Machine Translated by Google

WM_W800_SDK_OS Migration Guide

V1.0

WinnerMicro

Beijing Lianshengde Microelectronics Co., Ltd. (winner micro)

Address: Floor 18, Yindu Building, No. 67, Fucheng Road, Haidian District, Beijing

Tel: +86-10-62161900

Company website: www.winnermicro.com

Machine Translated by Google

Document modification record

Version revision time		revision history	author review	
V0.1	2019/9/25	[C] Create document	Zhangwl	
V0.2	2020/7/8	Unified font	Cuiyc	
V1.0	2020/8/10	Upgrade version number	Cuiyc	

WinnerMicro

Machine Translated by Google

Table of contents

Documentation Modification History 2

Table of contents..... 3

1 **Introduction** 5

1.1 Purpose of writing..... 5

1.2 Intended Reader 5

1.3 Definition of Terms..... 5

1.4 References..... 5

2 SDK Kernel porting..... 6

3 System scheduling related porting..... 6

4 Memory-related porting..... 7

5 printf function..... 7

6 OSAL Layer Migration..... 7

6.1 System clock..... 8

6.2 Operating system type..... 8

6.3 OSAL layer interface functions..... 8

6.3.1 Clock related functions..... 9

6.3.2 System initialization function..... 9

6.3.3 Task-related functions..... 10

6.3.4 Semaphore related functions..... 10

6.3.5 Queue-related functions..... 12

6.3.6 Mailbox message related functions..... 12

WinnerMicro

Machine Translated by Google

6.3.7 Critical Section Related Functions..... 13

6.3.8 System timer related functions..... 13

6.3.9 other..... 14

7 Switch Kernel in the project..... 15

WinnerMicro

Machine Translated by Google

1 **Introduction**

1.1 Purpose of writing

This document describes how to guide SDK developers of Winnermicro W800 chips to migrate or switch to a new OS.

1.2 Intended audience

W800 SDK developers and engineering implementers

1.3 Definition of terms

the term	illustrate
API	Application Program Interface
YOU	Operating System
SDK	Software Development Kit

1.4 References

none

WinnerMicro

Machine Translated by Google

2 SDK Kernel porting

As shown in Figure 1, the operating system-related codes of the SDK are placed in the ./Src/OS directory, and have been transplanted

FreeRTOS real-time operating system. If users need to use other real-time operating systems, they can create new

Folder, add the operating system you need



Figure 1 Place directory

3 system scheduling related transplantation

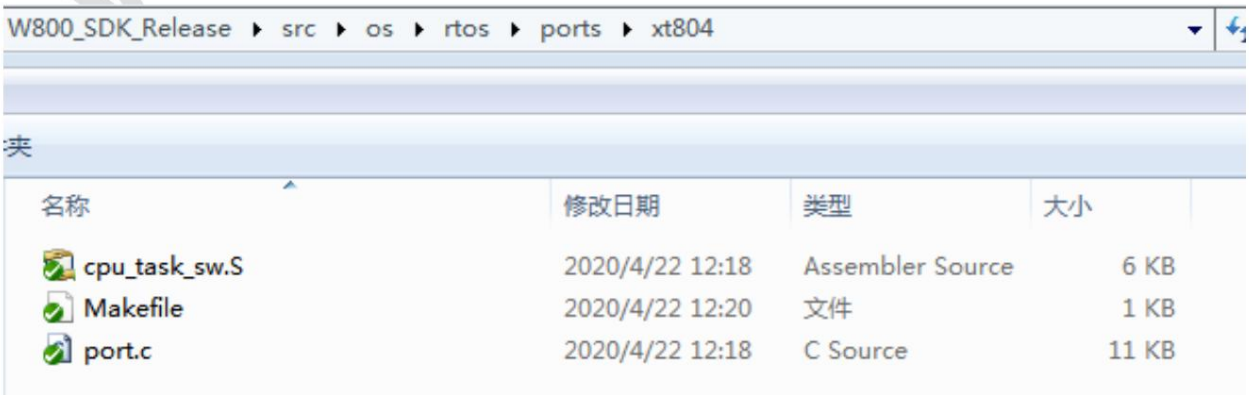
For related functions, see cpu_task_sw.S file under sdk_root_dir/src/os/rtos/ports/ck804/ and

port.c file. Here we take FreeRTOS as an example. When users use this operating system, they need to implement the

to the corresponding interface. If you need to use other operating systems, it is recommended to download the sdk of xt804 and find its sample code

The operating system that has been transplanted in the computer can be used directly; if there is no existing one available, it needs to be used according to the requirements of the operating system

To implement the corresponding interface, the technical support of the original kernel (XT804) factory (Pingtou brother) may be required here.



