

# CamelStudioX Library Documentaion

## 1.3

Generated by Doxygen 1.8.14

Tue Jul 3 2018 00:26:47

## Contents

# 1 CamelStudioX Library Documentation

## Version 1.3

### Digital

- [General Input Output](#)
- [UART](#)
- [SPI](#)
- [Time Counter 0 \(TC0\)](#)
- [Time Counter 1 \(TC1\)](#)
- [Time Counter 2 \(TC2\)](#)
- [Time Counter 4 \(TC4\)](#)
- [\[LCD\]\(\)](#)
- [Real Time Counter Library \(RTC\)](#)

### Analog

- [Amplifier](#)
- [Sigma Delta](#)
- [Voltage to Pulse Width](#)
- [Internal Temperature Sensor](#)
- [Analog Output](#)

### System

- [Micro Control Unit \(MCU\)](#)
- [Interrupt](#)
- [Watch Dog](#)
- [Flash Library](#)

### Security

- [Data Encryption Standard \(DES\)](#)

## Other

- [Standard Input Output Library \(stdio\)](#)
- [Standard Library \(stdlib\)](#)
- [string Library](#)
- [Soft Float Library](#)

## File List

## 2 Analog Output Library for M2

To use this library, please include [analogIO.h](#) and [soft\\_fp.h](#).

### ATTENTION

- Because analog value ranges from 0 to the voltage of p\_vdd5, you must use soft float library!
- To get a smoother output, please connect a low-pass filter, a capacity, to the pin you set as analog output.

### Description

This library provides the function of DAC (Digital to Analog Converter) by using the PWM0 of Timer Counter 0, 1 and 2. When `RT_ADC_analogWrite` is used, the frequency of the corresponding timer is changed. If you want to custom the timer frequency, please change it later.

### Interface

```
void RT_DAC_analogWrite(channel, value, p_vdd5);
```

### Example

```
// set analog output channel 0 to output 1.5v.  
RTC_DAC_analogWrite(ANALOG_OUTPUT_0, 1.5, 5.0);
```

## 3 Internal Temperature Sensor

To use this library, please include [analogIO.h](#).

### Interface

```
void RT_ADC_TemperatureSensorOn();  
void RT_ADC_TemperatureSensorOff();
```

### Example

```
#include "analogIO.h"

/* If you want to get an absolute value of the temperature,
 * you should make a calibration at first.
 */
void Example_InternalTemperatureSensor() {
    RT_ADC_TemperatureSensorOn();
    uint32_t result = RT_ADC_SD_Read();
}
```

## 4 Amplifier (OPO)

To use this library, please include [analogIO.h](#).

### Interface

```
void RT_OPO_On();
void RT_OPO_Off();
void RT_OPO_SetChannel(ch0, ch1, ch2, ch3);
void RT_OPO_SetAmplification(Pin, Pgain, Nin, Ngain);
void RT_OPO_SetPsideFeedback(mode);
void RT_OPO_ExchangeChannelPin(switch);
void RT_OPO_SelectChannelPin(pin);
void RT_OPO_SetSingleSideMode(ch0, ch1, ch2, ch3, pin);
void RT_OPO_SetDifferentialMode(ch0, ch1, ch2, ch3, exchange);
void RT_OPO_SetShort(mode);
void RT_OPO_SetBypass(mode);
void RT_OPO_SetVPGND(op);
void RT_ADC_Clear();
```

### Example

```
#include "analogIO.h"

void Example_OPO_SingleSide()
{
    RT_OPO_SetSingleSideMode(ON, OFF, OFF, OFF,
        OPO_PSIDE);
    // 20-time amplification
    RT_OPO_SetAmplification(OPO_PIN_RESISTOR_1K,
        OPO_GAIN_20K, \
        OPO_PIN_RESISTOR_1K, OPO_GAIN_20K);
    RT_ADC_Clear();
    RT_OPO_On();
}

void Example_OPO_DifferentialMode()
{
    RT_OPO_SetDifferentialMode(ON, OFF, OFF,
        OFF, OPO_NOT_EXCHANGE_PIN);
    // 20-time amplification
    RT_OPO_SetAmplification(OPO_PIN_RESISTOR_1K,
        OPO_GAIN_20K, \
        OPO_PIN_RESISTOR_1K, OPO_GAIN_20K);
    RT_ADC_Clear();
    RT_OPO_On();
}
```

```

void Example_OPO_Calibration()
{
    RT_OPO_SetShort(ON);
}

void Example_OPO_Bypass()
{
    RT_OPO_SetBypass(ON);
}

```

## 5 Sigma Delta

To use this library, please include [analogIO.h](#).

### Interface

```

void RT_ADC_SD_On();
void RT_ADC_SD_Off();
void RT_ADC_SD_SetSampleRate(mode);
void RT_ADC_SD_SetAdWidth(mode);
void RT_ADC_SD_SetTrigger(source);
void RT_ADC_SD_Setup(sampleRate, adWidth, triggerSource);
void RT_ADC_SD_Start();
uint32_t RT_ADC_SD_DataReady();
uint32_t RT_ADC_SD_Read();
void RT_ADC_Clear();

```

### Example

```

#include "analogIO.h"

// This example setups SD simply.
void Example_SD_SimpleSst()
{
    RT_ADC_SD_Setup(SD_CLK_3M, SD_20BIT,
        SD_TRIG_BY_WT2READ);
    // Turning on SD, clearing SD and starting accumulation are all done in RT_ADC_SD_Read
    uint32_t result = RT_ADC_SD_Read();
    // ...
}

// This example reveals more details about SD setup.
void Example_SD_Basic()
{
    RT_ADC_SD_SetSampleRate(SD_CLK_3M);
    RT_ADC_SD_SetAdWidth(SD_20BIT);
    RT_ADC_SD_SetTrigger(SD_TRIG_BY_TCOPWM);
    RT_ADC_Clear();
    RT_ADC_SD_On();
    for (register uint32_t i=0; i<200;i++)
        __asm__("nop");
    RT_ADC_SD_Start();
    while(!RT_ADC_SD_DataReady());
    uint32_t result = MemoryRead32(AD_READ_REG) & 0xffff;
    // ...
}

```

## 6 Voltage to Pulse Width

To use this library, please include [analogIO.h](#).

## Interface

```

void RT_ADC_V2P_On();
void RT_ADC_V2P_Off();
void RT_ADC_V2P_SetResistor();
uint32_t RT_ADC_V2P_Read();
void RT_ADC_Clear();

```

## Example

```

#include "analogIO.h"

void Example_V2P() {
    RT_ADC_V2P_On();
    RT_ADC_V2P_SetResistor(V2P_220K); // V2P_185K is set in single side
    mode of the amplifier.
    uint32_t result = RT_ADC_V2P_Read();
}

```

## 7 Data Encryption Standard (DES) Library for M2

Since M2 is a 32-bit chip, some special measurements should be used in the implementation.

To use this library, please include [DES.h](#).

### Data Structure

#### 1. [DES\\_Key](#)

This is a data structure defined by union. [DES\\_Key.key](#) is a 64-bit value. [DES\\_Key.apart](#) is a 2\*32-bit array.

To prepare a key for DES algorithm, declare an [DES\\_Key](#) at first.

```
DES_Key originalKey;
```

Then store high 32 bits and low 32 bits respectively.

```

originalKey.apart[1] = <#high32OfKey#>;
originalKey.apart[0] = <#low32OfKey#>;

```

#### 2. [MessageData](#)

This is a data structure for storing data to encrypt or decrypt.

[MessageData.data](#) is a 64-bit value. [MessageData.apart](#) is a 2\*32-bit array. [MessageData.bytes](#) is an 8\*8-bit array.

To prepare a [MessageData](#) for DES algorithm, the operation is similar to [DES\\_Key](#)'s.

You can load data byte by byte or uint32\_t by uint32\_t.

## Interface

### 1. DES\_generateSubKeys

Generate 16 subKeys with a provided original key.

#### Definition

```
DES_Key* DES_generateSubKeys(const DES_Key originalKey);
```

### 2. DES\_process

Encrypt or decrypt data, require 16 subKeys provided by DES\_generateSubKeys. When mode = DES\_ENCRYPT\_MODE, encrypt data. When DES\_DECRYPT\_MODE, decrypt data.

#### Definition

```
MessageData DES_process(MessageData originalData,
    DES_Key* subKeys, uint8_t mode);
```

### 3. DES

Encrypt or decrypt data, require original key. When mode = DES\_ENCRYPT\_MODE, encrypt data. When DES\_DECRYPT\_MODE, decrypt data.

#### Definition

```
MessageData DES(MessageData originalData, DES_Key originalKey, uint8_t mode
    );
```

## Example

```
#include "mcu.h"
#include "DES.h"
#include "stdio.h"
// This is the interrupt function for user
void user_interrupt() {
}

// This is the main function
int main() {
    DES_Key originalKey;
    originalKey.key = 0x133457799BBCDFF1;

    // Generate 16 subKeys
    DES_Key* subKeys = DES_generateSubKeys(originalKey);
    for (uint8_t index = 0; index < 17; index++) {
        printf("K%d = 0x%x\n", index, subKeys[index].apart[1], subKeys[index].apart[0]);
    }

    MessageData originalData;
    originalData.data = 0x123456789abcdef;
    printf("originalData = 0x%x\n", originalData.apart[1], originalData.apart[0]);

    // encrypt data
    MessageData encryptedData = DES_process(originalData, subKeys,
        DES_ENCRYPT_MODE);
    printf("encryptedData = 0x%x\n", encryptedData.apart[1], encryptedData.apart[0]);

    // decrypt data
    MessageData decryptedData = DES_process(encryptedData, subKeys,
        DES_DECRYPT_MODE);
    printf("decryptedData = 0x%x\n", decryptedData.apart[1], decryptedData.apart[0]);
}
```

## 8 Flash Library for M2

To use this library, please include [Flash.h](#).

### Interface

```
void RT_Flash_Write(value, address);  
void RT_Flash_Erase1k(address);  
void RT_Flash_EraseFrom(address);  
void RT_Flash_SetMAC(id);
```

### Example

```
RT_Flash_Write(0x123, 0x10000000);    // write 0x123 to address 0x10000000  
RT_Flash_Erase1k(0x10000000);        // erase 1k-byte, starting from address 0x10000000
```

## 9 Interrupt Library for M2

To use this library, please include [Interrupt.h](#).

### Interface

```
int  RT_SYSINT_GetFlag(device);  
void RT_SYSINT_On();  
void RT_SYSINT_Off();  
void RT_EXTINT_Setup(port, trigger);  
void RT_EXTINT_Off(port);  
void RT_EXTINT_Clear(port);  
void RT_EXTINT_ClearAll();  
int  RT_EXTINT_GetAllFlag();  
int  RT_EXTINT_GetFlag(port);
```

### Example

```
RT_SYSINT_On();           // turn on system interrupt  
RT_EXTINT_Clear(1);       // clear External Interrupt port 1
```

## 10 Input Output Library for M2

To use this library, please include [IO.h](#).



## Interface

```
void RT_IO_SetOutput(io);
void RT_IO_SetInput(io);
void RT_IO_SetInputOutput(io, mode);
void RT_IO_SetInputOutput16(mode);
void RT_IO_SetHigh(io);
void RT_IO_SetLow(io);
void RT_IO_SetLevel(io, level);
void RT_IO_SetLevel16(level);
uint8_t RT_IO_Read(io);
uint16_t RT_IO_Read16();
```

## Example

```
// set IO as Input
RT_IO_SetOutput(5);           // set io5 OUTPUT
RT_IO_SetInputOutput(5, OUTPUT); // also set io5 OUTPUT.
RT_IO_SetInputOutput16(OUTPUT<<5); // set io5 OUTPUT, but other io input.

// set IO output HIGH
RT_IO_SetHigh(5);           // set io5 output HIGH.
RT_IO_SetLevel(5, HIGH);    // also set io5 output HIGH.
RT_IO_SetLevel16(HIGH<<5); // set io5 output HIGH, but other io output LOW.

// read io
uint8_t levelOfOneIO = RT_IO_Read(5); // read io5
uint16_t levelOf16IO = RT_IO_Read16(); // read all io.
```

# 11 Micro Control Unit (MCU) Library for M2

To use this library, please include `mcu.h`.

## Interface

```
long MemoryRead32(addr)
void MemoryWrite32(addr, val)
void MemoryOr32(addr, val)
void MemoryAnd32(addr, val)
void MemoryBitAt(addr, val)
void MemoryBitOn(addr, val)
void MemoryBitOff(addr, val)
void MemoryBitSwitch(addr, val)
void JumpTo(address)
void RT_MCU_SetSystemClock(mode);
void RT_Clr_Sram();
```

## Example

```
long a = MemoryRead32(0x1000000); // read the value @0x1000000
JumpTo(0x1000000);               // jump the program to the address 0x1000000
RT_Clr_Sram();                   // clear the 2kx32 sram space
```

## 12 Soft Float Library for M2

To use this library, please include `soft_fp.h`.

### Interface

```
float fp_float32_neg(float a_fp);
float fp_float32_abs(float a_fp);
float fp_float32_add(float a_fp, float b_fp);
float fp_float32_sub(float a_fp, float b_fp);
float fp_float32_mult(float a_fp, float b_fp);
float fp_float32_div(float a_fp, float b_fp);
long fp_float32_to_int32(float a_fp);
float fp_int32_to_float32(long af);
float fp_uint32_to_float32(unsigned long af);
double fp_float32_to_float64(float a_fp);
float fp_float64_to_float32(double a_dfp);
int fp_float32_cmp(float a_fp, float b_fp);
float fp_float32_sqrt(float a);
float fp_float32_cos(float rad);
float fp_float32_sin(float rad);
float fp_float32_atan(float x);
float fp_float32_atan2(float y, float x);
float fp_float32_exp(float x);
float fp_float32_log(float x);
float fp_float32_pow(float x, float y);
```

### Example

```
float a = fp_int32_to_float32(-123) // convert int -123 -> float type
float b = fp_uint32_to_float32(2345); // convert unsigned int 2345 -> float type
float c = fp_float32_add(a, b);      // floating point add
float d = fp_float32_sqrt(b);        // floating point sqrt
```

## 13 SPI Library For M2

To use this library, please include `SPI.h`.

## Interface

```

void RT_SPI_On();
void RT_SPI_Off();
void RT_SPI_Modeset(MorS);
void RT_SPI_ChipSelect();
void RT_SPI_ChipRelease();
void RT_SPI_ClearState();
void RT_SPI_Busy();
void RT_SPI_Write_(val);
void RT_SPI_Write(val);
uint32_t RT_SPI_DataReady();
uint8_t RT_SPI_Read();
unsigned char RT_SPI_Read();
unsigned char RT_SPI_MasterTransfer(unsigned char c);
unsigned char RT_SPI_SlaveTransfer(unsigned char c);
void RT_SPI_Delay();

```

## Example

```

#include "SPI.h"
#include "stdio.h"

// This example shows how to set M2 SPI as master, send and receive data.
void master(){
    char c;
    RT_SPI_Modeset(MASTER);           // set M2 SPI as Master
    RT_SPI_ChipSelect();
    RT_SPI_ClearState();
    while(1){
        c=RT_SPI_MasterTransfer('A');
        putchar(c);
    }
}

// This example shows how to set M2 SPI as slave, send and receive data.
void slave(){
    char c;
    RT_SPI_Modeset(SLAVE);             // set M2 SPI as Slave
    RT_SPI_ClearState();
    RT_SPI_Write_('B');
    while(1){
        c=RT_SPI_SlaveTransfer('B');
        putchar(c);
    }
}

```

## 14 Standard Input Output Library for M2

To use this library, please include `stdlib.h`. If you want to use a stdio library supporting soft float, please include `stdio_fp.h` instead of `stdio.h`.

### Interface

```
void putchar(char c);  
void puts(const char* string);  
char getchar();  
long getnum();  
unsigned long getHexNum();  
char* itoa(int value, unsigned int base);  
void printf(const char *format, ...);  
void sprintf(char* buf, const char* format, ...);
```

### Example

```
char a = 'H'; putchar(a);           // print 'H' to Uart0  
char b = getchar();                 // get a char from Uart, then assign to b
```

## 15 Standard Library for M2

To use this library, please include `stdlib.h`. If you want to use a stdlib library supporting soft float, please include `stdlib_fp.h` instead of `stdlib.h`.

### Interface

```
long atoi(const char* str);  
unsigned long xtoi(const char* str);
```

### Example

```
long i = atoi("1234");              // convert string decimal "1234" to number  
long j = xtoi("1f80");              // convert string hex "1f80" to number
```

## 16 String Library for M2

To use this library, please include `string.h`.

## Interface

```
void *memchr(const void *str, int c, size_t n);

void memcmp(const void *str, int c, size_t n);

void * memcpy(void * dest, const void * src, size_t n);

void * memmove(void * dest, const void * src, size_t n);

void * memset(void *str, int c, size_t n);

char * strcat(char *dest, const char * src);

char * strncat(char *dest, const char * src, size_t n);

char * strchr(const char *str, int c);

int strcmp(const char * s1, const char * s2);

int strncmp(const char * s1, const char * s2, size_t n);

char * strcpy(char * dest, const char * src);

char * strncpy(char * dest, const char * src, size_t n);

size_t strlen(const char * str);

char * strstr(const char *s1, const char * s2);
```

## Example

```
#include "stdio.h"

void Example_strlen() {
    printf("%s has %d chars", s, strlen(s)); // strlen
}

void Example_strncmp() {
    int ptr;
    ptr=strncmp(buf2, buf1, 3); // string comparison
}
```

## 17 TC0 Library for M2

To use this library, please include [TC0.h](#).

## Interface

```
void RT_TC0_Stop();

void RT_TC0_ClearIrq();

void RT_TC0_ClearCnt();

void RT_TC0_ClearAll();

void RT_TC0_TcIrqOn();

void RT_TC0_TcIrqOff();

void RT_TC0_PWMIrqOn();

void RT_TC0_PWMIrqOff();

int RT_TC0_CheckTcFlag();

int RT_TC0_CheckPWMFlag();

void RT_TC0_TimerOn();

void RT_TC0_TimerOff();
```

```

void RT_TC0_TimerSetlus(T, irq);
void RT_TC0_SetCounter(n);
void RT_TC0_EcntOn();
void RT_TC0_EcntOff();
void RT_TC0_SetEcnt(n, trigger, irq);
void RT_TC0_PWMOn();
void RT_TC0_PWMOff();
void RT_TC0_SetPWM(div, ref, irq);
void RT_TC0_PWMMOn();
void RT_TC0_PWMMOff();
void RT_TC0_PWMMTriggerMode(mode);
void RT_TC0_SetPWMM(trigger, irq);
int RT_TC0_ReadCnt();

```

### Example

```

RT_TC0_ClearCnt();           // clear TC0 Counter
RT_TC0_TcIrqOn();           // set TC0 cnt-IRQ enable

```

## 18 TC1 Library for M2

To use this library, please include [TC1.h](#).

### Interface

```

void RT_TC1_Stop();
void RT_TC1_ClearIrq();
void RT_TC1_ClearCnt();
void RT_TC1_ClearAll();
void RT_TC1_TcIrqOn();
void RT_TC1_TcIrqOff();
void RT_TC1_PWMIrqOn();
void RT_TC1_PWMIrqOff();
int RT_TC1_CheckTcFlag();
int RT_TC1_CheckPWMFlag();
void RT_TC1_TimerOn();
void RT_TC1_TimerOff();
void RT_TC1_TimerSetlus(T, irq);
void RT_TC1_SetCounter(n);
void RT_TC1_EcntOn();
void RT_TC1_EcntOff();
void RT_TC1_SetEcnt(n, trigger, irq);
void RT_TC1_PWMOn();

```

```

void RT_TC1_PWMOFF();
void RT_TC1_SetPWM(div, ref, irq);
void RT_TC1_PWMMON();
void RT_TC1_PWMMOFF();
void RT_TC1_PWMMTriggerMode(mode);
void RT_TC1_SetPWMM(trigger, irq);
int RT_TC1_ReadCnt();

```

### Example

```

RT_TC1_ClearCnt();           // clear TC1 Counter
RT_TC1_TcIrqOn();           // set TC1 cnt-IRQ enable

```

## 19 TC2 Library for M2

To use this library, please include `TC2.h`.

### Interface

```

void RT_TC2_Stop();
void RT_TC2_ClearIrq();
void RT_TC2_ClearCnt();
void RT_TC2_ClearAll();
void RT_TC2_TcIrqOn();
void RT_TC2_TcIrqOff();
void RT_TC2_PWMIrqOn();
void RT_TC2_PWMIrqOff();
int RT_TC2_CheckTcFlag();
int RT_TC2_CheckPWMFlag();
void RT_TC2_TimerOn();
void RT_TC2_TimerOff();
void RT_TC2_TimerSetlus(T, irq);
void RT_TC2_SetCounter(n);
void RT_TC2_EcntOn();
void RT_TC2_EcntOff();
void RT_TC2_SetEcnt(n, trigger, irq);
void RT_TC2_PWM0On();
void RT_TC2_PWM0Off();
void RT_TC2_PWM1to3On();
void RT_TC2_PWM1to3Off();
void RT_TC2_SetAllPWM(div, duty0, duty1, duty2, duty3, phase0, phase1, phase2, phase3, pwm0,
    , pwm13);
void RT_TC2_PWMMON();

```

```

void RT_TC2_PWMMOff();
void RT_TC2_PWMTriggerMode(mode);
void RT_TC2_SetPWM(trigger, irq);
int RT_TC2_ReadCnt();

```

### Example

```

RT_TC2_ClearCnt();           // clear TC2 Counter
RT_TC2_TcIrqOn();           // set TC2 cnt-IRQ enable

```

## 20 TC4 Library for M2

To use this library, please include [TC4.h](#).

### Interface

```

void RT_TC4_AllPWM_On();
void RT_TC4_AllPWM_Off();
void RT_TC4_PWM_On(n);
void RT_TC4_PWM_Off(n);
void RT_TC4_SetAllPWM(pw0_en, div0, ref0, pwm1_en, div1, ref1);
void RT_TC4_SetPWM(n, pwm_en, div, ref);

```

### Example

```

RT_TC4_AllPWM_On();          // turn on TC4 pwm0 and pwm1
RT_TC4_SetPWM(0, ON, 2, 4);  // set pwm0, div=2, ref=4

```

## 21 Real Time Counter Library for M2

To use this library, please include [time.h](#).

### Interface

```

void RT_RTC_On()
void RT_RTC_Off()
void RT_RTC_SetTimeFormat(format)
void RT_RTC_SetTime(year, month, day, hour, min, sec)
long RT_RTC_Read32()
void RT_RTC_GetTime(unsigned char *d_year, unsigned char *d_mon, unsigned char *d_day,
                    unsigned char *d_hour, unsigned char *d_min, unsigned char *d_sec);
void RT_DelayMilliseconds(unsigned long ms);

```



### Example

```
RT_RTC_On();      // turn on RTC
RT_RTC_Off();     // turn off RTC
```

## 22 UART Library for M2

UART library provides interface to operate UART0 and UART1 of M2. To use this library, please include `UART.h`.

### Interface

```
void RT_UART_On(port);
void RT_UART_Off(port);
void RT_UART_Busy(port);
void RT_UART_Write(port);
uint32_t RT_UART_DataReady(port);
uint8_t RT_UART_Read(port);
void RT_UART_IrqOn(port);
void RT_UART_CompareOn(port);
void RT_UART_CompareOff(port);
void RT_UART_SetCompare(port, val);
void RT_UART_ClearIrq(port);
void RT_UART_RaiseIrq(port);
void RT_UART_CheckIrq(port);
void RT_UART_LinSyncOn(port);
void RT_UART_LinSyncOff(port);
void RT_UART_LinBreakNormal(port);
void RT_UART_LinBreakExtreme(port);
void RT_UART_LinSendBreak(port);
void RT_UART_LinCheckIrq(port);
uint32_t RT_UART_GetBrp(port);
void RT_UART_LinSlave(port);
void RT_UART_putchar(uint32_t port, unsigned char c);
void RT_UART_puts(uint32_t port, unsigned char *string);
unsigned char RT_UART_getchar(uint32_t port);
void RT_UART_LinMaster(uint32_t port, char pattern);
```

### Example

#### Send and receive data via UART0 and UART1

```
/*
 * Before running this example, please connect pins:
 * P_TX <-----> SPI_SI/RX1
 * P_RX <-----> SPI_SO/TX1
 */
```

```
#include "UART.h"

void user_interrupt() {}

int main() {
    char c;

    // Turn on UART0 and UART1
    RT_UART_On(UART0);
    RT_UART_On(UART1);

    while(1) {
        c = RT_UART_getchar(UART1);
        RT_UART_puts(UART0, "Get a character from UART1: ");
        RT_UART_putchar(UART0, c);
        RT_UART_putchar(UART0, '\n');
    }
}
```

### Send Lin break via UART1

```
#include "UART.h"

void user_interrupt()

int main() {
    RT_UART_On(UART1); // Turn on UART1
    // RT_UART_LinSlave(UART1); // set M2 as Lin slave
    // RT_UART_LinMaster(UART1, NORMAL_BREAK); // 13-bit break
    RT_UART_LinMaster(UART1, EXTREME_BREAK); // 26-bit break
    // ATTENTION: When RT_UART_LinMaster is used, a break will be sent automatically.
    RT_UART_LinSendBreak(UART1); // Send break again manually.

    // code after Lin sync
    // ...
}
```

## 23 Watch Dog Library for M2

To use this library, please include [WDT.h](#).

### Interface

```
void RT_WatchDog_Setup(n, irq, rst);

void RT_WatchDog_Clear();

uint4_t RT_WatchDog_ReadValue();
```

### Example

```
RT_WDT_Clr(); // clear watch dog
RT_WDT_Set(8,0,1); // set as 1s (8 * 1/8s), no interrupt, auto-reset
```

## 24 Class Index

### 24.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">DES_Key</a>	Type for storing key to be used by DES Algorithm	??
<a href="#">DES_Key_pair</a>	Store the C key and the D key of a DES key	??
<a href="#">MessageData</a>	Type for storing data to be processed by DES Algorithm	??

## 25 File Index

### 25.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/Flash.h</a>	Flash Operation Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/Interrupt.h</a>	Interrupt Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/IO.h</a>	General Input Output Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/TC0.h</a>	Timer0 Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/TC1.h</a>	Timer1 Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/TC2.h</a>	Timer2 Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/TC4.h</a>	Timer4 Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/WDT.h</a>	Watchdog Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/analogIO/analogIO.c</a>	Analog Input Output Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/analogIO/analogIO.h</a>	Analog Input Output Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/DES/DES.c</a>	Standard Encryption Standard Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/DES/DES.h</a>	Data Encryption Standard Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/LCD/LCD.c</a>	LCD Library for M2	??
<a href="#">/Users/daizhirui/Development/CamelStudio_Library/Official/src/LCD/LCD.h</a>	LCD Library for M2	??

