

# HT32F125x RTOS

D/N : HA0288E

## Introduction

### Keil RTX

The ARM-ARTX is a real-time operating system, otherwise known as RTOS, that uses Keil's RealView Microcontroller Development Kit, RVMDK, on Holtek's Cortex-M3 microcontrollers. A simple demo is also provided to indicate how to implement the RTOS on Holtek's 32-bit MCUs. For more detailed information and further documentation, refer to the website: <http://www.keil.com/rtos/>.

### FreeRTOS

FreeRTOS<sup>TM</sup> is a portable, open-source, royalty-free, compact Real Time Kernel. It is free for download and users can freely use the RTOS for commercial application development with no requirements to reveal the user proprietary source code. Downloaded more than 77,500 times during 2008, the FreeRTOS has become a cross platform defacto standard for embedded microcontrollers.

Each official port includes a pre-configured example application which is used for kernel feature demonstrations, expediting learning and facilitating 'out of the box' development. Further support is provided in the way of an active user community. For more information refer to the website: <http://www.freertos.org/main.html>.

### IAR PowerPac

IAR Systems offer one-stop-shopping for embedded development tools. RTOS and middleware components integrate seamlessly with the IAR Embedded Workbench and provide a complete development environment even for the most complex applications. Refer to the website for more information: <http://www.iar.com/website1/1.0.1.0/353/1/>.

### CooCox CoOS

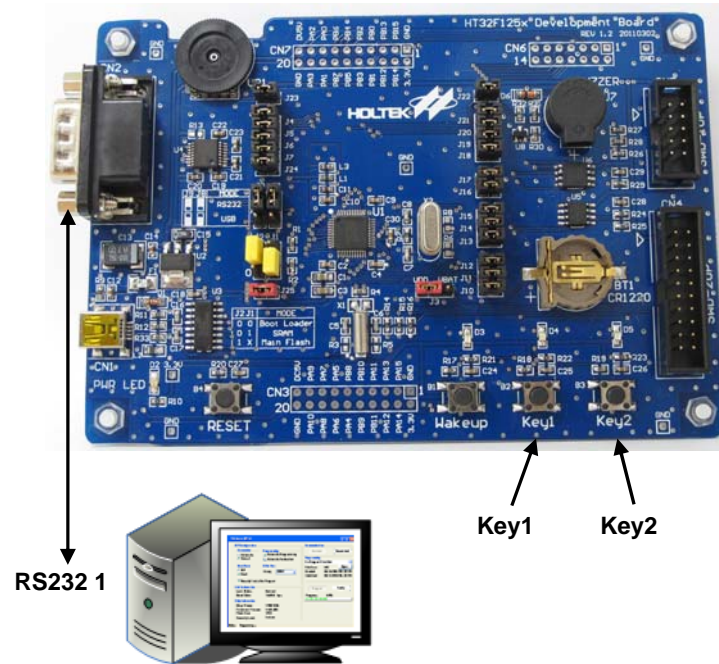
CooCox CoOS is an embedded real-time multi-task OS that is especially designed for the ARM Cortex- M series chipset. It is also a free and open RTOS.

In this document, simple example code is provided to indicate how to implement CooCox CoOS on Holtek's 32-bit MCUs. To understand more about CoOS , refer to the web site: <http://www.coocox.org/CoOS.htm>.

## Demonstration

### Hardware Setup

- Connect the Holtek 32bit MCU Development Board to the PC through the RS232 cable.
- Two switches on the board will be used later and will be known as "Key1" and a "Key2" during future demos.



### Demo Process

1. After startup, the OS will remain on hold until any key is pressed.
2. When one of the keys is pressed, the OS will start the demo process.
3. The "Buzzer" will then immediately start playing a song with the song title being transmitted to the PC through the RS232 interface. Run the hyper-terminal program on the PC to receive and show this information string.
4. Additionally, each LED represents a song. When "Song1" is playing, LED1 will also be turned on.
5. Press "Key1" to play the next song while "Key2" will play the previous song.

## Keil RTX

### RTOS Real-Time Kernel

Keil provides several RTOS and middleware components that are fully integrated into the C Compiler languages that it supports. Its kernel provides multitasking features for real-time applications.

The RTX provides all the resources to create and control multi-threaded, real-time applications and can be tailored to the exact system requirements. It fully supports ARM7, ARM9, and Cortex-M devices, and is included in the MDK-ARM. Refer to: <http://www.keil.com/rtos/>.

The RTX kernel is a real time operating system, RTOS, that is able to create applications that need to simultaneously perform multiple functions or tasks. This is very useful in embedded applications. An RTOS like an RTX can solve numerous scheduling, maintenance and timing issues.

An RTOS enables flexible scheduling of system resources like the CPU and the memory and offers ways of communication between tasks. The RTX kernel is a powerful RTOS that is easy to use and works with microcontrollers that are based on the ARM7™ TDMI, ARM9™ or Cortex™ M3 CPU core.

RTOS programs are written using standard C and compiled with the RealView® Compiler. The RTX.h header file defines the RTX functions and macros that allows for easy task declaration and access to all RTOS features.

Refer to: [http://www.keil.com/support/man/docs/rlarm/rlarm\\_ar\\_artxarm.htm](http://www.keil.com/support/man/docs/rlarm/rlarm_ar_artxarm.htm)

### Example Programs

The RL-ARM Kit includes several example programs that are configured for ARM® devices.

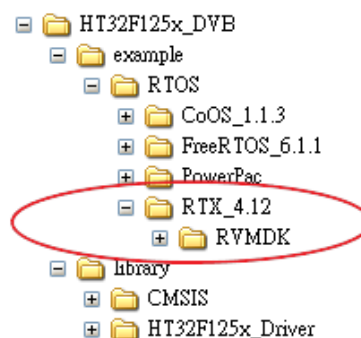
The example programs are located in the

\\Keil\ARM\RL\...\Examples\and\Keil\ARM\Boards\<vendor>\<board>\RL\... folders. Each example program is stored in a separate folder along with complete project files that will help to quickly rebuild these projects and run the programs.

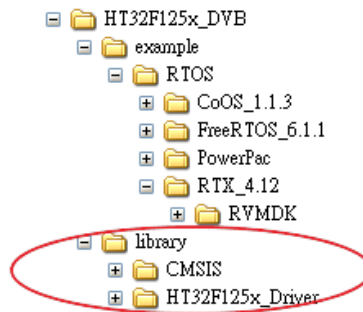
### Environment Setup

- Download the "Holtek RTOS Package" and extract the file into any known folder.
- Select "RTOS" under the working directory and copy "RTOS\_Config.c" from the "Keil installed path" to the working directory as shown below:

\\HT32Fxxx\example\RTOS\RTOS



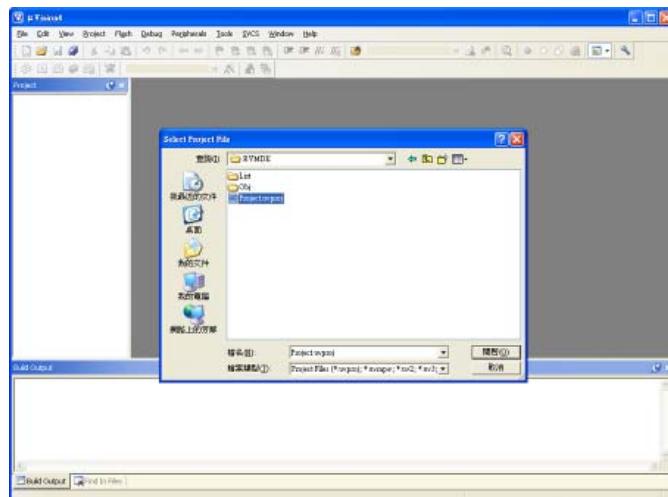
- Ensure that the "firmware library" exists under the working directory  
 \ HT32Fxxx\library



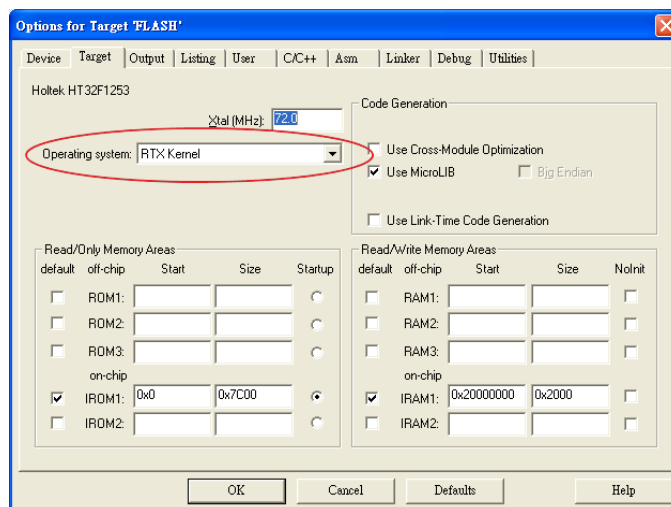
- Open the keil project file **xxxx.uvproj** as shown below:  
 \ HT32Fxxx\example\RTOS\RTX\RVMKD

