```
/*
  esp8266_peri.h - Peripheral registers exposed in more AVR style for esp8266

  Copyright (c) 2015 Hristo Gochkov. All rights reserved.
  This file is part of the esp8266 core for Arduino environment.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/
#ifndef ESP8266_PERI_H_INCLUDED
#define ESP8266_PERI_H_INCLUDED

// we expect mocking framework to provide these
#ifndef CORE_MOCK

#include "c_types.h"
#include "esp8266_undocumented.h"

#define ESP8266_REG(addr) *((volatile uint32_t *)(0x60000000+(addr)))
#define ESP8266_DREG(addr) *((volatile uint32_t *)(0x3FF00000+(addr)))
#define ESP8266_CLOCK 80000000UL

#define i2c_readReg_Mask(block, host_id, reg_add, Msb, Lsb)  rom_i2c_readReg_Mask(block, host_id,
reg_add, Msb, Lsb)
#define i2c_readReg_Mask_def(block, reg_add) i2c_readReg_Mask(block, block##_hostid,  reg_add,
reg_add##_msb,  reg_add##_lsb)
#define i2c_writeReg_Mask(block, host_id, reg_add, Msb, Lsb, indata)  rom_i2c_writeReg_Mask(block,
host_id, reg_add, Msb, Lsb, indata)
#define i2c_writeReg_Mask_def(block, reg_add, indata) i2c_writeReg_Mask(block, block##_hostid,
reg_add,  reg_add##_msb,  reg_add##_lsb,  indata)

//CPU Register
#define CPU2X     ESP8266_DREG(0x14) //when bit 0 is set, F_CPU = 160MHz

//OTP Registers
#define MAC0      ESP8266_DREG(0x50)
#define MAC1      ESP8266_DREG(0x54)
#define CHIPID    ESP8266_DREG(0x58)

//GPIO (0-15) Control Registers
#define GPO    ESP8266_REG(0x300) //GPIO_OUT R/W (Output Level)
#define GPOS   ESP8266_REG(0x304) //GPIO_OUT_SET WO
#define GPOC   ESP8266_REG(0x308) //GPIO_OUT_CLR WO
#define GPE    ESP8266_REG(0x30C) //GPIO_ENABLE R/W (Enable)
#define GPES   ESP8266_REG(0x310) //GPIO_ENABLE_SET WO
#define GPEC   ESP8266_REG(0x314) //GPIO_ENABLE_CLR WO
#define GPI    ESP8266_REG(0x318) //GPIO_IN RO (Read Input Level)
#define GPIE   ESP8266_REG(0x31C) //GPIO_STATUS R/W (Interrupt Enable)
#define GPIES  ESP8266_REG(0x320) //GPIO_STATUS_SET WO
#define GPIEC  ESP8266_REG(0x324) //GPIO_STATUS_CLR WO

#define GPOP(p) ((GPO & (1 << ((p) & 0xF))) != 0)
#define GPEP(p) ((GPE & (1 << ((p) & 0xF))) != 0)
```

```
#define GPIP(p) ((GPI & (1 << ((p) & 0xF))) != 0)
#define GPIEP(p) ((GPIE & (1 << ((p) & 0xF))) != 0)

//GPIO (0-15) PIN Control Registers
#define GPC(p) ESP8266_REG(0x328 + ((p & 0xF) * 4))
#define GPC0   ESP8266_REG(0x328) //GPIO_PIN0
#define GPC1   ESP8266_REG(0x32C) //GPIO_PIN1
#define GPC2   ESP8266_REG(0x330) //GPIO_PIN2
#define GPC3   ESP8266_REG(0x334) //GPIO_PIN3
#define GPC4   ESP8266_REG(0x338) //GPIO_PIN4
#define GPC5   ESP8266_REG(0x33C) //GPIO_PIN5
#define GPC6   ESP8266_REG(0x340) //GPIO_PIN6
#define GPC7   ESP8266_REG(0x344) //GPIO_PIN7
#define GPC8   ESP8266_REG(0x348) //GPIO_PIN8
#define GPC9   ESP8266_REG(0x34C) //GPIO_PIN9
#define GPC10  ESP8266_REG(0x350) //GPIO_PIN10
#define GPC11  ESP8266_REG(0x354) //GPIO_PIN11
#define GPC12  ESP8266_REG(0x358) //GPIO_PIN12
#define GPC13  ESP8266_REG(0x35C) //GPIO_PIN13
#define GPC14  ESP8266_REG(0x360) //GPIO_PIN14
#define GPC15  ESP8266_REG(0x364) //GPIO_PIN15

//GPIO (0-15) PIN Control Bits
#define GPCWE  10 //WAKEUP_ENABLE (can be 1 only when INT_TYPE is high or low)
#define GPCI   7  //INT_TYPE (3bits) 0:disable,1:rising,2:falling,3:change,4:low,5:high
#define GPCD   2  //DRIVER 0:normal,1:open drain
#define GPCS   0  //SOURCE 0:GPIO_DATA,1:SigmaDelta

#define GPMUX  ESP8266_REG(0x800)
//GPIO (0-15) PIN Function Registers
#define GPF0   ESP8266_REG(0x834)
#define GPF1   ESP8266_REG(0x818)
#define GPF2   ESP8266_REG(0x838)
#define GPF3   ESP8266_REG(0x814)
#define GPF4   ESP8266_REG(0x83C)
#define GPF5   ESP8266_REG(0x840)
#define GPF6   ESP8266_REG(0x81C)
#define GPF7   ESP8266_REG(0x820)
#define GPF8   ESP8266_REG(0x824)
#define GPF9   ESP8266_REG(0x828)
#define GPF10  ESP8266_REG(0x82C)
#define GPF11  ESP8266_REG(0x830)
#define GPF12  ESP8266_REG(0x804)
#define GPF13  ESP8266_REG(0x808)
#define GPF14  ESP8266_REG(0x80C)
#define GPF15  ESP8266_REG(0x810)

extern volatile uint32_t* const esp8266_gpioToFn[16];
#define GPF(p) (*esp8266_gpioToFn[(p & 0xF)])

//GPIO (0-15) PIN Function Bits
#define GPFSOE 0 //Sleep OE
#define GPFSS  1 //Sleep Sel
#define GPFSPD 2 //Sleep Pulldown
#define GPFSPU 3 //Sleep Pullup
#define GPFFS0 4 //Function Select bit 0
#define GPFFS1 5 //Function Select bit 1
#define GPFPD  6 //Pulldown
#define GPFPU  7 //Pullup
#define GPFFS2 8 //Function Select bit 2
#define GPFFS(f) (((((f) & 4) != 0) << GPFFS2) | ((((f) & 2) != 0) << GPFFS1) | ((((f) & 1) != 0) <<
GPFFS0))
#define GPFFS_GPIO(p) (((p)==0||(p)==2||(p)==4||(p)==5)?0:((p)==16)?1:3)
#define GPFFS_BUS(p) (((p)==1||(p)==3)?0:((p)==2||(p)==12||(p)==13||(p)==14||(p)==15)?2:((p)==0)?4:1)
```

```
//GPIO 16 Control Registers
#define GP16O  ESP8266_REG(0x768)
#define GP16E  ESP8266_REG(0x774)
#define GP16I  ESP8266_REG(0x78C)

//GPIO 16 PIN Control Register
#define GP16C  ESP8266_REG(0x790)
#define GPC16  GP16C

//GPIO 16 PIN Function Register
#define GP16F  ESP8266_REG(0x7A0)
#define GPF16  GP16F

