tree view is highlighted with a red box and connected by a red arrow to the corresponding section in the main content area.

```
* UART can receive data when soc enter power save mode  
baudrate: 110~500000  
  
* NOTICE: not support Tx/Rx DMA mode under Low Power Rx.  
*****  
* How to use Normal Uart  
*****  
* To use the normal uart mode, the following steps are mandatory:  
  
1. Enable peripheral clock using the follwoing functions.  
    RCC_PeriphClockCmd(APBPeriph_UARTx, APBPeriph_UARTx_CLOCK, ENABLE);  
  
2. configure the UART pinmux.  
    Pinmux_Config(Pin_Num, PINMUX_FUNCTION_UART)  
  
3. Program Word Length , Stop Bit, Parity, Hardware flow control and DMA ,  
    Mode(ENABLE/DISABLE) using the UART_StructInit() and UART_Init() function.  
  
4. Program the Baud Rate, using function UART_SetBaud().  
  
5. Enable the NVIC and the corresponding interrupt using following function if you need  
    to use interrupt mode.  
    UART_INTConfig(): UART IRQ Mask set  
    InterruptRegister(): register the uart irq handler  
    InterruptEn(): Enable the NVIC interrupt  
  
6. Enable uart rx path, using function UART_RxCmd().  
  
@note in UART_Normal_functions group, these functions below are about Interrupts  
and flags management.  
    UART_INTConfig()  
    UART_IntStatus()  
    UART_LineStatusGet()  
  
*****  
* How to use uart in DMA mode  
*****  
* To use the uart in DMA mode, the following steps are mandatory:  
  
1. Enable peripheral clock using the follwoing functions.  
    RCC_PeriphClockCmd(APBPeriph_UARTx, APBPeriph_UARTx_CLOCK, ENABLE);
```



XXX_InitTypeDef

AmebaZ_Peripheral_API

隐藏 查找 上一步 前进 停止 刷新 主页 字体 打印 选项

目录 | 索引 | 搜索 | 收藏夹 |

- + BKUP_REG
- + CLOCK
- + DELAY
- + OTA
- + PIN
- + PMC
- + CACHE
- + DIAG
- + EFUSE
- + PROTECTION
- + RCC
- + SYSCFG
- + IRQ
- + Debug
- AmebaZ_Periph_Driver
 - + CRYPTO
 - + DONGLE
 - + INIC
 - + SDIO
 - + USOC
 - + ADC
 - + FLASH
 - + GDMA
 - + GPIO
 - + I2S
 - + RTC
 - + SPI
 - + Timer
 - UART
 - + UART_Exported_Types
 - + UART_InitTypeDef
 - DmaModeCtrl
 - WordLen
 - StopBit
 - Parity
 - ParityType
 - StickParity
 - FlowControl
 - RxFifoTrigLevel
 - RxErReportCtrl
 - + LPUART_InitTypeDef
 - + I2C_InitTypeDef

u32 UART_InitTypeDef::DmaModeCtrl

Specifies the uart DMA mode state. This parameter can be ENABLE or DISABLE.

u32 UART_InitTypeDef::WordLen

Specifies the UART word length. This parameter can be a value of [UART_Word_length_define](#).

u32 UART_InitTypeDef::StopBit

Specifies the UART stop bit number. This parameter can be a value of [UART_Stop_Bit_define](#).

u32 UART_InitTypeDef::Parity

Specifies the UART parity. This parameter can be a value of [UART_Parity_Enable_define](#).

u32 UART_InitTypeDef::ParityType

Specifies the UART parity type. This parameter can be a value of [UART_Parity_Type_define](#).

u32 UART_InitTypeDef::StickParity

Specifies the UART stick parity. This parameter can be a value of [UART_Stick_Parity_Type_define](#).

u32 UART_InitTypeDef::FlowControl

Specifies the UART auto flow control. This parameter can be ENABLE or DISABLE.