**Software Setup**

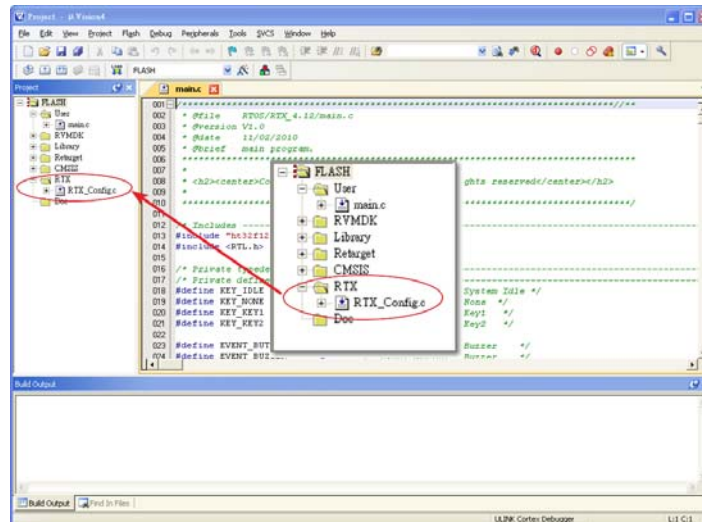
- Install Keil's RealView MDK-ARM. The updated evaluation version can be registered and downloaded at Keil's website.
- Open the RTOS Demo project.



- Ensure that the "RTX Kernel" has been set in the "Options for Target -Target" as shown below.



- An RTX configuration file RTX\_Config.c must also be added to the project.



## RTOS API

Below is part of the APIs that are used in the demo. For more complete information refer to the following below:

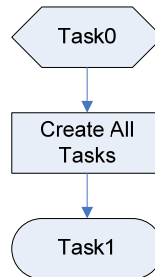
[http://www.keil.com/support/man/docs/rlarm/rlarm\\_library.htm](http://www.keil.com/support/man/docs/rlarm/rlarm_library.htm).

- `os_tsk_create` : Create the task identified by the task function pointer argument and then add the task to the ready queue. This will dynamically assign a task identifier value, TID, to the new task.
- `os_tsk_self` : Identify the currently running task by returning its task ID.
- `os_tsk_delete_self` : Stop and delete the currently running task. The program execution continues with the task with the next highest priority in the ready queue.
- `os_tsk_prio_self` : Change the priority of the currently running task to `new_prio`.
- `os_evt_set` : Set the event flags for the task identified by the function argument. The function only sets the event flags whose corresponding bit is set to 1 in the `event_flags` argument.
- `os_evt_wait_or` : Wait for one of the events specified in the `wait_flags` to occur. The function only waits on events whose corresponding flags have been set to 1 in the `wait_flags` parameter. The function can wait on as many as 16 different events.
- `os_dly_wait` : Pause the calling task. The argument `delay_time` specifies the length of the pause and is measured in number of `system_ticks`. The `delay_time` can be set to any value between 1 and 0xFFFFE.

## Tasks Description with Sample Code

### 1. Task0 - Init\_Task

- Functions
  - To create and define all the tasks
  - Create "Button Task"
  - Create "Buzzer Task"
  - Create "UART Task"
  - Create "LED Task"



- APIs and Flags

- Task\_Init : "Init Task" identifier value
- Task\_Button : "Button Task" identifier value
- Task\_Buzzer : "Buzzer Task" identifier value
- Task\_UART : "UART Task" identifier value
- Task\_LED : "LED Task" identifier value
- EVENT\_BUTTON : "Button Task" identifier value

```

/**
*****
* @brief Task 'Init_Task'
* @param[in] pdata D pointer to parameter passed to task.
* @param[out] None
* @retval None
*****
*/
__task void Init_Task (void )
{
    os_tsk_prio_self (1);

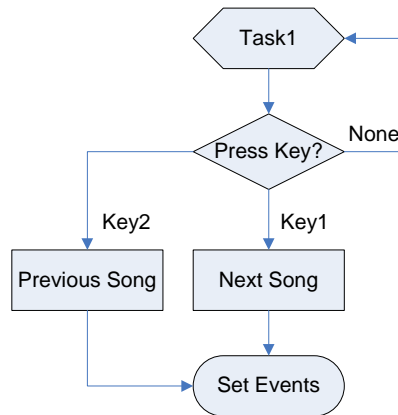
    /* Create four tasks for button, buzzer, UART, and LED function */
    Task_Init = os_tsk_self (); /* get own task identification number */
    Task_Button = os_tsk_create (Button_Task,BUTTON_TASK_PRIORITY); /* start task 1 */
    Task_Buzzer = os_tsk_create (Buzzer_Task,BUZZER_TASK_PRIORITY); /* start task 2 */
    Task_UART = os_tsk_create (UART_Task,UART_TASK_PRIORITY); /* start task 3 */
    Task_LED = os_tsk_create (LED_Task,LED_TASK_PRIORITY); /* start task 4 */

    /* Set event : Buzzer */
    os_evt_set(EVENT_BUTTON, Task_Button);

    /* Delete "init_task" task. */
    os_tsk_delete_self();
}
    
```

## 2. Task1 - Button\_Task

- Functions
  - To detect a key input depending on "ButtonNewStatus"
    - ButtonNewStatus = KEY\_KEY1  
Next song will be played.
    - ButtonNewStatus = KEY\_KEY2  
Previous song will be played.
  - If "ButtonNewStatus" has been changed, set the events of all other tasks.



- APIs and Flags
  - Get\_button\_state()
    - Get the current status of the key input.

```

/**
*****
* @brief Task 1 'Button_Task': Detect button
* @param[in] pdata A pointer to parameter passed to task.
* @param[out] None
* @retval None
*****
*/
__task void Button_Task (void)
{
    SongMsg = 0;

    for (;;)
    {
        /* Get button state*/
        ButtonNewStatus = Get_Button_State();

        if(ButtonNewStatus != KEY_NONE)
        {
            Tone = 0;//Reset song tone

            /* Detect Button and select song number */
            if(ButtonNewStatus == KEY_KEY1) SongMsg++;
            else if(ButtonNewStatus == KEY_KEY2) SongMsg--;
            SongMsg = SongMsg % 4;

            /* Set event : Buzzer */
            os_evt_set(EVENT_BUZZER, Task_Buzzer);
            /* Set event : UART */
            os_evt_set(EVENT_UART, Task_UART);
            /* Set event : LED */
            os_evt_set(EVENT_LED, Task_LED);
        }
    }
}

```

### 3. Task2 - Buzzer\_Task

- Functions

- Drive the buzzer depending on the event "EVENT\_BUZZER"

- Select the song depending on the flag "SongMsg"

SongMsg = 0

Stop buzzer driving

SongMsg = 1

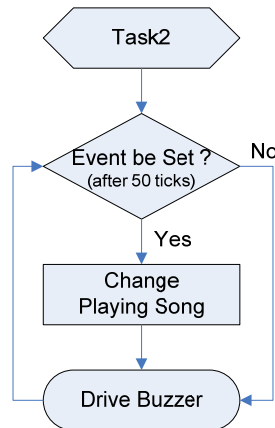
Song : "Twinkle Twinkle Little Star"

SongMsg = 2

Song : "Lightly Row"

SongMsg = 3

Song : "Here Comes the Train"



- APIs and Flags

- Buzzer\_Output (u16 SongNum)

Select which song is to be played.

```

/**
*****
* @brief Task 2 'Buzzer_Task': Buzzer
* @param[in] pdata B pointer to parameter passed to task.
* @param[out] None
* @retval None
*****
*/
__task void Buzzer_Task (void )
{
    for (;;)
    {
        os_evt_wait_or(EVENT_BUZZER,50);
        /* Output song */
        Buzzer_Output (SongMsg);
    }
}

```

#### 4. Task3 - UART\_Task

- Functions

- Send the song title via the UART to the PC

SongMsg = 0

Stop sending data to the PC.