//GPIO 16 PIN Function Bits
#define GP16FFS0 0 //Function Select bit 0
#define GP16FFS1 1 //Function Select bit 1
#define GP16FPD  3 //Pulldown
#define GP16FSPD 5 //Sleep Pulldown
#define GP16FFS2 6 //Function Select bit 2
#define GP16FFS(f) (((f) & 0x03) | (((f) & 0x04) << 4))

//Timer 1 Registers (23bit CountDown Timer)
#define T1L  ESP8266_REG(0x600) //Load Value (Starting Value of Counter) 23bit (0-8388607)
#define T1V  ESP8266_REG(0x604) //(RO) Current Value
#define T1C  ESP8266_REG(0x608) //Control Register
#define T1I  ESP8266_REG(0x60C) //Interrupt Status Register (1bit) write to clear
//edge interrupt enable register
#define TEIE    ESP8266_DREG(0x04)
#define TEIE1   0x02 //bit for timer 1

//Timer 2 Registers (32bit CountUp Timer)
#define T2L  ESP8266_REG(0x620) //Load Value (Starting Value of Counter)
#define T2V  ESP8266_REG(0x624) //(RO) Current Value
#define T2C  ESP8266_REG(0x628) //Control Register
#define T2I  ESP8266_REG(0x62C) //Interrupt Status Register (1bit) write to clear
#define T2A  ESP8266_REG(0x630) //Alarm Value

//Timer Control Bits
#define TCIS  8 //Interrupt Status
#define TCTE  7 //Timer Enable
#define TCAR  6 //AutoReload (restart timer when condition is reached)
#define TCPD  2 //Prescale Divider (2bit) 0:1(12.5ns/tick), 1:16(0.2us/tick), 2/3:256(3.2us/tick)
#define TCIT  0 //Interrupt Type 0:edge, 1:level

//RTC Registers
#define RTCSV     ESP8266_REG(0x704) //RTC SLEEP COUNTER Target Value
#define RTCCV     ESP8266_REG(0x71C) //RTC SLEEP COUNTER Value
#define RTCIS     ESP8266_REG(0x720) //RTC INT Status
#define RTCIC     ESP8266_REG(0x724) //RTC INT Clear
#define RTCIE     ESP8266_REG(0x728) //RTC INT Enable

#define RTC_USER_MEM ((volatile uint32_t*)0x60001200)

//IO SWAP Register
#define IOSWAP    ESP8266_DREG(0x28)
#define IOSWAPU   0 //Swaps UART
#define IOSWAPS   1 //Swaps SPI
#define IOSWAPU0  2 //Swaps UART 0 pins (u0rxd <-> u0cts), (u0txd <-> u0rts)
#define IOSWAPU1  3 //Swaps UART 1 pins (u1rxd <-> u1cts), (u1txd <-> u1rts)
#define IOSWAPHS  5 //Sets HSPI with higher prio
#define IOSWAP2HS 6 //Sets Two SPI Masters on HSPI
#define IOSWAP2CS 7 //Sets Two SPI Masters on CSPI

//UART INT Status
#define UIS   ESP8266_DREG(0x20020)
```

```
#define UIS0  0
#define UIS1  2

//UART 0 Registers
#define U0F   ESP8266_REG(0x000) //UART FIFO
#define U0IR  ESP8266_REG(0x004) //INT_RAW
#define U0IS  ESP8266_REG(0x008) //INT_STATUS
#define U0IE  ESP8266_REG(0x00c) //INT_ENABLE
#define U0IC  ESP8266_REG(0x010) //INT_CLEAR
#define U0D   ESP8266_REG(0x014) //CLKDIV
#define U0A   ESP8266_REG(0x018) //AUTOBAUD
#define U0S   ESP8266_REG(0x01C) //STATUS
#define U0C0  ESP8266_REG(0x020) //CONF0
#define U0C1  ESP8266_REG(0x024) //CONF1
#define U0LP  ESP8266_REG(0x028) //LOW_PULSE
#define U0HP  ESP8266_REG(0x02C) //HIGH_PULSE
#define U0PN  ESP8266_REG(0x030) //PULSE_NUM
#define U0DT  ESP8266_REG(0x078) //DATE
#define U0ID  ESP8266_REG(0x07C) //ID

//UART 1 Registers
#define U1F   ESP8266_REG(0xF00) //UART FIFO
#define U1IR  ESP8266_REG(0xF04) //INT_RAW
#define U1IS  ESP8266_REG(0xF08) //INT_STATUS
#define U1IE  ESP8266_REG(0xF0c) //INT_ENABLE
#define U1IC  ESP8266_REG(0xF10) //INT_CLEAR
#define U1D   ESP8266_REG(0xF14) //CLKDIV
#define U1A   ESP8266_REG(0xF18) //AUTOBAUD
#define U1S   ESP8266_REG(0xF1C) //STATUS
#define U1C0  ESP8266_REG(0xF20) //CONF0
#define U1C1  ESP8266_REG(0xF24) //CONF1
#define U1LP  ESP8266_REG(0xF28) //LOW_PULSE
#define U1HP  ESP8266_REG(0xF2C) //HIGH_PULSE
#define U1PN  ESP8266_REG(0xF30) //PULSE_NUM
#define U1DT  ESP8266_REG(0xF78) //DATE
#define U1ID  ESP8266_REG(0xF7C) //ID

//UART(uart) Registers
#define USF(u)   ESP8266_REG(0x000+(0xF00*(u&1))) //UART FIFO
#define USIR(u)  ESP8266_REG(0x004+(0xF00*(u&1))) //INT_RAW
#define USIS(u)  ESP8266_REG(0x008+(0xF00*(u&1))) //INT_STATUS
#define USIE(u)  ESP8266_REG(0x00c+(0xF00*(u&1))) //INT_ENABLE
#define USIC(u)  ESP8266_REG(0x010+(0xF00*(u&1))) //INT_CLEAR
#define USD(u)   ESP8266_REG(0x014+(0xF00*(u&1))) //CLKDIV
#define USA(u)   ESP8266_REG(0x018+(0xF00*(u&1))) //AUTOBAUD
#define USS(u)   ESP8266_REG(0x01C+(0xF00*(u&1))) //STATUS
#define USC0(u)  ESP8266_REG(0x020+(0xF00*(u&1))) //CONF0
#define USC1(u)  ESP8266_REG(0x024+(0xF00*(u&1))) //CONF1
#define USLP(u)  ESP8266_REG(0x028+(0xF00*(u&1))) //LOW_PULSE
#define USHP(u)  ESP8266_REG(0x02C+(0xF00*(u&1))) //HIGH_PULSE
#define USPN(u)  ESP8266_REG(0x030+(0xF00*(u&1))) //PULSE_NUM
#define USDT(u)  ESP8266_REG(0x078+(0xF00*(u&1))) //DATE
#define USID(u)  ESP8266_REG(0x07C+(0xF00*(u&1))) //ID

//UART INT Registers Bits
#define UITO    8 //RX FIFO TimeOut
#define UIBD    7 //Break Detected
#define UICTS   6 //CTS Changed
#define UIDSR   5 //DSR Change
#define UIOF    4 //RX FIFO OverFlow
#define UIFR    3 //Frame Error
#define UIPE    2 //Parity Error
#define UIFE    1 //TX FIFO Empty
#define UIFF    0 //RX FIFO Full
```

```
//UART STATUS Registers Bits
#define USTX    31 //TX PIN Level (Doesn't seem to work, always reads as 0 for both uarts. HW bug?