/Users/daizhirui/Development/CamelStudio_Library/Official/src/mcu/ <a href="#">mcu.c</a> M2 micro core unit	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/mcu/ <a href="#">mcu.h</a> M2 micro core unit	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/soft_fp/ <a href="#">soft_fp.c</a> Soft float library	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/soft_fp/ <a href="#">soft_fp.h</a> Soft float library	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/SPI/ <a href="#">SPI.c</a> SPI Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/SPI/ <a href="#">SPI.h</a> SPI Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdio.c</a> Standard Input Output Library For M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdio.h</a> Standard Input Outpt Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdio_fp.c</a> Standard Input Output Library For M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdio_fp.h</a> Standard Input Outpt Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdio_shared.h</a>	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdlib.c</a> Standard Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdlib.h</a> Standard Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdlib_fp.c</a> Standard Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/ <a href="#">stdlib_fp.h</a> Standard Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/string/ <a href="#">string.c</a> String Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/string/ <a href="#">string.h</a> String Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/time/ <a href="#">time.c</a> Real Time Module Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/time/ <a href="#">time.h</a> Real Time Module Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/UART/ <a href="#">UART.c</a> UART Library for M2	??
/Users/daizhirui/Development/CamelStudio_Library/Official/src/UART/ <a href="#">UART.h</a> UART Library for M2	??

## 26 Class Documentation

### 26.1 DES\_Key Union Reference

Type for storing key to be used by DES Algorithm.

```
#include <DES.h>
```

#### Public Attributes

- [uint64\\_t key](#)  
*64-bit key.*
- [uint32\\_t apart](#) [2]  
*two 32-bit key. [apart\[1\]](#) is high 32 bits, [apart\[0\]](#) is low 32 bits.*

#### 26.1.1 Detailed Description

Type for storing key to be used by DES Algorithm.

The documentation for this union was generated from the following file:

- /Users/daizhirui/Development/CamelStudio\_Library/Official/src/DES/[DES.h](#)

### 26.2 DES\_Key\_pair Struct Reference

Store the C key and the D key of a DES key.

#### Public Attributes

- [half\\_DES\\_Key cKey](#)  
*C key.*
- [half\\_DES\\_Key dKey](#)  
*D key.*

#### 26.2.1 Detailed Description

Store the C key and the D key of a DES key.

The documentation for this struct was generated from the following file:

- /Users/daizhirui/Development/CamelStudio\_Library/Official/src/DES/[DES.c](#)

### 26.3 MessageData Union Reference

Type for storing data to be processed by DES Algorithm.

```
#include <DES.h>
```

### Public Attributes

- uint64\_t [data](#)  
*64-bit key.*
- uint32\_t [apart](#) [2]  
*two 32-bit keys. [apart\[1\]](#) is high 32 bits, [apart\[0\]](#) is low 32 bits.*
- uint8\_t [bytes](#) [8]  
*eight 8-bit keys.*

#### 26.3.1 Detailed Description

Type for storing data to be processed by DES Algorithm.

The documentation for this union was generated from the following file:

- /Users/daizhirui/Development/CamelStudio\_Library/Official/src/DES/[DES.h](#)

## 27 File Documentation

### 27.1 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/analogIO/analogIO.c File Reference

Analog Input Output Library for M2.

```
#include "analogIO.h"
#include "TC0.h"
```

### Functions

- uint32\_t [RT\\_ADC\\_SD\\_Read](#) ()  
*Get the result of SD.*
- uint32\_t [RT\\_ADC\\_V2P\\_Read](#) ()  
*Get the result of V2P.*

#### 27.1.1 Detailed Description

Analog Input Output Library for M2.

### Author

Zhirui Dai

### Date

15 Jun 2018

### Copyright

2018 Zhirui

## 27.1.2 Function Documentation

### 27.1.2.1 RT\_ADC\_SD\_Read()

```
uint32_t RT_ADC_SD_Read ( )
```

Get the result of SD.

#### Returns

uint32\_t SD result.

### 27.1.2.2 RT\_ADC\_V2P\_Read()

```
uint32_t RT_ADC_V2P_Read ( )
```

Get the result of V2P.

#### Returns

int V2P result.

## 27.2 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/analogIO/analogIO.h File Reference

Analog Input Output Library for M2.

```
#include "mcu.h"
```

#### Macros

- #define **OPO\_PSIDE** 0x4  
*P-Side Keyword for OPO Module.*
- #define **OPO\_NSIDE** 0x0  
*N-Side Keyword for OPO Module.*
- #define **OPO\_EXCHANGE\_PIN** 0x4  
*Keyword for exchanging channel pins.*
- #define **OPO\_NOT\_EXCHANGE\_PIN** 0x0  
*Keyword for not exchanging channel pins.*
- #define **OPO\_PIN\_RESISTOR\_1K** 0x1  
*Pin resistor 1K of OPO Channel.*
- #define **OPO\_PIN\_RESISTOR\_10K** 0x0  
*Pin resistor 10K of OPO Channel.*
- #define **VCOM** 0x1  
*VCOM.*
- #define **GND** 0x0  
*Ground Pin Keyword.*

- #define [OPO\\_GAIN\\_480K](#) 0x1  
*Pin gain level 480K for OPO Module.*
- #define [OPO\\_GAIN\\_400K](#) 0x0  
*Pin gain level 400K for OPO Module.*
- #define [OPO\\_GAIN\\_320K](#) 0x2  
*Pin gain level 320K for OPO Module.*
- #define [OPO\\_GAIN\\_240K](#) 0x4  
*Pin gain level 240K for OPO Module.*
- #define [OPO\\_GAIN\\_100K](#) 0x6  
*Pin gain level 100K for OPO Module.*
- #define [OPO\\_GAIN\\_80K](#) 0x3  
*Pin gain level 80K for OPO Module.*
- #define [OPO\\_GAIN\\_40K](#) 0x5  
*Pin gain level 40K for OPO Module.*
- #define [OPO\\_GAIN\\_20K](#) 0x7  
*Pin gain level 20K for OPO Module.*
- #define [SD\\_CLK\\_3M](#) 0x0  
*Clock Divider Frequency 3M Hz of Sigma Delta Module.*
- #define [SD\\_CLK\\_1\\_5M](#) 0x1  
*Clock Divider Frequency 1.5M Hz of Sigma Delta Module.*
- #define [SD\\_CLK\\_781K](#) 0x2  
*Clock Divider Frequency 781K Hz of Sigma Delta Module.*
- #define [SD\\_CLK\\_390K](#) 0x3  
*Clock Divider Frequency 390K Hz of Sigma Delta Module.*
- #define [SD\\_14BIT](#) 0x0  
*Result Length 14-bit of Sigma Delta Module.*
- #define [SD\\_16BIT](#) 0x1  
*Result Length 16-bit of Sigma Delta Module.*
- #define [SD\\_18BIT](#) 0x2  
*Result Length 18-bit of Sigma Delta Module.*
- #define [SD\\_20BIT](#) 0x3  
*Result Length 20-bit of Sigma Delta Module.*
- #define [SD\\_TRIG\\_BY\\_TC0PWM](#) 0x1  
*Sampling Trigger By TC0 PWM Keyword of Sigma Delta Module.*
- #define [SD\\_TRIG\\_BY\\_WT2READ](#) 0x0  
*Sampling Trigger By wt<2> of SD\_READ\_REG Keyword of Sigma Delta Module.*
- #define [V2P\\_220K](#) 0x0  
*Resistor 220K Keyword of Voltage-To-PulseWidth Module.*
- #define [V2P\\_256K](#) 0x1  
*Resistor 256K Keyword of Voltage-To-PulseWidth Module.*
- #define [V2P\\_291K](#) 0x2  
*Resistor 291K Keyword of Voltage-To-PulseWidth Module.*
- #define [V2P\\_185K](#) 0x3  
*Resistor 185K Keyword of Voltage-To-PulseWidth Module.*
- #define [RT\\_ADC\\_Clear\(\)](#) [MemoryWrite32\(AD\\_CLR\\_REG, 1\)](#)  
*Clear the state of Analog Digit Converter.*
- #define [RT\\_ADC\\_V2P\\_On\(\)](#) [MemoryOr32\(AD\\_CTL0\\_REG, \(1 << 8\)\)](#)  
*Set ADC\_V2P on.*
- #define [RT\\_ADC\\_V2P\\_Off\(\)](#) [MemoryAnd32\(AD\\_CTL0\\_REG, ~\(1 << 8\)\)](#)  
*Set ADC\_V2P off.*
- #define [RT\\_ADC\\_V2P\\_SetResistor](#)(resistor)



- Set the resistor of V2P. When res = 3, single input mode is enabled, vcom will be fed into the P-side of OPO!*
- #define `RT_ADC_TemperatureSensorOn()` `MemoryOr32(AD_CTL0_REG, 1 << 5)`  
*Set ADC temperature sensor on.*
  - #define `RT_ADC_TemperatureSensorOff()` `MemoryAnd32(AD_CTL0_REG, ~(1 << 5))`  
*Set ADC temperature sensor off.*
  - #define `RT_OPO_On()` `MemoryOr32(AD_OPO_REG, 0x1)`  
*Turn the Amplifier on.*
  - #define `RT_OPO_Off()` `MemoryAnd32(AD_OPO_REG, ~0x1)`  
*Turn the Amplifier off.*
  - #define `RT_OPO_SetChannel(ch0, ch1, ch2, ch3)`  
*Set the state of every amplifier channel.*
  - #define `RT_OPO_SetAmplification(Pin, Pgain, Nin, Ngain)`  
*Set the amplification of the amplifier.*
  - #define `RT_OPO_SetPsideFeedback(mode)`  
*Set the feedback mode of P-Side of the amplifier. When mode = 1, feedback is on, vcom(1.5V) is connected to the resistor block at inp. When mode = 0, vcom is disconnected to the resistor block at inp.*
  - #define `RT_OPO_ExchangeChannelPin(switch)`  
*Set double side mode and exchange P side and N side in double side mode.*
  - #define `RT_OPO_SelectChannelPin(pin)`  
*Set single side mode and select P side or N side in single side mode.*
  - #define `RT_OPO_SetSingleSideMode(ch0, ch1, ch2, ch3, pin)`  
*Set OPO at SingleSide Mode, open specific channel and select specific pin.*
  - #define `RT_OPO_SetDifferentialMode(ch0, ch1, ch2, ch3, exchange)`  
*Set OPO at Differential Mode.*
  - #define `RT_OPO_SetShort(mode)`  
*Set whether to connect pin and nin to vcom. Channel 0-3 should be closed!*
  - #define `RT_OPO_SetBypass(mode)`  
*Set the bypass of opo. If opo is bypassed, input will be fed into SD directly. (select the filter resistor, 1 for 50k and 0 for 100k)*
  - #define `RT_OPO_SetVPGND(op)`  
*Set VPGND op=1, PsideGND = vcom; op=0, PsideGND = 0.*
  - #define `RT_ADC_SD_On()` `MemoryOr32(AD_CTL0_REG, 1)`  
*Turn on SD.*
  - #define `RT_ADC_SD_Off()` `MemoryAnd32(AD_CTL0_REG, ~1)`  
*Turn off SD.*
  - #define `RT_ADC_SD_SetSampleRate(mode)`  
*Set the sample rate of SD.*
  - #define `RT_ADC_SD_SetAdWidth(mode)`  
*Set the length of the result.*
  - #define `RT_ADC_SD_SetTrigger(source)`  
*Set the trigger source of SD.*
  - #define `RT_ADC_SD_Setup(sampleRate, adWidth, triggerSource)`  
*Set SD simply.*
  - #define `RT_ADC_SD_Start()` `MemoryWrite32(AD_READ_REG, 1)`  
*Start SD accumulate.*
  - #define `RT_ADC_SD_DataReady()` `((MemoryRead32(AD_CTL0_REG) & 0x80000000) >> 31)`  
*Check is the accumulation of SD is completed.*
  - #define `ANALOG_OUTPUT_0` `0x0`  
*Keyword ANALOG\_OUTPUT\_0.*
  - #define `ANALOG_OUTPUT_1` `0x100`  
*Keyword ANALOG\_OUTPUT\_1.*

- #define `ANALOG_OUTPUT_2` 0x300  
*Keyword `ANALOG_OUTPUT_2`.*
- #define `RT_DAC_analogWrite`(channel, value, p\_vdd5)  
*Output an analog value by a selected pwm.*

## Functions

- uint32\_t `RT_ADC_SD_Read` ()  
*Get the result of SD.*
- uint32\_t `RT_ADC_V2P_Read` ()  
*Get the result of V2P.*

### 27.2.1 Detailed Description

Analog Input Output Library for M2.

#### Author

Zhirui Dai

#### Date

15 Jun 2018

#### Copyright

2018 Zhirui

### 27.2.2 Macro Definition Documentation

#### 27.2.2.1 ANALOG\_OUTPUT\_0

```
#define ANALOG_OUTPUT_0 0x0
```

Keyword `ANALOG_OUTPUT_0`.

#### Note

Relative value of `T0_CTL0_REG` to `T0_CTL0_REG`.

### 27.2.2.2 ANALOG\_OUTPUT\_1

```
#define ANALOG_OUTPUT_1 0x100
```

Keyword ANALOG\_OUTPUT\_1.

#### Note

Relative value of T1\_CTL0\_REG to T0\_CTL0\_REG.

### 27.2.2.3 ANALOG\_OUTPUT\_2

```
#define ANALOG_OUTPUT_2 0x300
```

Keyword ANALOG\_OUTPUT\_2.

#### Note

Relative value of T2\_CTL0\_REG to T0\_CTL0\_REG.

### 27.2.2.4 RT\_ADC\_Clear

```
#define RT_ADC_Clear( ) MemoryWrite32(AD_CLR_REG, 1)
```

Clear the state of Analog Digit Converter.

#### Returns

void

### 27.2.2.5 RT\_ADC\_SD\_DataReady

```
#define RT_ADC_SD_DataReady( ) ((MemoryRead32(AD_CTL0_REG) & 0x80000000) >> 31)
```

Check is the accumulation of SD is completed.

#### Returns

The result if the accumulation is completed, 1=completed, 0=not completed

## 27.2.2.6 RT\_ADC\_SD\_Off

```
#define RT_ADC_SD_Off( ) MemoryAnd32(AD_CTL0_REG, ~1)
```

Turn off SD.

## Note

AD\_CTL0\_REG[0]: 1=on, 0=off

## Returns

void

## 27.2.2.7 RT\_ADC\_SD\_On

```
#define RT_ADC_SD_On( ) MemoryOr32(AD_CTL0_REG, 1)
```

Turn on SD.

## Note

AD\_CTL0\_REG[0]: 1=on, 0=off

## Returns

void

## 27.2.2.8 RT\_ADC\_SD\_SetAdWidth

```
#define RT_ADC_SD_SetAdWidth(  
    mode )
```

## Value:

```
{  
    MemoryAnd32(AD_CTL0_REG, ~(0x3 << 1)); \  
    MemoryOr32(AD_CTL0_REG, (mode << 1)); \  
}
```

Set the length of the result.

## Note

Set the filter frequency to adjust the ad width. 1K Hz is for 14bit, 512Hz for 16bit, 256Hz for 18bit, 128Hz for 20bit. The frequency mentioned is relative to the sd clock frequency.

**Parameters**

<i>mode</i>	optional value:  <code>SD_14BIT, SD_16BIT, SD_18BIT, SD_20BIT</code>
-------------	--

**Returns**

void

**27.2.2.9 RT\_ADC\_SD\_SetSampleRate**

```
#define RT_ADC_SD_SetSampleRate(  
    mode )
```

**Value:**

```
{  
    MemoryAnd32(AD_CTL0_REG, ~(0x3 << 3)); \  
    MemoryOr32(AD_CTL0_REG, (mode << 3)); \  
}
```

Set the sample rate of SD.

**Parameters**

<i>mode</i>	optional value:  <code>SD_CLK_3M, SD_CLK_1_5M, SD_CLK_781K, SD_CLK_390K</code>
-------------	--

**Returns**

void

**27.2.2.10 RT\_ADC\_SD\_SetTrigger**

```
#define RT_ADC_SD_SetTrigger(  
    source )
```

**Value:**

```
{  
    MemoryAnd32(AD_CTL0_REG, ~(1 << 6)); \  
    MemoryOr32(AD_CTL0_REG, source << 6); \  
}
```

Set the trigger source of SD.

Parameters

<i>source</i>	optional value:  SD_TRIG_BY_WT2READ, SD_TRIG_BY_TCOPWM
---------------	--

Returns

void

27.2.2.11 RT\_ADC\_SD\_Setup

```
#define RT_ADC_SD_Setup(  
    sampleRate,  
    adWidth,  
    triggerSource )
```

Value:

```
{  
    RT_ADC_SD_SetSampleRate(sampleRate); \  
    RT_ADC_SD_SetAdWidth(adWidth); \  
    RT_ADC_SD_SetTrigger(triggerSource); \  
}
```

Set SD simply.

Parameters

<i>sampleRate</i>	the sample rate of SD, optional value:  SD_CLK_3M, SD_CLK_1_5M, SD_CLK_781K, SD_CLK_390K  .
<i>adWidth</i>	the precision of the result optional value:  SD_14BIT, SD_16BIT, SD_18BIT, SD_20BIT  .
<i>triggerSource</i>	the source to trig sampling. optional value:  SD_TRIG_BY_TCOPWM, SD_TRIG_BY_WT2READ  .

Returns

void

#### 27.2.2.12 RT\_ADC\_SD\_Start

```
#define RT_ADC_SD_Start( ) MemoryWrite32(AD_READ_REG, 1)
```

Start SD accumulate.

##### Note

When writing to AD\_READ\_REG, it will start df conversion, if SD is enabled and choose trig pwm. SD flag or irq will be asserted upon conversion completed.

##### Returns

void

#### 27.2.2.13 RT\_ADC\_TemperatureSensorOff

```
#define RT_ADC_TemperatureSensorOff( ) MemoryAnd32(AD_CTL0_REG, ~(1 << 5))
```

Set ADC temperature sensor off.

##### Note

AD\_CTL0\_REG[5]: 1=on, 0=off

##### Returns

void

#### 27.2.2.14 RT\_ADC\_TemperatureSensorOn

```
#define RT_ADC_TemperatureSensorOn( ) MemoryOr32(AD_CTL0_REG, 1 << 5)
```

Set ADC temperature sensor on.

##### Note

AD\_CTL0\_REG[5]: 1=on, 0=off

##### Returns

void

### 27.2.2.15 RT\_ADC\_V2P\_Off

```
#define RT_ADC_V2P_Off( ) MemoryAnd32(AD_CTL0_REG, ~(1 << 8))
```

Set ADC\_V2P off.

#### Note

AD\_CLR\_REG[8]: 1=V2P on, 0=V2P off.

#### Returns

void

### 27.2.2.16 RT\_ADC\_V2P\_On

```
#define RT_ADC_V2P_On( ) MemoryOr32(AD_CTL0_REG, (1 << 8))
```

Set ADC\_V2P on.

#### Note

AD\_CLR\_REG[8]: 1=V2P on, 0=V2P off.

#### Returns

void

### 27.2.2.17 RT\_ADC\_V2P\_SetResistor

```
#define RT_ADC_V2P_SetResistor(  
    resistor )
```

#### Value:

```
{  
    MemoryAnd32(AD_CTL0_REG, ~(0x3 << 9)); \  
    MemoryOr32(AD_CTL0_REG, (resistor << 9)); \  
}
```

Set the resistor of V2P. When res = 3, single input mode is enabled, vcom will be fed into the P-side of OPO!

#### Parameters

<i>resistor</i>	optional value:  V2P_185K, V2P_220K, V2P_256K, V2P_291K
-----------------	---



**Returns**

void

**27.2.2.18 RT\_DAC\_analogWrite**

```
#define RT_DAC_analogWrite(
    channel,
    value,
    p_vdd5 )
```

**Value:**

```
{
    MemoryOr32((T0_CTL0_REG|channel), 1<<4);          \
    MemoryAnd32((T0_CTL0_REG|channel), ~(0x3 << 6)); \
    MemoryWrite32((T0_CLK_REG|channel), 1);          \
    MemoryWrite32((T0_REF_REG|channel), value * 255/p_vdd5); \
}
```

Output an analog value by a selected pwm.

**Note**

1. Turn the pwm for selected channel on.
2. Turn on the interrupt of the pwm.
3. Set the clock divider of the corresponding timer.
4. Set the duty of the pwm.

**Parameters**

<i>channel</i>	The analog channel to output, optional value: <a href="#">ANALOG_OUTPUT_0</a> , <a href="#">ANALOG_OUTPUT_1</a> , <a href="#">ANALOG_OUTPUT_2</a>
<i>value</i>	The analog value to output.
<i>p_vdd5</i>	The vlotage of external power supply, it should be connected to p_vdd5, a pin of M2. On the Oasis V1.1 (black board version) of M2, it is determined by Jumper 6 and Jumper 7. Choose Jumper 6 for 5V and Jumper 7 for 3.3V. ATTENTION! DO NOT CHOOSE JUMPER 6 AND JUMPER 7 AT THE SAME TIME!

**Returns**

void

**27.2.2.19 RT\_OPO\_ExchangeChannelPin**

```
#define RT_OPO_ExchangeChannelPin(
    switch )
```

**Value:**

```
{
    RT_ADC_V2P_SetResistor(0);
    MemoryAnd32(AD_OPO_REG, ~OPO_EXCHANGE_PIN);
    MemoryOr32(AD_OPO_REG, switch);
}
```

Set double side mode and exchange P side and N side in double side mode.

Note

AD\_OPO\_REG[2]: 1 for inn and inp switch, 0 for not (normal use). AD\_OPO\_REG[2]: 0 for not exchange, 1 for exchange.

Parameters

switch	optional value:  OPO_EXCHANGE_PIN, OPO_NOT_EXCHANGE_PIN
--------	---

Returns

void

27.2.2.20 RT\_OPO\_Off

```
#define RT_OPO_Off( ) MemoryAnd32(AD_OPO_REG, ~0x1)
```

Turn the Amplifier off.

Note

AD\_OPO\_REG[0]: 1=on, 0=off

Returns

void

27.2.2.21 RT\_OPO\_On

```
#define RT_OPO_On( ) MemoryOr32(AD_OPO_REG, 0x1)
```

Turn the Amplifier on.

Note

AD\_OPO\_REG[0]: 1=on, 0=off

Returns

void

### 27.2.2.22 RT\_OPO\_SelectChannelPin

```
#define RT_OPO_SelectChannelPin(
    pin )
```

#### Value:

```
{
    RT_ADC_V2P_SetResistor(V2P_185K); \
    RT_ADC_TemperatureSensorOff(); \
    MemoryAnd32(AD_OPO_REG, ~OPO_EXCHANGE_PIN); \
    MemoryOr32(AD_OPO_REG, pin); \
}
```

Set single side mode and select P side or N side in single side mode.

#### Note

AD\_CTL0\_REG[10:9]: should be 11(binary) when opo is required to be single mode. Temperature sensor should be turned off. AD\_OPO\_REG[15]: 1 for single side, 0 for double side. AD\_OPO\_REG[2]: 0 for n side, 1 for p side.

#### Parameters

<i>pin</i>	Pin to select, optional value:  OPO_NSIDE, OPO_PSIDE
------------	--

#### Returns

void

### 27.2.2.23 RT\_OPO\_SetAmplification

```
#define RT_OPO_SetAmplification(
    Pin,
    Pgain,
    Nin,
    Ngain )
```

#### Value:

```
{
    MemoryAnd32(AD_OPO_REG, ~(0xff << 8)); \
    MemoryOr32(AD_OPO_REG, ((Pin << 7) + (Pgain << 4) + (Nin << 3) + Ngain) << 8)); \
}
```

Set the amplification of the amplifier.

#### Note

AD\_OPO\_REG[15]: Pin, AD\_OPO\_REG[14:12]: Pgain; AD\_OPO\_REG[11]: Nin, AD\_OPO\_REG[10:8]: Ngain.

Parameters

<i>Pin</i>	Pin resistor, optional value:  OPO_PIN_RESISTOR_1K, OPO_PIN_RESISTOR_10K
<i>Pgain</i>	Pside amplification, optional value:  OPO_GAIN_20K, OPO_GAIN_40K, OPO_GAIN_80K, OPO_GAIN_100K, OPO_GAIN_240K, OPO_GAIN_320K, OPO_GAIN_400K, OPO_GAIN_480K
<i>Nin</i>	Nin resistor, optional value:  OPO_PIN_RESISTOR_1K, OPO_PIN_RESISTOR_10K
<i>Ngain</i>	Nside amplification, optional value:  OPO_GAIN_20K, OPO_GAIN_40K, OPO_GAIN_80K, OPO_GAIN_100K, OPO_GAIN_240K, OPO_GAIN_320K, OPO_GAIN_400K, OPO_GAIN_480K

Returns

void

27.2.2.24 RT\_OPO\_SetBypass

```
#define RT_OPO_SetBypass(  
    mode )
```

Value:

```
{  
    MemoryAnd32(AD_CTL0_REG, ~(1 << 15)); \br/>    MemoryOr32(AD_CTL0_REG, (mode << 15)); \br/>}
```

Set the bypass of opo. If opo is bypassed, input will be fed into SD directly. (select the filter resistor, 1 for 50k and 0 for 100k)

Note

AD\_CTL0\_REG[15]: 1=bypass, 0=no bypass

Parameters

<i>mode</i>	bypass switch, optional value:  ON, OFF
-------------	---

Returns

void

### 27.2.2.25 RT\_OPO\_SetChannel

```
#define RT_OPO_SetChannel(  
    ch0,  
    ch1,  
    ch2,  
    ch3 )
```

#### Value:

```
{  
    MemoryAnd32(AD_OPO_REG, ~(0xf << 4));  
    MemoryOr32(AD_OPO_REG, ((ch0 << 4) + (ch1 << 5) + (ch2 << 6) + (ch3 << 7)));  
}
```

Set the state of every amplifier channel.

#### Parameters

<i>ch0</i>	channel 0, optional value:  <i>ON</i> , <i>OFF</i>
<i>ch1</i>	channel 1, optional value:  <i>ON</i> , <i>OFF</i>
<i>ch2</i>	channel 2, optional value:  <i>ON</i> , <i>OFF</i>
<i>ch3</i>	channel 3, optional value:  <i>ON</i> , <i>OFF</i>

#### Returns

void

### 27.2.2.26 RT\_OPO\_SetDifferentialMode

```
#define RT_OPO_SetDifferentialMode(  
    ch0,  
    ch1,  
    ch2,  
    ch3,  
    exchange )
```

#### Value:

```
{
    RT_ADC_V2P_SetResistor(0);
    RT_OPO_ExchangeChannelPin(exchange);
    RT_OPO_SetChannel(ch0, ch1, ch2, ch3);
}
```

Set OPO at Differential Mode.