SongMsg = 1

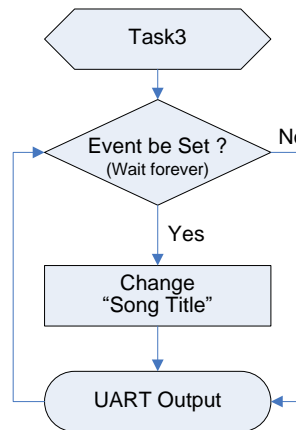
Title : "Twinkle Twinkle Little Star"

SongMsg = 2

Title : "Lightly Row"

SongMsg = 3

Title : "Here Comes the Train"



- APIs and Flags

- UART\_Display ( u16 OutputMode)

Select which title will be shown on the PC.

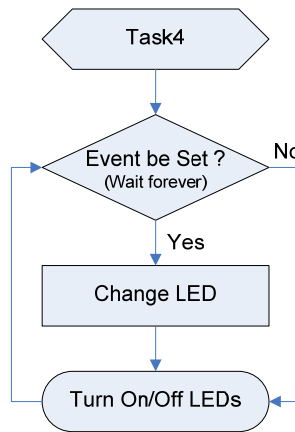
```

/**
*****
* @brief   Task 3 'UART_Task': UART Output
* @param[in]  pdata C pointer to parameter passed to task.
* @param[out] None
* @retval   None
*****
*/
__task void UART_Task (void )
{
  for (;;)
  {
    os_evt_wait_or(EVENT_UART, 0xffff);
    /* process message here */
    UART_Display(SongMsg);
  }
}

```

**5. Task4 - LED\_Task**

- Functions
  - Switch on/off the LEDs
    - SongMsg = 0  
LEDs are all off.
    - SongMsg = 1  
LED1 is on - other LEDs are off.
    - SongMsg = 2  
LED2 is on - other LEDs are off.
    - SongMsg = 3  
LED3 is on - other LEDs are off.



• APIs and Flags

LED\_Control (u16 LEDMode)

- Switch on the corresponding LED depending on LED Mode.

```

/**
*****
* @brief Task 4 'LED_Task': LED Output
* @param[in] pdata D pointer to parameter passed to task.
* @param[out] None
* @retval None
*****
*/
void LED_Task (void)
{
    for (;;)
    {
        os_evt_wait_or(EVENT_LED, 0xffff);
        /* process message here */
        LED_Control(SongMsg);
    }
}
    
```

## FreeRTOS

### FreeRTOS Kernel

FreeRTOS is licensed under a modified GPL and can be used in commercial applications under this license. An alternative commercial license option is also available at:

<http://www.freertos.org/a00114.html>.

FreeRTOS.org has been ported to many different architectures and compilers. Each port is accompanied by a pre-configured demo application to allow users to get up and running quickly. Each demo application is accompanied by a documentation page providing full information on locating the demo project source code, building the demo project and configuring the target hardware.

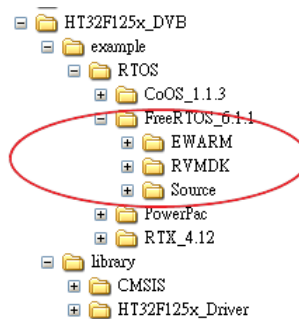
### Example Programs

Every demo application is located in a subdirectory of the FreeRTOS/Demo directory. The name of each such subdirectory describes the configuration of the demo application it contains. Refer to the FreeRTOS.org source code organization page for a full explanation of the FreeRTOS.org directory structure.

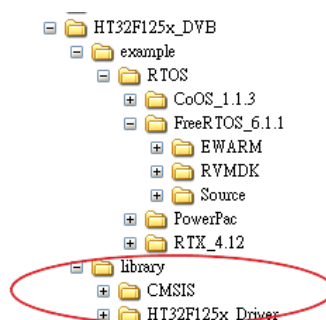
### Environment Setup

- Download the "FreeRTOS source code" and extract it.
- Download the "Holtek RTOS Package" and extract the file into any known folder.
- Select "FreeRTOS" under the working directory, and copy "Source" folder from "FreeRTOS source code directory" to the working directory as below:

\HT32Fxxx\example\RTOS\FreeRTOS



- Ensure that the "firmware library" exists under the working directory  
 \HT32Fxxx\library

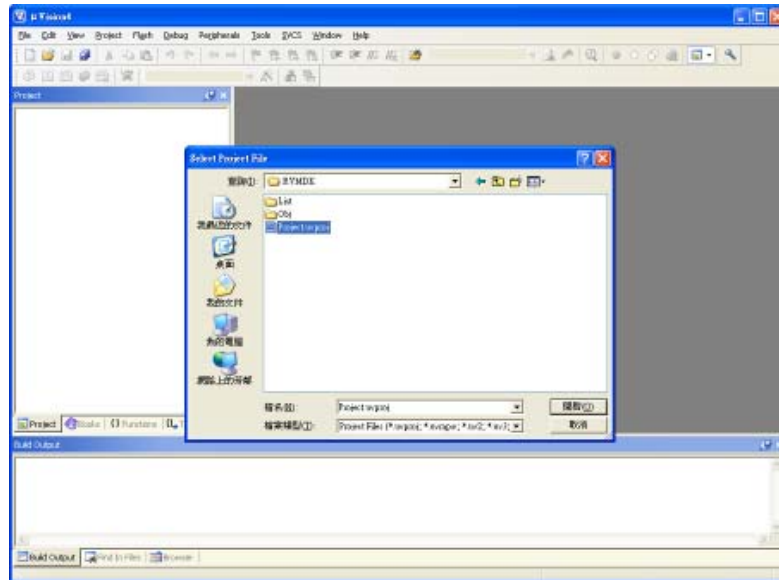


- Open the keil project file **xxxx.uvproj** as shown below:  
 \HT32Fxxx\example\RTOS\ FreeRTOS \RVMDK

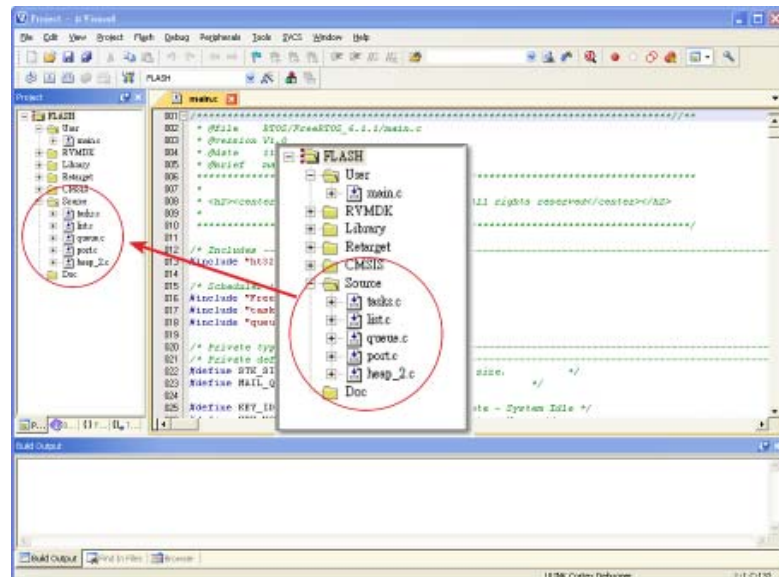
- The IAR project file **xxxx.eww** can also be found as shown below. In this document, only the keil environment is introduced.  
 \ HT32Fxxx\example\RTOS\ FreeRTOS \EWARM

### Software Setup

- Install Keil's RealView MDK-ARM. The updated evaluation version can be registered and downloaded at Keil's website.
- Open the RTOS Demo project.



- The "FreeRTOS kernel files" must be added from "Source" to the project.  
 \ HT32Fxxx\example\RTOS\FreeRTOS\_6.1.1\Source



## RTOS API

Below are some of the APIs that are used in the demo. For complete information, refer to :

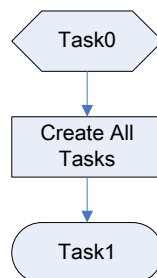
<http://www.freertos.org/a00106.html>

- xTaskCreate : Create a new task and add it to the list of tasks that are ready to run.
- xTaskDelete : Remove a task from the RTOS real time kernel management. The task being deleted will be removed from all ready, blocked, suspended and event lists.
- xQueueCreate : Create a new queue instance. This allocates the storage required by the new queue and returns a handle for the queue.
- xQueueSend : Post an item on a queue. The item is queued by copy, not by reference. This function must not be called from an interrupt service routine.
- xQueueReceive : Receive an item from a queue. The item is received by copy so a buffer of adequate size must be provided. The number of bytes copied into the buffer was defined when the queue was created.