Possible workaround: Enable loopback UxC0 |= 1<<UCLBE and read USRXD, see
https://github.com/esp8266/Arduino/issues/7256 for discussion.)
#define USRTS   30 //RTS PIN Level
#define USDTR   29 //DTR PIN Level
#define USTXC   16 //TX FIFO COUNT (8bit)
#define USRXD   15 //RX PIN Level
#define USCTS   14 //CTS PIN Level
#define USDSR   13 //DSR PIN Level
#define USRXC    0 //RX FIFO COUNT (8bit)

//UART CONF0 Registers Bits
#define UCDTRI  24 //Invert DTR
#define UCRTSI  23 //Invert RTS
#define UCTXI   22 //Invert TX
#define UCDSRI  21 //Invert DSR
#define UCCTSI  20 //Invert CTS
#define UCRXI   19 //Invert RX
#define UCTXRST 18 //Reset TX FIFO
#define UCRXRST 17 //Reset RX FIFO
#define UCTXHFE 15 //TX Hardware Flow Enable
#define UCLBE   14 //LoopBack Enable
#define UCBRK   8  //Send Break on the TX line
#define UCSWDTR 7  //Set this bit to assert DTR
#define UCSWRTS 6  //Set this bit to assert RTS
#define UCSBN   4  //StopBits Count (2bit) 0:disable, 1:1bit, 2:1.5bit, 3:2bit
#define UCBN    2  //DataBits Count (2bin) 0:5bit, 1:6bit, 2:7bit, 3:8bit
#define UCPAE   1  //Parity Enable
#define UCPA    0  //Parity 0:even, 1:odd

//UART CONF1 Registers Bits
#define UCTOE   31 //RX TimeOut Enable
#define UCTOT   24 //RX TimeOut Threshold (7bit)
#define UCRXHFE 23 //RX Hardware Flow Enable
#define UCRXHFT 16 //RX Hardware Flow Threshold (7bit)
#define UCFET   8  //TX FIFO Empty Threshold (7bit)
#define UCFFT   0  //RX FIFO Full Threshold (7bit)

//WDT Feed (the dog) Register
#define WDTFEED    ESP8266_REG(0x914)
#define WDT_FEED() (WDTFEED = 0x73)

//SPI_READY
#define SPIRDY    ESP8266_DREG(0x0C)
#define SPI_BUSY   9 //wait SPI idle

//SPI0 Registers (SPI0 is used for the flash)
#define SPI0CMD         ESP8266_REG(0x200)
#define SPI0A           ESP8266_REG(0x204)
#define SPI0C           ESP8266_REG(0x208)
#define SPI0C1          ESP8266_REG(0x20C)
#define SPI0RS          ESP8266_REG(0x210)
#define SPI0C2          ESP8266_REG(0x214)
#define SPI0CLK         ESP8266_REG(0x218)
#define SPI0U           ESP8266_REG(0x21C)
#define SPI0U1          ESP8266_REG(0x220)
#define SPI0U2          ESP8266_REG(0x224)
#define SPI0WS          ESP8266_REG(0x228)
#define SPI0P           ESP8266_REG(0x22C)
#define SPI0S           ESP8266_REG(0x230)
#define SPI0S1          ESP8266_REG(0x234)
#define SPI0S2          ESP8266_REG(0x238)
#define SPI0S3          ESP8266_REG(0x23C)
#define SPI0W0          ESP8266_REG(0x240)
```

```
#define SPI0W1          ESP8266_REG(0x244)
#define SPI0W2          ESP8266_REG(0x248)
#define SPI0W3          ESP8266_REG(0x24C)
#define SPI0W4          ESP8266_REG(0x250)
#define SPI0W5          ESP8266_REG(0x254)
#define SPI0W6          ESP8266_REG(0x258)
#define SPI0W7          ESP8266_REG(0x25C)
#define SPI0W8          ESP8266_REG(0x260)
#define SPI0W9          ESP8266_REG(0x264)
#define SPI0W10         ESP8266_REG(0x268)
#define SPI0W11         ESP8266_REG(0x26C)
#define SPI0W12         ESP8266_REG(0x270)
#define SPI0W13         ESP8266_REG(0x274)
#define SPI0W14         ESP8266_REG(0x278)
#define SPI0W15         ESP8266_REG(0x27C)
#define SPI0E3          ESP8266_REG(0x2FC)
#define SPI0W(p)  ESP8266_REG(0x240 + ((p & 0xF) * 4))

//SPI1 Registers
#define SPI1CMD         ESP8266_REG(0x100)
#define SPI1A           ESP8266_REG(0x104)
#define SPI1C           ESP8266_REG(0x108)
#define SPI1C1          ESP8266_REG(0x10C)
#define SPI1RS          ESP8266_REG(0x110)
#define SPI1C2          ESP8266_REG(0x114)
#define SPI1CLK         ESP8266_REG(0x118)
#define SPI1U           ESP8266_REG(0x11C)
#define SPI1U1          ESP8266_REG(0x120)
#define SPI1U2          ESP8266_REG(0x124)
#define SPI1WS          ESP8266_REG(0x128)
#define SPI1P           ESP8266_REG(0x12C)
#define SPI1S           ESP8266_REG(0x130)
#define SPI1S1          ESP8266_REG(0x134)
#define SPI1S2          ESP8266_REG(0x138)
#define SPI1S3          ESP8266_REG(0x13C)
#define SPI1W0          ESP8266_REG(0x140)
#define SPI1W1          ESP8266_REG(0x144)
#define SPI1W2          ESP8266_REG(0x148)
#define SPI1W3          ESP8266_REG(0x14C)
#define SPI1W4          ESP8266_REG(0x150)
#define SPI1W5          ESP8266_REG(0x154)
#define SPI1W6          ESP8266_REG(0x158)