Parameters

ch0	channel 0, optional value:  ON, OFF
ch1	channel 1, optional value:  ON, OFF
ch2	channel 2, optional value:  ON, OFF
ch3	channel 3, optional value:  ON, OFF
exchange	Exchange P side and N side or not. optional value:  OPO_EXCHANGE_PIN, OPO_NOT_EXCHANGE_PIN  .

Returns

void

27.2.2.27 RT\_OPO\_SetPsideFeedback

```
#define RT_OPO_SetPsideFeedback(  
    mode )
```

Value:

```
{
    MemoryAnd32(AD_OPO_REG, ~(1 << 3));
    MemoryOr32(AD_OPO_REG, mode << 3);
}
```

Set the feedback mode of P-Side of the amplifier. When mode = 1, feedback is on, vcom(1.5V) is connected to the resistor block at inp When mode = 0, vcom is disconnected to the resistor block at inp.

Note

Set AD\_OPO\_REG[3]

**Parameters**

<i>mode</i>	Pside feedback switch, optional value:  ON, OFF
-------------	--

**Returns**

void

**27.2.2.28 RT\_OPO\_SetShort**

```
#define RT_OPO_SetShort(
    mode )
```

**Value:**

```
{
    RT_OPO_SetChannel (OFF, OFF, OFF, OFF); \
    if (mode) \
    { \
        MemoryAnd32 (AD_OPO_REG, ~(0xf << 4)); \
        MemoryOr32 (AD_OPO_REG, (1 << 1)); \
    } \
    else \
        MemoryAnd32 (AD_OPO_REG, ~(1 << 1)); \
}
```

Set whether to connecte pin and nin to vcom. Channel 0-3 should be closed!

**Parameters**

<i>mode</i>	short switch, optional value:  ON, OFF
-------------	---

**Returns**

void

**27.2.2.29 RT\_OPO\_SetSingleSideMode**

```
#define RT_OPO_SetSingleSideMode(
    ch0,
    ch1,
    ch2,
    ch3,
    pin )
```

**Value:**

```
{
    RT_OPO_SelectChannelPin(pin);
    RT_OPO_SetChannel(ch0, ch1, ch2, ch3);
}
```

Set OPO at SingleSide Mode, open specific channel and select specific pin.

Parameters

ch0	channel 0, optional value:  ON, OFF
ch1	channel 1, optional value:  ON, OFF
ch2	channel 2, optional value:  ON, OFF
ch3	channel 3, optional value:  ON, OFF
pin	Select P side or N side. Optional value:  OPO_NSIDE, OPO_PSIDE

Returns

void

27.2.2.30 RT\_OPO\_SetVPGND

```
#define RT_OPO_SetVPGND(  
    op )
```

Value:

```
{
    MemoryAnd32(AD_CTL0_REG, ~(1 << 7));
    MemoryOr32(AD_CTL0_REG, op << 7);
}
```

Set VPGND op=1, PsideGND = vcom; op=0, PsideGND = 0.

Parameters

op	the option of gnd, optional value:  VCOM, GND
----	---



**Returns**

void

**27.2.3 Function Documentation****27.2.3.1 RT\_ADC\_SD\_Read()**

```
uint32_t RT_ADC_SD_Read ( )
```

Get the result of SD.

**Returns**

uint32\_t SD result.

**27.2.3.2 RT\_ADC\_V2P\_Read()**

```
uint32_t RT_ADC_V2P_Read ( )
```

Get the result of V2P.

**Returns**

int V2P result.

**27.3 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/DES/DES.c File Reference**

Standard Encryption Standard Library for M2.

```
#include "DES.h"
```

**Classes**

- struct [DES\\_Key\\_pair](#)  
*Store the C key and the D key of a DES key.*

**Typedefs**

- typedef uint32\_t [half\\_DES\\_Key](#)  
*Half of the key.*

## Functions

- `DES_Key * DES_generateSubKeys` (const `DES_Key` originalKey)  
*Generate 16 subKeys with the original DES key.*
- `MessageData DES_process` (`MessageData` originalData, `DES_Key` \*subKeys, `uint8_t` mode)  
*Process data with specific subkeys and specific mode.*
- `MessageData DES` (`MessageData` originalData, `DES_Key` originalKey, `uint8_t` mode)  
*Process data with specific original key and specific mode.*

### 27.3.1 Detailed Description

Standard Encryption Standard Library for M2.

#### Author

Zhirui Dai

#### Date

14 Jun 2018

#### Copyright

2018 Zhirui

### 27.3.2 Function Documentation

#### 27.3.2.1 DES()

```
MessageData DES (  
    MessageData originalData,  
    DES_Key originalKey,  
    uint8_t mode )
```

Process data with specific original key and specific mode.

#### Parameters

<i>originalData</i>	data to process.
<i>originalKey</i>	the original key.
<i>mode</i>	DES_ENCRYPT_MODE or DES_DECRYPT_MODE

#### Returns

`MessageData` Processed data.

### 27.3.2.2 DES\_generateSubKeys()

```
DES_Key* DES_generateSubKeys (
    const DES_Key originalKey )
```

Generate 16 subKeys with the original DES key.

#### Parameters

<i>originalKey</i>	The original DES key.
--------------------	-----------------------

#### Returns

DES\_Key\* A pointer to an array of 17 keys: 1 original key and 16 subKeys.

### 27.3.2.3 DES\_process()

```
MessageData DES_process (
    MessageData originalData,
    DES_Key * subKeys,
    uint8_t mode )
```

Process data with specific subkeys and specific mode.

#### Parameters

<i>originalData</i>	data to process.
<i>subKeys</i>	17 keys: 1 original key and 16 subkeys.
<i>mode</i>	DES_ENCRYPT_MODE or DES_DECRYPT_MODE

#### Returns

MessageData Processed data.

## 27.4 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/DES/DES.h File Reference

Data Encryption Standard Library for M2.

```
#include <stdint.h>
```

#### Classes

- union [DES\\_Key](#)  
Type for storing key to be used by DES Algorithm.
- union [MessageData](#)  
Type for storing data to be processed by DES Algorithm.

## Macros

- `#define DES_ENCRYPT_MODE 0x0`  
*Keyword DES\_ENCRYPT\_MODE.*
- `#define DES_DECRYPT_MODE 0x1`  
*Keyword DES\_DECRYPT\_MODE.*

## Functions

- `DES_Key * DES_generateSubKeys` (const `DES_Key` originalKey)  
*Generate 16 subKeys with the original DES key.*
- `MessageData DES_process` (`MessageData` originalData, `DES_Key` \*subKeys, `uint8_t` mode)  
*Process data with specific subkeys and specific mode.*
- `MessageData DES` (`MessageData` originalData, `DES_Key` originalKey, `uint8_t` mode)  
*Process data with specific original key and specific mode.*

### 27.4.1 Detailed Description

Data Encryption Standard Library for M2.

## Author

Zhirui Dai

## Date

14 Jun 2018

## Copyright

2018 Zhirui

### 27.4.2 Function Documentation

#### 27.4.2.1 DES()

```
MessageData DES (
    MessageData originalData,
    DES_Key originalKey,
    uint8_t mode )
```

Process data with specific original key and specific mode.

## Parameters

<i>originalData</i>	data to process.
<i>originalKey</i>	the original key.
<i>mode</i>	DES_ENCRYPT_MODE or DES_DECRYPT_MODE

**Returns**

[MessageData](#) Processed data.

**27.4.2.2 DES\_generateSubKeys()**

```
DES_Key* DES_generateSubKeys (
    const DES_Key originalKey )
```

Generate 16 subKeys with the original DES key.

**Parameters**

<i>originalKey</i>	The original DES key.
--------------------	-----------------------

**Returns**

DES\_Key\* A pointer to an array of 17 keys: 1 original key and 16 subKeys.

**27.4.2.3 DES\_process()**

```
MessageData DES_process (
    MessageData originalData,
    DES_Key * subKeys,
    uint8_t mode )
```

Process data with specific subkeys and specific mode.

**Parameters**

<i>originalData</i>	data to process.
<i>subKeys</i>	17 keys: 1 original key and 16 subkeys.
<i>mode</i>	DES_ENCRYPT_MODE or DES_DECRYPT_MODE

**Returns**

[MessageData](#) Processed data.

**27.5 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/Flash.h File Reference**

Flash Operation Library for M2.

```
#include "mcu.h"
```

## Macros

- `#define RT_Flash_Write(value, address)`  
*Flash Write procedure, 32-bit write.*
- `#define RT_Flash_Erase1k(address)`  
*Flash erase procedure, 1K-byte erase from the given address.*
- `#define RT_Flash_EraseFrom(address)`  
*Flash erase from the given address, to the end of the flash.*
- `#define MAC_ID 0x1001f3f0`  
*Chip ID location (this will be replaced by NVR chip ID)*
- `#define RT_Flash_SetMAC(id) RT_Flash_Write(id, MAC_ID)`  
*Set chip identity(MAC\_ID).*
- `#define getMAC() MemoryRead32(MAC_ID)`  
*Get chip identity(MAC\_ID).*

### 27.5.1 Detailed Description

Flash Operation Library for M2.

#### Author

Zhirui Dai

#### Date

22 Jun 2018

#### Copyright

2018 Zhirui

### 27.5.2 Macro Definition Documentation

#### 27.5.2.1 MAC\_ID

```
#define MAC_ID 0x1001f3f0
```

Chip ID location (this will be replaced by NVR chip ID)

#### Returns

void

### 27.5.2.2 RT\_Flash\_Erase1k

```
#define RT_Flash_Erase1k(  
    address )
```

#### Value:

```
{  
    unsigned long addr;  
    FuncPtr1 funcptr;  
    funcptr = (FuncPtr1)FLASH_ERASE_ADDRESS;  
    addr = ((address&0x1ffff) | 0x10100000);  
    uint32_t oldVal = MemoryRead32(SYS_CTL2_REG);  
    RT_MCU_SetSystemClock(SYS_CLK_6M);  
    funcptr(addr);  
    MemoryWrite32(SYS_CTL2_REG, oldVal);  
}
```

Flash erase procedure, 1K-byte erase from the given address.

**Parameters**

<i>address</i>	The starting address to erase, 1K-byte space to be erased
----------------	---

**Returns**

void

**27.5.2.3 RT\_Flash\_EraseFrom**

```
#define RT_Flash_EraseFrom(  
    address )
```

**Value:**

```
{  
    for(unsigned long addr=address; addr<0x10001f400; addr+=0x400) {  
        RT_Flash_Erase1k(addr);  
    }  
}
```

Flash erase from the given address, to the end of the flash.

**Parameters**

<i>address</i>	The starting address to the end (0x10001f400)
----------------	---

**Returns**

void

**27.5.2.4 RT\_Flash\_Write**

```
#define RT_Flash_Write(  
    value,  
    address )
```

**Value:**

```
{  
    FuncPtr2 funcptr;  
    funcptr = (FuncPtr2)FLASH_WRITE_ADDRESS;  
    uint32_t oldVal = MemoryRead32(SYS_CTL2_REG);  
    RT_MCU_SetSystemClock(SYS_CLK_6M);  
    funcptr(value, address);  
    MemoryWrite32(SYS_CTL2_REG, oldVal);  
}
```

Flash Write procedure, 32-bit write.



## Parameters

<i>address</i>	The address to write, starting from 0x10000000
<i>value</i>	The value to be written, a 32-bit value

## Returns

void

## 27.6 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/Interrupt.h File Reference

Interrupt Library for M2.

```
#include "mcu.h"
```

## Macros

- #define `SYSINT_SPIINT` 0x8  
*Keyword SYSINT\_SPIINT.*
- #define `SYSINT_UART1INT` 0x7  
*Keyword SYSINT\_UART1INT.*
- #define `SYSINT_WDTINT` 0x6  
*Keyword SYSINT\_WDTINT.*
- #define `SYSINT_EXTINT` 0x5  
*Keyword SYSINT\_EXTINT.*
- #define `SYSINT_DBGINT` 0x4  
*Keyword SYSINT\_DBGINT.*
- #define `SYSINT_TC2INT` 0x3  
*Keyword SYSINT\_TC2INT.*
- #define `SYSINT_TC1INT` 0x2  
*Keyword SYSINT\_TC1INT.*
- #define `SYSINT_TC0INT` 0x1  
*Keyword SYSINT\_TC0INT.*
- #define `SYSINT_UART0INT` 0x0  
*Keyword SYSINT\_UART0INT.*
- #define `RT_SYSINT_GetFlag(device)` (MemoryRead(`SYS_IRQ_REG`) & (0x1 << device) >> device)  
*Check interrupt flag of SPI, UART1, WatchDog, ExternalInterrupt, Debug, Timer2, Timer1, Timer0 or UART0.*
- #define `RT_SYSINT_On()` MemoryOr(`SYS_CTL0_REG`, 0x1)  
*Turn on system interrupt.*
- #define `RT_SYSINT_Off()` MemoryAnd(`SYS_CTL0_REG`, ~0x1)  
*Turn off system interrupt.*
- #define `EXINT0` 0x0  
*Keyword EXINT0.*
- #define `EXINT1` 0x1  
*Keyword EXINT1.*
- #define `EXINT2` 0x2  
*Keyword EXINT2.*
- #define `EXINT3` 0x3

- *Keyword EXINT3.*
- `#define EXINT4 0x4`
- *Keyword EXINT4.*
- `#define EXINT5 0x5`
- *Keyword EXINT5.*
- `#define RISING 0x1`
- *Keyword RISING.*
- `#define FALLING 0x0`
- *Keyword FALLING.*
- `#define RT_EXINT_Setup(port, trigger)`
- *Set the external interrupt.*
- `#define RT_EXINT_Off(port) MemoryAnd32(INT_CTL0_REG, ~(1 << port))`
- *Close an external interrupt port.*
- `#define RT_EXINT_Clear(port) MemoryWrite(INT_CLR_REG, 1 << port)`
- *Clear interrupt flag from specific external interrupt port.*
- `#define RT_EXINT_ClearAll() MemoryWrite(INT_CLR_REG, 0xff)`
- *Clear all external interrupt flag.*
- `#define RT_EXINT_GetAllFlag() MemoryRead(INT_CTL1_REG)`
- *Get the external interrupt flag table.*
- `#define RT_EXINT_GetFlag(port) ((RT_EXINT_GetAllFlag() >> port) & 0x1)`
- *Get the flag of specific external interrupt port.*

### 27.6.1 Detailed Description

Interrupt Library for M2.

#### Author

Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.6.2 Macro Definition Documentation

#### 27.6.2.1 EXINT0

```
#define EXINT0 0x0
```

Keyword EXINT0.

#### Note

Optional value used in RT\_EXINT\_Setup

### 27.6.2.2 EXINT1

```
#define EXINT1 0x1
```

Keyword EXINT1.

#### Note

Optional value used in RT\_EXINT\_Setup

### 27.6.2.3 EXINT2

```
#define EXINT2 0x2
```

Keyword EXINT2.

#### Note

Optional value used in RT\_EXINT\_Setup

### 27.6.2.4 EXINT3

```
#define EXINT3 0x3
```

Keyword EXINT3.

#### Note

Optional value used in RT\_EXINT\_Setup

### 27.6.2.5 EXINT4

```
#define EXINT4 0x4
```

Keyword EXINT4.

#### Note

Optional value used in RT\_EXINT\_Setup

#### 27.6.2.6 EXINT5

```
#define EXINT5 0x5
```

Keyword EXINT5.

##### Note

Optional value used in RT\_EXINT\_Setup

#### 27.6.2.7 FALLING

```
#define FALLING 0x0
```

Keyword FALLING.

##### Note

Optional value used in RT\_EXINT\_Setup

#### 27.6.2.8 RISING

```
#define RISING 0x1
```

Keyword RISING.

##### Note

Optional value used in RT\_EXINT\_Setup

#### 27.6.2.9 RT\_EXINT\_Clear

```
#define RT_EXINT_Clear(  
    port ) MemoryWrite(INT_CLR_REG, 1 << port)
```

Clear interrupt flag from specific external interrupt port.

##### Parameters

<i>port</i>	the external interrupt port to clear irq flag, optional value:  <code>EXINT0</code> , <code>EXINT1</code> , <code>EXINT2</code> , <code>EXINT3</code> , <code>EXINT4</code> , <code>EXINT5</code>  .
-------------	--

**Returns**

void

**27.6.2.10 RT\_EXINT\_ClearAll**

```
#define RT_EXINT_ClearAll( ) MemoryWrite(INT_CLR_REG, 0xff)
```

Clear all external interrupt flag.

**Returns**

void

**27.6.2.11 RT\_EXINT\_GetAllFlag**

```
#define RT_EXINT_GetAllFlag( ) MemoryRead(INT_CTL1_REG)
```

Get the external interrupt flag table.

**Returns**

the external interrupt flag table

**27.6.2.12 RT\_EXINT\_GetFlag**

```
#define RT_EXINT_GetFlag(  
    port ) ((RT_EXINT_GetAllFlag() >> port) & 0x1)
```

Get the flag of specific external interrupt port.

**Returns**

The flag of the external interrupt port.

**27.6.2.13 RT\_EXINT\_Off**

```
#define RT_EXINT_Off(  
    port ) MemoryAnd32(INT_CTL0_REG, ~(1 << port))
```

Close an external interrupt port.

**Parameters**

<i>port</i>	the external interrupt port to close, optional value:  <code>EXINT0</code> , <code>EXINT1</code> , <code>EXINT2</code> , <code>EXINT3</code> , <code>EXINT4</code> , <code>EXINT5</code>  .
-------------	---

**Returns**

`void`

**27.6.2.14 RT\_EXINT\_Setup**

```
#define RT_EXINT_Setup(  
    port,  
    trigger )
```

**Value:**

```
{  
    MemoryOr32(INT_CTL0_REG, 1 << port); \  
    MemoryAnd32(INT_CTL2_REG, ~(RISING << port)); \  
    MemoryOr32(INT_CTL2_REG, trigger << port); \  
}
```

Set the external interrupt.

**Parameters**

<i>port</i>	the port of external interrupt, optional value:  <code>EXINT0</code> , <code>EXINT1</code> , <code>EXINT2</code> , <code>EXINT3</code> , <code>EXINT4</code> , <code>EXINT5</code>  .
<i>trigger</i>	the trigger mode, optional value:  <code>RISING</code> , <code>FALLING</code>

**Returns**

`void`

**27.6.2.15 RT\_SYSINT\_GetFlag**

```
#define RT_SYSINT_GetFlag(  
    device ) (MemoryRead(SYS_IRQ_REG)&(0x1 << device) >> device)
```

Check interrupt flag of SPI, UART1, WatchDog, ExternalInterrupt, Debug, Timer2, Timer1, Timer0 or UART0.

**Note**

SYS\_IRQ\_REG[8:0]: 9 devices.

**Parameters**

<i>device</i>	Optional value:  <a href="#">SYSINT_SPIINT</a> , <a href="#">SYSINT_UART1INT</a> , <a href="#">SYSINT_UART0INT</a> , <a href="#">SYSINT_WDTINT</a> , <a href="#">SYSINT_EXTINT</a> , <a href="#">SYSINT_DBGINT</a> , <a href="#">SYSINT_TC2INT</a> , <a href="#">SYSINT_TC1INT</a> . <a href="#">SYSINT_TCOINT</a> .
---------------	--

**27.6.2.16 RT\_SYSINT\_Off**

```
#define RT_SYSINT_Off( ) MemoryAnd(SYS\_CTL0\_REG, ~0x1)
```

Turn off system interrupt.

**Note**

SYS\_CTL0\_REG[0]: 1=enable system interrupt, 0=disable system interrupt.

**27.6.2.17 RT\_SYSINT\_On**

```
#define RT_SYSINT_On( ) MemoryOr(SYS\_CTL0\_REG, 0x1)
```

Turn on system interrupt.

**Note**

SYS\_CTL0\_REG[0]: 1=enable system interrupt, 0=disable system interrupt.

**27.6.2.18 SYSINT\_DBGINT**

```
#define SYSINT_DBGINT 0x4
```

Keyword SYSINT\_DBGINT.

**Note**

Optional value for RT\_SYSINT\_Flag

#### 27.6.2.19 SYSINT\_EXTINT

```
#define SYSINT_EXTINT 0x5
```

Keyword SYSINT\_EXTINT.

##### Note

Optional value for RT\_SYSINT\_Flag

#### 27.6.2.20 SYSINT\_SPIINT

```
#define SYSINT_SPIINT 0x8
```

Keyword SYSINT\_SPIINT.

##### Note

Optional value for RT\_SYSINT\_Flag

#### 27.6.2.21 SYSINT\_TC0INT

```
#define SYSINT_TC0INT 0x1
```

Keyword SYSINT\_TC0INT.

##### Note

Optional value for RT\_SYSINT\_Flag

#### 27.6.2.22 SYSINT\_TC1INT

```
#define SYSINT_TC1INT 0x2
```

Keyword SYSINT\_TC1INT.

##### Note

Optional value for RT\_SYSINT\_Flag



#### 27.6.2.23 SYSINT\_TC2INT

```
#define SYSINT_TC2INT 0x3
```

Keyword SYSINT\_TC2INT.

##### Note

Optional value for RT\_SYSINT\_Flag

#### 27.6.2.24 SYSINT\_UART0INT

```
#define SYSINT_UART0INT 0x0
```

Keyword SYSINT\_UART0INT.

##### Note

Optional value for RT\_SYSINT\_Flag

#### 27.6.2.25 SYSINT\_UART1INT

```
#define SYSINT_UART1INT 0x7
```

Keyword SYSINT\_UART1INT.

##### Note

Optional value for RT\_SYSINT\_Flag

#### 27.6.2.26 SYSINT\_WDTINT

```
#define SYSINT_WDTINT 0x6
```

Keyword SYSINT\_WDTINT.

##### Note

Optional value for RT\_SYSINT\_Flag

### 27.7 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/IO.h File Reference

General Input Output Library for M2.

```
#include "mcu.h"
```

## Macros

- #define `OUTPUT` 0x1  
*Keyword OUTPUT.*
- #define `INPUT` 0x0  
*Keyword INPUT.*
- #define `HIGH` 0x1  
*Keyword HIGH.*
- #define `LOW` 0x0  
*Keyword LOW.*
- #define `RT_IO_SetOutput`(io) `MemoryOr32(SYS_IOCTL_REG, (1 << (io)))`  
*This function sets a specific channel at OUTPUT mode.*
- #define `RT_IO_SetInput`(io) `MemoryAnd32(SYS_IOCTL_REG, ~(1 << (io)))`  
*This function sets a specific channel at INPUT mode.*
- #define `RT_IO_SetInputOutput16`(mode) `MemoryWrite32(SYS_IOCTL_REG, mode)`  
*This function sets the mode of all GPIO channels.*
- #define `RT_IO_SetHigh`(io) `MemoryOr32(SYS_GPIO0_REG, (1 << (io)))`  
*This function sets a specific output io channel at HIGH level.*
- #define `RT_IO_SetLow`(io) `MemoryAnd32(SYS_GPIO0_REG, ~(1 << (io)))`  
*This function sets a specific output io channel at LOW level.*
- #define `RT_IO_SetLevel16`(level) `MemoryWrite32(SYS_GPIO0_REG, (level))`  
*This function sets the level of all 16 io channel.*
- #define `RT_IO_Read`(io) `((MemoryRead32(SYS_GPIO1_REG) >> (io)) & 0x1)`  
*This function returns the level of a specific io.*
- #define `RT_IO_Read16`() `MemoryRead32(SYS_GPIO1_REG)`  
*This function returns the level of all 16 io.*
- #define `RT_IO_SetInputOutput`(io, mode)  
*This function sets the mode of specific GPIO channel.*
- #define `RT_IO_SetLevel`(io, level)  
*This function sets the level of a specific io channel.*

### 27.7.1 Detailed Description

General Input Output Library for M2.

#### Author

Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.7.2 Macro Definition Documentation

#### 27.7.2.1 RT\_IO\_Read

```
#define RT_IO_Read(  
    io ) ((MemoryRead32(SYS_GPIO1_REG) >> (io)) & 0x1)
```

This function returns the level of a specific io.

**Parameters**

<i>io</i>	the io channel to read
-----------	------------------------

**Returns**

the io level

**27.7.2.2 RT\_IO\_Read16**

```
#define RT_IO_Read16( ) MemoryRead32(SYS_GPIO1_REG)
```

This function returns the level of all 16 io.

**Returns**

uint16\_t 16-bit number which defines 16 io channels' level

**27.7.2.3 RT\_IO\_SetHigh**

```
#define RT_IO_SetHigh(
    io ) MemoryOr32(SYS_GPIO0_REG, (1 << (io)))
```

This function sets a specific output io channel at HIGH level.

**Parameters**

<i>io</i>	Channel to set.
-----------	-----------------

**Returns**

void

**27.7.2.4 RT\_IO\_SetInput**

```
#define RT_IO_SetInput(
    io ) MemoryAnd32(SYS_IOCTL_REG, ~(1 << (io)))
```

This function sets a specific channel at INPUT mode.

**Parameters**

<i>io</i>	Channel to set.
-----------	-----------------

**Returns**

void

**27.7.2.5 RT\_IO\_SetInputOutput**

```
#define RT_IO_SetInputOutput (
    io,
    mode )
```

**Value:**

```
{
    if (!mode) \
        IO_SetInput(io); \
    else \
        IO_SetOutput(io); \
}
```

This function sets the mode of specific GPIO channel.

**Parameters**

<i>io</i>	the specific io channel to setup
<i>mode</i>	the mode, optional value: <code>INPUT</code> , <code>OUTPUT</code>

**Returns**

void

**27.7.2.6 RT\_IO\_SetInputOutput16**

```
#define RT_IO_SetInputOutput16(
    mode ) MemoryWrite32(SYS_IOCTL_REG, mode)
```

This function sets the mode of all GPIO channels.

**Parameters**

<i>mode</i>	16-bit number which defines 16 IO channels' mode
-------------	--

**Returns**

void

### 27.7.2.7 RT\_IO\_SetLevel

```
#define RT_IO_SetLevel(
    io,
    level )
```

#### Value:

```
{
    IO_SetOutput(io); \
    if (!level) \
        IO_SetLow(io); \
    else \
        IO_SetHigh(io); \
}
```

This function sets the level of a specific io channel.

#### Parameters

<i>io</i>	the io channel to setup
<i>level</i>	the io level, optional value:  HIGH, LOW

#### Returns

void

### 27.7.2.8 RT\_IO\_SetLevel16

```
#define RT_IO_SetLevel16(
    level ) MemoryWrite32(SYS_GPIO0_REG, (level))
```

This function sets the level of all 16 io channel.

#### Parameters

<i>level</i>	16-bit number which defines 16 IO channels' level
--------------	---

#### Returns

void

### 27.7.2.9 RT\_IO\_SetLow

```
#define RT_IO_SetLow(
    io ) MemoryAnd32(SYS_GPIO0_REG, ~(1 << (io)))
```

This function sets a specific output io channel at LOW level.