## Tasks Description with sample code

### 1. Task0 - Init\_Task

- Functions
  - To create and define all the tasks
    - Create "Button Task"
    - Create "Buzzer Task"
    - Create "UART Task"
    - Create "LED Task"



- APIs and Flags
  - vInit\_Task : "Init Task" identifier value
  - vButton\_Task : "Button Task" identifier value
  - vBuzzer\_Task : "Buzzer Task" identifier value
  - vUART\_Task : "UART Task" identifier value
  - vLED\_Task : "LED Task" identifier value
  - Buzzer\_Queue : "Buzzer Task" identifier value
  - UART\_Queue : "UART Task" identifier value
  - LED\_Queue : "LED Task" identifier value

```

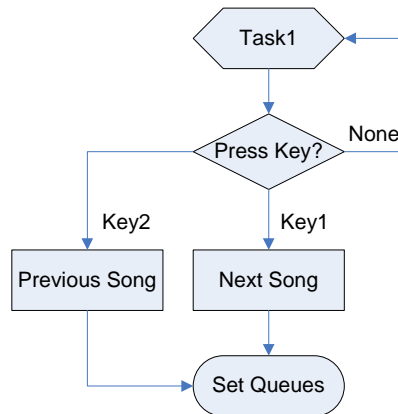
/**
*****
* @brief Task 'Init_Task'
* @param[in] pdata D pointer to parameter passed to task.
* @param[out] None
* @retval None
*****
*/
void Init_Task (void *pdata)
{
    /* Create three queues for buzzer, UART, and LED function */
    Buzzer_Queue = xQueueCreate(BUZZER_QUEUE_SIZE,4 );
    UART_Queue = xQueueCreate(UART_QUEUE_SIZE,4 );
    LED_Queue = xQueueCreate(LED_QUEUE_SIZE,4 );

    /* Create four tasks for button, buzzer, UART, and LED function */
    xTaskCreate(Button_Task, (signed portCHAR * ) "Button", BUTTON_TASK_STACK_SIZE, NULL, BUTTON_TASK_PRIORITY, &vButton_Task );
    xTaskCreate(Buzzer_Task, (signed portCHAR * ) "Buzzer", BUZZER_TASK_STACK_SIZE, NULL, BUZZER_TASK_PRIORITY, &vBuzzer_Task );
    xTaskCreate(UART_Task, (signed portCHAR * ) "UART", UART_TASK_STACK_SIZE, NULL, UART_TASK_PRIORITY, &vUART_Task );
    xTaskCreate(LED_Task, (signed portCHAR * ) "LED", LED_TASK_STACK_SIZE, NULL, LED_TASK_PRIORITY, &vLED_Task );

    /* Delete "init_task" task. */
    vTaskDelete( vInit_Task );
}
    
```

## 2. Task1 - Button\_Task

- Functions
  - To detect the key input depending on "ButtonNewStatus"
    - ButtonNewStatus = KEY\_KEY1  
Next song will be played.
    - ButtonNewStatus = KEY\_KEY2  
Previous song will be played.
  - If "ButtonNewStatus" has been changed, set the events of all other tasks.



- APIs and Flags
  - Get\_button\_state()
    - Get the current status of the key input.
  - Send\_Queue (u8 QueueType, u16\* QueueMessage)
    - Send queues that include "key state" to other tasks.

```

/**
*****
* @brief Task 1 'Button_Task': Detect button
* @param[in] pdata A pointer to parameter passed to task.
* @param[out] None
* @retval None
*****
*/
void Button_Task (void *pdata)
{
    SongMsg = 0;

    for (;;)
    {
        /* Get button state*/
        ButtonNewStatus = Get_Button_State();

        if(ButtonNewStatus != KEY_NONE)
        {
            Tone = 0; //Reset Tone

            /* Detect Button and select song number */
            if(ButtonNewStatus == KEY_KEY1) SongMsg++;
            else if(ButtonNewStatus == KEY_KEY2) SongMsg--;
            SongMsg = SongMsg % 4;

            /* Send Queue : UART */
            Send_Queue(Queue_UART, &SongMsg);
            /* Send Queue : LED */
            Send_Queue(Queue_LED, &SongMsg);
            /* Send Queue : Buzzer */
            Send_Queue(Queue_BUZZER, &SongMsg);
        }
    }
}

```

### 3. Task2 - Buzzer\_Task

- Functions

- Drive the buzzer depending on the event "EVENT\_BUZZER"

- Select the song depending on the flag "BuzzerMsg"

BuzzerMsg = 0

Stop driving the buzzer.

BuzzerMsg = 1

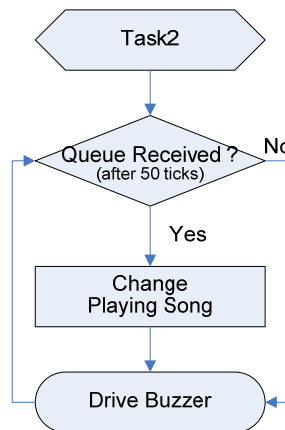
Song : "Twinkle Twinkle Little Star"

BuzzerMsg = 2

Song : "Lightly Row"

BuzzerMsg = 3

Song : "Here Comes the Train"



- APIs and Flags

- Buzzer\_Output (u16 SongNum)

Select which song is to be played.

```

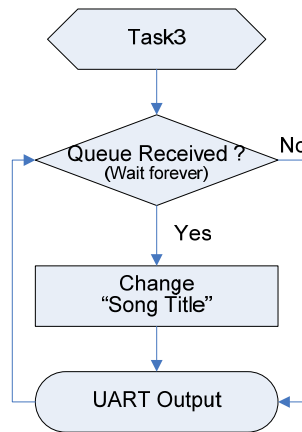
/**
*****
* @brief Task 2 'Buzzer_Task': Buzzer
* @param[in] pdata B pointer to parameter passed to task.
* @param[out] None
* @retval None
*****
*/
void Buzzer_Task (void *pdata)
{
    u16 BuzzerMsg = 0;;

    for (;;)
    {
        /*Receive Queue for "buzzer" function*/
        if( xQueueReceive( Buzzer_Queue, &BuzzerMsg , ( portTickType ) 50 ) );

        /* Output song */
        Buzzer_Output (BuzzerMsg);
    }
}
  
```

**4. Task3 - UART\_Task**

- Functions
  - Send the song title via the UART to the PC
    - UARTMsg = 0
      - Stop sending data to the PC.
    - UARTMsg = 1
      - Title : "Twinkle Twinkle Little Star"
    - UARTMsg = 2
      - Title : "Lightly Row"
    - UARTMsg = 3
      - Title : "Here Comes the Train"



- APIs and Flags
  - UART\_Display (u16 OutputMode)
    - Select which song title is to be shown on the PC.

```

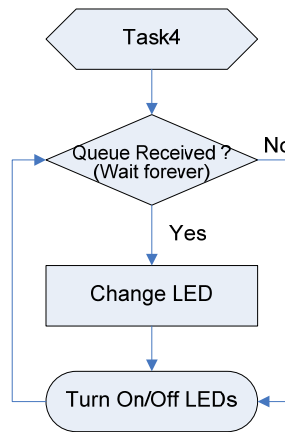
/**
*****
* @brief   Task 3 'UART_Task': UART Output
* @param[in]  pdata C pointer to parameter passed to task.
* @param[out] None
* @retval   None
*****
*/
void UART_Task (void *pdata)
{
    u16 UARTMsg = 0;

    for (;;)
    {
        /*Receive Queue for "UART" function*/
        if( xQueueReceive( UART_Queue, &UARTMsg , portMAX_DELAY ) )
        {
            /* Output song title*/
            UART_Display(UARTMsg);
        }
    }
}

```

**5. Task4 - LED\_Task**

- Functions
  - Switch on/off the LEDs
    - LEDMsg = 0  
The LEDs are all off
    - LEDMsg = 1  
LED1 is on - other LEDs are off
    - LEDMsg = 2  
LED2 is on - other LEDs are off
    - LEDMsg = 3  
LED3 is on - other LEDs are off



- APIs and Flags
  - LED\_Control (u16 LEDMode)
    - Turn on the corresponding LED depending on LEDMode.

```

/**
*****
* @brief Task 4 'LED_Task': LED Output
* @param[in] pdata D pointer to parameter passed to task.
* @param[out] None
* @retval None
*****
*/
void LED_Task (void *pdata)
{
    u16 LEDMsg = 0;

    for (;;)
    {
        /*Receive Queue for "LED" function*/
        if( xQueueReceive( LED_Queue, &LEDMsg , portMAX_DELAY ) )
        {
            /*Enable LED output*/
            LED_Control(LEDMsg);
        }
    }
}
  
```

## IAR PowerPac

### IAR PowerPac Kernel

The IAR PowerPac is a fully-featured real-time operating system, RTOS, combined with a high performance file system. The IAR PowerPac is tightly integrated with the IAR Embedded Workbench and comes with sample projects and board support packages for devices from different manufacturers.

