**Porting FreeRTOS to SiM3U167 Cortex M3**

RTOS is just an application which can execute difference function base on different trigger conditions, something like system ticket, interrupt, and message queue.etc. And also need protect some share resource would be accessed by difference function, or we call it task at one time.

For Arm Cortex M3, there two popular RTOS, one it uC/OSII, the other one is FreeRtos. We pickup FreeRTOS as our target since it free and open source. The poring procedure is pretty easy, let start it.

1. Get Latest FreeRtos 7.3.0 source code.

Go to <http://www.freertos.org/>, and download it to your PC, extract it. We can see directory structure like below.

\FreeRTOS

\FreeRTOS-Plus

\readme.txt

Just go through the directory, there are a lot of difference platform support codes. We need Cortex M3 and GCC. So here are the resources we need.

\FreeRTOS\Source\\*.\*

\FreeRTOS\Source\include\\*.\*

\FreeRTOS\Source\portable\GCC\ARM\CM3\\*.\*

\FreeRTOS\Source\portable\MemMang\heap\_4.c

And one head file needed, FreeRTOSConfig.h

I just choose \FreeRTOS\Demo\CORTEX\_LM3S811\_GCC\ FreeRTOSConfig.h, and copy it into \FreeRTOS\Source\include\

1. Prepare our RTOS directory.
2. Created a directory “FreeRTOSV7.3.0”.
3. Copy a Blinky example code as code base and copy startup file in it.
4. Copy FreeRTOS directory into this directory.
5. The directory structure now looks like this

\FreeRTOS\Source\\*.\*

\FreeRTOS\Source\include\\*.\*

\FreeRTOS\Source\portable\GCC\ARM\CM3\\*.\*

\FreeRTOS\Source\portable\MemMang\heap\_4.c

\src\\*.\*

\src\generated\\*.\*

1. Precistion32 environment setting and interrupt handler functions modification.

Include paths looks like this:

"${workspace\_loc:/sim3u1xx\_FreeRtos/src}"

"${workspace\_loc:/sim3u1xx\_FreeRtos/src/generated}"

"${workspace\_loc:/sim3u1xx\_FreeRtos/FreeRTOS/Source/include}"

"${workspace\_loc:/sim3u1xx\_FreeRtos/FreeRTOS/Source/portable/GCC/ARM\_CM3}"

"${SI32\_PATH}/si32Hal/sim3u1xx"

"${SI32\_PATH}/si32Hal/SI32\_Modules"

"${SI32\_PATH}/si32Hal/CPU"

Open start up file src\startup\_sim3u1xx\_p32.c, changed three handler functions. In vector table “void (\* const g\_pfnVectors[])(void)”,

1. Change SVCall handler to “vPortSVCHandler”
2. Change PendSV handler to “xPortPendSVHandler”
3. Change SysTick handler to “xPortSysTickHandler”
4. Added task test code in main.c

**void** **task1**(**void** \* parameters)

{

uint32\_t count = 0;

**const** portCHAR \*pcPassMessage = "PASS";

**while**(1)

{

**if**(SI32\_PBSTD\_A\_read\_pin(SI32\_PBSTD\_2, 8)==0)

{

**printf**("Task1 running %d\n",count++);

xQueueSend( xLEDQueue, &pcPassMessage, portMAX\_DELAY );

}

}

}

**void** **task2**(**void** \* parameters)

{

uint32\_t count = 0;

**while**(1)

{

**printf**("Task2 running %d\n",count++);

vTaskDelay( 1000 );

}

}

**void** **task3**(**void** \* parameters)

{

uint32\_t count = 0;

portCHAR \*pcMessage;

**while**(1)

{

/\* Wait for a message to arrive. \*/

xQueueReceive( xLEDQueue, &pcMessage, portMAX\_DELAY );

SI32\_PBSTD\_A\_toggle\_pins(SI32\_PBSTD\_2,0xc00);

**printf**("Task3 running %d %s\n",count++,pcMessage);

}

}

**int** **main**()

{

**while**(SI32\_PBSTD\_A\_read\_pin(SI32\_PBSTD\_2, 8) == 0);

// Enter the default operating mode for this application (gModes.c)

gModes\_enter\_my\_default\_mode();

// Print a starting message

**printf**("hello world\n");

/\* Create the queue used to pass message to vPrintTask. \*/

xLEDQueue = xQueueCreate( 3, **sizeof**( portCHAR \* ) );

xTaskCreate( task1, "task1", 200, NULL, 1, NULL );

xTaskCreate( task2, "task2", 200, NULL, 2, NULL );

xTaskCreate( task3, "task3", 200, NULL, 2, NULL );

/\* Start the scheduler running the tasks and co-routines just created. \*/

vTaskStartScheduler();

**while**(1);

// Loop forever...

}

1. Run the code,

We can task output from serial view window.

hello world

Task2 running 0

Task2 running 1

If we press SW2(PB2.8), we can see task1 and task3 active and output message

Task1 running 0

Task3 running 0 PASS

Task1 running 1

Task3 running 1 PASS

1. Source code

We can get source code from http://github.org/MarkDing/sim3u1xx\_FreeRtos