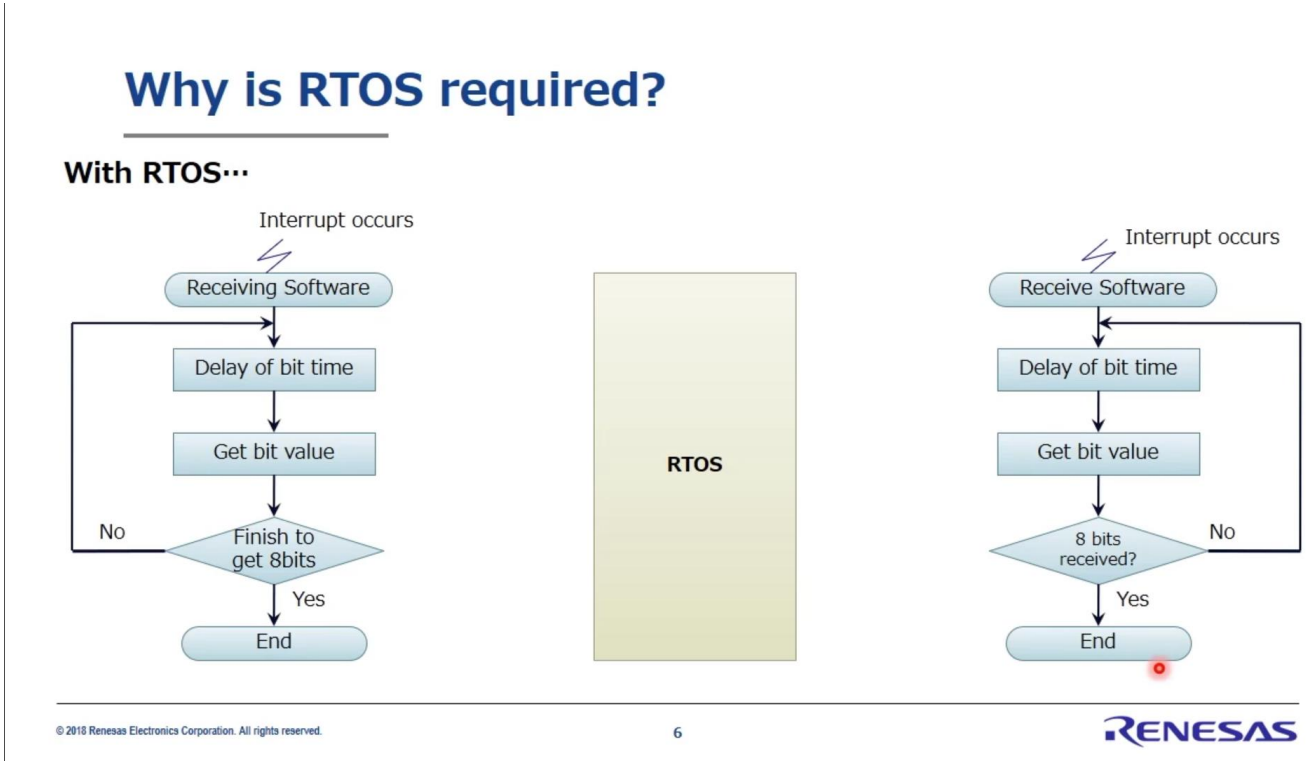
임베디드 시스템

최 민

학습열기

- 다음 **RTOS**에 관한 동영상을 학습하세요.
- **https://www.youtube.com/watch?v=ECEvUEkSSLg**



목차

- 학습목표
  - 타이머와 RTOS
- 학습활동
  - ARM lpc1768의 타이머 사용방법을 이해합니다.
  - ARM lpc1768에서 RTOS를 사용하는 방법을 이해합니다.
  - RTOS가 지원하는 다양한 API 활용에 대하여 학습합니다.