Machine Translated by Google

4 memory-related porting

In the wm_mem.c file, you can see the malloc memory application method provided by the SDK for FreeRTOS. because

Because the FreeRTOS system has special requirements, the memory application methods of interrupt functions and ordinary functions are different.

If the user's operating system has relevant requirements, it also needs to be processed; if there are no relevant requirements, it will be closed

The TLS_OS_FREERTOS macro definition in wm_os_config.h is the malloc function of the C library to apply for memory.

5 printf function

You can see two functions in the retarget.c file:

1ÿint wm_vprintf(const char *fmt, va_list arg_ptr)ÿ

2ÿint wm_printf(const char *fmt,...)ÿ

This is the printing function that will be used in the two Wi-Fi. Users can implement this function by themselves according to the compilation environment, or they can

Use the implemented function to distinguish it from the printf function.

6 OSAL Layer Migration

In order to facilitate the transplantation of the operating system, a layer of abstraction is encapsulated between the operating system and the W80X SDK application program.

That is, the interface functions of the operating system that need to be used in the program are repackaged, and the program uses a unified interface call operation

System related functions.

The interface functions that need to be implemented are defined in wm_osal.h in the ./include/OS directory

WinnerMicro

Machine Translated by Google

6.1 System Clock

Declare HZ as a constant:

```
const unsigned int HZ = 500;
```

Referenced in the wm_osal.h file

```
extern const unsigned int HZ;
```

The operating system clock definition, here indicates that 500 system clocks have a frequency of 1HZ, that is, a sysTick is generated in 2ms

Interrupt, users can modify it according to their needs. For some operating systems, it is necessary to configure the corresponding time in the operating system.

The settings are consistent with what we defined, for example, the settings in freeRTOSConfig.h need to be modified in FreeRTOS

```
#define configTICK_RATE_HZ          ( ( portTickType ) 500u )   //时 间 片 中 断 的 频 率
```

6.2 Operating system type

```
/** ENUMERATION of OS */
enum TLS_OS_TYPE{
    OS_UCOSII = 0,
    OS_FREERTOS = 1,
    OS_MAX_NUM
};
```

Operating system type definition, used to indicate the type of the current operating system, users can add their own operating system types.

It is mainly used for compatibility design on special occasions (temporarily not used).

6.3 OSAL layer interface functions

OSAL layer interface functions, users can refer to the implementation methods in FreeRTOS supported by the existing SDK to convert their own

Your own operating system-related functions are encapsulated in these interfaces.

The functions of FreeRTOS are encapsulated in the wm_osal_rtos.c file.

Machine Translated by Google

6.3.1 Clock related functions

function prototype	void tls_os_timer_init(void); void tls_os_time_tick(void *p);
Function introduces two functions related to the clock, which have been implemented in our project.	

function prototype	u32 tls_os_get_time(void);
Function introduction Get the tick value that the current operating system has been running.	

6.3.2 System initialization function

function prototype	void tls_os_init(void *arg);
The function introduces the operating system initialization function, which is called when the program is initialized. The operating system used needs to be implemented.	

function prototype	int tls_os_get_type(void);
Function introduction Get the current operating system type, different operating systems may be used in some programs The processing method may be different, the program will judge the return value of this function to determine How the program is handled.	

WinnerMicro

Machine Translated by Google

6.3.3 Task-related functions

function prototype	void tls_os_start_scheduler(void);
The function introduces the start task scheduling, and calls to start the execution of the system task after the task is created.	

function original type	tls_os_status_t tls_os_task_create(tls_os_task_t *task,const char* name, void (*entry)(void* param),void* param,u8 *stk_start,u32 stk_size,u32 prio,u32 flag);
function introduce	Task creation function, need to pay attention to two points: 1. The stk_start task stack cannot use the memory used by other peripherals The memory used by peripherals is defined in wm_ram_config.h 2. The larger the prio task priority value, the lower the priority, 0 is the highest priority, Some operating systems need adjustments not in this order , can refer to Package method for FreeRTOS (configMAX_PRIORITIES - prio).

function prototype	tls_os_status_t tls_os_task_del(u8 prio, void (*freefun)(void));
The function introduces the deletion task. The deletion task is not used in the program at present, so it is not necessary to implement it.	