u32 UART_InitTypeDef::RxFifoTrigLevel



XXX_functions

AmebaZ_Peripheral_API

隐藏 查找 上一步 前进 停止 刷新 主页 打印 选项

目录 索引 搜索 收藏夹

Timer
UART
 UART Exported Types
 UART Exported Constants
 UART Exported Functions
 UART_Normal_functions
 UART_Delnit
 UART_StructInit
 UART_Init
 UART_BaudParaGet
 UART_BaudParaGetFull
 UART_SetBaudExt
 UART_SetBaud
 UART_SetRxLevel
 UART_RxCmd
 UART_Writable
 UART_Readable
 UART_CharPut
 UART_CharGet
 UART_ReceiveData
 UART_SendData
 UART_ReceiveDataTO
 UART_SendDataTO
 UART_RxByteCntClear
 UART_RxByteCntGet
 UART_BreakCtl
 UART_ClearRxFifo
 UART_ClearTxFifo
 UART_INTConfig
 UART_IntStatus
 UART_ModemStatusGet
 UART_LineStatusGet
 UART_WaitBusy
 UART_PinMuxInit
 UART_PinMuxDeinit
 UART_DMA_functions
 UART_Low_Power_functions
 UART_IRDA_functions
 UART Register Definitions
 DLH_INTCR
 INTID
 FCR

`_LONG_CALL_ void UART_Init (UART_TypeDef * UARTx,
 UART_InitTypeDef * UART_InitStruct
)`

Initializes the UARTx peripheral according to the specified parameters in the UART_InitStruct.

Parameters

 UARTx where x can be 0~2.
 UART_InitStruct pointer to a **UART_InitTypeDef** structure that contains the configuration information for the specified USART peripheral.

Return values

 None

`_LONG_CALL_ u32 UART_BaudParaGet (u32 baudrate,
 u32 * ovsr,
 u32 * ovsr_adj
)`

get ovsr & ovsr_adj parameters for the given baudrate according to the Baudrate Table BAUDRATE_TABLE_40M.

Parameters

 UARTx where x can be 0~2.
 baudrate the desired baudrate
 ovsr the pointer to ovsr parameter
 ovsr_adj the pointer to ovsr_adj parameter

Return values

 searching status:

- 1: found
- 0: not found



XXX_registers

AmebaZ_Peripheral_API

隐藏 查找 上一步 前进 停止 刷新 主页 打印 选项①

目录 (C) | 索引 (I) | 搜索 (S) | 收藏夹 (D)

- SYSCFG
- IRQ
- Debug
- AmebaZPeriph_Driver
 - CRYPTO
 - DONGLE
 - INIC
 - SDIO
 - USOC
 - ADC
 - FLASH
 - GDMA
 - GPIO
 - I2S
 - RTC
 - SPI
 - Timer
- UART
 - UART Exported Types
 - UART Exported Constants
 - UART Exported Functions
 - UART Register Definitions
 - DLH_INTCR
 - INTID
 - FCR
 - MCR
 - LCR
 - LSR
 - SPR
 - STSR
 - MISCR
 - TXPLSR
 - RXPLSR
 - REG_RX_PATH_CTRL
 - REG_MON_BAUD_CTRL
 - REG_MON_BAUD_STS
 - REG_RX_BYTE_CNT
 - WDG
 - I2C
- Classes
- Files

REALTEK

AmebaZ_Peripheral_API 1

Main Page Modules Classes Files

DLH_INTCR

AmebaZPeriph_Driver » UART » UART Register Definitions

Macros

```
#define RUART_IER_ERBI ((u32)0x00000001 /*BIT[0], Enable received data available interrupt (rx trigger)*/
#define RUART_IER_ETBEI ((u32)0x00000001<<1 /*BIT[1], Enable transmitter FIFO empty interrupt (tx fifo empty)*/
#define RUART_IER_ELSI ((u32)0x00000001<<2 /*BIT[2], Enable receiver line status interrupt (receiver line status)*/
#define RUART_IER_EDSSI ((u32)0x00000001<<3 /*BIT[3], Enable modem status interrupt (modem status transition)*/
#define RUART_IER_EDMI ((u32)0x00000001<<4 /*BIT[4], Enable low power rx monitor done interrupt (monitor done)*/
#define RUART_IERETOI ((u32)0x00000001<<5 /*BIT[5], Enable rx time out interrupt*/
```

Detailed Description



Pin Map



Pin Map (rtl8710b_pinmapcfg.c)