Timer

- 특징
  - 32비트 프리스케일러를 내장한 32비트 타이머/카운터
  - 카운터 또는 타이머로 동작
  - 32비트 Capture 채널 : 각 타이머별로 입력 신호가 변하는 순간의 타이머 값을 저장할 수 있는 레지스터(2 개).
  - 32비트 Match Register : 타이머 일치 동작을 설정, 모든 타이머 일치 동작에서 타이머 값 일치 시 인터럽트를 발생가능(4개)
    - 1) 타이머값 일치 후에도 계속되는 타이머 동작
    - 2) 타이머값 일치 시 정지하는 타이머 동작
    - 3) 타이머값 일치 시 리셋되는 타이머 동작
- 관련 구조체 및 함수의 정의
  - Timer관련 구조체와 함수들은 drivers/include/lpc17xx_timer.c와 drivers/source/lpc17xx_timer.h에 정의되어 있다

Timer

- **Timer 0~3 설정 레지스터**
  - Interrupt Register
  - Timer Control Register
  - Timer Counter
  - Prescale Register
  - Prescale Counter
  - Match Control Register
  - Match Register 0 ~ 3
  - Capture Control Register
  - Capture Register 0 ~ 1

Generic Name	Description	Access	Reset Value[1]	TIMERn Register/ Name & Address
IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	R/W	0	T0IR - 0x4000 4000 T1IR - 0x4000 8000 T2IR - 0x4009 0000 T3IR - 0x4009 4000
TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0	T0TCR - 0x4000 4004 T1TCR - 0x4000 8004 T2TCR - 0x4009 0004 T3TCR - 0x4009 4004
TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	R/W	0	T0TC - 0x4000 4008 T1TC - 0x4000 8008 T2TC - 0x4009 0008 T3TC - 0x4009 4008
PR	Prescale Register. The Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	R/W	0	T0PR - 0x4000 400C T1PR - 0x4000 800C T2PR - 0x4009 000C T3PR - 0x4009 400C
PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	R/W	0	T0PC - 0x4000 4010 T1PC - 0x4000 8010 T2PC - 0x4009 0010 T3PC - 0x4009 4010
MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0	T0MCR - 0x4000 4014 T1MCR - 0x4000 8014 T2MCR - 0x4009 0014 T3MCR - 0x4009 4014
MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0	T0MR0 - 0x4000 4018 T1MR0 - 0x4000 8018 T2MR0 - 0x4009 0018 T3MR0 - 0x4009 4018
MR1	Match Register 1. See MR0 description.	R/W	0	T0MR1 - 0x4000 401C T1MR1 - 0x4000 801C T2MR1 - 0x4009 001C T3MR1 - 0x4009 401C
MR2	Match Register 2. See MR0 description.	R/W	0	T0MR2 - 0x4000 4020 T1MR2 - 0x4000 8020 T2MR2 - 0x4009 0020

Timer

- **IR (Interrupt Register)**
  - 각 채널에서 인터럽트 발생시 대응하는 비트가 set됨.
  - Bit clear 방법 : 해당하는 IR bit에 1값을 write 하면 interrupt reset 됨

Interrupt Register (T[0/1/2/3]IR - addresses 0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000) bit description

Bit	Symbol	Description	Reset Value
0	MR0 Interrupt	Interrupt flag for match channel 0.	0
1	MR1 Interrupt	Interrupt flag for match channel 1.	0
2	MR2 Interrupt	Interrupt flag for match channel 2.	0
3	MR3 Interrupt	Interrupt flag for match channel 3.	0
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.	0
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.	0
31:6	-	Reserved	-

Timer

- 네 개의 **32-Bit Match Registers**
  - 타이머는 계속 동작하면서 인터럽트를 발생시키도록 만들 수 있다.
  - 타이머를 정지시키면서 인터럽트를 발생시키도록 만들 수 있다.
  - 타이머를 리셋 하면서 인터럽트를 발생시키도록 만들 수 있다.
- **Match Registers**에 해당하는 **External Outputs**이 최대 **4** 개
  - Set Low On Match : 타이머값 일치 시 Low 출력으로 설정
  - Set High On Match : 타이머값 일치 시 High 출력으로 설정
  - Toggle On Match : 타이머값 일치 시 출력 값을 반전하도록 설정
  - Do Nothing On Match : 타이머값 일치 시 출력 없음