#define SPI1W7          ESP8266_REG(0x15C)
#define SPI1W8          ESP8266_REG(0x160)
#define SPI1W9          ESP8266_REG(0x164)
#define SPI1W10         ESP8266_REG(0x168)
#define SPI1W11         ESP8266_REG(0x16C)
#define SPI1W12         ESP8266_REG(0x170)
#define SPI1W13         ESP8266_REG(0x174)
#define SPI1W14         ESP8266_REG(0x178)
#define SPI1W15         ESP8266_REG(0x17C)
#define SPI1E0          ESP8266_REG(0x1F0)
#define SPI1E1          ESP8266_REG(0x1F4)
#define SPI1E2          ESP8266_REG(0x1F8)
#define SPI1E3          ESP8266_REG(0x1FC)
#define SPI1W(p)  ESP8266_REG(0x140 + ((p & 0xF) * 4))

//SPI0, SPI1 & I2S Interrupt Register
#define SPIIR     ESP8266_DREG(0x20)
#define SPII0     4 //SPI0 Interrupt
#define SPII1     7 //SPI1 Interrupt
#define SPII2     9 //I2S Interrupt

//SPI CMD
#define SPICMDREAD (1 << 31) //SPI_FLASH_READ
```

```
#define SPICMDWREN (1 << 30) //SPI_FLASH_WREN
#define SPICMDWRDI (1 << 29) //SPI_FLASH_WRDI
#define SPICMDRDID (1 << 28) //SPI_FLASH_RDID
#define SPICMDRDSR (1 << 27) //SPI_FLASH_RDSR
#define SPICMDWRSR (1 << 26) //SPI_FLASH_WRSR
#define SPICMDPP  (1 << 25) //SPI_FLASH_PP
#define SPICMDSE  (1 << 24) //SPI_FLASH_SE
#define SPICMDBE  (1 << 23) //SPI_FLASH_BE
#define SPICMDCE  (1 << 22) //SPI_FLASH_CE
#define SPICMDDP  (1 << 21) //SPI_FLASH_DP
#define SPICMDRES (1 << 20) //SPI_FLASH_RES
#define SPICMDHPM (1 << 19) //SPI_FLASH_HPM
#define SPICMDUSR (1 << 18) //SPI_FLASH_USR
#define SPIBUSY         (1 << 18) //SPI_USR

//SPI CTRL (SPIxC)
#define SPICWBO       (1 << 26) //SPI_WR_BIT_ODER
#define SPICRBO       (1 << 25) //SPI_RD_BIT_ODER
#define SPICQIO       (1 << 24) //SPI_QIO_MODE
#define SPICDIO       (1 << 23) //SPI_DIO_MODE
#define SPIC2BSE      (1 << 22) //SPI_TWO_BYTE_STATUS_EN
#define SPICWPR       (1 << 21) //SPI_WP_REG
#define SPICQOUT      (1 << 20) //SPI_QOUT_MODE
#define SPICSHARE     (1 << 19) //SPI_SHARE_BUS
#define SPICHOLD      (1 << 18) //SPI_HOLD_MODE
#define SPICAHB       (1 << 17) //SPI_ENABLE_AHB
#define SPICSSTAAI    (1 << 16) //SPI_SST_AAI
#define SPICRESANDRES (1 << 15) //SPI_RESANDRES
#define SPICDOUT      (1 << 14) //SPI_DOUT_MODE
#define SPICFASTRD    (1 << 13) //SPI_FASTRD_MODE

//SPI CTRL1 (SPIxC1)
#define SPIC1TCSH   0xF //SPI_T_CSH
#define SPIC1TCSH_S 28 //SPI_T_CSH_S
#define SPIC1TRES   0xFFF //SPI_T_RES
#define SPIC1TRES_S 16 //SPI_T_RES_S
#define SPIC1BTL    0xFFFF //SPI_BUS_TIMER_LIMIT
#define SPIC1BTL_S  0 //SPI_BUS_TIMER_LIMIT_S

//SPI Status (SPIxRS)
#define SPIRSEXT    0xFF //SPI_STATUS_EXT
#define SPIRSEXT_S  24 //SPI_STATUS_EXT_S
#define SPIRSWB     0xFF //SPI_WB_MODE
#define SPIRSWB_S   16 //SPI_WB_MODE_S
#define SPIRSSP     (1 << 7) //SPI_FLASH_STATUS_PRO_FLAG
#define SPIRSTBP    (1 << 5) //SPI_FLASH_TOP_BOT_PRO_FLAG
#define SPIRSBP2    (1 << 4) //SPI_FLASH_BP2
#define SPIRSBP1    (1 << 3) //SPI_FLASH_BP1
#define SPIRSBP0    (1 << 2) //SPI_FLASH_BP0
#define SPIRSWRE    (1 << 1) //SPI_FLASH_WRENABLE_FLAG
#define SPIRSBUSY   (1 << 0) //SPI_FLASH_BUSY_FLAG

//SPI CTRL2 (SPIxC2)
#define SPIC2CSDN     0xF //SPI_CS_DELAY_NUM
#define SPIC2CSDN_S   28  //SPI_CS_DELAY_NUM_S
#define SPIC2CSDM     0x3 //SPI_CS_DELAY_MODE
#define SPIC2CSDM_S   26  //SPI_CS_DELAY_MODE_S
#define SPIC2MOSIDN   0x7 //SPI_MOSI_DELAY_NUM
#define SPIC2MOSIDN_S 23  //SPI_MOSI_DELAY_NUM_S
#define SPIC2MOSIDM   0x3 //SPI_MOSI_DELAY_MODE
#define SPIC2MOSIDM_S 21  //SPI_MOSI_DELAY_MODE_S
#define SPIC2MISODN   0x7 //SPI_MISO_DELAY_NUM
#define SPIC2MISODN_S 18  //SPI_MISO_DELAY_NUM_S
#define SPIC2MISODM   0x3 //SPI_MISO_DELAY_MODE
#define SPIC2MISODM_S 16  //SPI_MISO_DELAY_MODE_S
```

```
#define SPIC2CKOHM    0xF //SPI_CK_OUT_HIGH_MODE
#define SPIC2CKOHM_S  12  //SPI_CK_OUT_HIGH_MODE_S
#define SPIC2CKOLM    0xF //SPI_CK_OUT_LOW_MODE
#define SPIC2CKOLM_S  8   //SPI_CK_OUT_LOW_MODE_S
#define SPIC2HT       0xF //SPI_HOLD_TIME
#define SPIC2HT_S     4   //SPI_HOLD_TIME_S
#define SPIC2ST       0xF //SPI_SETUP_TIME
#define SPIC2ST_S     0   //SPI_SETUP_TIME_S

//SPI CLK (SPIxCLK)
#define SPICLK_EQU_SYSCLK (1 << 31) //SPI_CLK_EQU_SYSCLK
#define SPICLKDIVPRE 0x1FFF //SPI_CLKDIV_PRE
#define SPICLKDIVPRE_S 18 //SPI_CLKDIV_PRE_S
#define SPICLKCN 0x3F //SPI_CLKCNT_N
#define SPICLKCN_S 12 //SPI_CLKCNT_N_S