Pin Name	Func Select	Func PU/PD	Slp PU/PD	DrvStrength
{_PA_14,	PINMUX_FUNCTION_SWD,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SWD_CLK
{_PA_15,	PINMUX_FUNCTION_SWD,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SWD_DATA
{_PA_13,	PINMUX_FUNCTION_PWM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //PWM4
{_PA_0,	PINMUX_FUNCTION_PWM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_DOWN,	PAD_DRV_STRENGTH_0}, //PWM2
{_PA_16,	PINMUX_FUNCTION_PWM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //PWM1
{_PA_17,	PINMUX_FUNCTION_PWM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //PWM2
{_PA_25,	PINMUX_FUNCTION_UART,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //UART1_RXD
{_PA_26,	PINMUX_FUNCTION_UART,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //UART1_TXD
{_PA_28,	PINMUX_FUNCTION_I2C,	GPIO_PuPd_UP,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //I2C1_SCL
{_PA_27,	PINMUX_FUNCTION_I2C,	GPIO_PuPd_UP,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //I2C1_SDA
{_PA_12,	PINMUX_FUNCTION_PWM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //PWM3
{_PA_4,	PINMUX_FUNCTION_UART,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //UART0_TXD
{_PA_1,	PINMUX_FUNCTION_UART,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //UART0_RXD
{_PA_3,	PINMUX_FUNCTION_UART,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //UART0_RTS
{_PA_2,	PINMUX_FUNCTION_UART,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //UART0_CTS
{_PA_6,	PINMUX_FUNCTION_SPIF,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPIC_CS
{_PA_7,	PINMUX_FUNCTION_SPIF,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPIC_DATA1
{_PA_8,	PINMUX_FUNCTION_SPIF,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPIC_DATA2
{_PA_9,	PINMUX_FUNCTION_SPIF,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPIC_DATA0
{_PA_10,	PINMUX_FUNCTION_SPIF,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPIC_CLK
{_PA_11,	PINMUX_FUNCTION_SPIF,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPIC_DATA3
{_PA_5,	PINMUX_FUNCTION_PWM,	GPIO_PuPd_UP,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //PWM4
{_PA_18,	PINMUX_FUNCTION_SDIOD,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SD_D2
{_PA_19,	PINMUX_FUNCTION_SDIOD,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SD_D3
{_PA_20,	PINMUX_FUNCTION_SDIOD,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SD_CMD
{_PA_21,	PINMUX_FUNCTION_SDIOD,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SD_CLK
{_PA_22,	PINMUX_FUNCTION_SDIOD,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SD_D0
{_PA_23,	PINMUX_FUNCTION_SDIOD,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SD_D1
{_PB_0,	PINMUX_FUNCTION_SPIM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPI1_CS
{_PB_1,	PINMUX_FUNCTION_SPIM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPI1_CLK
{_PB_2,	PINMUX_FUNCTION_SPIM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPI1_MISO
{_PB_3,	PINMUX_FUNCTION_SPIM,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //SPI1_MOSI
{_PB_4,	PINMUX_FUNCTION_I2S,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //I2S_MCK
{_PB_5,	PINMUX_FUNCTION_I2S,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //I2S_SD_TX
{_PA_24,	PINMUX_FUNCTION_I2S,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //I2S_SD_RX
{_PA_31,	PINMUX_FUNCTION_I2S,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //I2S_CLK
{_PB_6,	PINMUX_FUNCTION_I2S,	GPIO_PuPd_NOPULL,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //I2S_WS
{_PA_30,	PINMUX_FUNCTION_UART,	GPIO_PuPd_UP,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //UART2_log_TXD
{_PA_29,	PINMUX_FUNCTION_UART,	GPIO_PuPd_UP,	GPIO_PuPd_UP,	PAD_DRV_STRENGTH_0}, //UART2_log_RXD
{_PNC,	PINMUX_FUNCTION_GPIO,	GPIO_PuPd_NOPULL,	GPIO_PuPd_NOPULL,	PAD_DRV_STRENGTH_0}, //table end



Pin Map settings

- Func Select
 - Set pinmux function based on your board
- Func PU/PD
 - Set pull control based on the function of this pin
- Slp PU/PD
 - Set pull control under sleep mode to prevent power leakage
 - Set this option based on your board
- DrvStrength
 - Driver strength when PD
 - Don't set it if not needed