Timer

- **TCR (Timer Control Register)**
  - Timer/Counter의 operation을 제어하기 위한 레지스터
  - Enable, Reset 동작

Table 427. Timer Control Register (TCR, TIMERn: TnTCR - addresses 0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004) bit description

Bit	Symbol	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When 1, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

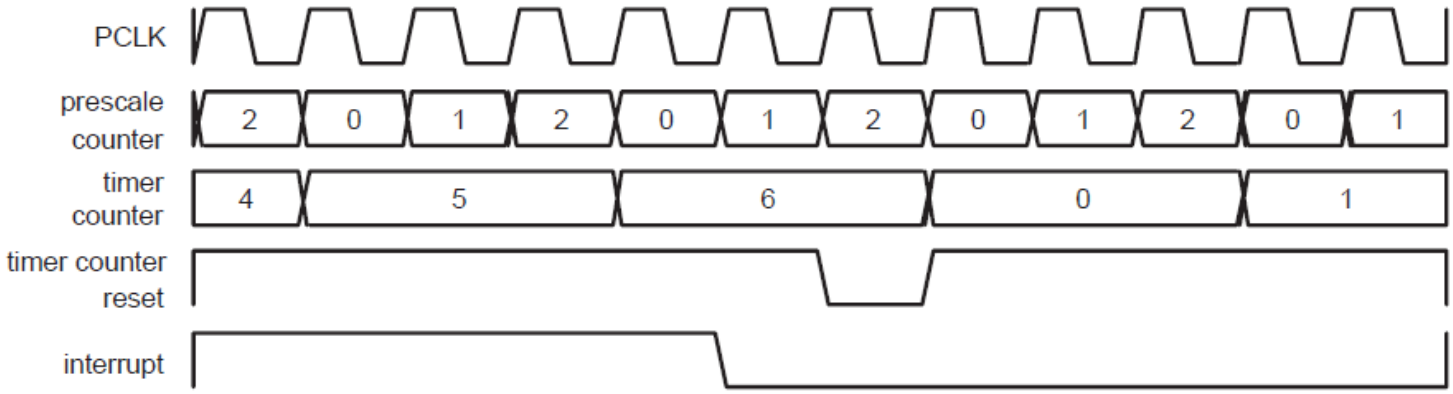
Timer

- **Count Control Register**
  - Timer Mode, Counter Mode 설정
  - Rising/falling edge, 또는 모두에서 count할 것인지 결정

Bit	Symbol	Value	Description	Reset Value
1:0	Counter/ Timer Mode		This field selects which rising PCLK edges can increment the Timer's Prescale Counter (PC), or clear the PC and increment the Timer Counter (TC).	00
		00	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register. The Prescale Counter is incremented on every rising PCLK edge.	
		01	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		10	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		11	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.	

Timer

- **Prescale Register**
  - Prescale Counter에 대한 최대 값 설정
- **Prescale Counter Register**
  - Prescale Counter는 매 PCLK 클럭 펄스 당 1값이 증가함
  - Prescale Counter Register의 값이 Prescale Register 값과 같아지면 Timer Counter 레지스터에 1값이 증가하며, Prescale Counter는 다음번 PCLK에서 reset됨.



A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.