#define SPICLKCH 0x3F //SPI_CLKCNT_H
#define SPICLKCH_S 6 //SPI_CLKCNT_H_S
#define SPICLKCL 0x3F //SPI_CLKCNT_L
#define SPICLKCL_S 0 //SPI_CLKCNT_L_S

//SPI Phases (SPIxU)
#define SPIUCOMMAND     (1 << 31) //COMMAND pahse, SPI_USR_COMMAND
#define SPIUADDR        (1 << 30) //ADDRESS phase, SPI_FLASH_USR_ADDR
#define SPIUDUMMY       (1 << 29) //DUMMY phase, SPI_FLASH_USR_DUMMY
#define SPIUMISO        (1 << 28) //MISO phase, SPI_FLASH_USR_DIN
#define SPIUMOSI        (1 << 27) //MOSI phase, SPI_FLASH_DOUT
#define SPIUDUMMYIDLE (1 << 26) //SPI_USR_DUMMY_IDLE
#define SPIUMOSIH (1 << 25) //MOSI phase uses W8-W15, SPI_USR_DOUT_HIGHPART
#define SPIUMISOH (1 << 24) //MISO pahse uses W8-W15, SPI_USR_DIN_HIGHPART
#define SPIUPREPHOLD (1 << 23) //SPI_USR_PREP_HOLD
#define SPIUCMDHOLD (1 << 22) //SPI_USR_CMD_HOLD
#define SPIUADDRHOLD (1 << 21) //SPI_USR_ADDR_HOLD
#define SPIUDUMMYHOLD (1 << 20) //SPI_USR_DUMMY_HOLD
#define SPIUMISOHOLD (1 << 19) //SPI_USR_DIN_HOLD
#define SPIUMOSIHOLD (1 << 18) //SPI_USR_DOUT_HOLD
#define SPIUHOLDPOL (1 << 17) //SPI_USR_HOLD_POL
#define SPIUSIO (1 << 16) //SPI_SIO
#define SPIUFWQIO (1 << 15) //SPI_FWRITE_QIO
#define SPIUFWDIO (1 << 14) //SPI_FWRITE_DIO
#define SPIUFWQUAD (1 << 13) //SPI_FWRITE_QUAD
#define SPIUFWDUAL (1 << 12) //SPI_FWRITE_DUAL
#define SPIUWRBYO (1 << 11) //SPI_WR_BYTE_ORDER
#define SPIURDBYO (1 << 10) //SPI_RD_BYTE_ORDER
#define SPIUAHBEM 0x3 //SPI_AHB_ENDIAN_MODE
#define SPIUAHBEM_S 8 //SPI_AHB_ENDIAN_MODE_S
#define SPIUSME (1 << 7) //SPI Master Edge (0:falling, 1:rising), SPI_CK_OUT_EDGE
#define SPIUSSE   (1 << 6) //SPI Slave Edge (0:falling, 1:rising), SPI_CK_I_EDGE
#define SPIUCSSETUP (1 << 5) //SPI_CS_SETUP
#define SPIUCSHOLD (1 << 4) //SPI_CS_HOLD
#define SPIUAHBUCMD (1 << 3) //SPI_AHB_USR_COMMAND
#define SPIUAHBUCMD4B (1 << 1) //SPI_AHB_USR_COMMAND_4BYTE
#define SPIUDUPLEX (1 << 0) //SPI_DOUTDIN

//SPI Phase Length Locations
#define SPILCOMMAND     28 //4 bit in SPIxU2 default 7 (8bit)
#define SPILADDR        26 //6 bit in SPIxU1 default:23 (24bit)
#define SPILDUMMY       0  //8 bit in SPIxU1 default:0 (0 cycles)
#define SPILMISO        8  //9 bit in SPIxU1 default:0 (1bit)
#define SPILMOSI        17 //9 bit in SPIxU1 default:0 (1bit)
//SPI Phase Length Masks
#define SPIMCOMMAND     0xF
#define SPIMADDR        0x3F
#define SPIMDUMMY       0xFF
#define SPIMMISO        0x1FF
#define SPIMMOSI        0x1FF
```

```
//SPI Slave (SPIxS)
#define SPISSRES    (1 << 31) //SYNC RESET, SPI_SYNC_RESET
#define SPISE       (1 << 30) //Slave Enable, SPI_SLAVE_MODE
#define SPISBE      (1 << 29) //WR/RD BUF enable, SPI_SLV_WR_RD_BUF_EN
#define SPISSE      (1 << 28) //STA enable, SPI_SLV_WR_RD_STA_EN
#define SPISCD      (1 << 27) //CMD define, SPI_SLV_CMD_DEFINE
#define SPISTRCNT   0xF //SPI_TRANS_CNT
#define SPISTRCNT_S 23 //SPI_TRANS_CNT_S
#define SPISSLS     0x7 //SPI_SLV_LAST_STATE
#define SPISSLS_S   20 //SPI_SLV_LAST_STATE_S
#define SPISSLC     0x7 //SPI_SLV_LAST_COMMAND
#define SPISSLC_S   17 //SPI_SLV_LAST_COMMAND_S
#define SPISCSIM    0x3 //SPI_CS_I_MODE
#define SPIDCSIM_S  10 //SPI_CS_I_MODE_S
#define SPISTRIE    (1 << 9)  //TRANS interrupt enable
#define SPISWSIE    (1 << 8)  //WR_STA interrupt enable
#define SPISRSIE    (1 << 7)  //RD_STA interrupt enable
#define SPISWBIE    (1 << 6)  //WR_BUF interrupt enable
#define SPISRBIE    (1 << 5)  //RD_BUF interrupt enable
#define SPISTRIS    (1 << 4)  //TRANS interrupt status
#define SPISWSIS    (1 << 3)  //WR_STA interrupt status
#define SPISRSIS    (1 << 2)  //RD_STA interrupt status
#define SPISWBIS    (1 << 1)  //WR_BUF interrupt status
#define SPISRBIS    (1 << 0)  //RD_BUF interrupt status

//SPI Slave1 (SPIxS1)
#define SPIS1LSTA   27 //5 bit in SPIxS1 default:0 (1bit), SPI_SLV_STATUS_BITLEN
#define SPIS1FE     (1 << 26) //SPI_SLV_STATUS_FAST_EN
#define SPIS1RSTA   (1 << 25) //default:0 enable STA read from Master, SPI_SLV_STATUS_READBACK
#define SPIS1LBUF   16 //9 bit in SPIxS1 default:0 (1bit), SPI_SLV_BUF_BITLEN
#define SPIS1LRBA   10 //6 bit in SPIxS1 default:0 (1bit), SPI_SLV_RD_ADDR_BITLEN
#define SPIS1LWBA   4  //6 bit in SPIxS1 default:0 (1bit), SPI_SLV_WR_ADDR_BITLEN
#define SPIS1WSDE   (1 << 3) //SPI_SLV_WRSTA_DUMMY_EN
#define SPIS1RSDE   (1 << 2) //SPI_SLV_RDSTA_DUMMY_EN