## Parameters

<i>io</i>	Channel to set.
-----------	-----------------

## Returns

void

## 27.7.2.10 RT\_IO\_SetOutput

```
#define RT_IO_SetOutput(  
    io ) MemoryOr32(SYS_IOCTL_REG, (1 << (io)))
```

This function sets a specific channel at OUTPUT mode.

## Parameters

<i>io</i>	Channel to set.
-----------	-----------------

## Returns

void

## 27.8 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/LCD/LCD.c File Reference

LCD Library for M2.

```
#include "LCD.h"  
#include "mcu.h"
```

## Functions

- void **RT\_LCD\_Initialize** ()
- void **RT\_LCD\_ShowAll** ()
- void **RT\_LCD\_DisOn** ()
- void **RT\_LCD\_DisHiPressure** (unsigned int b)
- void **RT\_LCD\_DisLoPressure** (unsigned int b)
- void **RT\_LCD\_DisHeartRate** (unsigned int b)
- void **RT\_LCD\_DisTime** (unsigned int hour, unsigned int minute)
- void **LRT\_CD\_DisDate** (unsigned int month, unsigned int day)
- void **RT\_LCD\_DisSign** (unsigned long x, unsigned long y)
- void **RT\_LCD\_ClearSign** (unsigned long x, unsigned long y)
- void **RT\_LCD\_GetSegCode1** (unsigned int b)
- void **RT\_LCD\_GetSegCode2** (unsigned int b)
- void **RT\_LCD\_GetSegCode3** (unsigned int b)
- void **RT\_LCD\_ExtractDigit** (unsigned int b)
- void **RT\_LCD\_BlinkSign** (unsigned long x, unsigned long y)
- void **RT\_LCD\_BlinkMinute** ()
- void **RT\_LCD\_BlinkHour** ()
- void **RT\_LCD\_BlinkDay** ()
- void **RT\_LCD\_BlinkMonth** ()

### 27.8.1 Detailed Description

LCD Library for M2.

#### Author

John & Jack, Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

## 27.9 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/LCD/LCD.h File Reference

LCD Library for M2.

#### Macros

- `#define LCD_M2 0,0`
- `#define kPa 0,2`
- `#define DP1 0,2`
- `#define mmHg 0,4`
- `#define DN 0,6`
- `#define UP 4,6`
- `#define LB 8,6`
- `#define M1 12,6`
- `#define L3 0,7`
- `#define L4 4,7`
- `#define L5 8,7`
- `#define L6 12,7`
- `#define L2 0,8`
- `#define L1 4,8`
- `#define IHB 8,8`
- `#define AVG 12,8`
- `#define elAGD 12,10`
- `#define DP2 12,12`
- `#define E13 12,14`
- `#define AM 12,15`
- `#define C9 8,15`
- `#define B9 4,15`
- `#define nineAGDE 0,15`
- `#define HT 12,17`
- `#define colon 12,19`
- `#define PM 12,21`
- `#define No 12,22`
- `#define HR 12,24`
- `#define C13 0,26`

- `#define M_2 12,28`
- `#define B13 12,30`
- `#define segment 0x01000600`
- `#define digit 0x01000610`
- `#define days 0x01000613`
- `#define LCDCode1 0x01000620`
- `#define LCDCode2 0x01000630`
- `#define LCDCode3 0x01000640`
- `#define flaga 0x0100006f`
- `#define subsecond 0x01000650`
- `#define second 0x01000654`
- `#define minuteg 0x01000658`
- `#define hourg 0x01000678`
- `#define dayg 0x01000660`
- `#define monthg 0x01000664`
- `#define SetMode (*(volatile unsigned long *) (0x01000664))`

## Functions

- `void RT_LCD_Initialize ()`
- `void RT_LCD_ShowAll ()`
- `void RT_LCD_DispOn ()`
- `void RT_LCD_DisHiPressure (unsigned int b)`
- `void RT_LCD_DisLoPressure (unsigned int b)`
- `void RT_LCD_DisHeartRate (unsigned int b)`
- `void RT_LCD_DisDate (unsigned int month, unsigned int day)`
- `void RT_LCD_DisTime (unsigned int hour, unsigned int minute)`
- `void RT_LCD_DisSign (unsigned long x, unsigned long y)`
- `void RT_LCD_ClearSign (unsigned long x, unsigned long y)`
- `void RT_LCD_GetSegCode1 (unsigned int b)`
- `void RT_LCD_GetSegCode2 (unsigned int b)`
- `void RT_LCD_GetSegCode3 (unsigned int b)`
- `void RT_LCD_ExtractDigit (unsigned int b)`
- `void RT_LCD_BlinkSign (unsigned long x, unsigned long y)`
- `void RT_LCD_BlinkMinute ()`
- `void RT_LCD_BlinkHour ()`
- `void RT_LCD_BlinkDay ()`
- `void RT_LCD_BlinkMonth ()`
- `void RT_LCD_delay (unsigned int a)`

### 27.9.1 Detailed Description

LCD Library for M2.

#### Author

John & Jack, Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui



## 27.10 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/mcu/mcu.c File Reference

M2 micro core unit.

```
#include "mcu.h"
```

### Functions

- void [RT\\_Clr\\_Sram](#) ()  
*This function is to clear the sram.*

### 27.10.1 Detailed Description

M2 micro core unit.

#### Author

Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.10.2 Function Documentation

#### 27.10.2.1 RT\_Clr\_Sram()

```
void RT_Clr_Sram ( )
```

This function is to clear the sram.

#### Returns

void

## 27.11 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/mcu/mcu.h File Reference

M2 micro core unit.

```
#include <stdint.h>
```

## Macros

- `#define ON 0x1`  
*Keyword ON.*
- `#define OFF 0x0`  
*Keyword OFF.*
- `#define RAISE_TRIGGER 0x1`  
*Keyword RAISE\_TRIGGER.*
- `#define FALL_TRIGGER 0x0`  
*Keyword FALL\_TRIGGER.*
- `#define SYS_CTL0_REG (unsigned long)0x1f800700`  
*External register address.*
- `#define SYS_CTL2_REG (unsigned long)0x1f800702`  
*External register address.*
- `#define SYS_IOCTL_REG (unsigned long)0x1f800704`  
*External register address.*
- `#define SYS_IRQ_REG (unsigned long)0x1f800707`  
*External register address.*
- `#define USER_INT (unsigned long)0x01001FFC`  
*External register address.*
- `#define PC_LOC (unsigned long)0x01001FF8`  
*External register address.*
- `#define INT_COUNT (unsigned long)0x01001FF4`  
*External register address.*
- `#define INT_CTL0_REG (unsigned long)0x1f800500`  
*External register address.*
- `#define INT_CTL1_REG (unsigned long)0x1f800501`  
*External register address.*
- `#define INT_CTL2_REG (unsigned long)0x1f800502`  
*External register address.*
- `#define INT_CLR_REG (unsigned long)0x1f800503`  
*External register address.*
- `#define UART0_READ_REG (unsigned long)0x1f800000`  
*External register address.*
- `#define UART0_BUSY_REG (unsigned long)0x1f800001`  
*External register address.*
- `#define UART0_WRITE_REG (unsigned long)0x1f800002`  
*External register address.*
- `#define UART0_IRQ_ACK_REG (unsigned long)0x1f800003`  
*External register address.*
- `#define UART0_CTL_REG (unsigned long)0x1f800004`  
*External register address.*
- `#define UART0_DATA_RDY_REG (unsigned long)0x1f800005`  
*External register address.*
- `#define UART0_LIN_BREAK_REG (unsigned long)0x1f800006`  
*External register address.*
- `#define UART0_BRP_REG (unsigned long)0x1f800007`  
*External register address.*
- `#define UART1_READ_REG (unsigned long)0x1f800800`  
*External register address.*
- `#define UART1_BUSY_REG (unsigned long)0x1f800801`

- External register address.*

  - #define [UART1\\_WRITE\\_REG](#) (unsigned long)0x1f800802

*External register address.*
- #define [UART1\\_IRQ\\_ACK\\_REG](#) (unsigned long)0x1f800803

*External register address.*
- #define [UART1\\_CTL\\_REG](#) (unsigned long)0x1f800804

*External register address.*
- #define [UART1\\_DATA\\_RDY\\_REG](#) (unsigned long)0x1f800805

*External register address.*
- #define [UART1\\_LIN\\_BREAK\\_REG](#) (unsigned long)0x1f800806

*External register address.*
- #define [UART1\\_BRP\\_REG](#) (unsigned long)0x1f800807

*External register address.*
- #define [SPI\\_READ\\_REG](#) (unsigned long)0x1f800d00

*External register address.*
- #define [SPI\\_BUSY\\_REG](#) (unsigned long)0x1f800d01

*External register address.*
- #define [SPI\\_WRITE\\_REG](#) (unsigned long)0x1f800d02

*External register address.*
- #define [SPI\\_IRQ\\_ACK\\_REG](#) (unsigned long)0x1f800d03

*External register address.*
- #define [SPI\\_CTL\\_REG](#) (unsigned long)0x1f800d04

*External register address.*
- #define [SPI\\_DATA\\_RDY\\_REG](#) (unsigned long)0x1f800d05

*External register address.*
- #define [SYS\\_GDR\\_REG](#) (unsigned long)0x1f800703

*External register address.*
- #define [SYS\\_IOCTL\\_REG](#) (unsigned long)0x1f800704

*External register address.*
- #define [SYS\\_GPIO0\\_REG](#) (unsigned long)0x1f800705

*External register address.*
- #define [SYS\\_GPIO1\\_REG](#) (unsigned long)0x1f800706

*External register address.*
- #define [T0\\_CTL0\\_REG](#) (unsigned long)0x1f800100

*External register address.*
- #define [T0\\_REF\\_REG](#) (unsigned long)0x1f800101

*External register address.*
- #define [T0\\_READ\\_REG](#) (unsigned long)0x1f800102

*External register address.*
- #define [T0\\_CLRIRQ\\_REG](#) (unsigned long)0x1f800103

*External register address.*
- #define [T0\\_CLK\\_REG](#) (unsigned long)0x1f800104

*External register address.*
- #define [T0\\_CLRCNT\\_REG](#) (unsigned long)0x1f800105

*External register address.*
- #define [T1\\_CTL0\\_REG](#) (unsigned long)0x1f800200

*External register address.*
- #define [T1\\_REF\\_REG](#) (unsigned long)0x1f800201

*External register address.*
- #define [T1\\_READ\\_REG](#) (unsigned long)0x1f800202

*External register address.*

- `#define T1_CLRIRQ_REG` (unsigned long)0x1f800203  
*External register address.*
- `#define T1_CLK_REG` (unsigned long)0x1f800204  
*External register address.*
- `#define T1_CLRCNT_REG` (unsigned long)0x1f800205  
*External register address.*
- `#define T2_CTL0_REG` (unsigned long)0x1f800400  
*External register address.*
- `#define T2_REF_REG` (unsigned long)0x1f800401  
*External register address.*
- `#define T2_READ_REG` (unsigned long)0x1f800402  
*External register address.*
- `#define T2_CLRIRQ_REG` (unsigned long)0x1f800403  
*External register address.*
- `#define T2_CLK_REG` (unsigned long)0x1f800404  
*External register address.*
- `#define T2_CLRCNT_REG` (unsigned long)0x1f800405  
*External register address.*
- `#define T2_PHASE_REG` (unsigned long)0x1f800406  
*External register address.*
- `#define T4_CTL0_REG` (unsigned long)0x1f800a00  
*External register address.*
- `#define T4_REF0_REG` (unsigned long)0x1f800a01  
*External register address.*
- `#define T4_CLK0_REG` (unsigned long)0x1f800a02  
*External register address.*
- `#define T4_REF1_REG` (unsigned long)0x1f800a03  
*External register address.*
- `#define T4_CLK1_REG` (unsigned long)0x1f800a04  
*External register address.*
- `#define AD_CTL0_REG` (unsigned long)0x1f800600  
*External register address.*
- `#define AD_OPO_REG` (unsigned long)0x1f800601  
*External register address.*
- `#define AD_READ_REG` (unsigned long)0x1f800602  
*External register address.*
- `#define AD_CLR_REG` (unsigned long)0x1f800603  
*External register address.*
- `#define LCD_CTL0_REG` (unsigned long)0x1f800300  
*External register address.*
- `#define LCD_RAM_REG` (unsigned long)0x1f800380  
*External register address.*
- `#define LCD_RAM_LINE0` (unsigned long)0x1f800380  
*External register address.*
- `#define LCD_RAM_LINE1` (unsigned long)0x1f800384  
*External register address.*
- `#define LCD_RAM_LINE2` (unsigned long)0x1f800388  
*External register address.*
- `#define LCD_RAM_LINE3` (unsigned long)0x1f80038c  
*External register address.*
- `#define WDT_CTL0_REG` (unsigned long)0x1f800b00

- External register address.
  - #define `WDT_CLR_REG` (unsigned long)0x1f800b03
- External register address.
  - #define `WDT_READ_REG` (unsigned long)0x1f800b02
- External register address.
  - #define `RTC_CTL_REG` (unsigned long)0x1f800f00
- External register address.
  - #define `RTC_TIME_REG` (unsigned long)0x1f800f01
- External register address.
  - #define `RTC_CLR_REG` (unsigned long)0x1f800f03
- External register address.
  - #define `MemoryRead32`(addr) (\*(volatile unsigned long\*)(addr))
 Read 32-bit data from a specific address.
- #define `MemoryWrite32`(addr, val) (\*(volatile unsigned long\*)(addr))=(val)
 Write 32-bit data to a specific address.
- #define `MemoryOr32`(addr, val) (\*(volatile unsigned long\*)(addr))|=(val)
 Logical OR operation on 32-bit data at a specific address.
- #define `MemoryAnd32`(addr, val) (\*(volatile unsigned long\*)(addr))&=(val)
 Logical AND operation on 32-bit data from a specific address.
- #define `MemoryBitAt`(addr, val) ((\*(volatile unsigned long\*)(addr))&=(1<<(val)))>>(val)
 Get a specific bit of a 32-bit data from a specific address.
- #define `MemoryBitOn`(addr, val) `MemoryOr32`(addr,1<<(val))
 Set a specific bit of a 32-bit data from a specific address to 1.
- #define `MemoryBitOff`(addr, val) `MemoryAnd32`(addr,~(1<<(val)))
 Set a specific bit of a 32-bit data from a specific address to 0.
- #define `MemoryBitSwitch`(addr, val) (\*(volatile unsigned long\*)(addr))^(1<<(val)))
 Set a specific bit of a 32-bit data from a specific address to opposite state.
- #define `JumpTo`(address)
 Jump to a specific address.
- #define `SYS_CLK_3M` (0x0<<12)
 Keyword `SYS_CLK_3M`.
- #define `SYS_CLK_6M` (0x1<<12)
 Keyword `SYS_CLK_6M`.
- #define `SYS_CLK_12M` (0x3<<12)
 Keyword `SYS_CLK_12M`.
- #define `RT_MCU_SetSystemClock`(mode)
 Set the frequency of the system clock.

## Typedefs

- typedef void(\* `FuncPtr`) (void)
  - Function pointer type (void)-> void definition.
- typedef void(\* `FuncPtr1`) (unsigned long)
  - Function pointer type (unsigned long)-> void definition.
- typedef void(\* `FuncPtr2`) (unsigned long, unsigned long)
  - Function pointer type (unsigned long, unsigned long)-> void definition.

## Functions

- void `RT_Clr_Sram` ()
  - This function is to clear the sram.

### 27.11.1 Detailed Description

M2 micro core unit.

#### Author

Zhirui Dai

#### Date

2018-05-25

### 27.11.2 Macro Definition Documentation

#### 27.11.2.1 JumpTo

```
#define JumpTo(  
    address )
```

#### Value:

```
{  
    FuncPtr funcptr;  
    funcptr = (FuncPtr)address;  
    funcptr();  
}
```

Jump to a specific address.

#### Parameters

<i>address</i>	the address to jump
----------------	---------------------

#### Returns

void

#### 27.11.2.2 MemoryAnd32

```
#define MemoryAnd32(  
    addr,  
    val ) (*(volatile unsigned long*) (addr) &= (val))
```

Logical addrND operation on 32-bit data from a specific address.

**Parameters**

<i>addr</i>	the address to be AND
<i>val</i>	the AND mask

**Returns**

void

**27.11.2.3 MemoryBitAt**

```
#define MemoryBitAt(  
    addr,  
    val ) ((*(volatile unsigned long*)(addr) &= (1<<(val)))>>(val))
```

Get a specific bit of a 32-bit data from a specific address.

**Parameters**

<i>addr</i>	the address
<i>val</i>	the bit location (in the 32-bit data)

**Returns**

long the bit

**27.11.2.4 MemoryBitOff**

```
#define MemoryBitOff(  
    addr,  
    val ) MemoryAnd32(addr, ~(1<<(val)))
```

Set a specific bit of a 32-bit data from a specific address to 0.

**Parameters**

<i>addr</i>	the address
<i>val</i>	the bit location (in the 32-bit data)

**Returns**

void

### 27.11.2.5 MemoryBitOn

```
#define MemoryBitOn(  
    addr,  
    val ) MemoryOr32(addr,1<<(val))
```

Set a specific bit of a 32-bit data from a specific address to 1.

#### Parameters

<i>addr</i>	the address
<i>val</i>	the bit location (in the 32-bit data)

#### Returns

void

### 27.11.2.6 MemoryBitSwitch

```
#define MemoryBitSwitch(  
    addr,  
    val ) (*(volatile unsigned long*)(addr)^(1<<(val)))
```

Set a specific bit of a 32-bit data from a specific address to opposite state.

#### Parameters

<i>addr</i>	the address
<i>val</i>	the bit location (in the 32-bit data)

#### Returns

void

### 27.11.2.7 MemoryOr32

```
#define MemoryOr32(  
    addr,  
    val ) (*(volatile unsigned long*)(addr)|=(val))
```

Logical OR operation on 32-bit data at a specific address.

#### Parameters

<i>addr</i>	the address to be OR
<i>val</i>	the OR mask



**Returns**

void

**27.11.2.8 MemoryRead32**

```
#define MemoryRead32(
    addr ) (*(volatile unsigned long*)(addr))
```

Read 32-bit data from a specific address.

**Parameters**

<i>addr</i>	the address to read
-------------	---------------------

**Returns**

long the read value

**27.11.2.9 MemoryWrite32**

```
#define MemoryWrite32(
    addr,
    val ) (*(volatile unsigned long*)(addr)=(val))
```

Write 32-bit data to a specific address.

**Parameters**

<i>addr</i>	the address to write
<i>val</i>	the value to be written

**Returns**

void

**27.11.2.10 RT\_MCU\_SetSystemClock**

```
#define RT_MCU_SetSystemClock(
    mode )
```

**Value:**

```
{
    MemoryAnd32(SYS_CTL2_REG, ~SYS_CLK_12M); \
    MemoryOr32(SYS_CTL2_REG, mode);          \
}
```

Set the frequency of the system clock.

**Note**

This is used in UART to adjust the baudrate.

**Parameters**

<i>mode</i>	<p>The frequency of system clock to set. Optional value:</p> <p><code>SYS_CLK_3M</code>, <code>SYS_CLK_6M</code>, <code>SYS_CLK_12M</code></p> <p>.</p>
-------------	---

**Returns**

void

**27.11.3 Function Documentation****27.11.3.1 RT\_Clr\_Sram()**

```
void RT_Clr_Sram ( )
```

This function is to clear the sram.

**Returns**

void

**27.12 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/soft\_fp/soft\_fp.c File Reference**

Soft float library.

```
#include "soft_fp.h"
```

**Functions**

- float `fp_float32_neg` (float *a\_fp*)  
*Calculate the negative value of a\_fp.*
- float `fp_float32_abs` (float *a\_fp*)  
*Calculate absolute value.*
- float `fp_float32_add` (float *a\_fp*, float *b\_fp*)  
*Add arithmetic operation.*
- float `fp_float32_sub` (float *a\_fp*, float *b\_fp*)  
*Subtraction arithmetic operation.*
- float `fp_float32_mult` (float *a\_fp*, float *b\_fp*)  
*Multiplication.*
- float `fp_float32_div` (float *a\_fp*, float *b\_fp*)

*Float Division.*

- long `fp_float32_to_int32` (float a\_fp)  
*Convert float to signed long.*
- float `fp_int32_to_float32` (long af)  
*Convert signed long to float.*
- float `fp_uint32_to_float32` (unsigned long af)  
*Convert unsigned long to float.*
- int `fp_float32_cmp` (float a\_fp, float b\_fp)  
*Compare two float value.*
- float `fp_float32_sqrt` (float x)  
*Returns the square root of x.*
- float `fp_float32_cos` (float rad)  
*Returns the cosine of an angle of x radians.*
- float `fp_float32_sin` (float rad)  
*Returns the sine of an angle of x radians.*
- float `fp_float32_atan` (float x)  
*Returns the principal value of the arc tangent of x, expressed in radians.*
- float `fp_float32_atan2` (float y, float x)  
*Returns the principal value of the arc tangent of y/x, expressed in radians.*
- float `fp_float32_exp` (float x)  
*Returns the base-e exponential function of x, which is e raised to the power x.*
- float `fp_float32_log` (float x)  
*The natural logarithm is the base-e logarithm: the inverse of the natural exponential function (fp\_float32\_exp).*
- float `fp_float32_pow` (float x, float y)  
*Compute power.*

### 27.12.1 Detailed Description

Soft float library.

Author

Zhirui Dai

Date

1 Jun 2018

Copyright

2018 Zhirui

### 27.12.2 Function Documentation

#### 27.12.2.1 `fp_float32_abs()`

```
float fp_float32_abs (
    float a_fp )
```

Calculate absolute value.

**Parameters**

$a_{fp}$	The value to calculate absolute value.
----------	--

**Returns**

float The absolute value of  $a_{fp}$ .

**27.12.2.2 fp\_float32\_add()**

```
float fp_float32_add (
    float  $a_{fp}$ ,
    float  $b_{fp}$  )
```

Add arithmetic operation.

**Parameters**

$a_{fp}$	Number 1 to add.
$b_{fp}$	Number 2 to add.

**Returns**

The sum of  $a_{fp}$  and  $b_{fp}$ .

**27.12.2.3 fp\_float32\_atan()**

```
float fp_float32_atan (
    float  $x$  )
```

Returns the principal value of the arc tangent of  $x$ , expressed in radians.

**Parameters**

$x$	Value whose arc tangent is computed.
-----	--------------------------------------

**Returns**

float Principal arc tangent of  $x$ , in the interval  $[-\pi/2, +\pi/2]$  radians. One radian is equivalent to  $180/\pi$  degrees.

#### 27.12.2.4 `fp_float32_atan2()`

```
float fp_float32_atan2 (
    float y,
    float x )
```

Returns the principal value of the arc tangent of y/x, expressed in radians.

##### Parameters

<i>y</i>	Value representing the proportion of the y-coordinate.
<i>x</i>	Value representing the proportion of the x-coordinate.

##### Returns

float Principal arc tangent of y/x, in the interval  $[-\pi, +\pi]$  radians. One radian is equivalent to  $180/\pi$  degrees. If x is zero, return 0.

#### 27.12.2.5 `fp_float32_cmp()`

```
int fp_float32_cmp (
    float a_fp,
    float b_fp )
```

Compare two float value.

##### Parameters

<i>a</i> ↔ <i>_fp</i>	float value a.
<i>b</i> ↔ <i>_fp</i>	float value b.

##### Returns

int 0 if a==b; 1 if a>b; -1 if a<b;

#### 27.12.2.6 `fp_float32_cos()`

```
float fp_float32_cos (
    float rad )
```

Returns the cosine of an angle of x radians.

##### Parameters

<i>rad</i>	Value representing an angle expressed in radians.
------------	---

**Returns**

float Cosine of x radians.

**27.12.2.7 fp\_float32\_div()**

```
float fp_float32_div (
    float a_fp,
    float b_fp )
```

Float Division.

**Parameters**

$a \leftrightarrow$ _fp	Dividend.
$b \leftrightarrow$ _fp	Divisor.

**Returns**

float Quotient.

**27.12.2.8 fp\_float32\_exp()**

```
float fp_float32_exp (
    float x )
```

Returns the base-e exponential function of x, which is e raised to the power x.

**Parameters**

x	Value of the exponent.
---	------------------------

**Returns**

float Exponential value of x.

**27.12.2.9 fp\_float32\_log()**

```
float fp_float32_log (
    float x )
```

The natural logarithm is the base-e logarithm: the inverse of the natural exponential function (fp\_float32\_exp).

**Parameters**

$x$	Value whose logarithm is calculated.
-----	--------------------------------------

**Returns**

float Natural logarithm of x. If  $x \leq 0$ , return 0.

**27.12.2.10 fp\_float32\_mult()**

```
float fp_float32_mult (
    float a_fp,
    float b_fp )
```

Multiplication.

**Parameters**

$a_{\leftarrow}$ _fp	The number 1 to multiply.
$b_{\leftarrow}$ _fp	The number 2 to multiply.

**Returns**

The product.

Actual fraction in calculation has 24 bits, and hi uses 16 bits at most. What's more, bit15 is the exceeding bit, so it should be moved to bit24.

**27.12.2.11 fp\_float32\_neg()**

```
float fp_float32_neg (
    float a_fp )
```

Calculate the negative value of a\_fp.

**Parameters**

$a_{\leftarrow}$ _fp	The value to calculate.
-------------------------	-------------------------

**Returns**

The negative value of a\_fp.

**27.12.2.12 fp\_float32\_pow()**

```
float fp_float32_pow (
    float x,
    float y )
```

Compute power.

**Parameters**

<i>x</i>	Base value.
<i>y</i>	Exponent value.

**Returns**

float The result of raising base to the power exponent.

**27.12.2.13 fp\_float32\_sin()**

```
float fp_float32_sin (
    float rad )
```

Returns the sine of an angle of x radians.

**Parameters**

<i>rad</i>	Value representing an angle expressed in radians.
------------	---

**Returns**

float Sine of x radians.

**27.12.2.14 fp\_float32\_sqrt()**

```
float fp_float32_sqrt (
    float x )
```

Returns the square root of x.

**Parameters**

<i>x</i>	Value whose square root is computed.
----------	--------------------------------------

**Returns**

float Square root of x.If  $x < 0$ , return 0.



**27.12.2.15 fp\_float32\_sub()**

```
float fp_float32_sub (
    float a_fp,
    float b_fp )
```

Subtraction arithmetic operation.

**Parameters**

$a \leftrightarrow$ _fp	The minuend.
$b \leftrightarrow$ _fp	The subtrahend.

**Returns**

The result.

**27.12.2.16 fp\_float32\_to\_int32()**

```
long fp_float32_to_int32 (
    float a_fp )
```

Convert float to signed long.

**Parameters**

$a \leftrightarrow$ _fp	float to be converted to long.
----------------------------	--------------------------------

**Returns**

long the long integer.

**27.12.2.17 fp\_int32\_to\_float32()**

```
float fp_int32_to_float32 (
    long af )
```

Convert signed long to float.