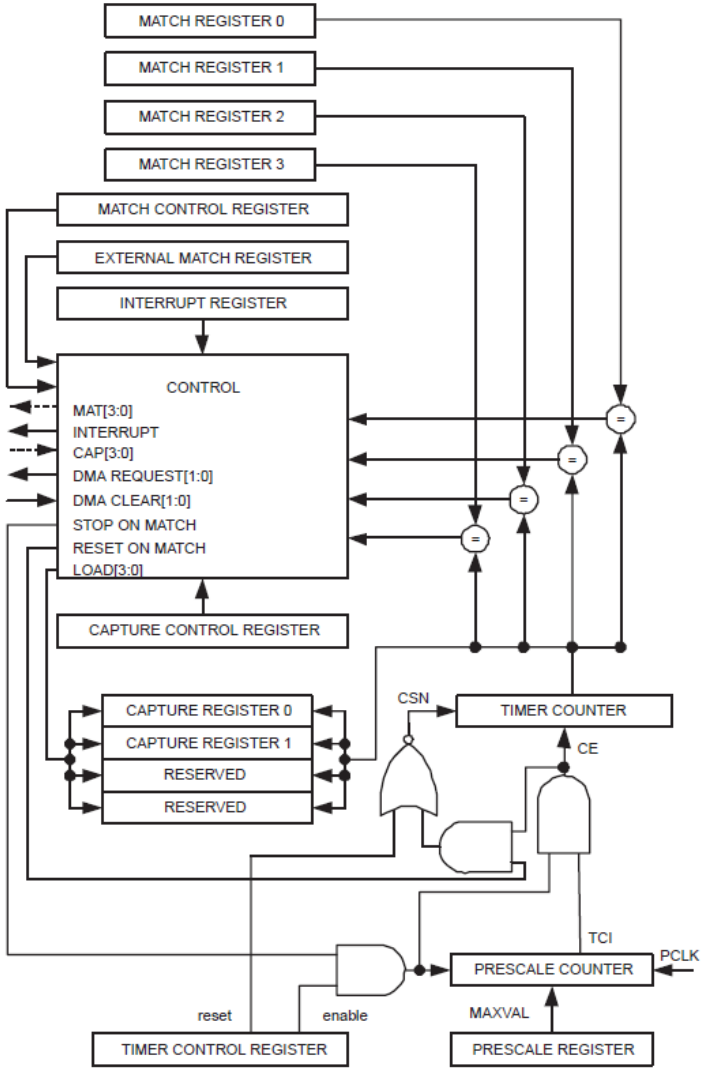
Timer

- **Match Register**
  - Timer Counter 레지스터의 값과 계속적으로 비교되며, 값이 일치할 경우 action(인터럽트, TC reset, timer 정지 등)이 trigger됨.
- **Match Control Register**
  - 여러 개 중 하나의 Match 레지스터와 Timer Counter와의 값이 일치할 때 어떠한 동작을 할 것인지 설정

Bit	Symbol	Value	Description	Reset Value
0	MR0I	1	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		0	This interrupt is disabled	
1	MR0R	1	Reset on MR0: the TC will be reset if MR0 matches it.	0
		0	Feature disabled.	
2	MR0S	1	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		0	Feature disabled.	
3	MR1I	1	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		0	This interrupt is disabled	
4	MR1R	1	Reset on MR1: the TC will be reset if MR1 matches it.	0
		0	Feature disabled.	
5	MR1S	1	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		0	Feature disabled.	
6	MR2I	1	Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0