#define SPIS1WBDE   (1 << 1) //SPI_SLV_WRBUF_DUMMY_EN
#define SPIS1RBDE   (1 << 0) //SPI_SLV_RDBUF_DUMMY_EN

//SPI Slave2 (SPIxS2)
#define SPIS2WBDL 0xFF //SPI_SLV_WRBUF_DUMMY_CYCLELEN
#define SPIS2WBDL_S 24 //SPI_SLV_WRBUF_DUMMY_CYCLELEN_S
#define SPIS2RBDL 0xFF //SPI_SLV_RDBUF_DUMMY_CYCLELEN
#define SPIS2RBDL_S 16 //SPI_SLV_RDBUF_DUMMY_CYCLELEN_S
#define SPIS2WSDL 0xFF //SPI_SLV_WRSTA_DUMMY_CYCLELEN
#define SPIS2WSDL_S 8 //SPI_SLV_WRSTA_DUMMY_CYCLELEN_S
#define SPIS2RSDL 0xFF //SPI_SLV_RDSTA_DUMMY_CYCLELEN
#define SPIS2RSDL_S 0 //SPI_SLV_RDSTA_DUMMY_CYCLELEN_S

//SPI Slave3 (SPIxS3)
#define SPIS3WSCV 0xFF //SPI_SLV_WRSTA_CMD_VALUE
#define SPIS3WSCV_S 24 //SPI_SLV_WRSTA_CMD_VALUE_S
#define SPIS3RSCV 0xFF //SPI_SLV_RDSTA_CMD_VALUE
#define SPIS3RSCV_S 16 //SPI_SLV_RDSTA_CMD_VALUE_S
#define SPIS3WBCV 0xFF //SPI_SLV_WRBUF_CMD_VALUE
#define SPIS3WBCV_S 8 //SPI_SLV_WRBUF_CMD_VALUE_S
#define SPIS3RBCV 0xFF //SPI_SLV_RDBUF_CMD_VALUE
#define SPIS3RBCV_S 0 //SPI_SLV_RDBUF_CMD_VALUE_S

//SPI EXT0 (SPIxE0)
#define SPIE0TPPEN (1 << 31) //SPI_T_PP_ENA
#define SPIE0TPPS 0xF //SPI_T_PP_SHIFT
#define SPIE0TPPS_S 16 //SPI_T_PP_SHIFT_S
#define SPIE0TPPT 0xFFF //SPI_T_PP_TIME
#define SPIE0TPPT_S 0 //SPI_T_PP_TIME_S
```

```
//SPI EXT1 (SPIxE1)
#define SPIE1TEREN (1 << 31) //SPI_T_ERASE_ENA
#define SPIE1TERS 0xF //SPI_T_ERASE_SHIFT
#define SPIE1TERS_S 16 //SPI_T_ERASE_SHIFT_S
#define SPIE1TERT 0xFFF //SPI_T_ERASE_TIME
#define SPIE1TERT_S 0 //SPI_T_ERASE_TIME_S

//SPI EXT2 (SPIxE2)
#define SPIE2ST 0x7 //SPI_ST
#define SPIE2ST_S 0 //SPI_ST_S

//SPI EXT3 (SPIxE3)
#define SPIE2IHEN 0x3 //SPI_INT_HOLD_ENA
#define SPIE2IHEN_S 0 //SPI_INT_HOLD_ENA_S

//SPI PIN (SPIxP)
#define SPIPCS2DIS (1 << 2)
#define SPIPCS1DIS (1 << 1)
#define SPIPCS0DIS (1 << 0)

//SLC (DMA) Registers
#define SLCC0     ESP8266_REG(0xB00) //SLC_CONF0
#define SLCIR     ESP8266_REG(0xB04) //SLC_INT_RAW
#define SLCIS     ESP8266_REG(0xB08) //SLC_INT_STATUS
#define SLCIE     ESP8266_REG(0xB0C) //SLC_INT_ENA
#define SLCIC     ESP8266_REG(0xB10) //SLC_INT_CLR
#define SLCRXS    ESP8266_REG(0xB14) //SLC_RX_STATUS
#define SLCRXP    ESP8266_REG(0xB18) //SLC_RX_FIFO_PUSH
#define SLCTXS    ESP8266_REG(0xB1C) //SLC_TX_STATUS
#define SLCTXP    ESP8266_REG(0xB20) //SLC_TX_FIFO_POP
#define SLCRXL    ESP8266_REG(0xB24) //SLC_RX_LINK
#define SLCTXL    ESP8266_REG(0xB28) //SLC_TX_LINK
#define SLCIVTH   ESP8266_REG(0xB2C) //SLC_INTVEC_TOHOST
#define SLCT0     ESP8266_REG(0xB30) //SLC_TOKEN0
#define SLCT1     ESP8266_REG(0xB34) //SLC_TOKEN1
#define SLCC1     ESP8266_REG(0xB38) //SLC_CONF1
#define SLCS0     ESP8266_REG(0xB3C) //SLC_STATE0
#define SLCS1     ESP8266_REG(0xB40) //SLC_STATE1
#define SLCBC     ESP8266_REG(0xB44) //SLC_BRIDGE_CONF
#define SLCRXEDA  ESP8266_REG(0xB48) //SLC_RX_EOF_DES_ADDR
#define SLCTXEDA  ESP8266_REG(0xB4C) //SLC_TX_EOF_DES_ADDR
#define SLCRXEBDA ESP8266_REG(0xB50) //SLC_RX_EOF_BFR_DES_ADDR
#define SLCAT     ESP8266_REG(0xB54) //SLC_AHB_TEST
#define SLCSS     ESP8266_REG(0xB58) //SLC_SDIO_ST
#define SLCRXDC   ESP8266_REG(0xB5C) //SLC_RX_DSCR_CONF
#define SLCTXD    ESP8266_REG(0xB60) //SLC_TXLINK_DSCR
#define SLCTXDB0  ESP8266_REG(0xB64) //SLC_TXLINK_DSCR_BF0
#define SLCTXDB1  ESP8266_REG(0xB68) //SLC_TXLINK_DSCR_BF1
#define SLCRXD    ESP8266_REG(0xB6C) //SLC_RXLINK_DSCR
#define SLCRXDB0  ESP8266_REG(0xB70) //SLC_RXLINK_DSCR_BF0
#define SLCRXDB1  ESP8266_REG(0xB74) //SLC_RXLINK_DSCR_BF1
#define SLCDT     ESP8266_REG(0xB78) //SLC_DATE
#define SLCID     ESP8266_REG(0xB7C) //SLC_ID
#define SLCHIR    ESP8266_REG(0xB88) //SLC_HOST_INTR_RAW
#define SLCHC0    ESP8266_REG(0xB94) //SLC_HOST_CONF_W0
#define SLCHC1    ESP8266_REG(0xB98) //SLC_HOST_CONF_W1
#define SLCHIS    ESP8266_REG(0xB9C) //SLC_HOST_INTR_ST
#define SLCHC2    ESP8266_REG(0xBA0) //SLC_HOST_CONF_W2
#define SLCHC3    ESP8266_REG(0xBA4) //SLC_HOST_CONF_W3
#define SLCHC4    ESP8266_REG(0xBA8) //SLC_HOST_CONF_W4
#define SLCHIC    ESP8266_REG(0xBB0) //SLC_HOST_INTR_CLR
#define SLCHIE    ESP8266_REG(0xBB4) //SLC_HOST_INTR_ENA
#define SLCHC5    ESP8266_REG(0xBBC) //SLC_HOST_CONF_W5

//SLC (DMA) CONF0
```

```
#define SLCMM     (0x3) //SLC_MODE
#define SLCM      (12) //SLC_MODE_S
#define SLCDTBE   (1 << 9) //SLC_DATA_BURST_EN