**Parameters**

af	long value to be converted to float.
----	--------------------------------------

**Returns**

float the float.

**27.12.2.18 fp\_uint32\_to\_float32()**

```
float fp_uint32_to_float32 (
    unsigned long af )
```

Convert unsigned long to float.

**Parameters**

<i>af</i>	unsigned long value to be converted to float.
-----------	---

**Returns**

float the float.

**27.13 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/soft\_fp/soft\_fp.h File Reference**

Soft float library.

```
#include <stdint.h>
```

**Macros**

- #define [abs fp\\_float32\\_abs](#)  
*fp\_float32\_abs*
- #define [sqrt fp\\_float32\\_sqrt](#)  
*fp\_float32\_sqrt*
- #define [cos fp\\_float32\\_cos](#)  
*fp\_float32\_cos*
- #define [sin fp\\_float32\\_sin](#)  
*fp\_float32\_sin*
- #define [atan fp\\_float32\\_atan](#)  
*fp\_float32\_atan*
- #define [atan2 fp\\_float32\\_atan2](#)  
*fp\_float32\_atan2*
- #define [exp fp\\_float32\\_exp](#)  
*fp\_float32\_exp*
- #define [log fp\\_float32\\_log](#)  
*fp\_float32\_log*
- #define [pow fp\\_float32\\_pow](#)  
*fp\_float32\_pow*
- #define [PI](#) ((float)3.1415926)  
*Keyword PI.*

- `#define HALF_PI ((float)(PI/2.0))`  
*Keyword HALF\_PI,  $\pi/2.0$ .*
- `#define EIGHTH_PI ((float)(PI/8.0))`  
*Keyword EIGHTH\_PI,  $\pi/8.0$ .*
- `#define TWO_PI ((float)(PI*2.0))`  
*Keyword TWO\_PI,  $2 * \pi$ .*
- `#define ATAN_EIGHTH_PI ((float)0.37419668)`  
*Keyword ATAN\_EIGHTH\_PI,  $\text{atan}(\pi/8.0)$ .*
- `#define E_SQUARE ((float)7.38905609)`  
*Keyword E\_SQUARE,  $e^2$ .*
- `#define INV_E_SQUARE ((float)0.13533528)`  
*Keyword INV\_E\_SQUARE, the reciprocal of  $e^2$ ,  $1/e^2$ .*
- `#define LN_2 ((float)0.69314718)`  
*Keyword LN\_2,  $\ln(2)$ .*

## Functions

- `float fp_float32_neg (float a_fp)`  
*Calculate the negative value of a\_fp.*
- `float fp_float32_abs (float a_fp)`  
*Calculate absolute value.*
- `float fp_float32_add (float a_fp, float b_fp)`  
*Add arithmetic operation.*
- `float fp_float32_sub (float a_fp, float b_fp)`  
*Subtraction arithmetic operation.*
- `float fp_float32_mult (float a_fp, float b_fp)`  
*Multiplication.*
- `float fp_float32_div (float a_fp, float b_fp)`  
*Float Division.*
- `long fp_float32_to_int32 (float a_fp)`  
*Convert float to signed long.*
- `float fp_int32_to_float32 (long af)`  
*Convert signed long to float.*
- `float fp_uint32_to_float32 (unsigned long af)`  
*Convert unsigned long to float.*
- `int fp_float32_cmp (float a_fp, float b_fp)`  
*Compare two float value.*
- `float fp_float32_sqrt (float a)`  
*Returns the square root of x.*
- `float fp_float32_cos (float rad)`  
*Returns the cosine of an angle of x radians.*
- `float fp_float32_sin (float rad)`  
*Returns the sine of an angle of x radians.*
- `float fp_float32_atan (float x)`  
*Returns the principal value of the arc tangent of x, expressed in radians.*
- `float fp_float32_atan2 (float y, float x)`  
*Returns the principal value of the arc tangent of y/x, expressed in radians.*
- `float fp_float32_exp (float x)`  
*Returns the base-e exponential function of x, which is e raised to the power x.*
- `float fp_float32_log (float x)`  
*The natural logarithm is the base-e logarithm: the inverse of the natural exponential function (fp\_float32\_exp).*
- `float fp_float32_pow (float x, float y)`  
*Compute power.*

### 27.13.1 Detailed Description

Soft float library.

#### Author

Zhirui Dai

#### Date

1 Jun 2018

#### Copyright

2018 Zhirui

### 27.13.2 Function Documentation

#### 27.13.2.1 fp\_float32\_abs()

```
float fp_float32_abs (  
    float a_fp )
```

Calculate absolute value.

#### Parameters

$a_{fp}$	The value to calculate absolute value.
----------	--

#### Returns

float The absolute value of  $a_{fp}$ .

#### 27.13.2.2 fp\_float32\_add()

```
float fp_float32_add (  
    float a_fp,  
    float b_fp )
```

Add arithmetic operation.

#### Parameters

$a_{fp}$	Number 1 to add.
$b_{fp}$	Number 2 to add.

**Returns**

The sum of `a_fp` and `b_fp`.

**27.13.2.3 fp\_float32\_atan()**

```
float fp_float32_atan (
    float x )
```

Returns the principal value of the arc tangent of `x`, expressed in radians.

**Parameters**

<code>x</code>	Value whose arc tangent is computed.
----------------	--------------------------------------

**Returns**

float Principal arc tangent of `x`, in the interval  $[-\pi/2, +\pi/2]$  radians. One radian is equivalent to  $180/\pi$  degrees.

**27.13.2.4 fp\_float32\_atan2()**

```
float fp_float32_atan2 (
    float y,
    float x )
```

Returns the principal value of the arc tangent of `y/x`, expressed in radians.

**Parameters**

<code>y</code>	Value representing the proportion of the y-coordinate.
<code>x</code>	Value representing the proportion of the x-coordinate.

**Returns**

float Principal arc tangent of `y/x`, in the interval  $[-\pi, +\pi]$  radians. One radian is equivalent to  $180/\pi$  degrees. If `x` is zero, return 0.

**27.13.2.5 fp\_float32\_cmp()**

```
int fp_float32_cmp (
    float a_fp,
    float b_fp )
```

Compare two float value.

**Parameters**

$a \leftrightarrow$ _fp	float value a.
$b \leftrightarrow$ _fp	float value b.

**Returns**

int 0 if a==b; 1 if a>b; -1 if a<b;

**27.13.2.6 fp\_float32\_cos()**

```
float fp_float32_cos (
    float rad )
```

Returns the cosine of an angle of x radians.

**Parameters**

<i>rad</i>	Value representing an angle expressed in radians.
------------	---

**Returns**

float Cosine of x radians.

**27.13.2.7 fp\_float32\_div()**

```
float fp_float32_div (
    float a_fp,
    float b_fp )
```

Float Division.

**Parameters**

$a \leftrightarrow$ _fp	Dividend.
$b \leftrightarrow$ _fp	Divisor.

**Returns**

float Quotient.

**27.13.2.8 fp\_float32\_exp()**

```
float fp_float32_exp (
    float x )
```

Returns the base-e exponential function of x, which is e raised to the power x.

**Parameters**

x	Value of the exponent.
---	------------------------

**Returns**

float Exponential value of x.

**27.13.2.9 fp\_float32\_log()**

```
float fp_float32_log (
    float x )
```

The natural logarithm is the base-e logarithm: the inverse of the natural exponential function (fp\_float32\_exp).

**Parameters**

x	Value whose logarithm is calculated.
---	--------------------------------------

**Returns**

float Natural logarithm of x. If x <= 0, return 0.

**27.13.2.10 fp\_float32\_mult()**

```
float fp_float32_mult (
    float a_fp,
    float b_fp )
```

Multiplication.

**Parameters**

$a \leftrightarrow$ _fp	The number 1 to multiply.
$b \leftrightarrow$ _fp	The number 2 to multiply.

**Returns**

The product.

Actual fraction in calculation has 24 bits, and hi uses 16 bits at most. What's more, bit15 is the exceeding bit, so it should be moved to bit24.

**27.13.2.11 fp\_float32\_neg()**

```
float fp_float32_neg (
    float a_fp )
```

Calculate the negative value of a\_fp.

**Parameters**

$a_{fp}$	The value to calculate.
----------	-------------------------

**Returns**

The negative value of a\_fp.

**27.13.2.12 fp\_float32\_pow()**

```
float fp_float32_pow (
    float x,
    float y )
```

Compute power.

**Parameters**

$x$	Base value.
$y$	Exponent value.

**Returns**

float The result of raising base to the power exponent.

**27.13.2.13 fp\_float32\_sin()**

```
float fp_float32_sin (
    float rad )
```

Returns the sine of an angle of x radians.



**Parameters**

<i>rad</i>	Value representing an angle expressed in radians.
------------	---

**Returns**

float Sine of x radians.

**27.13.2.14 fp\_float32\_sqrt()**

```
float fp_float32_sqrt (
    float x )
```

Returns the square root of x.

**Parameters**

<i>x</i>	Value whose square root is computed.
----------	--------------------------------------

**Returns**

float Square root of x. If x < 0, return 0.

**27.13.2.15 fp\_float32\_sub()**

```
float fp_float32_sub (
    float a_fp,
    float b_fp )
```

Subtraction arithmetic operation.

**Parameters**

<i>a<sub>fp</sub></i>	The minuend.
<i>b<sub>fp</sub></i>	The subtrahend.

**Returns**

The result.

**27.13.2.16 fp\_float32\_to\_int32()**

```
long fp_float32_to_int32 (
    float a_fp )
```

Convert float to signed long.

**Parameters**

$a \leftrightarrow$ _fp	float to be converted to long.
----------------------------	--------------------------------

**Returns**

long the long integer.

**27.13.2.17 fp\_int32\_to\_float32()**

```
float fp_int32_to_float32 (
    long af )
```

Convert signed long to float.

**Parameters**

af	long value to be converted to float.
----	--------------------------------------

**Returns**

float the float.

**27.13.2.18 fp\_uint32\_to\_float32()**

```
float fp_uint32_to_float32 (
    unsigned long af )
```

Convert unsigned long to float.

**Parameters**

af	unsigned long value to be converted to float.
----	---

**Returns**

float the float.

## 27.14 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/SPI/SPI.c File Reference

SPI Library for M2.

```
#include "SPI.h"
```

### Functions

- unsigned char [RT\\_SPI\\_Read](#) ()  
*This function returns 1 byte from SPI.*
- unsigned char [RT\\_SPI\\_MasterTransfer](#) (unsigned char c)  
*This function transfers data when M2's SPI works as Master.*
- unsigned char [RT\\_SPI\\_SlaveTransfer](#) (unsigned char c)  
*This function transfers data when M2's SPI works as Slave.*
- void [RT\\_SPI\\_Delay](#) ()  
*Insert 2 cycle delay.*

### 27.14.1 Detailed Description

SPI Library for M2.

#### Author

Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.14.2 Function Documentation

#### 27.14.2.1 RT\_SPI\_Delay()

```
void RT_SPI_Delay ( )
```

Insert 2 cycle delay.

#### Note

This function is used only when the other SPI device is not fast enough.

#### Returns

void

#### 27.14.2.2 RT\_SPI\_MasterTransfer()

```
unsigned char RT_SPI_MasterTransfer (  
    unsigned char c )
```

This function transfers data when M2's SPI works as Master.

**Parameters**

<code>c</code>	1-byte data to send
----------------	---------------------

**Returns**

unsigned char 1-byte data received

**27.14.2.3 RT\_SPI\_Read()**

```
unsigned char RT_SPI_Read ( )
```

This function returns 1 byte from SPI.

**Returns**

unsigned char 1-byte data from SPI

**27.14.2.4 RT\_SPI\_SlaveTransfer()**

```
unsigned char RT_SPI_SlaveTransfer (
    unsigned char c )
```

This function transfers data when M2's SPI works as Slave.

**Parameters**

<code>c</code>	1-byte data to send
----------------	---------------------

**Returns**

unsigned char 1-byte data received

**27.15 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/SPI/SPI.h File Reference**

SPI Library for M2.

```
#include "mcu.h"
#include "UART.h"
```

**Macros**

- `#define MASTER 0x1`  
Keyword *MASTER*.

- #define `SLAVE` 0x0  
*Keyword SLAVE.*
- #define `RT_SPI_ClearState()` `MemoryWrite32(SPI_IRQ_ACK_REG, 0x6)`  
*Clear SPI internal.*
- #define `RT_SPI_Modeset(MorS)`  
*SPI mode: master or slave.*
- #define `RT_SPI_ChipSelect()` `MemoryWrite32(SPI_CTL_REG, 0x6)`  
*SPI CS on.*
- #define `RT_SPI_ChipRelease()` `MemoryWrite32(SPI_CTL_REG, 0x4)`  
*SPI CS off, remain master mode.*
- #define `RT_SPI_Busy()` `MemoryRead32(SPI_BUSY_REG)`  
*Read SPI Tx busy status.*
- #define `RT_SPI_Write_(val)` `MemoryWrite32(SPI_WRITE_REG, val)`  
*Send a byte to SPI tx, start sending.*
- #define `RT_SPI_Write(val)`  
*SPI byte tx procedure: wait when SPI is busy, return after the sending is completed.*
- #define `RT_SPI_DataReady()` `MemoryRead32(SPI_DATA_RDY_REG)`  
*Check if SPI Rx has received a byte.*
- #define `RT_SPI_Read_()` `MemoryRead32(SPI_READ_REG)`  
*Read SPI Rx recived byte.*
- #define `RT_SPI_Off()` `MemoryOr32(SPI_CTL_REG, 0x8)`  
*Turn off SPI.*
- #define `RT_SPI_On()`  
*Turn on SPI.*

## Functions

- unsigned char `RT_SPI_Read ()`  
*This function returns 1 byte from SPI.*
- unsigned char `RT_SPI_MasterTransfer` (unsigned char c)  
*This function transfers data when M2's SPI works as Master.*
- unsigned char `RT_SPI_SlaveTransfer` (unsigned char c)  
*This function transfers data when M2's SPI works as Slave.*
- void `RT_SPI_Delay ()`  
*Insert 2 cycle delay.*

## 27.15.1 Detailed Description

SPI Library for M2.

Author

Zhirui Dai

Date

16 Jun 2018

Copyright

2018 Zhirui

## 27.15.2 Macro Definition Documentation

### 27.15.2.1 RT\_SPI\_Busy

```
#define RT_SPI_Busy( ) MemoryRead32(SPI_BUSY_REG)
```

Read SPI Tx busy status.

#### Returns

int 1=busy 0=idle

### 27.15.2.2 RT\_SPI\_ChipRelease

```
#define RT_SPI_ChipRelease( ) MemoryWrite32(SPI_CTL_REG, 0x4)
```

SPI CS off, remain master mode.

#### Returns

void

### 27.15.2.3 RT\_SPI\_ChipSelect

```
#define RT_SPI_ChipSelect( ) MemoryWrite32(SPI_CTL_REG, 0x6)
```

SPI CS on.

#### Returns

void

### 27.15.2.4 RT\_SPI\_ClearState

```
#define RT_SPI_ClearState( ) MemoryWrite32(SPI_IRQ_ACK_REG, 0x6)
```

Clear SPI internal.

#### Returns

void

### 27.15.2.5 RT\_SPI\_DataReady

```
#define RT_SPI_DataReady( ) MemoryRead32(SPI_DATA_RDY_REG)
```

Check if SPI Rx has received a byte.

#### Returns

int 1=data\_ready 0=no\_data

### 27.15.2.6 RT\_SPI\_Modeset

```
#define RT_SPI_Modeset (
    MorS )
```

#### Value:

```
{
    RT_UART_Off(UART1);
    if (MorS)
        MemoryWrite32(SPI_CTL, 0x4);
    else
        MemoryWrite32(SPI_CTL, 0x0);
    RT_SPI_ClearState();
}
```

SPI mode: master or slave.

#### Parameters

<i>MorS</i>	Optional value:  MASTER, SLAVE
-------------	---

#### Returns

void

### 27.15.2.7 RT\_SPI\_Off

```
#define RT_SPI_Off( ) MemoryOr32(SPI_CTL_REG, 0x8)
```

Turn off SPI.

#### Returns

void

### 27.15.2.8 RT\_SPI\_On

```
#define RT_SPI_On( )
```

#### Value:

```
{
    RT_UART_Off(UART1);
    MemoryAnd32(SPI_CTL_REG, ~0x8);
}
```

Turn on SPI.

#### Returns

void

### 27.15.2.9 RT\_SPI\_Read\_

```
#define RT_SPI_Read_ ( ) MemoryRead32(SPI_READ_REG)
```

Read SPI Rx received byte.

#### Note

Before using this function, [RT\\_SPI\\_DataReady\(\)](#) should be used to check if data is ready. Recommend [RT\\_SPI\\_Read\(\)](#).

#### Returns

int received Rx data

### 27.15.2.10 RT\_SPI\_Write

```
#define RT_SPI_Write(
    val )
```

#### Value:

```
{
    while (RT_SPI_Busy())
    ;
    RT_SPI_Write_(val);
    while (RT_SPI_Busy())
    ;
}
```

SPI byte tx procedure: wait when SPI is busy, return after the sending is completed.



**Parameters**

<i>val</i>	byte to send
------------	--------------

**Returns**

void

**27.15.2.11 RT\_SPI\_Write\_**

```
#define RT_SPI_Write_(  
    val ) MemoryWrite32(SPI_WRITE_REG, val)
```

Send a byte to SPI tx, start sending.

**Parameters**

<i>val</i>	byte to send
------------	--------------

**Returns**

void

**27.15.3 Function Documentation****27.15.3.1 RT\_SPI\_Delay()**

```
void RT_SPI_Delay ( )
```

Insert 2 cycle delay.

**Note**

This function is used only when the other SPI device is not fast enough.

**Returns**

void

**27.15.3.2 RT\_SPI\_MasterTransfer()**

```
unsigned char RT_SPI_MasterTransfer (  
    unsigned char c )
```

This function transfers data when M2's SPI works as Master.

**Parameters**

<code>c</code>	1-byte data to send
----------------	---------------------

**Returns**

unsigned char 1-byte data received

**27.15.3.3 RT\_SPI\_Read()**

```
unsigned char RT_SPI_Read ( )
```

This function returns 1 byte from SPI.

**Returns**

unsigned char 1-byte data from SPI

**27.15.3.4 RT\_SPI\_SlaveTransfer()**

```
unsigned char RT_SPI_SlaveTransfer (
    unsigned char c )
```

This function transfers data when M2's SPI works as Slave.

**Parameters**

<code>c</code>	1-byte data to send
----------------	---------------------

**Returns**

unsigned char 1-byte data received

**27.16 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/std/stdio.c File Reference**

Standard Input Output Library For M2.

```
#include <stdarg.h>
#include <stddef.h>
#include <stdio.h>
#include "stdio_shared.c"
```

### 27.16.1 Detailed Description

Standard Input Output Library For M2.

#### Author

Zhirui

#### Date

2018-05-26

## 27.17 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/std/stdio.h File Reference

Standard Input Outpt Library for M2.

### Macros

- `#define xtoa(value) itoa((value), 16)`  
*Convert an integer to a hex string.*

### Functions

- void `putchar` (char c)  
*Print a character to uart0.*
- void `puts` (const char \*string)  
*Print a string to uart0.*
- char `getchar` ()  
*Get a character from uart0.*
- long `getnum` ()  
*Get a decimal integer from uart0.*
- unsigned long `getHexNum` ()  
*Get a hexadecimal integer from uart0.*
- char \* `itoa` (int value, unsigned int base)  
*Convert an integer to a string.*
- void `printf` (const char \*format,...)  
*Print a formatted string to uart0.*
- void `sprintf` (char \*buf, const char \*format,...)  
*Generate and store a formatted string.*

### 27.17.1 Detailed Description

Standard Input Outpt Library for M2.

#### Author

Zhirui Dai

#### Date

2018-05-26

## 27.17.2 Macro Definition Documentation

### 27.17.2.1 xtoa

```
#define xtoa(  
    value ) itoa((value), 16)
```

Convert an integer to a hex string.

#### Parameters

<i>value</i>	Value to be converted to a string.
--------------	------------------------------------

#### Returns

char\* A pointer to the resulting null-terminated string.

## 27.17.3 Function Documentation

### 27.17.3.1 getchar()

```
char getchar ( )
```

Get a character from uart0.

#### Returns

char The character got from uart0.

### 27.17.3.2 getHexNum()

```
unsigned long getHexNum ( )
```

Get a hexadecimal integer from uart0.

#### Returns

unsigned long An unsigned integer got from uart0.

### 27.17.3.3 getnum()

```
long getnum ( )
```

Get a decimal integer from uart0.

#### Returns

long The integer got from uart0.

### 27.17.3.4 itoa()

```
char* itoa (
    int value,
    unsigned int base )
```

Convert an integer to a string.

#### Parameters

<i>value</i>	Value to be converted to a string.
<i>base</i>	Numerical base used to represent the value as a string, between 2 and 36, where 10 means decimal base, 16 hexadecimal, 8 octal, and 2 binary.

#### Returns

char\* A pointer to the resulting null-terminated string.

### 27.17.3.5 printf()

```
void printf (
    const char * format,
    ... )
```

Print a formatted string to uart0.

#### Parameters

<i>format</i>	pointer to a null-terminated multibyte string specifying how to interpret the data.
<i>...</i>	values to be interpreted.

### 27.17.3.6 putchar()

```
void putchar (
    char c )
```

Print a character to uart0.

**Parameters**

<i>c</i>	Character to be printed.
----------	--------------------------

**27.17.3.7 puts()**

```
void puts (
    const char * string )
```

Print a string to uart0.

**Parameters**

<i>string</i>	String to be printed.
---------------	-----------------------

**27.17.3.8 sprintf()**

```
void sprintf (
    char * buf,
    const char * format,
    ... )
```

Generate and store a formatted string.

**Parameters**

<i>buf</i>	Buffer to store the string.
<i>format</i>	pointer to a null-terminated multibyte string specifying how to interpret the data.
...	values to be interpreted.

**27.18 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/std/stdio\_fp.c File Reference**

Standard Input Output Library For M2.

```
#include <stdarg.h>
#include <soft_fp.h>
#include <stdio_fp.h>
#include "stdio_shared.c"
```

**Functions**

- char \* [ftoa](#) (float a\_fp)  
Convert a float value to a string.

## 27.18.1 Detailed Description

Standard Input Output Library For M2.

## Author

Zhirui

## Date

2018-05-26

## 27.18.2 Function Documentation

## 27.18.2.1 ftoa()

```
char* ftoa (
    float a_fp )
```

Convert a float value to a string.

## Parameters

$a_{fp}$	Value to be converted to a string.
----------	------------------------------------

## Returns

char\* A pointer to the resulting null-terminated string.

## 27.19 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/std/stdio\_fp.h File Reference

Standard Input Outpt Library for M2.

```
#include <stdio.h>
```

## Functions

- char \* [ftoa](#) (float a\_fp)  
*Convert a float value to a string.*



### 27.19.1 Detailed Description

Standard Input Output Library for M2.

#### Author

Zhirui Dai

#### Date

2018-05-26

### 27.19.2 Function Documentation

#### 27.19.2.1 ftoa()

```
char* ftoa (
    float a_fp )
```

Convert a float value to a string.

#### Parameters

<i>a<sub>←</sub></i> <i>_fp</i>	Value to be converted to a string.
------------------------------------	------------------------------------

#### Returns

char\* A pointer to the resulting null-terminated string.

## 27.20 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/std/stdio\_shared.h File Reference

### 27.20.1 Detailed Description

#### Author

your name

#### Date

2018-05-27

## 27.21 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/std/stdlib.c File Reference

Standard Library for M2.

```
#include <stdbool.h>
```

## Functions

- unsigned long [xtoi](#) (const char \*str)  
*Convert a string (hexadecimal) to an integer value.*
- long [atoi](#) (const char \*str)  
*Convert a string (decimal) to an integer value.*

### 27.21.1 Detailed Description

Standard Library for M2.

#### Author

Zhirui Dai

#### Date

27 May 2018

#### Copyright

2018 Zhirui

### 27.21.2 Function Documentation

#### 27.21.2.1 [atoi\(\)](#)

```
long atoi (
    const char * str )
```

Convert a string (decimal) to an integer value.

#### Parameters

<i>str</i>	String (decimal) to be converted to an integer.
------------	---

#### Returns

long A long integer.

#### 27.21.2.2 [xtoi\(\)](#)

```
unsigned long xtoi (
    const char * str )
```

Convert a string (hexadecimal) to an integer value.

**Parameters**

<i>str</i>	String (hexadecimal) to be converted to an integer.
------------	---

**Returns**

unsigned long An unsigned long integer.

**27.22 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/std/stdlib.h File Reference**

Standard Library for M2.

**Functions**

- long [atoi](#) (const char \*str)  
*Convert a string (decimal) to an integer value.*
- unsigned long [xtoi](#) (const char \*str)  
*Convert a string (hexadecimal) to an integer value.*

**27.22.1 Detailed Description**

Standard Library for M2.

**Author**

Zhirui Dai

**Date**

27 May 2018

**Copyright**

2018 Zhirui

**27.22.2 Function Documentation****27.22.2.1 atoi()**

```
long atoi (
    const char * str )
```

Convert a string (decimal) to an integer value.

**Parameters**

<i>str</i>	String (hexadecimal) to be converted to an integer.
------------	---

**Returns**

unsigned long An unsigned long integer.

**Parameters**

<i>str</i>	String (decimal) to be converted to an integer.
------------	---

**Returns**

long A long integer.

**27.22.2.2 xtoi()**

```
unsigned long xtoi (
    const char * str )
```

Convert a string (hexadecimal) to an integer value.

**Parameters**

<i>str</i>	String (decimal) to be converted to an integer.
------------	---

**Returns**

long A long integer.

**Parameters**

<i>str</i>	String (hexadecimal) to be converted to an integer.
------------	---

**Returns**

unsigned long An unsigned long integer.

**27.23 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/std/stdlib\_fp.c File Reference**

Standard Library for M2.

```
#include <stddef.h>
#include <math.h>
#include <stdlib.h>
#include <stdlib_fp.h>
```

## Functions

- float `atof` (char \*str)  
*Convert a string (decimal) to a float value.*

### 27.23.1 Detailed Description

Standard Library for M2.

#### Author

Zhirui Dai

#### Date

27 May 2018

#### Copyright

2018 Zhirui

### 27.23.2 Function Documentation

#### 27.23.2.1 `atof()`

```
float atof (  
    char * str )
```

Convert a string (decimal) to a float value.

#### Parameters

<code>str</code>	String (decimal) to be converted to a float value.
------------------	--

#### Returns

float A float value.

### 27.24 `/Users/daizhirui/Development/CamelStudio_Library/Official/src/std/stdlib_fp.h` File Reference

Standard Library for M2.

```
#include <stdlib.h>
```

## Functions

- float `atof` (char \*str)  
*Convert a string (decimal) to a float value.*

### 27.24.1 Detailed Description

Standard Library for M2.