6.3.4 Semaphore related functions

WinnerMicro

Machine Translated by Google

function prototype	tls_os_mutex_create tls_os_mutex_delete tls_os_mutex_acquire tls_os_mutex_release
The function introduces the creation, deletion, acquisition and release of mutex semaphores. Note: For the release of semaphores, some operating systems need to distinguish whether they are interrupted In the operation, if the operation is in the interrupt, you need to call the special interface for interrupt processing mouth. And the calls in our application are all unified interfaces, so users need Inside the wrapper function, it is judged whether the current function is interrupted. refer to The implementation in FreeRTOS uses the tls_get_isr_count interface to judge the current Whether to call in the interrupt function.	

function prototype	tls_os_sem_create tls_os_sem_delete tls_os_sem_acquire tls_os_sem_release tls_os_sem_set
The function introduces the creation, deletion, acquisition, release, and setting of the counting semaphore. Note: For the release of semaphores, some operating systems need to distinguish whether they are interrupted In the operation, if the operation is in the interrupt, you need to call the special interface for interrupt processing mouth. And the calls in our application are all unified interfaces, so users need Inside the wrapper function, it is judged whether the current function is interrupted. refer to	

WinnerMicro

Machine Translated by Google

	The implementation in FreeRTOS uses the tls_get_isr_count interface to judge the current Whether to call in the interrupt function.

6.3.5 Queue related functions

function prototype	tls_os_queue_create tls_os_queue_delete tls_os_queue_send tls_os_queue_flush tls_os_queue_receive
Functions introduce the creation, deletion, sending, refreshing, and receiving of message queues. Note: For the sending of message queues, some operating systems need to distinguish whether they are in Interrupt operation, if it is operating in an interrupt, it needs to call the special interrupt processing interface. And the calls in our application program are all unified interfaces, so users need to It is necessary to judge whether the current function is interrupted inside the encapsulated function. refer to The implementation in FreeRTOS uses the tls_get_isr_count interface to judge the current Whether to call in the interrupt function.	

6.3.6 Mailbox message related functions

function prototype	tls_os_mailbox_create tls_os_mailbox_delete

WinnerMicro

Machine Translated by Google

	tls_os_mailbox_send tls_os_mailbox_receive
Functions introduce the creation, deletion, sending, and receiving of mailbox messages. Note: For the sending of mailbox messages, some operating systems need to distinguish whether they are in Interrupt operation, if it is operating in an interrupt, it needs to call the special interrupt processing interface. And the calls in our application program are all unified interfaces, so users need to It is necessary to judge whether the current function is interrupted inside the encapsulated function. refer to The implementation in FreeRTOS uses the tls_get_isr_count interface to judge the current Whether to call in the interrupt function.	

6.3.7 Critical Section Related Functions

function prototype	tls_os_set_critical(void)
Function introduction Close all interrupts of the current system.	

function prototype	void tls_os_release_critical(u32 cpu_sr)
The function introduction re-enables all interrupts of the current system.	

6.3.8 System timer related functions

function prototype	tls_os_timer_create
The function introduces the timer callback task creation.	

Machine Translated by Google

	Note: After the timer task is created, before calling tls_os_timer_start, The timer is not started. For the operating system that needs to be encapsulated, it must also meet this One point, if the operating system can choose whether to auto_run, choose not to start.

function prototype	tls_os_timer_start tls_os_timer_stop tls_os_timer_delete
The function introduces the start, stop and delete of the timer task.	

function prototype	tls_os_timer_change
The function introduces to modify the timing value of the timer task. **Note: After calling this function, the timer will be restarted, and this needs to be paid attention to in the package** For one point, you can refer to the implementation in FreeRTOS.	

function prototype	void tls_os_time_delay(u32 ticks)
The function introduces the task delay, the unit is tick.	

6.3.9 Others

function prototype	void tls_os_disp_task_stat_info(void)
Function introduction Print the running status of the current operating system, which can be used during debugging. To achieve.	

WinnerMicro

Machine Translated by Google

7Switch the Kernel in the project

To replace the Kernel in the Keil project, you need to find the wm_os_config.h file and select the current operating system to be used. After modifying the Kernel to be selected, replace the .c files related to the OS operating system under the OS group in the keil project with Kernel files transplanted by users.

To change the operating system of GCC, just modify the wm_config_gcc.inc file directly. If the user uses his own operating system The operating system also needs to modify the corresponding Makefile, see "WM_W800_SDK Script Compilation Guide" for details.

WinnerMicro