#define SLCDBE    (1 << 8) //SLC_DSCR_BURST_EN
#define SLCRXNRC  (1 << 7) //SLC_RX_NO_RESTART_CLR
#define SLCRXAW   (1 << 6) //SLC_RX_AUTO_WRBACK
#define SLCRXLT   (1 << 5) //SLC_RX_LOOP_TEST
#define SLCTXLT   (1 << 4) //SLC_TX_LOOP_TEST
#define SLCAR     (1 << 3) //SLC_AHBM_RST
#define SLCAFR    (1 << 2) //SLC_AHBM_FIFO_RST
#define SLCRXLR   (1 << 1) //SLC_RXLINK_RST
#define SLCTXLR   (1 << 0) //SLC_TXLINK_RST

//SLC (DMA) INT
#define SLCITXDE  (1 << 21) //SLC_TX_DSCR_EMPTY_INT
#define SLCIRXDER (1 << 20) //SLC_RX_DSCR_ERR_INT
#define SLCITXDER (1 << 19) //SLC_TX_DSCR_ERR_INT
#define SLCITH    (1 << 18) //SLC_TOHOST_INT
#define SLCIRXEOF (1 << 17) //SLC_RX_EOF_INT
#define SLCIRXD   (1 << 16) //SLC_RX_DONE_INT
#define SLCITXEOF (1 << 15) //SLC_TX_EOF_INT
#define SLCITXD   (1 << 14) //SLC_TX_DONE_INT
#define SLCIT0    (1 << 13) //SLC_TOKEN1_1TO0_INT
#define SLCIT1    (1 << 12) //SLC_TOKEN0_1TO0_INT
#define SLCITXO   (1 << 11) //SLC_TX_OVF_INT
#define SLCIRXU   (1 << 10) //SLC_RX_UDF_INT
#define SLCITXS   (1 << 9) //SLC_TX_START_INT
#define SLCIRXS   (1 << 8) //SLC_RX_START_INT
#define SLCIFH7   (1 << 7) //SLC_FRHOST_BIT7_INT
#define SLCIFH6   (1 << 6) //SLC_FRHOST_BIT6_INT
#define SLCIFH5   (1 << 5) //SLC_FRHOST_BIT5_INT
#define SLCIFH4   (1 << 4) //SLC_FRHOST_BIT4_INT
#define SLCIFH3   (1 << 3) //SLC_FRHOST_BIT3_INT
#define SLCIFH2   (1 << 2) //SLC_FRHOST_BIT2_INT
#define SLCIFH1   (1 << 1) //SLC_FRHOST_BIT1_INT
#define SLCIFH0   (1 << 0) //SLC_FRHOST_BIT0_INT

//SLC (DMA) RX_STATUS
#define SLCRXE    (1 << 1) //SLC_RX_EMPTY
#define SLCRXF    (1 << 0) //SLC_RX_FULL

//SLC (DMA) TX_STATUS
#define SLCTXE    (1 << 1) //SLC_TX_EMPTY
#define SLCTXF    (1 << 0) //SLC_TX_FULL

//SLC (DMA) RX_FIFO_PUSH
#define SLCRXFP   (1 << 16) //SLC_RXFIFO_PUSH
#define SLCRXWDM  (0x1FF)   //SLC_RXFIFO_WDATA
#define SLCRXWD   (0)       //SLC_RXFIFO_WDATA_S

//SLC (DMA) TX_FIFO_POP
#define SLCTXFP   (1 << 16) //SLC_TXFIFO_POP
#define SLCTXRDM  (0x7FF)   //SLC_TXFIFO_RDATA
#define SLCTXRD   (0)       //SLC_TXFIFO_RDATA_S

//SLC (DMA) RX_LINK
#define SLCRXLP   (1 << 31) //SLC_RXLINK_PARK
#define SLCRXLRS  (1 << 30) //SLC_RXLINK_RESTART
#define SLCRXLS   (1 << 29) //SLC_RXLINK_START
#define SLCRXLE   (1 << 28) //SLC_RXLINK_STOP
#define SLCRXLAM  (0xFFFF)  //SLC_RXLINK_DESCADDR_MASK
#define SLCRXLA   (0)       //SLC_RXLINK_ADDR_S

//SLC (DMA) TX_LINK
#define SLCTXLP   (1 << 31) //SLC_TXLINK_PARK
```

```
#define SLCTXLRS  (1 << 30) //SLC_TXLINK_RESTART
#define SLCTXLS   (1 << 29) //SLC_TXLINK_START
#define SLCTXLE   (1 << 28) //SLC_TXLINK_STOP
#define SLCTXLAM  (0xFFFF)  //SLC_TXLINK_DESCADDR_MASK
#define SLCTXLA   (0)       //SLC_TXLINK_ADDR_S

//SLC (DMA) TOKENx
#define SLCTM  (0xFFF)   //SLC_TOKENx_MASK
#define SLCTT  (16)      //SLC_TOKENx_S
#define SLCTIM (1 << 14) //SLC_TOKENx_LOCAL_INC_MORE
#define SLCTI  (1 << 13) //SLC_TOKENx_LOCAL_INC
#define SLCTW  (1 << 12) //SLC_TOKENx_LOCAL_WR
#define SLCTDM (0xFFF)   //SLC_TOKENx_LOCAL_WDATA
#define SLCTD  (0)       //SLC_TOKENx_LOCAL_WDATA_S

//SLC (DMA) BRIDGE_CONF
#define SLCBFMEM  (0xF)   //SLC_FIFO_MAP_ENA
#define SLCBFME   (8)     //SLC_FIFO_MAP_ENA_S
#define SLCBTEEM  (0x3F)  //SLC_TXEOF_ENA
#define SLCBTEE   (0)     //SLC_TXEOF_ENA_S

//SLC (DMA) AHB_TEST
#define SLCATAM  (0x3)  //SLC_AHB_TESTADDR
#define SLCATA   (4)    //SLC_AHB_TESTADDR_S
#define SLCATMM  (0x7)  //SLC_AHB_TESTMODE
#define SLCATM   (0)    //SLC_AHB_TESTMODE_S

//SLC (DMA) SDIO_ST
#define SLCSBM  (0x7)     //SLC_BUS_ST
#define SLCSB   (12)      //SLC_BUS_ST_S
#define SLCSW   (1 << 8)  //SLC_SDIO_WAKEUP
#define SLCSFM  (0xF)     //SLC_FUNC_ST
#define SLCSF   (4)       //SLC_FUNC_ST_S
#define SLCSCM  (0x7)     //SLC_CMD_ST
#define SLCSC   (0)       //SLC_CMD_ST_S

//SLC (DMA) RX_DSCR_CONF
#define SLCBRXFE  (1 << 20) //SLC_RX_FILL_EN
#define SLCBRXEM  (1 << 19) //SLC_RX_EOF_MODE
#define SLCBRXFM  (1 << 18) //SLC_RX_FILL_MODE
#define SLCBINR   (1 << 17) //SLC_INFOR_NO_REPLACE
#define SLCBTNR   (1 << 16) //SLC_TOKEN_NO_REPLACE
#define SLCBPICM  (0xFFFF)  //SLC_POP_IDLE_CNT
#define SLCBPIC   (0)       //SLC_POP_IDLE_CNT_S


// I2S Registers
#define i2c_bbpll                       0x67
#define i2c_bbpll_hostid                4