#### Author

Zhirui Dai

#### Date

27 May 2018

#### Copyright

2018 Zhirui

### 27.24.2 Function Documentation

#### 27.24.2.1 `atof()`

```
float atof (
    char * str )
```

Convert a string (decimal) to a float value.

#### Parameters

<code>str</code>	String (decimal) to be converted to a float value.
------------------	--

#### Returns

float A float value.

### 27.25 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/string/string.c File Reference

String Library for M2.

```
#include "string.h"
```

## Functions

- void \* [memchr](#) (const void \*str, int c, [size\\_t](#) n)  
*Search for character c in the first n bytes in string \*str.*
- int [memcmp](#) (const void \*str1, const void \*str2, [size\\_t](#) n)  
*String compare the first n count in str1 and str2.*
- void \* [memcpy](#) (void \*dest, const void \*src, [size\\_t](#) n)  
*Copy n count from src to dest.*
- void \* [memmove](#) (void \*dest, const void \*src, [size\\_t](#) n)  
*Same as memcpy, except that it take care of the overlapped area case.*
- void \* [memset](#) (void \*str, int c, [size\\_t](#) n)  
*Set n count in \*str with character c.*
- char \* [strcat](#) (char \*dest, const char \*src)  
*Append \*str to the end of \*dest.*
- char \* [strncat](#) (char \*dest, const char \*src, [size\\_t](#) n)  
*Append n count of \*src to \*dest.*
- char \* [strchr](#) (const char \*str, int c)  
*Find the location of the 1st occuring of c in \*str.*
- int [strcmp](#) (const char \*s1, const char \*s2)  
*String compare str1 and str2.*
- int [strncmp](#) (const char \*s1, const char \*s2, [size\\_t](#) n)  
*String compare the first n count in str1 and str2.*
- char \* [strcpy](#) (char \*dest, const char \*src)  
*Copy string src to dest.*
- char \* [strncpy](#) (char \*dest, const char \*src, [size\\_t](#) n)  
*Copy n count from src to dest.*
- [size\\_t](#) [strlen](#) (const char \*str)  
*String length.*
- char \* [strstr](#) (const char \*s1, const char \*s2)  
*Check if \*s2 is part of \*s1.*

### 27.25.1 Detailed Description

String Library for M2.

#### Author

John & Jack

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.25.2 Function Documentation

### 27.25.2.1 memchr()

```
void* memchr (
    const void * str,
    int c,
    size_t n )
```

Search for character c in the first n bytes in string \*str.



**Parameters**

<i>*str</i>	the pointer to the string
<i>c</i>	the matching character
<i>n</i>	the count

**Returns**

void when the 1st c is found, turn the pointer to the location; return NULL if not found

**27.25.2.2 memcmp()**

```
int memcmp (
    const void * str1,
    const void * str2,
    size_t n )
```

String compare the first n count in str1 and str2.

**Parameters**

<i>str1</i>	the pointer to the 1st string
<i>str2</i>	the pointer to the 2nd string
<i>n</i>	the first n count to compare

**Returns**

int return <0 if str1 < str2; return =0 if str1=str2; return >0 if str1>str2

**27.25.2.3 memcpy()**

```
void* memcpy (
    void * dest,
    const void * src,
    size_t n )
```

Copy n count from src to dest.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string
<i>n</i>	the count

**Returns**

\* the pointer to the dest string

**27.25.2.4 memmove()**

```
void* memmove (
    void * dest,
    const void * src,
    size_t n )
```

Same as memcpy, except that it take care of the overlapped area case.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string
<i>n</i>	the count

**Returns**

\* the pointer to the dest string

**27.25.2.5 memset()**

```
void* memset (
    void * str,
    int c,
    size_t n )
```

Set *n* count in *\*str* with character *c*.

**Parameters**

<i>*str</i>	the pointer to the src string
<i>c</i>	the character to be set
<i>n</i>	the count

**Returns**

\* the pointer of the new string

**27.25.2.6 strcat()**

```
char* strcat (
    char * dest,
    const char * src )
```

Append `*str` to the end of `*dest`.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the string to be appended to <i>*dest</i>

**Returns**

\* the pointer to the dest string

**27.25.2.7 strchr()**

```
char* strchr (
    const char * str,
    int c )
```

Find the location of the 1st occuring of c in \*str.

**Parameters**

<i>*str</i>	the pointer to the src string
<i>c</i>	the matching character

**Returns**

\* the pointer to the first match in \*str; NULL if not found

**27.25.2.8 strcmp()**

```
int strcmp (
    const char * s1,
    const char * s2 )
```

String compare str1 and str2.

**Parameters**

<i>s1</i>	the pointer to the 1st string
<i>s2</i>	the pointer to the 2nd string

**Returns**

int return <0 if str1 < str2; reutnr =0 if str1=str2; return >0 if str1>str2

### 27.25.2.9 strcpy()

```
char* strcpy (
    char * dest,
    const char * src )
```

Copy string *src* to *dest*.

#### Parameters

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string

#### Returns

\* the pointer to the dest string

### 27.25.2.10 strlen()

```
size_t strlen (
    const char * str )
```

String length.

#### Parameters

<i>*str</i>	the pointer to the string <i>src</i>
-------------	--------------------------------------

#### Returns

size\_t the string length

### 27.25.2.11 strncat()

```
char* strncat (
    char * dest,
    const char * src,
    size_t n )
```

Append *n* count of *\*src* to *\*dest*.

#### Parameters

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string (use to append)
<i>n</i>	the count

**Returns**

\* the pointer to the dest string

**27.25.2.12 strncmp()**

```
int strncmp (
    const char * s1,
    const char * s2,
    size_t n )
```

String compare the first n count in str1 and str2.

**Parameters**

<i>s1</i>	the pointer to the 1st string
<i>s2</i>	the pointer to the 2nd string
<i>n</i>	the first n count to compare

**Returns**

int return <0 if str1 < str2; reutnr =0 if str1=str2; return >0 if str1>str2

**27.25.2.13 strncpy()**

```
char* strncpy (
    char * dest,
    const char * src,
    size_t n )
```

Copy n count from src to dest.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string
<i>n</i>	the count

**Returns**

\* the pointer to the dest string

**27.25.2.14 strstr()**

```
char* strstr (
    const char * s1,
    const char * s2 )
```

Check if \*s2 is part of \*s1.

#### Parameters

*s1	the pointer to the string to be checked
*s2	the pointer to the substring

#### Returns

\* the pointer to the 1st occuring of s2 in s1; NULL if not found

## 27.26 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/string/string.h File Reference

String Library for M2.

#### Macros

- #define [size\\_t](#) unsigned int  
*Keyword size\_t.*

#### Functions

- void \* [memchr](#) (const void \*str, int c, [size\\_t](#) n)  
*Search for character c in the first n bytes in string \*str.*
- int [memcmp](#) (const void \*str1, const void \*str2, [size\\_t](#) n)  
*String compare the first n count in str1 and str2.*
- void \* [memcpy](#) (void \*dest, const void \*src, [size\\_t](#) n)  
*Copy n count from src to dest.*
- void \* [memmove](#) (void \*dest, const void \*src, [size\\_t](#) n)  
*Same as memcpy, except that it take care of the overlapped area case.*
- void \* [memset](#) (void \*str, int c, [size\\_t](#) n)  
*Set n count in \*str with character c.*
- char \* [strcat](#) (char \*dest, const char \*src)  
*Append \*str to the end of \*dest.*
- char \* [strncat](#) (char \*dest, const char \*src, [size\\_t](#) n)  
*Append n count of \*src to \*dest.*
- char \* [strchr](#) (const char \*str, int c)  
*Find the location of the 1st occuring of c in \*str.*
- int [strcmp](#) (const char \*s1, const char \*s2)  
*String compare str1 and str2.*
- int [strncmp](#) (const char \*s1, const char \*s2, [size\\_t](#) n)  
*String compare the first n count in str1 and str2.*
- char \* [strcpy](#) (char \*dest, const char \*src)  
*Copy string src to dest.*
- char \* [strncpy](#) (char \*dest, const char \*src, [size\\_t](#) n)  
*Copy n count from src to dest.*
- [size\\_t](#) [strlen](#) (const char \*str)  
*String length.*
- char \* [strstr](#) (const char \*s1, const char \*s2)  
*Check if \*s2 is part of \*s1.*

### 27.26.1 Detailed Description

String Library for M2.

#### Author

John & Jack

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.26.2 Function Documentation

#### 27.26.2.1 memchr()

```
void* memchr (
    const void * str,
    int c,
    size_t n )
```

Search for character *c* in the first *n* bytes in string *\*str*.

#### Parameters

<i>*str</i>	the pointer to the string
<i>c</i>	the matching character
<i>n</i>	the count

#### Returns

void when the 1st *c* is found, turn the pointer to the location; return NULL if not found

#### 27.26.2.2 memcmp()

```
int memcmp (
    const void * str1,
    const void * str2,
    size_t n )
```

String compare the first *n* count in *str1* and *str2*.



**Parameters**

<i>str1</i>	the pointer to the 1st string
<i>str2</i>	the pointer to the 2nd string
<i>n</i>	the first n count to compare

**Returns**

int return <0 if *str1* < *str2*; return =0 if *str1*=*str2*; return >0 if *str1*>*str2*

**27.26.2.3 memcpy()**

```
void* memcpy (
    void * dest,
    const void * src,
    size_t n )
```

Copy n count from src to dest.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string
<i>n</i>	the count

**Returns**

\* the pointer to the dest string

**27.26.2.4 memmove()**

```
void* memmove (
    void * dest,
    const void * src,
    size_t n )
```

Same as memcpy, except that it take care of the overlapped area case.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string
<i>n</i>	the count

**Returns**

\* the pointer to the dest string

**27.26.2.5 memset()**

```
void* memset (
    void * str,
    int c,
    size_t n )
```

Set *n* count in *\*str* with character *c*.

**Parameters**

<i>*str</i>	the pointer to the src string
<i>c</i>	the character to be set
<i>n</i>	the count

**Returns**

\* the pointer of the new string

**27.26.2.6 strcat()**

```
char* strcat (
    char * dest,
    const char * src )
```

Append *\*str* to the end of *\*dest*.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the string to be appended to <i>*dest</i>

**Returns**

\* the pointer to the dest string

**27.26.2.7 strchr()**

```
char* strchr (
    const char * str,
    int c )
```

Find the location of the 1st occuring of *c* in *\*str*.

**Parameters**

<i>*str</i>	the pointer to the src string
<i>c</i>	the matching character

**Returns**

\* the pointer to the first match in \*str; NULL if not found

**27.26.2.8 strcmp()**

```
int strcmp (
    const char * s1,
    const char * s2 )
```

String compare str1 and str2.

**Parameters**

<i>s1</i>	the pointer to the 1st string
<i>s2</i>	the pointer to the 2nd string

**Returns**

int return <0 if str1 < str2; return =0 if str1=str2; return >0 if str1>str2

**27.26.2.9 strcpy()**

```
char* strcpy (
    char * dest,
    const char * src )
```

Copy string src to dest.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string

**Returns**

\* the pointer to the dest string

## 27.26.2.10 strlen()

```
size_t strlen (
    const char * str )
```

String length.

## Parameters

<i>*str</i>	the pointer to the string src
-------------	-------------------------------

## Returns

size\_t the string length

## 27.26.2.11 strncat()

```
char* strncat (
    char * dest,
    const char * src,
    size_t n )
```

Append n count of \*src to \*dest.

## Parameters

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string (use to append)
<i>n</i>	the count

## Returns

\* the pointer to the dest string

## 27.26.2.12 strncmp()

```
int strncmp (
    const char * s1,
    const char * s2,
    size_t n )
```

String compare the first n count in str1 and str2.

## Parameters

<i>s1</i>	the pointer to the 1st string
<i>s2</i>	the pointer to the 2nd string
<i>n</i>	the first n count to compare

**Returns**

int return <0 if str1 < str2; reutnr =0 if str1=str2; return >0 if str1>str2

**27.26.2.13 strncpy()**

```
char* strncpy (
    char * dest,
    const char * src,
    size_t n )
```

Copy n count from src to dest.

**Parameters**

<i>*dest</i>	the pointer to the dest string
<i>*src</i>	the pointer to the src string
<i>n</i>	the count

**Returns**

\* the pointer to the dest string

**27.26.2.14 strstr()**

```
char* strstr (
    const char * s1,
    const char * s2 )
```

Check if \*s2 is part of \*s1.

**Parameters**

<i>*s1</i>	the pointer to the string to be checked
<i>*s2</i>	the pointer to the substring

**Returns**

\* the pointer to the 1st occuring of s2 in s1; NULL if not found

**27.27 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/TC0.h File Reference**

Timer0 Library for M2.

```
#include "mcu.h"
```

## Macros

- `#define RT_TC0_Stop() MemoryWrite32(T0_CTL0_REG, 0)`  
*Stop TC0.*
- `#define RT_TC0_ClearIrq() MemoryWrite32(T0_CLRIRQ_REG, 0)`  
*Clear TC0 TC-IRQ and PWM-IRQ.*
- `#define RT_TC0_ClearCnt() MemoryWrite32(T0_CLRCNT_REG, 0)`  
*Clear TC0 Counter value.*
- `#define RT_TC0_ClearAll()`  
*Clear TC0 TC-IRQ, PWM-IRQ and Counter value.*
- `#define RT_TC0_TclrqOn() MemoryOr32(T0_CTL0_REG, 1 << 7)`  
*Turn on TC0 TC-IRQ.*
- `#define RT_TC0_TclrqOff() MemoryAnd32(T0_CTL0_REG, ~(1 << 7))`  
*Turn off TC0 TC-IRQ.*
- `#define RT_TC0_PWMlrqOn() MemoryOr32(T0_CTL0_REG, 1 << 6)`  
*Turn on TC0 PWM-IRQ.*
- `#define RT_TC0_PWMlrqOff() MemoryAnd32(T0_CTL0_REG, ~(1 << 6))`  
*Turn off TC0 PWM-IRQ.*
- `#define RT_TC0_CheckTcFlag() ((MemoryRead32(T0_CTL0_REG) & 0x80000000) >> 31)`  
*Read TC0 TC-flag.*
- `#define RT_TC0_CheckPWMFlag() ((MemoryRead32(T0_CTL0_REG) & 0x40000000) >> 30)`  
*Read TC0 PWM-flag.*
- `#define RT_TC0_TimerOn() MemoryOr32(T0_CTL0_REG, 1 << 1)`  
*Turn on TC0 Timer.*
- `#define RT_TC0_TimerOff() MemoryAnd32(T0_CTL0_REG, ~(1 << 1))`  
*Turn off TC0 Timer.*
- `#define RT_TC0_TimerSet1us(T, irq)`  
*This function sets the timer function of TC0.*
- `#define RT_TC0_SetCounter(n)`  
*This function sets the frequency counter of TC0 The base frequency of the counter is 45Hz.*
- `#define RT_TC0_EcntOn() MemoryOr32(T0_CTL0_REG, 1)`  
*Turn on TC0 Event Counter.*
- `#define RT_TC0_EcntOff() MemoryAnd32(T0_CTL0_REG, ~1)`  
*Turn off TC0 Event Counter.*
- `#define RT_TC0_SetEcnt(n, trigger, irq)`  
*This function sets the ECNT function of TC0.*
- `#define RT_TC0_PWMOn() MemoryOr32(T0_CTL0_REG, 1 << 4)`  
*Turn on TC0 PWM.*
- `#define RT_TC0_PWMOff() MemoryAnd32(T0_CTL0_REG, ~(1 << 4))`  
*Turn off TC0 PWM.*
- `#define RT_TC0_SetPWM(div, ref, irq)`  
*This function sets the PWM function of TC0.*
- `#define RT_TC0_PWMMOn() MemoryOr32(T0_CTL0_REG, 1 << 3)`  
*Turn on TC0 Pulse Width Measurement.*
- `#define RT_TC0_PWMMOff() MemoryAnd32(T0_CTL0_REG, ~(1 << 3))`  
*Turn off TC0 Pulse Width Measurement.*
- `#define RT_TC0_PWMMTriggerMode(mode)`  
*Set TC0 Trigger Mode of Pulse Width Measurement.*
- `#define RT_TC0_SetPWMM(trigger, irq)`  
*This function sets the Pulse width measure for TC0.*
- `#define RT_TC0_ReadCnt() MemoryRead32(T0_READ_REG)`  
*Read TC0 Counter Register Value.*

### 27.27.1 Detailed Description

Timer0 Library for M2.

#### Author

Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.27.2 Macro Definition Documentation

#### 27.27.2.1 RT\_TC0\_CheckPWMFlag

```
#define RT_TC0_CheckPWMFlag( ) ((MemoryRead32(T0_CTL0_REG) & 0x40000000) >> 30)
```

Read TC0 PWM-flag.

#### Returns

int PWM-flag

#### 27.27.2.2 RT\_TC0\_CheckTcFlag

```
#define RT_TC0_CheckTcFlag( ) ((MemoryRead32(T0_CTL0_REG) & 0x80000000) >> 31)
```

Read TC0 TC-flag.

#### Returns

int TC-flag

### 27.27.2.3 RT\_TC0\_ClearAll

```
#define RT_TC0_ClearAll( )
```

#### Value:

```
{  
    RT_TC0_ClearIrq(); \  
    RT_TC0_ClearCnt(); \  
}
```

Clear TC0 TC-IRQ, PWM-IRQ and Counter value.

#### Returns

void

### 27.27.2.4 RT\_TC0\_ClearCnt

```
#define RT_TC0_ClearCnt( ) MemoryWrite32(T0_CLRCNT_REG, 0)
```

Clear TC0 Counter value.

#### Returns

void

### 27.27.2.5 RT\_TC0\_ClearIrq

```
#define RT_TC0_ClearIrq( ) MemoryWrite32(T0_CLRIRQ_REG, 0)
```

Clear TC0 TC-IRQ and PWM-IRQ.

#### Returns

void

### 27.27.2.6 RT\_TC0\_EcntOff

```
#define RT_TC0_EcntOff( ) MemoryAnd32(T0_CTL0_REG, ~1)
```

Turn off TC0 Event Counter.

#### Returns

void



### 27.27.2.7 RT\_TC0\_EcntOn

```
#define RT_TC0_EcntOn( ) MemoryOr32(T0_CTL0_REG, 1)
```

Turn on TC0 Event Counter.

#### Returns

void

### 27.27.2.8 RT\_TC0\_PWMIrqOff

```
#define RT_TC0_PWMIrqOff( ) MemoryAnd32(T0_CTL0_REG, ~(1 << 6))
```

Turn off TC0 PWM-IRQ.

#### Returns

void

### 27.27.2.9 RT\_TC0\_PWMIrqOn

```
#define RT_TC0_PWMIrqOn( ) MemoryOr32(T0_CTL0_REG, 1 << 6)
```

Turn on TC0 PWM-IRQ.

#### Returns

void

### 27.27.2.10 RT\_TC0\_PWMMOff

```
#define RT_TC0_PWMMOff( ) MemoryAnd32(T0_CTL0_REG, ~(1 << 3))
```

Turn off TC0 Pulse Width Measurement.

#### Returns

void

### 27.27.2.11 RT\_TC0\_PWMMOn

```
#define RT_TC0_PWMMOn( ) MemoryOr32(T0_CTL0_REG, 1 << 3)
```

Turn on TC0 Pulse Width Measurement.

#### Returns

void

### 27.27.2.12 RT\_TC0\_PWMMTriggerMode

```
#define RT_TC0_PWMMTriggerMode(  
    mode )
```

#### Value:

```
{  
    MemoryAnd32(T0_CTL0_REG, ~(1 << 2)); \  
    MemoryOr32(T0_CTL0_REG, mode << 2); \  
}
```

Set TC0 Trigger Mode of Pulse Width Measurement.

**Parameters**

<i>mode</i>	Trigger mode, optional values:  <code>RAISE_TRIGGER</code> , <code>FALL_TRIGGER</code>
-------------	--

**Returns**

void

**27.27.2.13 RT\_TC0\_PWMOff**

```
#define RT_TC0_PWMOff( ) MemoryAnd32(T0_CTL0_REG, ~(1 << 4))
```

Turn off TC0 PWM.

**Returns**

void

**27.27.2.14 RT\_TC0\_PWMon**

```
#define RT_TC0_PWMon( ) MemoryOr32(T0_CTL0_REG, 1 << 4)
```

Turn on TC0 PWM.

**Returns**

void

**27.27.2.15 RT\_TC0\_ReadCnt**

```
#define RT_TC0_ReadCnt( ) MemoryRead32(T0_READ_REG)
```

Read TC0 Counter Register Value.

**Returns**

int TC0 Counter Register Value

**27.27.2.16 RT\_TC0\_SetCounter**

```
#define RT_TC0_SetCounter(  
    n )
```

**Value:**

```
{  
    MemoryAnd32(T0_CTL0_REG, ~(1 << 7)); \  
    MemoryWrite32(T0_CLK_REG, n); \  
    MemoryWrite32(T0_REF_REG, 0x0); \  
    MemoryOr32(T0_CTL0_REG, (0x02)); \  
}
```

This function sets the frequency counter of TC0 The base frequency of the counter is 45Hz.

## Parameters

<i>n</i>	times of 45Hz
----------	---------------

## Returns

void

## 27.27.2.17 RT\_TC0\_SetEcnt

```
#define RT_TC0_SetEcnt(
    n,
    trigger,
    irq )
```

## Value:

```
{
    MemoryAnd32(TC0_CTL0_REG, ~(0x1 << 7) + (0x1 << 2)); \
    MemoryOr32(TC0_CTL0_REG, ((trigger << 2) | (irq << 7) | 0x1)); \
    MemoryWrite32(TC0_REF_REG, n); \
    MemoryOr32(SYS_CTL0_REG, irq); \
}
```

This function sets the ECNT function of TC0.

## Parameters

<i>n</i>	the target value to reach
<i>trigger</i>	the trigger mode, RISING or FALLING
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

## Returns

void

## 27.27.2.18 RT\_TC0\_SetPWM

```
#define RT_TC0_SetPWM(
    div,
    ref,
    irq )
```

## Value:

```
{
    MemoryAnd32(TC0_CTL0_REG, ~(0x3 << 6)); \
    MemoryWrite32(TC0_CLK_REG, div); \
    MemoryWrite32(TC0_REF_REG, ref); \
    MemoryOr32(TC0_CTL0_REG, (0x10 | (irq << 6) | (irq << 7))); \
}
```

This function sets the PWM function of TC0.

**Parameters**

<i>div</i>	the clock freq divider
<i>ref</i>	0-255, the clock high length
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

**Returns**

void

**27.27.2.19 RT\_TC0\_SetPWMM**

```
#define RT_TC0_SetPWMM(
    trigger,
    irq )
```

**Value:**

```
{
    MemoryAnd32(T0_CTL0_REG, ~((0x1 << 7) + (0x1 << 2))); \
    MemoryOr32(T0_CTL0_REG, (0x18 | (irq << 7) | (rise << 2))); \
    MemoryOr32(SYS_CTL0_REG, irq); \
}
```

This function sets the Pulse width measure for TC0.

**Parameters**

<i>trigger</i>	Trigger mode, optional values:  <code>RAISE_TRIGGER</code> , <code>FALL_TRIGGER</code>
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

**Returns**

void

**27.27.2.20 RT\_TC0\_Stop**

```
#define RT_TC0_Stop( ) MemoryWrite32(T0_CTL0_REG, 0)
```

Stop TC0.

**Returns**

void

**27.27.2.21 RT\_TC0\_TcIrqOff**

```
#define RT_TC0_TcIrqOff( ) MemoryAnd32(T0_CTL0_REG, ~(1 << 7))
```

Turn off TC0 TC-IRQ.

**Returns**

void

**27.27.2.22 RT\_TC0\_TcIrqOn**

```
#define RT_TC0_TcIrqOn( ) MemoryOr32(T0_CTL0_REG, 1 << 7)
```

Turn on TC0 TC-IRQ.

**Returns**

void

**27.27.2.23 RT\_TC0\_TimerOff**

```
#define RT_TC0_TimerOff( ) MemoryAnd32(T0_CTL0_REG, ~(1 << 1))
```

Turn off TC0 Timer.

**Returns**

void

**27.27.2.24 RT\_TC0\_TimerOn**

```
#define RT_TC0_TimerOn( ) MemoryOr32(T0_CTL0_REG, 1 << 1)
```

Turn on TC0 Timer.

**Returns**

void

**27.27.2.25 RT\_TC0\_TimerSet1us**

```
#define RT_TC0_TimerSet1us(  
    T,  
    irq )
```

**Value:**

```
{  
    MemoryAnd32(T0_CTL0_REG, ~(1 << 7));  
    MemoryWrite32(T0_CLK_REG, T / 81 + 1);  
    MemoryWrite32(T0_REF_REG, 243 * T / (T + 81));  
    MemoryOr32(T0_CTL0_REG, (0x02 | (irq << 7)));  
    MemoryOr32(SYS_CTL0_REG, irq);  
}
```

This function sets the timer function of TC0.