### Example Programs

The IAR-PowerPac RTOS Kit includes several example programs for several evaluation boards.

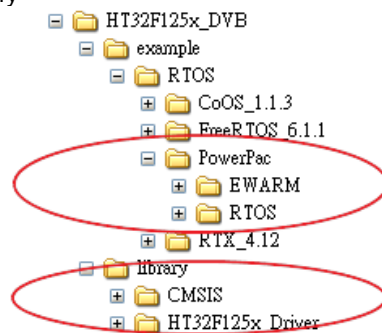
The example programs are located in \ IAR Systems\...\Arm\PowerPac \BoardSupport\... folders. Each example program is stored in a separate folder along with complete project files that will assist with rapid project rebuilding and program execution.

### Environment Setup

- Install the IAR Embedded Workbench Version 5.4 or above and also install the IAR PowerPac. The updated evaluation version can be downloaded through IAR's website after online registration.
- Download the "Holtek RTOS Package" and extract the file into any known folder.
- Select "PowerPac" under the working directory, and ensure that the "firmware library" exists under the working directory as shown below:

\ HT32Fxxx\example\RTOS\PowerPac

\ HT32Fxxx\library



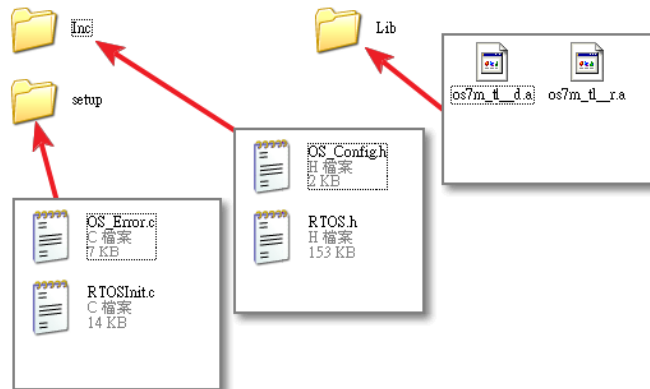
- Copy "os7m\_tl\_d.a" and "os7m\_tl\_r.a" from the "IAR install directory" to the project directory as shown below:

\ HT32Fxxx\example\RTOS\PowerPac\RTOSLib\

- Copy "OS\_Config.h" and "RTOS.h" from the "IAR install directory" to the project directory as shown below:

\ HT32Fxxx\example\RTOS\PowerPac\RTOS\Inc\

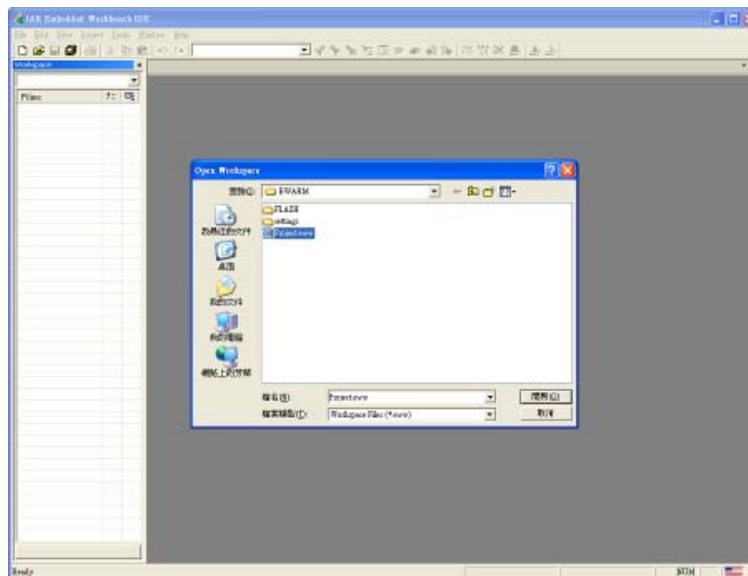
- Ensure that "OS\_Error.c" and "RTOSInit.c" exist in the following directory:  
 \HT32Fxxx\example\RTOS\PowerPac\RTOS\setup\



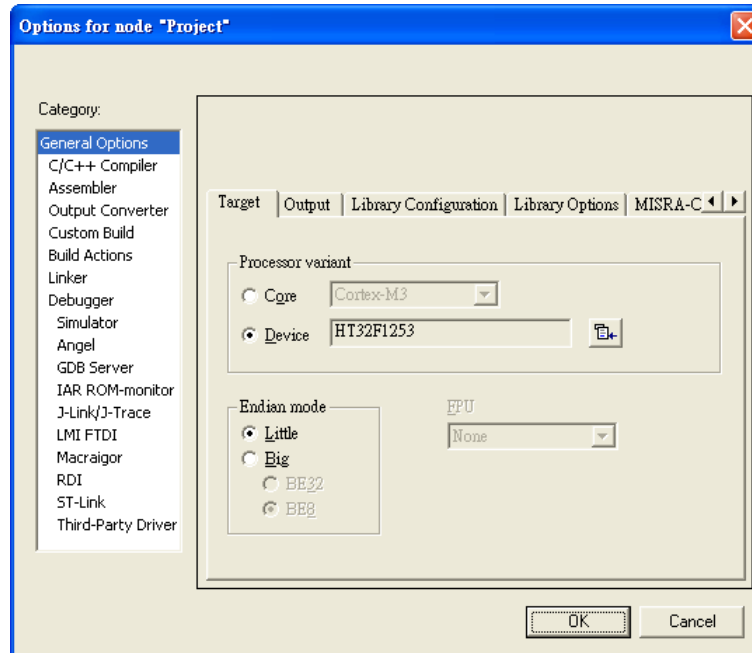
- The IAR project file **xxx.eww** can be found as shown below and opened:  
 \HT32Fxxx\example\RTOS\PowerPac\EWARM

### Software Setup

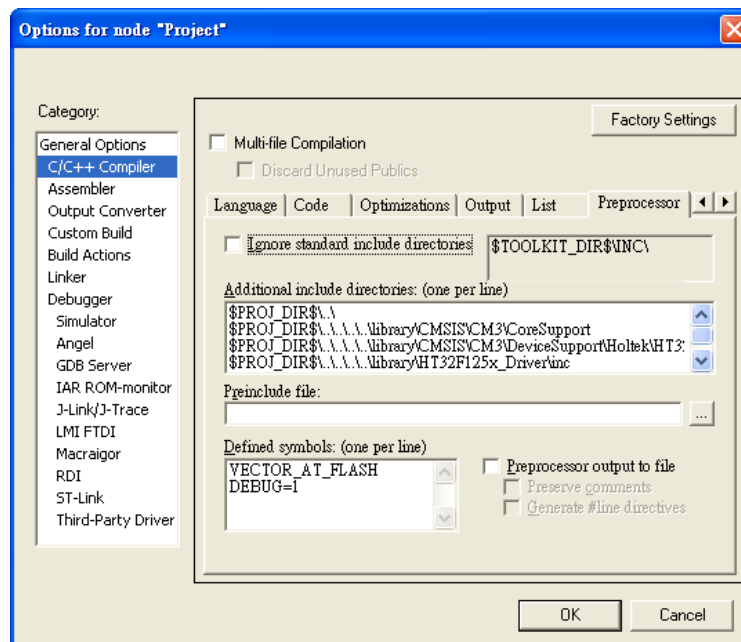
- Install the IAR Embedded Workbench and the IAR PowerPac. The updated evaluation version can be registered and downloaded at IAR's website.
- Open the RTOS Demo project.