Sleep Configuration



Power Management

■ rtl8710b_sleepcfg.c

```
const PWRCFG_TypeDef sleep_pwrctrl_config[] =  
{  
    // Module           Status  
    {BIT_SYSON_PMOPT_SLP_MEM2_EN,      ON}, /* SRAM 192K~256K: 6uA */  
    {BIT_SYSON_PMOPT_SLP_MEM1_EN,      ON}, /* SRAM 128K~192K: 6uA */  
    {BIT_SYSON_PMOPT_SLP_MEM0_EN,      ON}, /* SRAM 0~128K: 12uA */  
    {BIT_SYSON_PMOPT_SLP_SYSPLL_EN,    OFF}, /* Syspem PLL: 6.5mA */  
    {BIT_SYSON_PMOPT_SLP_XTAL_EN,      OFF}, /* XTAL: 2.2mA */  
    {BIT_SYSON_PMOPT_SLP_EN_SOC,       OFF}, /* SoC(CPU) domain, 200uA */  
    {BIT_SYSON_PMOPT_SLP_EN_PWM,       OFF}, /* SWR/LDO output heavy loading current mode */  
    {BIT_SYSON_PMOPT_SLP_EN_SWR,       OFF}, /* SWR/LDO 1.2V */  
    {BIT_SYSON_PMOPT_SLP_LPLDO_SEL,    OFF}, /* V12H LDO: 50uA */  
    {0xFFFFFFFF,                      OFF}, /* Table end */  
};
```



Wake Event configuration

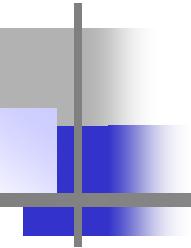
```
const PWRCFG_TypeDef sleep_wevent_config[] =  
{  
    // Module  
    {BIT_SYSON_WEVT_GPIO_DSTBY_MSK, ON}, /* dstandby: wakepin 0~3 wakeup */  
    {BIT_SYSON_WEVT_A33_AND_A33GPIO_MSK, ON}, /* dsleep: REGU A33 Timer & A33 wakepin wakeup */  
    {BIT_SYSON_WEVT_ADC_MSK, OFF}, /* sleep: ADC Wakeup */  
    {BIT_SYSON_WEVT_SDIO_MSK, OFF}, /* sleep: SDIO Wakeup */  
    {BIT_SYSON_WEVT_RTC_MSK, ON}, /* dstandby: RTC Wakeup */  
    {BIT_SYSON_WEVT_UART1_MSK, OFF}, /* sleep: UART1 Wakeup */  
    {BIT_SYSON_WEVT_UART0_MSK, OFF}, /* sleep: UART0 Wakeup */  
    {BIT_SYSON_WEVT_I2C1_MSK, OFF}, /* sleep: I2C1 Wakeup */  
    {BIT_SYSON_WEVT_I2C0_MSK, OFF}, /* sleep: I2C0 Wakeup */  
    {BIT_SYSON_WEVT_WLAN_MSK, ON}, /* sleep: WLAN Wakeup */  
    {BIT_SYSON_WEVT_I2C1_ADDRMATCH_MSK, OFF}, /* sleep: ADC Wakeup */  
    {BIT_SYSON_WEVT_I2C0_ADDRMATCH_MSK, OFF}, /* sleep: I2C1 Slave RX address Wakeup */  
    {BIT_SYSON_WEVT_USB_MSK, OFF}, /* sleep: I2C0 Slave RX address Wakeup */  
    {BIT_SYSON_WEVT_GPIO_MSK, ON}, /* sleep: USB Wakeup */  
    {BIT_SYSON_WEVT_CHIP_EN_MSK, OFF}, /* sleep: ChipEN Wakeup */  
    {BIT_SYSON_WEVT_OVER_CURRENT_MSK, OFF}, /* sleep: REGU OVER_CURRENT Wakeup */  
    {BIT_SYSON_WEVT_GTIM_MSK, ON}, /* sleep: Gtimer 4/5 Wakeup */  
    {BIT_SYSON_WEVT_SYSTIM_MSK, ON}, /* dstandby: SYS Timer(ANA Timer) Wakeup */  
  
    {0xFFFFFFFF, OFF}, /* Table end */  
};
```



Wake PIN configuration

```
const WAKEPIN_TypeDef sleep_wakepin_config[]=
{
    //  Module        Status      Polarity
    {WAKUP_0,      OFF,        0}, /* wakeup_0: GPIOA_18 */
    {WAKUP_1,      ON,         0}, /* wakeup_1: GPIOA_5 */
    {WAKUP_2,      OFF,        0}, /* wakeup_2: GPIOA_22 */
    {WAKUP_2,      OFF,        0}, /* wakeup_3: GPIOA_23 */

    {0xFFFFFFFF,  OFF,        0}, /* Table end */
};
```



Thank you!