#define i2c_bbpll_en_audio_clock_out    4
#define i2c_bbpll_en_audio_clock_out_msb  7
#define i2c_bbpll_en_audio_clock_out_lsb  7
#define I2S_CLK_ENABLE()                i2c_writeReg_Mask_def(i2c_bbpll,
i2c_bbpll_en_audio_clock_out, 1)
#define I2SBASEFREQ                     (160000000L)

#define I2STXF  ESP8266_REG(0xe00) //I2STXFIFO (32bit)
#define I2SRXF  ESP8266_REG(0xe04) //I2SRXFIFO (32bit)
#define I2SC    ESP8266_REG(0xe08) //I2SCONF
#define I2SIR   ESP8266_REG(0xe0C) //I2SINT_RAW
#define I2SIS   ESP8266_REG(0xe10) //I2SINT_ST
#define I2SIE   ESP8266_REG(0xe14) //I2SINT_ENA
#define I2SIC   ESP8266_REG(0xe18) //I2SINT_CLR
#define I2ST    ESP8266_REG(0xe1C) //I2STIMING
#define I2SFC   ESP8266_REG(0xe20) //I2S_FIFO_CONF
```

```
#define I2SRXEN ESP8266_REG(0xe24) //I2SRXEOF_NUM (32bit)
#define I2SCSD  ESP8266_REG(0xe28) //I2SCONF_SIGLE_DATA (32bit)
#define I2SCC   ESP8266_REG(0xe2C) //I2SCONF_CHAN

// I2S CONF
#define I2SBDM  (0x3F)    //I2S_BCK_DIV_NUM
#define I2SBD   (22)      //I2S_BCK_DIV_NUM_S
#define I2SCDM  (0x3F)    //I2S_CLKM_DIV_NUM
#define I2SCD   (16)      //I2S_CLKM_DIV_NUM_S
#define I2SBMM  (0xF)     //I2S_BITS_MOD
#define I2SBM   (12)      //I2S_BITS_MOD_S
#define I2SRMS  (1 << 11) //I2S_RECE_MSB_SHIFT
#define I2STMS  (1 << 10) //I2S_TRANS_MSB_SHIFT
#define I2SRXS  (1 << 9)  //I2S_I2S_RX_START
#define I2STXS  (1 << 8)  //I2S_I2S_TX_START
#define I2SMR   (1 << 7)  //I2S_MSB_RIGHT
#define I2SRF   (1 << 6)  //I2S_RIGHT_FIRST
#define I2SRSM  (1 << 5)  //I2S_RECE_SLAVE_MOD
#define I2STSM  (1 << 4)  //I2S_TRANS_SLAVE_MOD
#define I2SRXFR (1 << 3)  //I2S_I2S_RX_FIFO_RESET
#define I2STXFR (1 << 2)  //I2S_I2S_TX_FIFO_RESET
#define I2SRXR  (1 << 1)  //I2S_I2S_RX_RESET
#define I2STXR  (1 << 0)  //I2S_I2S_TX_RESET
#define I2SRST  (0xF)     //I2S_I2S_RESET_MASK

//I2S INT
#define I2SITXRE  (1 << 5) //I2S_I2S_TX_REMPTY_INT
#define I2SITXWF  (1 << 4) //I2S_I2S_TX_WFULL_INT
#define I2SIRXRE  (1 << 3) //I2S_I2S_RX_REMPTY_INT
#define I2SIRXWF  (1 << 2) //I2S_I2S_RX_WFULL_INT
#define I2SITXPD  (1 << 1) //I2S_I2S_TX_PUT_DATA_INT
#define I2SIRXTD  (1 << 0) //I2S_I2S_RX_TAKE_DATA_INT

//I2S TIMING
#define I2STBII  (1 << 22)  //I2S_TRANS_BCK_IN_INV
#define I2SRDS   (1 << 21)  //I2S_RECE_DSYNC_SW
#define I2STDS   (1 << 20)  //I2S_TRANS_DSYNC_SW
#define I2SRBODM (0x3)      //I2S_RECE_BCK_OUT_DELAY
#define I2SRBOD  (18)       //I2S_RECE_BCK_OUT_DELAY_S
#define I2SRWODM (0x3)      //I2S_RECE_WS_OUT_DELAY
#define I2SRWOD  (16)       //I2S_RECE_WS_OUT_DELAY_S
#define I2STSODM (0x3)      //I2S_TRANS_SD_OUT_DELAY
#define I2STSOD  (14)       //I2S_TRANS_SD_OUT_DELAY_S
#define I2STWODM (0x3)      //I2S_TRANS_WS_OUT_DELAY
#define I2STWOD  (12)       //I2S_TRANS_WS_OUT_DELAY_S
#define I2STBODM (0x3)      //I2S_TRANS_BCK_OUT_DELAY
#define I2STBOD  (10)       //I2S_TRANS_BCK_OUT_DELAY_S
#define I2SRSIDM (0x3)      //I2S_RECE_SD_IN_DELAY
#define I2SRSID  (8)        //I2S_RECE_SD_IN_DELAY_S
#define I2SRWIDM (0x3)      //I2S_RECE_WS_IN_DELAY
#define I2SRWID  (6)        //I2S_RECE_WS_IN_DELAY_S
#define I2SRBIDM (0x3)      //I2S_RECE_BCK_IN_DELAY
#define I2SRBID  (4)        //I2S_RECE_BCK_IN_DELAY_S
#define I2STWIDM (0x3)      //I2S_TRANS_WS_IN_DELAY
#define I2STWID  (2)        //I2S_TRANS_WS_IN_DELAY_S
#define I2STBIDM (0x3)      //I2S_TRANS_BCK_IN_DELAY
#define I2STBID  (0)        //I2S_TRANS_BCK_IN_DELAY_S

//I2S FIFO CONF
#define I2SRXFMM (0x7)      //I2S_I2S_RX_FIFO_MOD
#define I2SRXFM  (16)       //I2S_I2S_RX_FIFO_MOD_S
#define I2STXFMM (0x7)      //I2S_I2S_TX_FIFO_MOD
#define I2STXFM  (13)       //I2S_I2S_TX_FIFO_MOD_S
#define I2SDE    (1 << 12)  //I2S_I2S_DSCR_EN
#define I2STXDNM (0x3F)     //I2S_I2S_TX_DATA_NUM
```

```
#define I2STXDN  (6)        //I2S_I2S_TX_DATA_NUM_S
#define I2SRXDNM (0x3F)     //I2S_I2S_RX_DATA_NUM
#define I2SRXDN  (0)        //I2S_I2S_RX_DATA_NUM_S

//I2S CONF CHAN
#define I2SRXCMM (0x3)      //I2S_RX_CHAN_MOD
#define I2SRXCM  (3)        //I2S_RX_CHAN_MOD_S
#define I2STXCMM (0x7)      //I2S_TX_CHAN_MOD
#define I2STXCM  (0)        //I2S_TX_CHAN_MOD_S

/**
 Random Number Generator 32bit
 http://esp8266-re.foogod.com/wiki/Random_Number_Generator
**/
#define RANDOM_REG32  ESP8266_DREG(0x20E44)

#endif // ifndef CORE_MOCK

#endif
```