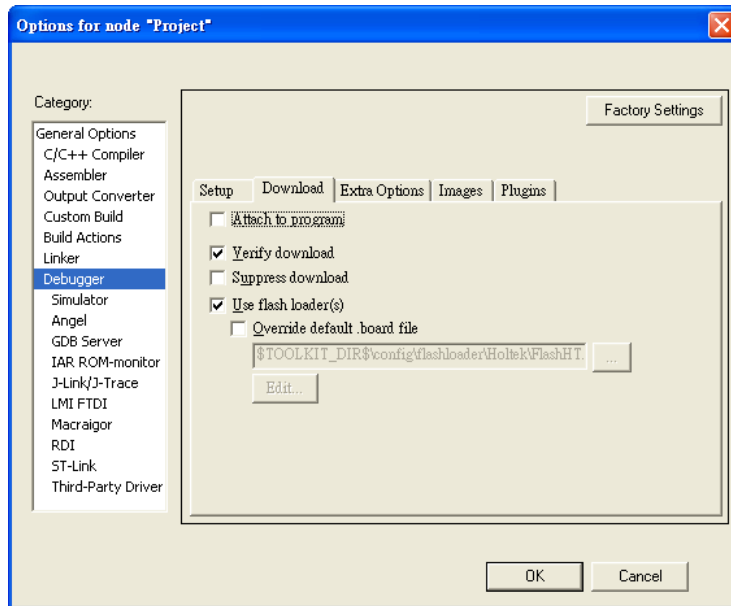
- Ensure that a "Holtek device" has been selected in the "Processor variant" area as shown below.



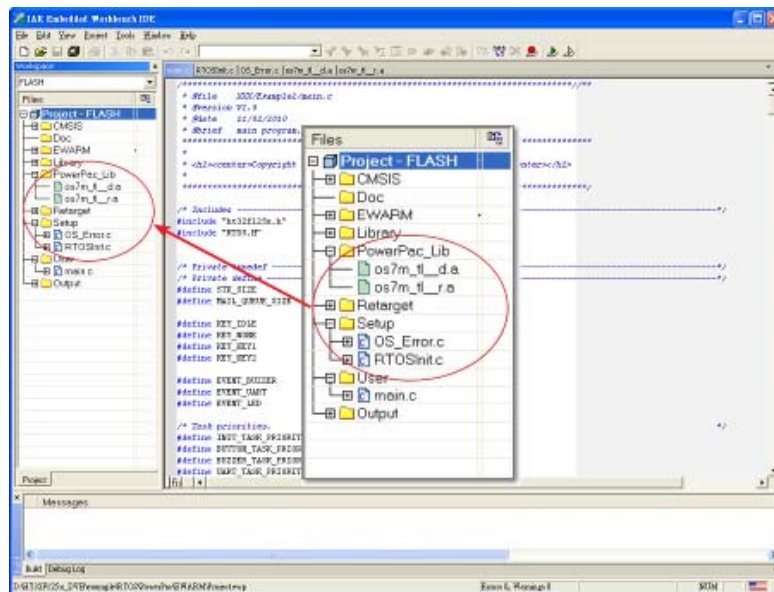
- Set the "Additional include directories" and "define symbols" as shown below.



- Setup the "flash loader file" in the "Download" section. If not setup the system will not be able to transfer the file to the MCU.



- Some PowerPac configuration files as shown below must also be added to the project.



For detailed information, refer to the IAR reference documents.

## RTOS API

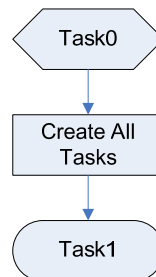
Below are some of the APIs that are used in the demo. Detailed information is located in the \\IAR Systems\...\arm\PowerPac\Doc\ folder.

- OS\_IncDI : Short for Increment and Disable Interrupts. Increments the interrupt disable counter, OS\_DICnt, and disables the interrupts.
- OS\_InitKern : Initialize all RTOS variables which require a non-zero initial value for arithmetic types or a non NULL pointer value for pointer types.
- OS\_InitHW : Initialize the hardware timer used for generating interrupts. The IAR PowerPac RTOS needs a timer-interrupt to determine when to activate tasks that wait for the expiration of a delay, when to call a software timer and to keep the time variable up to date.
- OS\_CREATETASK : Create a task.
- OS\_Q\_Create : Create and initialize a message queue.
- OS\_Q\_Put : Store a new message of given size in a queue.
- OS\_Q\_GetPtr : Retrieve a message from a queue.
- OS\_Q\_Purge : Delete the last retrieved message in a queue.
- OS\_EnterRegion : Indicate to the OS the beginning of a critical region.
- OS\_LeaveRegion : Indicate to the OS the end of a critical region.

## Tasks Description with Sample Code

### 1. Task0 - Init\_Task

- Functions
  - To create and define all the tasks
  - Create "Button Task"
  - Create "Buzzer Task"
  - Create "UART Task"
  - Create "LED Task"



- APIs and Flags
  - TCB\_Init\_Task : "Init Task" identifier value
  - TCB\_Button\_Task : "Button Task" identifier value
  - TCB\_Buzzer\_Task : "Buzzer Task" identifier value
  - TCB\_UART\_Task : "UART Task" identifier value
  - TCB\_LED\_Task : "LED Task" identifier value
  - Buzzer\_EVENT : "Buzzer Task" identifier value
  - UART\_EVENT : "UART Task" identifier value
  - LED\_EVENT : "LED Task" identifier value

```

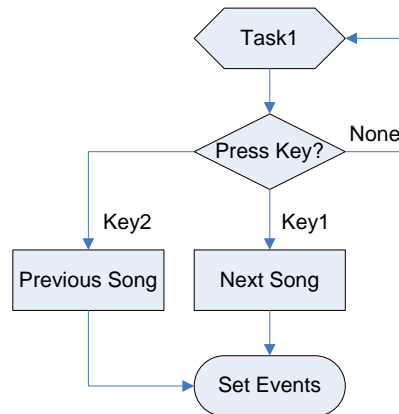
static void Init_Task(void)
{
    /* Create Events*/
    OS_EVENT_Create(&Buzzer_Event);
    OS_EVENT_Create(&UART_Event);
    OS_EVENT_Create(&LED_Event);

    /* Create Tasks */
    OS_CREATETASK(&TCB_Button_Task, "Button_Task", Button_Task, 50, Button_Task_stk);
    OS_CREATETASK(&TCB_Buzzer_Task, "Buzzer_Task", Buzzer_Task, 200, Buzzer_Task_stk);
    OS_CREATETASK(&TCB_UART_Task, "UART_Task", UART_Task, 150, UART_Task_stk);
    OS_CREATETASK(&TCB_LED_Task, "LED_Task", LED_Task, 100, LED_Task_stk);

    /*end this task */
    OS_Terminate(&TCB_Init_Task);
}
    
```

## 2. Task1 - Button\_Task

- Functions
  - Detect the key input depending on "ButtonNewStatus"
    - ButtonNewStatus = KEY\_KEY1  
Next song will be played
    - ButtonNewStatus = KEY\_KEY2  
Previous song will be played
  - If "ButtonNewStatus" has been changed, set the events of all other tasks.



- APIs and Flags
  - Get\_button\_state()
    - Get the current status of the key input.

```

static void Button_Task(void)
{
    SongMsg = 0;

    for (;;)
    {
        /* Get button state*/
        ButtonNewStatus = Get_Button_State();

        if(ButtonNewStatus != KEY_NONE)
        {
            Tone = 0; //Reset Tone

            /* Detect Button and select song number */
            if(ButtonNewStatus == KEY_KEY1) SongMsg++;
            else if(ButtonNewStatus == KEY_KEY2) SongMsg--;
            SongMsg = SongMsg % 4;

            /* Send Event : UART */
            Send_Event(EVENT_UART,&SongMsg);
            /* Send Event : LED */
            Send_Event(EVENT_LED,&SongMsg);
            /* Send Event : Buzzer */
            Send_Event(EVENT_BUZZER,&SongMsg);
        }
    }
}
    
```

### 3. Task2 - Buzzer\_Task

- Functions

- Drive the buzzer depending on the event "EVENT\_BUZZER"
- Select the song depending on the flag "SongMsg"

SongMsg = 0

Stop driving the buzzer.