## Parameters

<i>T</i>	the target time to reach
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

## Returns

void

## 27.28 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/TC1.h File Reference

Timer1 Library for M2.

```
#include "mcu.h"
```

## Macros

- `#define RT_TC1_Stop() MemoryWrite32(T1_CTL0_REG, 0)`  
*Stop TC1.*
- `#define RT_TC1_ClearIrq() MemoryWrite32(T1_CLRIRQ_REG, 0)`  
*Clear TC1 TC-IRQ and PWM-IRQ.*
- `#define RT_TC1_ClearCnt() MemoryWrite32(T1_CLRCNT_REG, 0)`  
*Clear TC1 Counter value.*
- `#define RT_TC1_ClearAll()`  
*Clear TC1 TC-IRQ, PWM-IRQ and Counter value.*
- `#define RT_TC1_TcIrqOn() MemoryOr32(T1_CTL0_REG, 1 << 7)`  
*Turn on TC1 TC-IRQ.*
- `#define RT_TC1_TcIrqOff() MemoryAnd32(T1_CTL0_REG, ~(1 << 7))`  
*Turn off TC1 TC-IRQ.*
- `#define RT_TC1_PwmIrqOn() MemoryOr32(T1_CTL0_REG, 1 << 6)`  
*Turn on TC1 PWM-IRQ.*
- `#define RT_TC1_PwmIrqOff() MemoryAnd32(T1_CTL0_REG, ~(1 << 6))`  
*Turn off TC1 PWM-IRQ.*
- `#define RT_TC1_CheckTcFlag() ((MemoryRead32(T1_CTL0_REG) & 0x80000000) >> 31)`  
*Read TC1 TC-flag.*
- `#define RT_TC1_CheckPwmFlag() ((MemoryRead32(T1_CTL0_REG) & 0x40000000) >> 30)`  
*Read TC1 PWM-flag.*
- `#define RT_TC1_TimerOn() MemoryOr32(T1_CTL0_REG, 1 << 1)`  
*Turn on TC1 Timer.*
- `#define RT_TC1_TimerOff() MemoryAnd32(T1_CTL0_REG, ~(1 << 1))`  
*Turn off TC1 Timer.*
- `#define RT_TC1_TimerSet1us(T, irq)`  
*This function sets the timer function of TC1.*
- `#define RT_TC1_SetCounter(n)`  
*This function sets the frequency counter of TC1 The base frequency of the counter is 45Hz.*
- `#define RT_TC1_EcntOn() MemoryOr32(T1_CTL0_REG, 1)`  
*Turn on TC1 Event Counter.*
- `#define RT_TC1_EcntOff() MemoryAnd32(T1_CTL0_REG, ~1)`

- Turn off TC1 Event Counter.*

  - #define `RT_TC1_SetEcnt`(n, trigger, irq)

*This function sets the ECNT function of TC1.*
- #define `RT_TC1_PWMOn`() `MemoryOr32(T1_CTL0_REG, 1 << 4)`

*Turn on TC1 PWM.*
- #define `RT_TC1_PWMOff`() `MemoryAnd32(T1_CTL0_REG, ~(1 << 4))`

*Turn off TC1 PWM.*
- #define `RT_TC1_SetPWM`(div, ref, irq)

*This function sets the PWM function of TC1.*
- #define `RT_TC1_PWMMOn`() `MemoryOr32(T1_CTL0_REG, 1 << 3)`

*Turn on TC1 Pulse Width Measurement.*
- #define `RT_TC1_PWMMOff`() `MemoryAnd32(T1_CTL0_REG, ~(1 << 3))`

*Turn off TC1 Pulse Width Measurement.*
- #define `RT_TC1_PWMMTriggerMode`(mode)

*Set TC1 Trigger Mode of Pulse Width Measurement.*
- #define `RT_TC1_SetPWMM`(trigger, irq)

*This function sets the Pulse width measure for TC1.*
- #define `RT_TC1_ReadCnt`() `MemoryRead32(T1_READ_REG)`

*Read TC1 Counter Register Value.*

### 27.28.1 Detailed Description

Timer1 Library for M2.

#### Author

Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.28.2 Macro Definition Documentation

#### 27.28.2.1 RT\_TC1\_CheckPWMFlag

```
#define RT_TC1_CheckPWMFlag( ) ( (MemoryRead32(T1_CTL0_REG) & 0x40000000) >> 30)
```

Read TC1 PWM-flag.

#### Returns

int PWM-flag



### 27.28.2.2 RT\_TC1\_CheckTcFlag

```
#define RT_TC1_CheckTcFlag( ) ((MemoryRead32(T1_CTL0_REG) & 0x80000000) >> 31)
```

Read TC1 TC-flag.

#### Returns

int TC-flag

### 27.28.2.3 RT\_TC1\_ClearAll

```
#define RT_TC1_ClearAll( )
```

#### Value:

```
{  
    RT_TC1_ClearIrq(); \  
    RT_TC1_ClearCnt(); \  
}
```

Clear TC1 TC-IRQ, PWM-IRQ and Counter value.

#### Returns

void

### 27.28.2.4 RT\_TC1\_ClearCnt

```
#define RT_TC1_ClearCnt( ) MemoryWrite32(T1_CLRCNT_REG, 0)
```

Clear TC1 Counter value.

#### Returns

void

### 27.28.2.5 RT\_TC1\_ClearIrq

```
#define RT_TC1_ClearIrq( ) MemoryWrite32(T1_CLRIRQ_REG, 0)
```

Clear TC1 TC-IRQ and PWM-IRQ.

#### Returns

void

#### 27.28.2.6 RT\_TC1\_EcntOff

```
#define RT_TC1_EcntOff( ) MemoryAnd32(T1_CTL0_REG, ~1)
```

Turn off TC1 Event Counter.

##### Returns

void

#### 27.28.2.7 RT\_TC1\_EcntOn

```
#define RT_TC1_EcntOn( ) MemoryOr32(T1_CTL0_REG, 1)
```

Turn on TC1 Event Counter.

##### Returns

void

#### 27.28.2.8 RT\_TC1\_PWMIrqOff

```
#define RT_TC1_PWMIrqOff( ) MemoryAnd32(T1_CTL0_REG, ~(1 << 6))
```

Turn off TC1 PWM-IRQ.

##### Returns

void

#### 27.28.2.9 RT\_TC1\_PWMIrqOn

```
#define RT_TC1_PWMIrqOn( ) MemoryOr32(T1_CTL0_REG, 1 << 6)
```

Turn on TC1 PWM-IRQ.

##### Returns

void

### 27.28.2.10 RT\_TC1\_PWMMOff

```
#define RT_TC1_PWMMOff( ) MemoryAnd32(T1_CTL0_REG, ~(1 << 3))
```

Turn off TC1 Pulse Width Measurement.

#### Returns

void

### 27.28.2.11 RT\_TC1\_PWMMOn

```
#define RT_TC1_PWMMOn( ) MemoryOr32(T1_CTL0_REG, 1 << 3)
```

Turn on TC1 Pulse Width Measurement.

#### Returns

void

### 27.28.2.12 RT\_TC1\_PWMMTriggerMode

```
#define RT_TC1_PWMMTriggerMode(  
    mode )
```

#### Value:

```
{  
    MemoryAnd32(T1_CTL0_REG, ~(1 << 2)); \  
    MemoryOr32(T1_CTL0_REG, mode << 2); \  
}
```

Set TC1 Trigger Mode of Pulse Width Measurement.

#### Parameters

<i>mode</i>	Trigger mode, optional values:  <code>RAISE_TRIGGER</code> , <code>FALL_TRIGGER</code>
-------------	--

#### Returns

void

**27.28.2.13 RT\_TC1\_PWMOff**

```
#define RT_TC1_PWMOff( ) MemoryAnd32(T1_CTL0_REG, ~(1 << 4))
```

Turn off TC1 PWM.

**Returns**

void

**27.28.2.14 RT\_TC1\_PWMON**

```
#define RT_TC1_PWMON( ) MemoryOr32(T1_CTL0_REG, 1 << 4)
```

Turn on TC1 PWM.

**Returns**

void

**27.28.2.15 RT\_TC1\_ReadCnt**

```
#define RT_TC1_ReadCnt( ) MemoryRead32(T1_READ_REG)
```

Read TC1 Counter Register Value.

**Returns**

int TC0 Counter Register Value

**27.28.2.16 RT\_TC1\_SetCounter**

```
#define RT_TC1_SetCounter(  
    n )
```

**Value:**

```
{  
    MemoryAnd32(T1_CTL0_REG, ~(1 << 7)); \  
    MemoryWrite32(T1_CLK_REG, n); \  
    MemoryWrite32(T1_REF_REG, 0x0); \  
    MemoryOr32(T1_CTL0_REG, (0x02)); \  
}
```

This function sets the frequency counter of TC1 The base frequency of the counter is 45Hz.

**Parameters**

<i>n</i>	times of 45Hz
----------	---------------

**Returns**

void

**27.28.2.17 RT\_TC1\_SetEcnt**

```
#define RT_TC1_SetEcnt(
    n,
    trigger,
    irq )
```

**Value:**

```
{
    MemoryAnd32(T1_CTL0_REG, ~(0x1 << 7) + (0x1 << 2)); \
    MemoryOr32(T1_CTL0_REG, ((trigger << 2) | (irq << 7) | 0x1)); \
    MemoryWrite32(T1_REF_REG, n); \
    MemoryOr32(SYS_CTL0_REG, irq); \
}
```

This function sets the ECNT function of TC1.

**Parameters**

<i>n</i>	the target value to reach
<i>trigger</i>	the trigger mode, RISING or FALLING
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

**Returns**

void

**27.28.2.18 RT\_TC1\_SetPWM**

```
#define RT_TC1_SetPWM(
    div,
    ref,
    irq )
```

**Value:**

```
{
    MemoryAnd32(T1_CTL0_REG, ~(0x3 << 6)); \
    MemoryWrite32(T1_CLK_REG, div); \
    MemoryWrite32(T1_REF_REG, ref); \
    MemoryOr32(T1_CTL0_REG, (0x10 | (irq << 6) | (irq << 7))); \
}
```

This function sets the PWM function of TC1.

**Parameters**

<i>div</i>	the clock freq divider
<i>ref</i>	0-255, the clock high length
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

**Returns**

void

**27.28.2.19 RT\_TC1\_SetPWMM**

```
#define RT_TC1_SetPWMM(  
    trigger,  
    irq )
```

**Value:**

```
{  
    MemoryAnd32(T1_CTL0_REG, ~((0x1 << 7) + (0x1 << 2))); \br/>    MemoryOr32(T1_CTL0_REG, (0x18 | (irq << 7) | (rise << 2))); \br/>    MemoryOr32(SYS_CTL0_REG, irq); \br/>}
```

This function sets the Pulse width measure for TC1.

**Parameters**

<i>trigger</i>	the trigger mode, RISING or FALLING
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

**Returns**

void

**27.28.2.20 RT\_TC1\_Stop**

```
#define RT_TC1_Stop( ) MemoryWrite32(T1_CTL0_REG, 0)
```

Stop TC1.

**Returns**

void

**27.28.2.21 RT\_TC1\_TcIrqOff**

```
#define RT_TC1_TcIrqOff( ) MemoryAnd32(T1_CTL0_REG, ~(1 << 7))
```

Turn off TC1 TC-IRQ.

**Returns**

void

**27.28.2.22 RT\_TC1\_TcIrqOn**

```
#define RT_TC1_TcIrqOn( ) MemoryOr32(T1_CTL0_REG, 1 << 7)
```

Turn on TC1 TC-IRQ.

**Returns**

void

**27.28.2.23 RT\_TC1\_TimerOff**

```
#define RT_TC1_TimerOff( ) MemoryAnd32(T1_CTL0_REG, ~(1 << 1))
```

Turn off TC1 Timer.

**Returns**

void

**27.28.2.24 RT\_TC1\_TimerOn**

```
#define RT_TC1_TimerOn( ) MemoryOr32(T1_CTL0_REG, 1 << 1)
```

Turn on TC1 Timer.

**Returns**

void

**27.28.2.25 RT\_TC1\_TimerSet1us**

```
#define RT_TC1_TimerSet1us(  
    T,  
    irq )
```

**Value:**

```
{  
    MemoryAnd32(T1_CTL0_REG, ~(1 << 7));  
    MemoryWrite32(T1_CLK_REG, T / 81 + 1);  
    MemoryWrite32(T1_REF_REG, 243 * T / (T + 81));  
    MemoryOr32(T1_CTL0_REG, (0x02 | (irq << 7)));  
    MemoryOr32(SYS_CTL0_REG, irq);  
}
```

This function sets the timer function of TC1.

## Parameters

<i>T</i>	the target time to reach
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

## Returns

void

## 27.29 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/TC2.h File Reference

Timer2 Library for M2.

```
#include "mcu.h"
```

## Macros

- `#define RT_TC2_Stop() MemoryWrite32(T2_CTL0_REG, 0)`  
*Stop TC2.*
- `#define RT_TC2_ClearIrq() MemoryWrite32(T2_CLRIRQ_REG, 0)`  
*Clear TC2 TC-IRQ and PWM-IRQ.*
- `#define RT_TC2_ClearCnt() MemoryWrite32(T2_CLRCNT_REG, 0)`  
*Clear TC2 Counter value.*
- `#define RT_TC2_ClearAll()`  
*Clear TC2 TC-IRQ, PWM-IRQ and Counter value.*
- `#define RT_TC2_TclrqOn() MemoryOr32(T2_CTL0_REG, 1 << 7)`  
*Turn on TC2 TC-IRQ.*
- `#define RT_TC2_TclrqOff() MemoryAnd32(T2_CTL0_REG, ~(1 << 7))`  
*Turn off TC2 TC-IRQ.*
- `#define RT_TC2_PWM0lrqOn() MemoryOr32(T2_CTL0_REG, 1 << 6)`  
*Turn on TC2 PWM-IRQ.*
- `#define RT_TC2_PWM0lrqOff() MemoryAnd32(T2_CTL0_REG, ~(1 << 6))`  
*Turn off TC2 PWM-IRQ.*
- `#define RT_TC2_CheckTcFlag() ((MemoryRead32(T2_CTL0_REG) & 0x80000000) >> 31)`  
*Read TC2 TC-flag.*
- `#define RT_TC2_CheckPWM0Flag() ((MemoryRead32(T2_CTL0_REG) & 0x40000000) >> 30)`  
*Read TC2 PWM-flag.*
- `#define RT_TC2_TimerOn() MemoryOr32(T2_CTL0_REG, 1 << 1)`  
*Turn on TC2 Timer.*
- `#define RT_TC2_TimerOff() MemoryAnd32(T2_CTL0_REG, ~(1 << 1))`  
*Turn off TC2 Timer.*
- `#define RT_TC2_TimerSet1us(T, irq)`  
*This function sets the timer function of TC2.*
- `#define RT_TC2_SetCounter(n)`  
*This function sets the frequency counter of TC2 The base frequency of the counter is 45Hz.*
- `#define RT_TC2_EcntOn() MemoryOr32(T2_CTL0_REG, 1)`  
*Turn on TC2 Event Counter.*
- `#define RT_TC2_EcntOff() MemoryAnd32(T2_CTL0_REG, ~1)`



- Turn off TC2 Event Counter.
- #define [RT\\_TC2\\_SetEcnt](#)(n, trigger, irq)
  - Turn on TC2 Event Counter.
- #define [RT\\_TC2\\_PWM0On](#)() [MemoryOr32](#)(T2\_CTL0\_REG, 1 << 4)
  - Turn on TC2 PWM0.
- #define [RT\\_TC2\\_PWM0Off](#)() [MemoryAnd32](#)(T2\_CTL0\_REG, ~(1 << 4))
  - Turn off TC2 PWM0.
- #define [RT\\_TC2\\_PWM1to3On](#)() [MemoryOr32](#)(T2\_CTL0\_REG, 1 << 5)
  - Turn on TC2 PWM1-3.
- #define [RT\\_TC2\\_PWM1to3Off](#)() [MemoryAnd32](#)(T2\_CTL0\_REG, ~(1 << 5))
  - Turn off TC2 PWM1-3.
- #define [RT\\_TC2\\_SetAllPWM](#)(div, duty0, duty1, duty2, duty3, phase0, phase1, phase2, phase3, pwm0, pwm13)
  - Turn on TC2 PWM.
- #define [RT\\_TC2\\_PWMMOn](#)() [MemoryOr32](#)(T2\_CTL0\_REG, 1 << 3)
  - Turn on TC2 Pulse Width Measurement.
- #define [RT\\_TC2\\_PWMMOff](#)() [MemoryAnd32](#)(T2\_CTL0\_REG, ~(1 << 3))
  - Turn off TC2 Pulse Width Measurement.
- #define [RT\\_TC2\\_PWMMTriggerMode](#)(mode)
  - Set TC2 Trigger Mode of Pulse Width Measurement.
- #define [RT\\_TC2\\_SetPWMM](#)(trigger, irq)
  - Turn on TC2 Pulse Width Measurement.
- #define [RT\\_TC2\\_ReadCnt](#)() [MemoryRead32](#)(T2\_READ\_REG)
  - Read TC2 Counter Register Value.

### 27.29.1 Detailed Description

Timer2 Library for M2.

#### Author

Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

### 27.29.2 Macro Definition Documentation

### 27.29.2.1 RT\_TC2\_CheckPWM0Flag

```
#define RT_TC2_CheckPWM0Flag( ) ((MemoryRead32(T2_CTL0_REG) & 0x40000000) >> 30)
```

Read TC2 PWM-flag.

**Returns**

int PWM-flag

### 27.29.2.2 RT\_TC2\_CheckTcFlag

```
#define RT_TC2_CheckTcFlag( ) ((MemoryRead32(T2_CTL0_REG) & 0x80000000) >> 31)
```

Read TC2 TC-flag.

**Returns**

int TC-flag

### 27.29.2.3 RT\_TC2\_ClearAll

```
#define RT_TC2_ClearAll( )
```

**Value:**

```
{  
    MemoryWrite32(T2_CLRIRQ_REG, 0); \  
    MemoryWrite32(T2_CLRCNT_REG, 0); \  
}
```

Clear TC2 TC-IRQ, PWM-IRQ and Counter value.

**Returns**

void

### 27.29.2.4 RT\_TC2\_ClearCnt

```
#define RT_TC2_ClearCnt( ) MemoryWrite32(T2_CLRCNT_REG, 0)
```

Clear TC2 Counter value.

**Returns**

void

#### 27.29.2.5 RT\_TC2\_ClearIrq

```
#define RT_TC2_ClearIrq( ) MemoryWrite32(T2_CLRIRQ_REG, 0)
```

Clear TC2 TC-IRQ and PWM-IRQ.

##### Returns

void

#### 27.29.2.6 RT\_TC2\_EcntOff

```
#define RT_TC2_EcntOff( ) MemoryAnd32(T2_CTL0_REG, ~1)
```

Turn off TC2 Event Counter.

##### Returns

void

#### 27.29.2.7 RT\_TC2\_EcntOn

```
#define RT_TC2_EcntOn( ) MemoryOr32(T2_CTL0_REG, 1)
```

Turn on TC2 Event Counter.

##### Returns

void

#### 27.29.2.8 RT\_TC2\_PWM0IrqOff

```
#define RT_TC2_PWM0IrqOff( ) MemoryAnd32(T2_CTL0_REG, ~(1 << 6))
```

Turn off TC2 PWM-IRQ.

##### Returns

void

### 27.29.2.9 RT\_TC2\_PWM0IrqOn

```
#define RT_TC2_PWM0IrqOn( ) MemoryOr32(T2_CTL0_REG, 1 << 6)
```

Turn on TC2 PWM-IRQ.

#### Returns

void

### 27.29.2.10 RT\_TC2\_PWM0Off

```
#define RT_TC2_PWM0Off( ) MemoryAnd32(T2_CTL0_REG, ~(1 << 4))
```

Turn off TC2 PWM0.

#### Returns

void

### 27.29.2.11 RT\_TC2\_PWM0On

```
#define RT_TC2_PWM0On( ) MemoryOr32(T2_CTL0_REG, 1 << 4)
```

Turn on TC2 PWM0.

#### Returns

void

### 27.29.2.12 RT\_TC2\_PWM1to3Off

```
#define RT_TC2_PWM1to3Off( ) MemoryAnd32(T2_CTL0_REG, ~(1 << 5))
```

Turn off TC2 PWM1-3.

#### Returns

void

**27.29.2.13 RT\_TC2\_PWM1to3On**

```
#define RT_TC2_PWM1to3On( ) MemoryOr32(T2_CTL0_REG, 1 << 5)
```

Turn on TC2 PWM1-3.

**Returns**

void

**27.29.2.14 RT\_TC2\_PWMMOff**

```
#define RT_TC2_PWMMOff( ) MemoryAnd32(T2_CTL0_REG, ~(1 << 3))
```

Turn off TC2 Pulse Width Measurement.

**Returns**

void

**27.29.2.15 RT\_TC2\_PWMMOn**

```
#define RT_TC2_PWMMOn( ) MemoryOr32(T2_CTL0_REG, 1 << 3)
```

Turn on TC2 Pulse Width Measurement.

**Returns**

void

**27.29.2.16 RT\_TC2\_PWMMTriggerMode**

```
#define RT_TC2_PWMMTriggerMode(  
    mode )
```

**Value:**

```
{  
    MemoryAnd32(T2_CTL0_REG, ~(1 << 2)); \  
    MemoryOr32(T2_CTL0_REG, mode << 2); \  
}
```

Set TC2 Trigger Mode of Pulse Width Measurement.

## Parameters

<i>mode</i>	Trigger mode, optional values:  <code>RAISE_TRIGGER</code> , <code>FALL_TRIGGER</code>
-------------	--

## Returns

void

## 27.29.2.17 RT\_TC2\_ReadCnt

```
#define RT_TC2_ReadCnt( ) MemoryRead32(T2_READ_REG)
```

Read TC2 Counter Register Value.

## Returns

int TC0 Counter Register Value

## 27.29.2.18 RT\_TC2\_SetAllPWM

```
#define RT_TC2_SetAllPWM(  
    div,  
    duty0,  
    duty1,  
    duty2,  
    duty3,  
    phase0,  
    phase1,  
    phase2,  
    phase3,  
    pwm0,  
    pwm13 )
```

## Value:

```
{  
    MemoryAnd32(T2_CTL0_REG, ~(0x3 << 4));  
    MemoryOr32(T2_CTL0_REG, ((pwm0 << 4) | (pwm13 << 5)));  
    MemoryWrite32(T2_CLK_REG, div);  
    MemoryWrite32(T2_REF_REG, (duty0 + (duty1 << 8) + (duty2 << 16) + (duty3 << 24)));  
    MemoryWrite32(T2_PHASE_REG, (phase0 + (phase1 << 8) + (phase2 << 16) + (phase3 << 24)));  
};
```

This function sets the PWM function of TC2.

**Parameters**

<i>div</i>	the clock freq divider
<i>duty0</i>	2-255, the duty of pwm0
<i>duty1</i>	2-255, the duty of pwm1
<i>duty2</i>	2-255, the duty of pwm2
<i>duty3</i>	2-255, the duty of pwm3
<i>phase0</i>	0-255, pwm0's relative phase among 4 pwm
<i>phase1</i>	0-255, pwm1's relative phase among 4 pwm
<i>phase2</i>	0-255, pwm2's relative phase among 4 pwm
<i>phase3</i>	0-255, pwm3's relative phase among 4 pwm
<i>pwm0</i>	the switch of pwm0, ON or OFF
<i>pwm13</i>	the switch of pwm13, ON or OFF

**Returns**

void

**27.29.2.19 RT\_TC2\_SetCounter**

```
#define RT_TC2_SetCounter(
    n )
```

**Value:**

```
{
    MemoryAnd32(T2_CTL0_REG, ~(1 << 7)); \
    MemoryWrite32(T2_CLK_REG, n); \
    MemoryWrite32(T2_REF_REG, 0x0); \
    MemoryOr32(T2_CTL0_REG, (0x02)); \
}
```

This function sets the frequency counter of TC2 The base frequency of the counter is 45Hz.

**Parameters**

<i>n</i>	times of 45Hz
----------	---------------

**Returns**

void

**27.29.2.20 RT\_TC2\_SetEcnt**

```
#define RT_TC2_SetEcnt(
    n,
    trigger,
    irq )
```

**Value:**

```

{
    MemoryAnd32(T2_CTL0_REG, ~(0x1 << 7) + (0x1 << 2)); \
    MemoryOr32(T2_CTL0_REG, ((trigger << 2) | (irq << 7) | 0x1)); \
    MemoryWrite32(T2_REF_REG, n); \
    MemoryOr32(SYS_CTL0_REG, irq); \
}

```

This function sets the ECNT function of TC2.

#### Parameters

<i>n</i>	the target value to reach
<i>trigger</i>	the trigger mode, RISING or FALLING
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

#### Returns

void

#### 27.29.2.21 RT\_TC2\_SetPWMM

```

#define RT_TC2_SetPWMM(
    trigger,
    irq )

```

#### Value:

```

{
    MemoryAnd32(T2_CTL0_REG, ~(0x1 << 7) + (0x1 << 2)); \
    MemoryOr32(T2_CTL0_REG, (0x18 | (irq << 7) | (rise << 2))); \
    MemoryOr32(SYS_CTL0_REG, irq); \
}

```

This function sets the Pulse width measure for TC2.

#### Parameters

<i>trigger</i>	the trigger mode, RISING or FALLING
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

#### Returns

void

#### 27.29.2.22 RT\_TC2\_Stop

```

#define RT_TC2_Stop( ) MemoryWrite32(T2_CTL0_REG, 0)

```

Stop TC2.



**Returns**

void

**27.29.2.23 RT\_TC2\_TcIrqOff**

```
#define RT_TC2_TcIrqOff( ) MemoryAnd32(T2_CTL0_REG, ~(1 << 7))
```

Turn off TC2 TC-IRQ.

**Returns**

void

**27.29.2.24 RT\_TC2\_TcIrqOn**

```
#define RT_TC2_TcIrqOn( ) MemoryOr32(T2_CTL0_REG, 1 << 7)
```

Turn on TC2 TC-IRQ.

**Returns**

void

**27.29.2.25 RT\_TC2\_TimerOff**

```
#define RT_TC2_TimerOff( ) MemoryAnd32(T2_CTL0_REG, ~(1 << 1))
```

Turn off TC2 Timer.

**Returns**

void

**27.29.2.26 RT\_TC2\_TimerOn**

```
#define RT_TC2_TimerOn( ) MemoryOr32(T2_CTL0_REG, 1 << 1)
```

Turn on TC2 Timer.

**Returns**

void

**27.29.2.27 RT\_TC2\_TimerSet1us**

```
#define RT_TC2_TimerSet1us(  
    T,  
    irq )
```

**Value:**

```
{  
    MemoryAnd32(T2_CTL0_REG, ~(1 << 7));  
    MemoryWrite32(T2_CLK_REG, T / 81 + 1);  
    MemoryWrite32(T2_REF_REG, 243 * T / (T + 81));  
    MemoryOr32(T2_CTL0_REG, (0x02 | (irq << 7)));  
    MemoryOr32(SYS_CTL0_REG, irq);  
}
```

This function sets the timer function of TC2.

## Parameters

<i>T</i>	the target time to reach
<i>irq</i>	when ON, the interrupt is enabled; when OFF, disabled

## Returns

void

## 27.30 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/TC4.h File Reference

Timer4 Library for M2.

```
#include "mcu.h"
```

## Macros

- `#define RT_TC4_AIIPWM_On() MemoryWrite32(T4_CTL0_REG, 3)`  
*Turn on TC4 PWM0 and PWM1.*
- `#define RT_TC4_AIIPWM_Off() MemoryWrite32(T4_CTL0_REG, 0)`  
*Turn off TC4 PWM0 and PWM1.*
- `#define RT_TC4_PWM_On(n) MemoryOr32(T4_CLK0_REG, 1 << n)`  
*Turn on TC4 PWM based on selection.*
- `#define RT_TC4_PWM_Off(n) MemoryAnd32(T4_CTL0_REG, ~(1 << n))`  
*Turn off TC4 PWM based on selection.*
- `#define RT_TC4_SetAIIPWM(pwm0_en, div0, ref0, pwm1_en, div1, ref1)`  
*This function sets all the PWM function of TC4.*
- `#define RT_TC4_SetPWM(n, pwm_en, div, ref)`  
*This function sets single pwm of TC4.*

## 27.30.1 Detailed Description

Timer4 Library for M2.

## Author

Zhirui Dai

## Date

16 Jun 2018

## Copyright

2018 Zhirui

## 27.30.2 Macro Definition Documentation

### 27.30.2.1 RT\_TC4\_AIIPWM\_Off

```
#define RT_TC4_AIIPWM_Off( ) MemoryWrite32(T4_CTL0_REG, 0)
```

Turn off TC4 PWM0 and PWM1.