SongMsg = 1

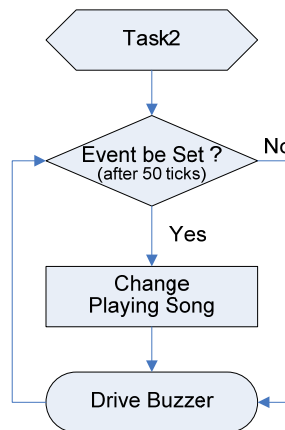
Stop : "Twinkle Twinkle Little Star"

SongMsg = 2

Stop : "Lightly Row"

SongMsg = 3

Stop : "Here Comes the Train"



- APIs and Flags

- Buzzer\_Output (u16 SongNum)

Select which song is to be played.

```

static void Buzzer_Task(void)
{
    for (;;)
    {
        /* Wait for Event messages */
        if(OS_EVENT_WaitTimed(&Buzzer_Event,5) == 0);

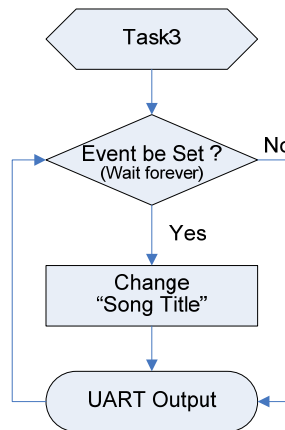
        OS_EnterRegion();

        /* Output song */
        Buzzer_Output(SongMsg);

        OS_LeaveRegion();
    }
}
  
```

#### 4. Task3 - UART\_Task

- Functions
  - Send the song title via the UART to the PC
    - SongMsg = 0
      - Stop sending data to the PC
    - SongMsg = 1
      - Title : "Twinkle Twinkle Little Star"
    - SongMsg = 2
      - Title : "Lightly Row"
    - SongMsg = 3
      - Title : "Here Comes the Train"



- APIs and Flags
  - UART\_Display (u16 OutputMode)
    - Select which song title is to be shown on the PC.

```

static void UART_Task(void)
{
  for (;;)
  {
    /*Receive Event for "UART" function*/
    OS_EVENT_Wait(&UART_Event);

    OS_EnterRegion();

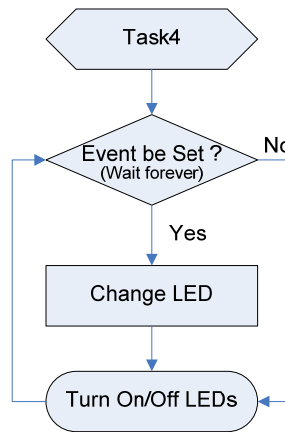
    /* Output song title*/
    UART_Display(SongMsg);

    OS_LeaveRegion();
  }
}

```

**5. Task4 - LED\_Task**

- Functions
  - Switch on/off the LEDs
    - SongMsg = 0  
The LEDs are all off.
    - SongMsg = 1  
LED1 is on - other LEDs are off.
    - SongMsg = 2  
LED2 is on - other LEDs are off.
    - SongMsg = 3  
LED3 is on - other LEDs are off.



- APIs and Flags
  - LED\_Control (u16 LEDMode)
    - Turn on the corresponding LED depending on the LEDMode.

```

static void LED_Task(void)
{
    for (;;)
    {
        /*Receive Event for "LED" function*/
        OS_EVENT_Wait(&LED_Event);

        OS_EnterRegion();

        /*Enable LED output*/
        LED_Control(SongMsg);

        OS_LeaveRegion();
    }
}
  
```

## CooCox CoOS

### CooCox CoOS Kernel

CooCox CoOS is a free and open Real-time Operating System, RTOS, combined with an adaptive task scheduling algorithm and support preemptive priority and round-robin. CoOS supports all ARM Cortex M3 based devices for which the source code can be downloaded from the official website.

### Example Programs

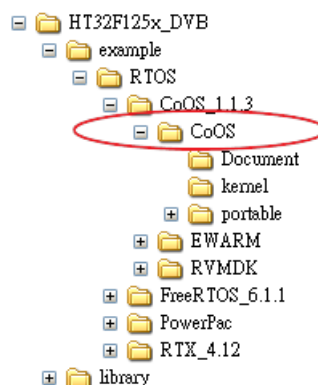
TheCooCox-CoOS RTOS Kit includes several example programs for several evaluation boards. The example programs are located on the CooCox's website: <http://www.coocox.org/CoOS.htm>.

Holtek provides example code to assist with the rapid development of embedded applications based on CoOS.

### Environment Setup

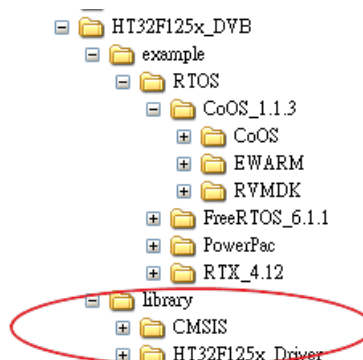
- Download and install the CoOS source code.
- Download the "Holtek RTOS Package" and extract the file into any known folder.
- Select "CoOS" under the working directory, and copy the "CoOS" folder from the "CoOS" source code installed path" to the working directory as shown below:

\HT32Fxxxx\example\RTOS\CoOS



- Ensure that the "firmware library" exists under the working directory

\HT32Fxxxx\library

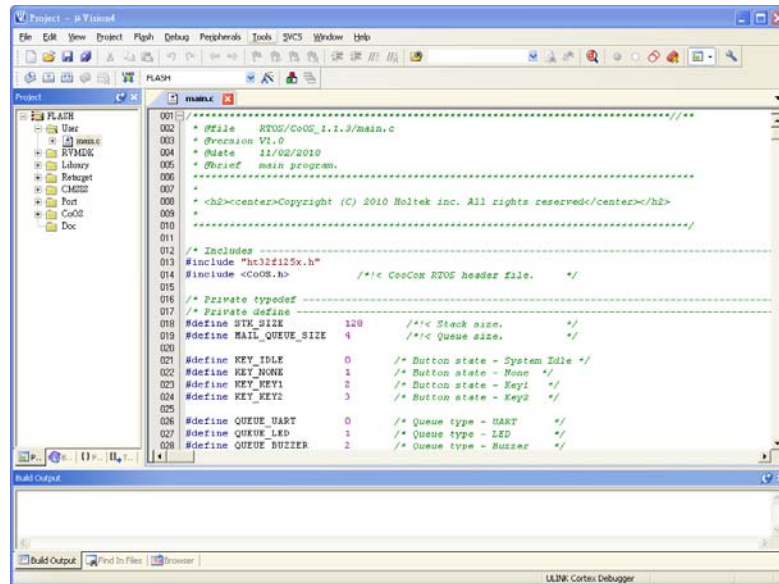


- Open the keil project file **xxxx.uvproj** as shown below:  
 \ HT32Fxxx\example\RTOS\CoOS \RVMDK
- The IAR project file **xxxx.eww** can also be found as shown below.  
 \ HT32Fxxx\example\RTOS\CoOS \EWARM

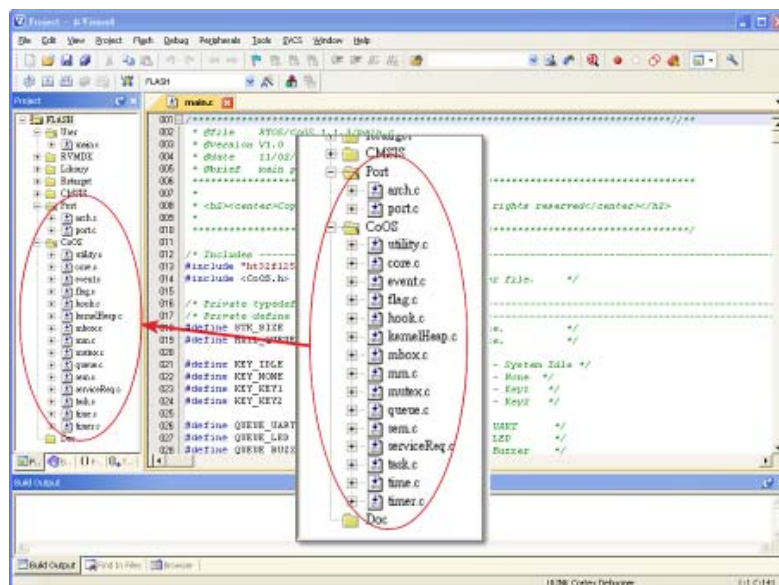
The Keil environment will also be introduced in this document.

### Software Setup

- Install the Keil RealView MDK-ARM. The updated evaluation version can be registered and downloaded at the Keil website.
- Open the RTOS Demo project.



- Add the "CoOS kernel" from "CoOS" to the project as shown below:  
 \ HT32Fxxx\example\RTOS\CoOS\CoOSkernel  
 \ HT32Fxxx\example\RTOS\CoOS\CoOSportable  
 \ HT32Fxxx\example\RTOS\CoOS\CoOSportable\Keil



- For more detailed information, refer to the "CooCox CoOS User's Guide".

## RTOS API

Below are some of the APIs that are used in the demo. For complete information, refer to

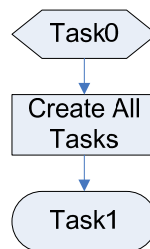
<http://www.coccox.org/CoOS.htm>

- ColnitOS : Initialise CoCox CoOS.
- CoStartOS : Start a OS multitask.
- CoCreateTask : Create a task.
- CoExitTask : Delete the current running task.
- CoCreateQueue : Create a message queue.
- CoPendQueueMail : Obtain a message from the message queue.

## Tasks Description with Sample Code

### 1. Task0 - Init\_Task

- Functions
  - To create and define all the tasks and queues
    - Create "Button Task"
    - Create "Buzzer Task"
    - Create "UART Task"
    - Create "LED Task"



- APIs and Flags
  - Init\_Task : "Init Task" identifier value
  - Button\_Task : "Button Task" identifier value
  - Buzzer\_Task : "Buzzer Task" identifier value
  - UART\_Task : "UART Task" identifier value
  - LED\_Task : "LED Task" identifier value
  - Buzzer\_Queue : "Buzzer Task" identifier value
  - UART\_Queue : "UART Task" identifier value
  - LED\_Queue : "LED Task" identifier value

```

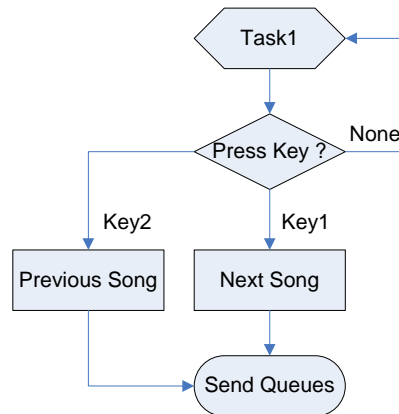
void Init_Task (void *pdata)
{
    /* Create four tasks for button, buzzer, UART, and LED function */
    CoCreateTask(Button_Task,0,BUTTON_TASK_PRIORITY,&Button_Task_stk[STK_SIZE-1],STK_SIZE);
    CoCreateTask(Buzzer_Task,0,BUZZER_TASK_PRIORITY,&Buzzer_Task_stk[STK_SIZE-1],STK_SIZE);
    CoCreateTask(UART_Task,0,UART_TASK_PRIORITY,&UART_Task_stk[STK_SIZE-1],STK_SIZE);
    CoCreateTask(LED_Task,0,LED_TASK_PRIORITY,&LED_Task_stk[STK_SIZE-1],STK_SIZE);

    /* Create three queues for buzzer, UART, and LED function */
    UART_Queue = CoCreateQueue(UARTQueue,MAIL_QUEUE_SIZE,EVENT_SORT_TYPE_FIFO);
    LED_Queue = CoCreateQueue(LEDQueue,MAIL_QUEUE_SIZE,EVENT_SORT_TYPE_FIFO);
    Buzzer_Queue = CoCreateQueue(BuzzerQueue,MAIL_QUEUE_SIZE,EVENT_SORT_TYPE_FIFO);

    /* Delete "init_task" task. */
    CoExitTask();
}
    
```

## 2. Task1 - Button\_Task

- Functions
  - Detect the key input depending on "ButtonNewStatus"
    - ButtonNewStatus = KEY\_KEY1  
Next song will be played.
    - ButtonNewStatus = KEY\_KEY2  
Previous song will be played.
  - If "ButtonNewStatus" has been changed, set the events of all other tasks.



- APIs and Flags
  - Get\_button\_state()
    - Get the current status of the key input.
  - Send\_Queue (u8 QueueType, u16\* QueueMessage)
    - Send queues that include "key state" to other tasks.

```

void Button_Task (void *pdata)
{
    SongMsg = 0;

    for (;;)
    {
        /* Get button state*/
        ButtonNewStatus = Get_Button_State();

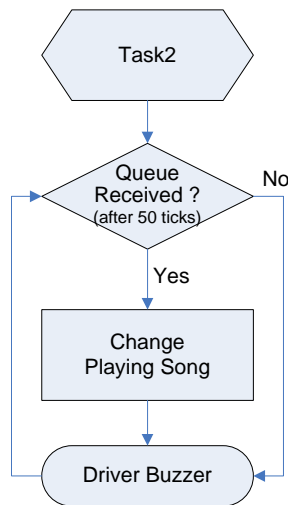
        if (ButtonNewStatus != KEY_NONE)
        {
            Tone = 0;                                     //Reset Tone

            /* Detect Button and select song number */
            if (ButtonNewStatus == KEY_KEY1) SongMsg++;
            else if (ButtonNewStatus == KEY_KEY2) SongMsg--;
            SongMsg = SongMsg % 4;

            /* Send Queue : UART */
            Send_Queue (QUEUE_UART, &SongMsg);
            /* Send Queue : LED */
            Send_Queue (QUEUE_LED, &SongMsg);
            /* Send Queue : Buzzer */
            Send_Queue (QUEUE_BUZZER, &SongMsg);
        }
    }
}
    
```

### 3. Task2 - Buzzer\_Task

- Functions
  - Drive the buzzer depending on the event "EVENT\_BUZZER"
  - Select a song depending on the flag "BuzzerMsg"
    - BuzzerMsg = 0  
Stop driving the buzzer.
    - BuzzerMsg = 1  
Song : "Twinkle Twinkle Little Star"
    - BuzzerMsg = 2  
Song : "Lightly Row"
    - BuzzerMsg = 3  
Song : "Here Comes the Train"



- APIs and Flags
  - Buzzer\_Output (u16 SongNum)  
Select which song is to be played.

```

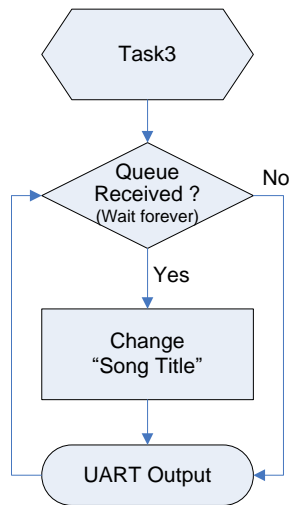
void Buzzer_Task (void *pdata)
{
    StatusType result;
    u16 *BuzzerMsg;

    for (;;)
    {
        /*Receive Queue for "buzzer" function*/
        BuzzerMsg = CoPendQueueMail(Buzzer_Queue,50,&result);
        if( result != E_OK)
        {
            if( result == E_INVALID_ID) printf("Invalid Queue ID:Buzzer\n");
        }
        else ;

        /* Output song */
        Buzzer_Output(*BuzzerMsg);
    }
}
    
```

#### 4. Task3 - UART\_Task

- Functions
  - Send the song title via the UART to the PC
    - UARTMsg = 0
      - Stop sending data to the PC.
    - UARTMsg = 1
      - Title : "Twinkle Twinkle Little Star"
    - UARTMsg = 2
      - Title : "Lightly Row"
    - UARTMsg = 3
      - Title : "Here Comes the Train"



- APIs and Flags
  - UART\_Display ( u16 OutputMode)
    - Select which song title is to be shown on the PC.

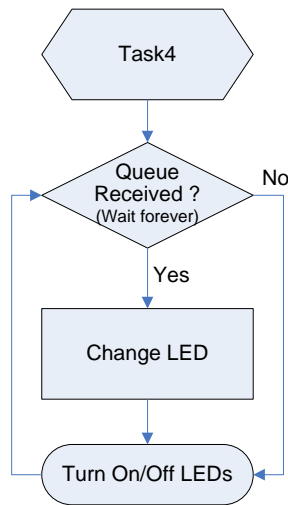
```

void UART_Task (void *pdata)
{
    StatusType result;
    u16 *UARTMsg;

    for (;;)
    {
        /*Receive Queue for "UART" function*/
        UARTMsg = CoPendQueueMail(UART_Queue,0,&result);
        if( result != E_OK)
        {
            if( result == E_INVALID_ID) printf("Invalid Queue ID:UART\n");
        }
        else
        {
            /* Output song title*/
            UART_Display(*UARTMsg);
        }
    }
}
    
```

**5. Task4 - LED\_Task**

- Functions
  - Switch on/off the LEDs
    - LEDMsg = 0  
The LEDs are all off.
    - LEDMsg = 1  
LED1 is on - other LEDs are off.
    - LEDMsg = 2  
LED2 is on - other LEDs are off.
    - LEDMsg = 3  
LED3 is on - other LEDs are off.



- APIs and Flags
  - LED\_Control (u16 LEDMode)
    - Turn on the corresponding LED depending on the LEDMode.

```

void LED_Task (void *pdata)
{
    StatusType result;
    u16 *LEDMsg;

    for (;;)
    {
        /*Receive Queue for "LED" function*/
        LEDMsg = CoPendQueueMail(LED_Queue,0,&result);
        if( result != E_OK)
        {
            if( result == E_INVALID_ID) printf("Invalid Queue ID :LED\n");
        }
        else
        {
            /*Enable LED output*/
            LED_Control(*LEDMsg);
        }
    }
}
    
```