#### Returns

void

### 27.30.2.2 RT\_TC4\_AIIPWM\_On

```
#define RT_TC4_AIIPWM_On( ) MemoryWrite32(T4_CTL0_REG, 3)
```

Turn on TC4 PWM0 and PWM1.

#### Returns

void

### 27.30.2.3 RT\_TC4\_PWM\_Off

```
#define RT_TC4_PWM_Off(  
    n ) MemoryAnd32(T4_CTL0_REG, ~(1 << n))
```

Turn off TC4 PWM based on selection.

#### Parameters

<i>n</i>	0=PWM0 1=PWM1
----------	---------------

#### Returns

void

### 27.30.2.4 RT\_TC4\_PWM\_On

```
#define RT_TC4_PWM_On(  
    n ) MemoryOr32(T4_CLK0_REG, 1 << n)
```

Turn on TC4 PWM based on selection.

**Parameters**

<i>n</i>	0=PWM0 1=PWM1
----------	---------------

**Returns**

void

**27.30.2.5 RT\_TC4\_SetAllPWM**

```
#define RT_TC4_SetAllPWM(
    pwm0_en,
    div0,
    ref0,
    pwm1_en,
    div1,
    ref1 )
```

**Value:**

```
{
    MemoryWrite32(T4_CLK0_REG, div0);
    MemoryWrite32(T4_REF0_REG, ref0);
    MemoryWrite32(T4_CLK1_REG, div1);
    MemoryWrite32(T4_REF1_REG, ref1);
    MemoryAnd32(T4_CTL0_REG, ~(0x3));
    MemoryOr32(T4_CTL0_REG, (pwm0_en | pwm1_en << 1));
}
```

This function sets all the PWM function of TC4.

**Parameters**

<i>pwm0_en</i>	the switch of TC4-pwm0, ON or OFF
<i>div0</i>	the clock freq divider of TC4-pwm0
<i>ref0</i>	15-0, the duty of TC4-pwm0
<i>pwm1_en</i>	the switch of TC4-pwm1, On or OFF
<i>div1</i>	the clock freq divider of TC4-pwm1
<i>ref1</i>	15-0, the duty of TC4-pwm1

**Returns**

void

**27.30.2.6 RT\_TC4\_SetPWM**

```
#define RT_TC4_SetPWM(
    n,
```

```

    pwm_en,
    div,
    ref )

```

**Value:**

```

{
    MemoryWrite32(T4_CLK0_REG + 2 * n, div); \
    MemoryWrite32(T4_REF0_REG + 2 * n, ref); \
    MemoryAnd32(T4_CTL0_REG, ~(1 << n)); \
    MemoryOr32(T4_CTL0_REG, pwm_en << n); \
}

```

This function sets single pwm of TC4.

**Parameters**

<i>n</i>	the number of pwm, 0 or 1
<i>pwm_en</i>	the switch of the pwm, optional value:  ON, OFF
<i>div</i>	the clock freq divider of pwm
<i>ref</i>	the duty of pwm

**Returns**

void

**27.31 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/time/time.c File Reference**

Real Time Module Library for M2.

```

#include "mcu.h"
#include "time.h"

```

**Functions**

- void [RT\\_RTC\\_GetTime](#) (unsigned char \*d\_year, unsigned char \*d\_mon, unsigned char \*d\_day, unsigned char \*d\_hour, unsigned char \*d\_min, unsigned char \*d\_sec)  
*This function returns the RTC time.*
- void [RT\\_DelayMilliseconds](#) (unsigned long ms)  
*This function delays specific time.*

**27.31.1 Detailed Description**

Real Time Module Library for M2.

**Author**

Zhirui Dai

**Date**

16 Jun 2018

**Copyright**

2018 Zhirui

**27.31.2 Function Documentation****27.31.2.1 RT\_DelayMilliseconds()**

```
void RT_DelayMilliseconds (
    unsigned long ms )
```

This function delays specific time.

**Parameters**

<i>ms</i>	time to delay, the unit is ms
-----------	-------------------------------

**27.31.2.2 RT\_RTC\_GetTime()**

```
void RT_RTC_GetTime (
    unsigned char * d_year,
    unsigned char * d_mon,
    unsigned char * d_day,
    unsigned char * d_hour,
    unsigned char * d_min,
    unsigned char * d_sec )
```

This function returns the RTC time.

**Parameters**

<i>d_year</i>	year
<i>d_mon</i>	month
<i>d_day</i>	day
<i>d_hour</i>	hour
<i>d_min</i>	minute
<i>d_sec</i>	second

## 27.32 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/time/time.h File Reference

Real Time Module Library for M2.

```
#include "mcu.h"
```

### Macros

- `#define RT_RTC_On() MemoryOr32(RTC_CTL_REG, 1)`  
*Turn on RTC.*
- `#define RT_RTC_Off() MemoryAnd32(RTC_CTL_REG, ~1)`  
*Turn off RTC.*
- `#define RTC_12HOUR 0x1`  
*Keyword RTC\_12HOUR.*
- `#define RTC_24HOUR 0x3`  
*Keyword RTC\_24HOUR.*
- `#define RT_RTC_SetTimeFormat(format)`  
*Set the time format of RTC.*
- `#define RT_RTC_SetTime(year, month, day, hour, min, sec) MemoryWrite32(RTC_TIME_REG, year << 26 | mon << 22 | day << 17 | hour << 12 | min << 6 | sec);`  
*This function sets the time of RTC.*
- `#define RT_RTC_Read32() MemoryRead32(RTC_TIME_REG)`  
*This function returns the RTC time raw value.*

### Functions

- `void RT_RTC_GetTime (unsigned char *d_year, unsigned char *d_mon, unsigned char *d_day, unsigned char *d_hour, unsigned char *d_min, unsigned char *d_sec)`  
*This function returns the RTC time.*
- `void RT_DelayMilliseconds (unsigned long ms)`  
*This function delays specific time.*

#### 27.32.1 Detailed Description

Real Time Module Library for M2.

#### Author

Zhirui Dai

#### Date

16 Jun 2018

#### Copyright

2018 Zhirui

## 27.32.2 Macro Definition Documentation

### 27.32.2.1 RT\_RTC\_Off

```
#define RT_RTC_Off( ) MemoryAnd32(RTC_CTL_REG, ~1)
```

Turn off RTC.

#### Returns

void

### 27.32.2.2 RT\_RTC\_On

```
#define RT_RTC_On( ) MemoryOr32(RTC_CTL_REG, 1)
```

Turn on RTC.

#### Returns

void

### 27.32.2.3 RT\_RTC\_Read32

```
#define RT_RTC_Read32( ) MemoryRead32(RTC_TIME_REG)
```

This function returns the RTC time raw value.

#### Returns

void

### 27.32.2.4 RT\_RTC\_SetTime

```
#define RT_RTC_SetTime(  
    year,  
    month,  
    day,  
    hour,  
    min,  
    sec ) MemoryWrite32(RTC_TIME_REG, year << 26 | mon << 22 | day << 17 | hour <<  
12 | min << 6 | sec);
```

This function sets the time of RTC.



**Parameters**

<i>year</i>	year value
<i>month</i>	month value
<i>day</i>	day value
<i>hour</i>	hour value
<i>min</i>	minute value
<i>sec</i>	second value

**Returns**

void

**27.32.2.5 RT\_RTC\_SetTimeFormat**

```
#define RT_RTC_SetTimeFormat(  
    format )
```

**Value:**

```
{  
    MemoryWrite32(RTC_CTL_REG, format); \  
    MemoryWrite32(RTC_TIME_REG, 0x00420000); \  
}
```

Set the time format of RTC.

**Parameters**

<i>format</i>	The time format, optional value:  <code>RTC_12HOUR</code> , <code>RTC_24HOUR</code>  .
---------------	--

**Returns**

void

**27.32.2.6 RTC\_12HOUR**

```
#define RTC_12HOUR 0x1
```

Keyword `RTC_12HOUR`.**Note**Time format used in `RT_RTC_SetTimeFormat`

### 27.32.2.7 RTC\_24HOUR

```
#define RTC_24HOUR 0x3
```

Keyword RTC\_24HOUR.

#### Note

Time format used in RT\_RTC\_SetTimeFormat

## 27.32.3 Function Documentation

### 27.32.3.1 RT\_DelayMilliseconds()

```
void RT_DelayMilliseconds (
    unsigned long ms )
```

This function delays specific time.

#### Parameters

<i>ms</i>	time to delay, the unit is ms
-----------	-------------------------------

### 27.32.3.2 RT\_RTC\_GetTime()

```
void RT_RTC_GetTime (
    unsigned char * d_year,
    unsigned char * d_mon,
    unsigned char * d_day,
    unsigned char * d_hour,
    unsigned char * d_min,
    unsigned char * d_sec )
```

This function returns the RTC time.

#### Parameters

<i>d_year</i>	year
<i>d_mon</i>	month
<i>d_day</i>	day
<i>d_hour</i>	hour
<i>d_min</i>	minute
<i>d_sec</i>	second

## 27.33 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/UART/UART.c File Reference

UART Library for M2.

```
#include "UART.h"
```

### Functions

- void [RT\\_UART\\_putchar](#) (uint32\_t port, unsigned char c)  
*Send a character via Uart, to print via UART0, [\\_putchar\(\)](#) is recommended.*
- void [RT\\_UART\\_puts](#) (uint32\_t port, unsigned char \*string)  
*Send a string via Uart, to print via UART0, [puts\(\)](#) is recommended.*
- unsigned char [RT\\_UART\\_getchar](#) (uint32\_t port)  
*Get a character via Uart, to get a byte via UART0, [getchar\(\)](#) is recommended.*
- void [RT\\_UART\\_LinMaster](#) (uint32\_t port, char pattern)  
*Send LIN break mode.*

### 27.33.1 Detailed Description

UART Library for M2.

#### Author

Zhirui Dai

#### Date

1 Nov 2017

#### Copyright

2018 Zhirui

### 27.33.2 Function Documentation

#### 27.33.2.1 RT\_UART\_getchar()

```
unsigned char RT_UART_getchar (  
    uint32_t port )
```

Get a character via Uart, to get a byte via UART0, [getchar\(\)](#) is recommended.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0</code> , <code>UART1</code>
-------------	---

**Returns**

char 1-byte data from Uart

**27.33.2.2 RT\_UART\_LinMaster()**

```
void RT_UART_LinMaster (
    uint32_t port,
    char pattern )
```

Send LIN break mode.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0</code> , <code>UART1</code>
<i>pattern</i>	The length of the break, optional value:  <code>NORMAL_BREAK</code> , <code>EXTREME_BREAK</code>

**Returns**

void

**27.33.2.3 RT\_UART\_putchar()**

```
void RT_UART_putchar (
    uint32_t port,
    unsigned char c )
```

Send a character via Uart, to print via UART0, `putchar()` is recommended.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0</code> , <code>UART1</code>
<i>c</i>	character

**Returns**

void

**27.33.2.4 RT\_UART\_puts()**

```
void RT_UART_puts (
    uint32_t port,
    unsigned char * string )
```

Send a string via Uart, to print via UART0, [puts\(\)](#) is recommended.

**Parameters**

<i>port</i>	Port to use, optional value:  <a href="#">UART0</a> , <a href="#">UART1</a>
<i>string</i>	string

**Returns**

void

**27.34 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/UART/UART.h File Reference**

UART Library for M2.

```
#include "mcu.h"
```

**Macros**

- `#define UART0 0x0`  
*Keyword [UART0](#).*
- `#define UART1 0x1`  
*Keyword [UART0](#).*
- `#define EXTREME\_BREAK 0x1`  
*Keyword [EXTREME\\_BREAK](#).*
- `#define NORMAL\_BREAK 0x0`  
*Keyword [NORMAL\\_BREAK](#).*
- `#define RT\_UART\_Off(port) MemoryOr32(UART_CTL_REG(port),0x10)`  
*Turn off Uart.*
- `#define RT\_UART\_On(port) {MemoryAnd32(UART_CTL_REG(port),~0x10);}`  
*Turn on Uart.*
- `#define BAUDRATE\_9600 (0x0<<12)`  
*Keyword [BAUDRATE\\_9600](#).*
- `#define BAUDRATE\_19200 (0x1<<12)`

- Keyword BAUDRATE\_19200.*
- #define `BAUDRATE_38400` (0x3<<12)
- Keyword BAUDRATE\_38400.*
- #define `RT_UART_SetBaudrate`(mode) `RT_MCU_SetSystemClock`(mode)
- Set the baudrate of UART.*
- #define `RT_UART_Busy`(port) `MemoryRead32`(UART\_BUSY\_REG(port))
- Check if Uart Tx is busy.*
- #define `RT_UART_Write`(port, val) `MemoryWrite32`(UART\_WRITE\_REG(port),val)
- Sent data via Uart Tx.*
- #define `RT_UART_DataReady`(port) `MemoryRead32`(UART\_DATA\_RDY\_REG(port))
- Check if Uart Rx has received data.*
- #define `RT_UART_Read`(port) `MemoryRead32`(UART\_READ\_REG(port))
- Read Uart Rx data.*
- #define `RT_UART_IrqOn`(port) `MemoryOr32`(UART\_CTL\_REG(port),1)
- Turn on Uart Irq enable.*
- #define `RT_UART_IrqOff`(port) `MemoryAnd32`(UART\_CTL\_REG(port),~1)
- Turn off Uart Irq (interrupt)*
- #define `RT_UART_CompareOn`(port) `MemoryOr32`(UART\_CTL\_REG(port),0x2)
- Set UART compare irq on, irq flag will be raised when the port receives a byte the same as the compare value.*
- #define `RT_UART_CompareOff`(port) `MemoryAnd32`(UART\_CTL\_REG(port),~0x2)
- This function sets UART compare irq off.*
- #define `RT_UART_SetCompare`(port, val) `MemoryOr32`(UART\_CTL\_REG(port),val<<8)
- Set UART port's compare value.*
- #define `RT_UART_ClearIrq`(port) `MemoryWrite32`(UART\_IRQ\_ACK\_REG(port),0x0)
- Clear Uart Irq (interrupt)*
- #define `RT_UART_RaiseIrq`(port) `MemoryOr32`(UART\_CTL\_REG(port),0x1)
- Turn on Uart Irq enable.*
- #define `RT_UART_CheckIrq`(port) `MemoryBitAt`(UART\_CTL\_REG(port),0)
- Check if Uart Irq enable.*
- #define `RT_UART_LinSyncOn`(port) `MemoryOr32`(UART\_CTL\_REG(port),0x8)
- Turn on LIN sync.*
- #define `RT_UART_LinSyncOff`(port) `MemoryAnd32`(UART\_CTL\_REG(port),~0x8)
- Turn off LIN sync.*
- #define `RT_UART_LinBreakNormal`(port) `MemoryAnd32`(UART\_CTL\_REG(port),~0x20)
- Turn on LIN break normal length (13-bit)*
- #define `RT_UART_LinBreakExtreme`(port) `MemoryOr32`(UART\_CTL\_REG(port),0x20)
- Turn on LIN break extra long length (26-bit)*
- #define `RT_UART_LinSendBreak`(port) `MemoryWrite32`(UART\_LIN\_BREAK\_REG(port),0x0)
- Send LIN break.*
- #define `RT_UART_LinCheckIrq`(port) ((`MemoryRead32`(UART\_CTL\_REG(port))&0x4)>>3)
- Check LIN irq enable.*
- #define `RT_UART_GetBrp`(port) `MemoryRead32`(UART\_BRP\_REG(port))
- Read Uart Brp value.*
- #define `RT_UART_LinSlave`(port) `UART_LinSyncOn`(port)
- Set LIN slave mode.*

## Functions

- void `RT_UART_putchar` (uint32\_t port, unsigned char c)  
*Send a character via Uart, to print via UART0, `_putchar()` is recommended.*
- void `RT_UART_puts` (uint32\_t port, unsigned char \*string)  
*Send a string via Uart, to print via UART0, `puts()` is recommended.*
- unsigned char `RT_UART_getchar` (uint32\_t port)  
*Get a character via Uart, to get a byte via UART0, `getchar()` is recommended.*
- void `RT_UART_LinMaster` (uint32\_t port, char pattern)  
*Send LIN break mode.*

### 27.34.1 Detailed Description

UART Library for M2.

#### Author

Zhirui Dai

#### Date

31 Oct 2017

#### Copyright

2018 Zhirui

### 27.34.2 Macro Definition Documentation

#### 27.34.2.1 RT\_UART\_Busy

```
#define RT_UART_Busy(  
    port ) MemoryRead32 (UART_BUSY_REG (port))
```

Check if Uart Tx is busy.

#### Parameters

<i>port</i>	Port to use, optional value:  <code>UART0</code> , <code>UART1</code>
-------------	---

#### Returns

int 1 = Uart Tx busy 0 = Uart Tx not busy

## 27.34.2.2 RT\_UART\_CheckIrq

```
#define RT_UART_CheckIrq(  
    port ) MemoryBitAt (UART_CTL_REG(port),0)
```

Check if Uart Irq enable.

## Parameters

<i>port</i>	Port to use, optional value:  UART0, UART1
-------------	--

## Returns

int 1=enable 0=disable

## 27.34.2.3 RT\_UART\_ClearIrq

```
#define RT_UART_ClearIrq(  
    port ) MemoryWrite32 (UART_IRQ_ACK_REG(port),0x0)
```

Clear Uart Irq (interrupt)

## Parameters

<i>port</i>	Port to use, optional value:  UART0, UART1
-------------	--

## Returns

void

## 27.34.2.4 RT\_UART\_CompareOff

```
#define RT_UART_CompareOff(  
    port ) MemoryAnd32 (UART_CTL_REG(port),~0x2)
```

This function sets UART compare irq off.



**Parameters**

<i>port</i>	Port to use, optional value:  UART0, UART1
-------------	--

**Returns**

void

**27.34.2.5 RT\_UART\_CompareOn**

```
#define RT_UART_CompareOn(  
    port ) MemoryOr32 (UART_CTL_REG (port), 0x2)
```

Set UART compare irq on, irq flag will be raised when the port receives a byte the same as the compare value.

**Parameters**

<i>port</i>	Port to use, optional value:  UART0, UART1
-------------	--

**Returns**

void

**27.34.2.6 RT\_UART\_DataReady**

```
#define RT_UART_DataReady(  
    port ) MemoryRead32 (UART_DATA_RDY_REG (port))
```

Check if Uart Rx has received data.

**Parameters**

<i>port</i>	Port to use, optional value:  UART0, UART1
-------------	--

**Returns**

int 1=ready 0=not ready

### 27.34.2.7 RT\_UART\_GetBrp

```
#define RT_UART_GetBrp(  
    port ) MemoryRead32 (UART_BRP_REG (port))
```

Read Uart Brp value.

#### Parameters

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

#### Returns

int brp value

### 27.34.2.8 RT\_UART\_IrqOff

```
#define RT_UART_IrqOff(  
    port ) MemoryAnd32 (UART_CTL_REG (port), ~1)
```

Turn off Uart Irq (interrupt)

#### Parameters

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

#### Returns

void

### 27.34.2.9 RT\_UART\_IrqOn

```
#define RT_UART_IrqOn(  
    port ) MemoryOr32 (UART_CTL_REG (port), 1)
```

Turn on Uart Irq enable.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

**Returns**

void

**27.34.2.10 RT\_UART\_LinBreakExtreme**

```
#define RT_UART_LinBreakExtreme(  
    port ) MemoryOr32 (UART_CTL_REG (port), 0x20)
```

Turn on LIN break extra long length (26-bit)

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

**Returns**

void

**27.34.2.11 RT\_UART\_LinBreakNormal**

```
#define RT_UART_LinBreakNormal(  
    port ) MemoryAnd32 (UART_CTL_REG (port), ~0x20)
```

Turn on LIN break normal length (13-bit)

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

**Returns**

void

**27.34.2.12 RT\_UART\_LinCheckIrq**

```
#define RT_UART_LinCheckIrq(  
    port ) ((MemoryRead32(UART_CTL_REG(port)) & 0x4) >> 3)
```

Check LIN irq enable.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0</code> , <code>UART1</code>
-------------	---

**Returns**

int 1=enable 0=disable

**27.34.2.13 RT\_UART\_LinSendBreak**

```
#define RT_UART_LinSendBreak(  
    port ) MemoryWrite32(UART_LIN_BREAK_REG(port), 0x0)
```

Send LIN break.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0</code> , <code>UART1</code>
-------------	---

**Returns**

void

**27.34.2.14 RT\_UART\_LinSlave**

```
#define RT_UART_LinSlave(  
    port ) UART_LinSyncOn(port)
```

Set LIN slave mode.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

**Returns**

void

**27.34.2.15 RT\_UART\_LinSyncOff**

```
#define RT_UART_LinSyncOff(  
    port ) MemoryAnd32(UART_CTL_REG(port), ~0x8)
```

Turn off LIN sync.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

**Returns**

void

**27.34.2.16 RT\_UART\_LinSyncOn**

```
#define RT_UART_LinSyncOn(  
    port ) MemoryOr32(UART_CTL_REG(port), 0x8)
```

Turn on LIN sync.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

**Returns**

void

#### 27.34.2.17 RT\_UART\_Off

```
#define RT_UART_Off(  
    port ) MemoryOr32 (UART_CTL_REG (port), 0x10)
```

Turn off Uart.

##### Parameters

<i>port</i>	Port to use, optional value:  UART0, UART1
-------------	--

##### Returns

void

#### 27.34.2.18 RT\_UART\_On

```
#define RT_UART_On(  
    port ) {MemoryAnd32 (UART_CTL_REG (port), ~0x10);}
```

Turn on Uart.

##### Parameters

<i>port</i>	Port to use, optional value:  UART0, UART1
-------------	--

##### Returns

void

#### 27.34.2.19 RT\_UART\_RaiseIrq

```
#define RT_UART_RaiseIrq(  
    port ) MemoryOr32 (UART_CTL_REG (port), 0x1)
```

Turn on Uart Irq enable.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

**Returns**

void

**27.34.2.20 RT\_UART\_Read**

```
#define RT_UART_Read(  
    port ) MemoryRead32 (UART_READ_REG (port))
```

Read Uart Rx data.

**Parameters**

<i>port</i>	Port to use, optional value:  <code>UART0, UART1</code>
-------------	---

**Returns**

int Rx data

**27.34.2.21 RT\_UART\_SetBaudrate**

```
#define RT_UART_SetBaudrate(  
    mode ) RT_MCU_SetSystemClock (mode)
```

Set the baudrate of UART.

**Parameters**

<i>mode</i>	Baudrate to set, optional value:  <code>BAUDRATE_9600, BAUDRATE_19200, BAUDRATE_38400</code>
-------------	--

**Returns**

void

## 27.34.2.22 RT\_UART\_SetCompare

```
#define RT_UART_SetCompare(  
    port,  
    val ) MemoryOr32 (UART_CTL_REG (port) , val<<8)
```

Set UART port's compare value.

## Parameters

<i>port</i>	Port to use, optional value:  UART0, UART1
<i>val</i>	Value to be compared.

## Returns

void

## 27.34.2.23 RT\_UART\_Write

```
#define RT_UART_Write(  
    port,  
    val ) MemoryWrite32 (UART_WRITE_REG (port) , val)
```

Sent data via Uart Tx.

## Parameters

<i>port</i>	Port to use, optional value:  UART0, UART1
<i>val</i>	data to Tx

## Returns

void

## 27.34.3 Function Documentation

## 27.34.3.1 RT\_UART\_getchar()

```
unsigned char RT_UART_getchar (  
    uint32_t port )
```

Get a character via Uart, to get a byte via UART0, [getchar\(\)](#) is recommended.



**Parameters**

<i>port</i>	Port to use, optional value:  <a href="#">UART0</a> , <a href="#">UART1</a>
-------------	---

**Returns**

char 1-byte data from Uart

**27.34.3.2 RT\_UART\_LinMaster()**

```
void RT_UART_LinMaster (
    uint32_t port,
    char pattern )
```

Send LIN break mode.

**Parameters**

<i>port</i>	Port to use, optional value:  <a href="#">UART0</a> , <a href="#">UART1</a>
<i>pattern</i>	The length of the break, optional value:  <a href="#">NORMAL_BREAK</a> , <a href="#">EXTREME_BREAK</a>

**Returns**

void

**27.34.3.3 RT\_UART\_putchar()**

```
void RT_UART_putchar (
    uint32_t port,
    unsigned char c )
```

Send a character via Uart, to print via UART0, [putchar\(\)](#) is recommended.

**Parameters**

<i>port</i>	Port to use, optional value:  <a href="#">UART0</a> , <a href="#">UART1</a>
<i>c</i>	character

**Returns**

void

**27.34.3.4 RT\_UART\_puts()**

```
void RT_UART_puts (
    uint32_t port,
    unsigned char * string )
```

Send a string via Uart, to print via UART0, [puts\(\)](#) is recommended.

**Parameters**

<i>port</i>	Port to use, optional value:  <a href="#">UART0</a> , <a href="#">UART1</a>
<i>string</i>	string

**Returns**

void

**27.35 /Users/daizhirui/Development/CamelStudio\_Library/Official/src/WDT.h File Reference**

Watchdog Library for M2.

```
#include "mcu.h"
```

**Macros**

- `#define RT\_WatchDog\_Setup(n, irq, rst)`  
*This function set the WDT app.*
- `#define RT\_WatchDog\_Clear()` `MemoryWrite32(WDT_CLR_REG, 1)`  
*Clear Watchdog.*
- `#define RT\_WatchDog\_ReadValue()` `MemoryRead32(WDT_READ_REG)`  
*Read watchdog value.*

**27.35.1 Detailed Description**

Watchdog Library for M2.

**Author**

John & Jack, Zhirui Dai

**Date**

16 Jun 2018

**Copyright**

2018 Zhirui

## 27.35.2 Macro Definition Documentation

### 27.35.2.1 RT\_WatchDog\_Clear

```
#define RT_WatchDog_Clear( ) MemoryWrite32(WDT_CLR_REG, 1)
```

Clear Watchdog.

#### Returns

void

### 27.35.2.2 RT\_WatchDog\_ReadValue

```
#define RT_WatchDog_ReadValue( ) MemoryRead32(WDT_READ_REG)
```

Read watchdog value.

#### Returns

uint4\_t The number of 1/8s.

### 27.35.2.3 RT\_WatchDog\_Setup

```
#define RT_WatchDog_Setup(  
    n,  
    irq,  
    rst )
```

#### Value:

```
{  
    MemoryWrite32(WDT_CTL0_REG, (n << 4 | irq << 2 | rst << 1 | 1)); \  
    MemoryOr32(SYS_CTL0_REG, irq); \  
}
```

This function set the WDT app.

#### Parameters

<i>n</i>	the number of 1/8s n=1~16
<i>irq</i>	trigger interrupt, ON or oFF
<i>rst</i>	whether reset the system, ON or OFF

Returns

void