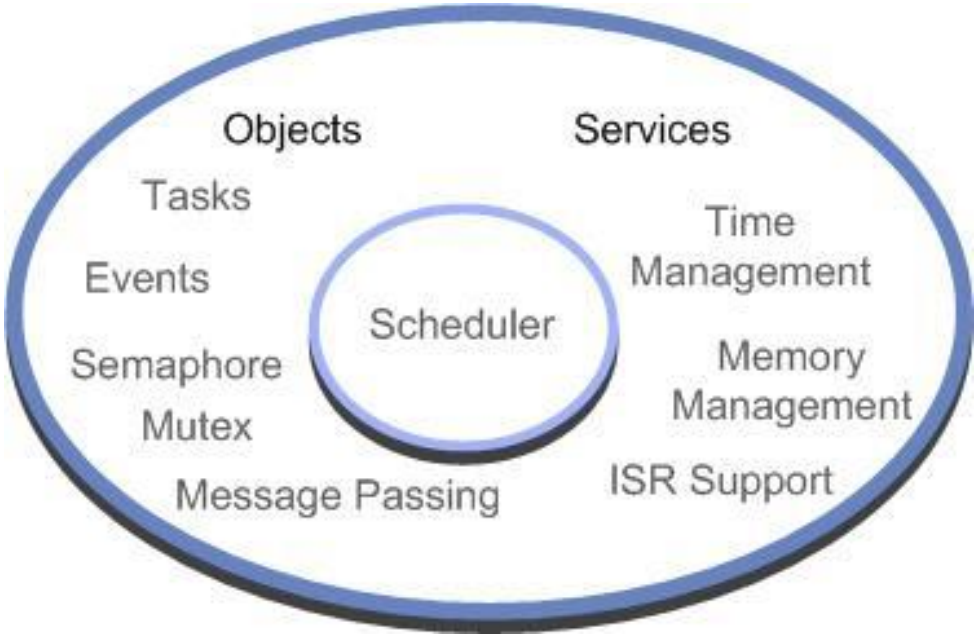
Timer

- **Timer Block Diagram**



RTOS

- **CMSIS-RTOS**
  - RTOS itself consists of a scheduler which supports round-robin, pre-emptive and co-operative multitasking of program threads, as well as time and memory management services



RTOS

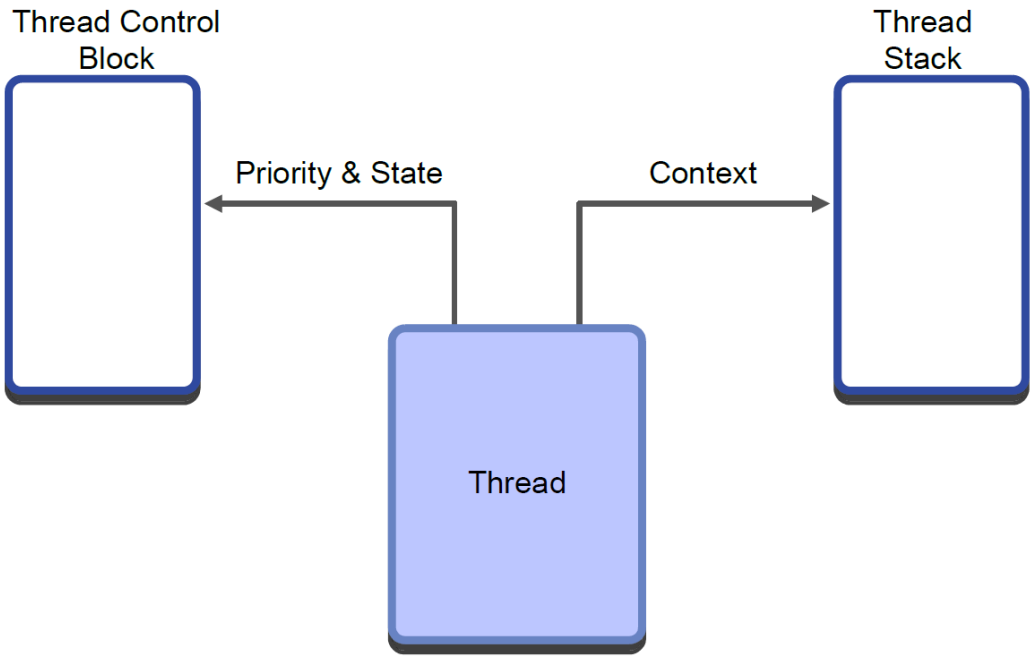
- To access any of the CMSIS-RTOS features in our application code it is necessary to include the following header file
- #include <cmsis_os.h>
- This header file is maintained by ARM as part of the CMSIS-RTOS standard. For the CMSIS-RTOS Keil RTX this is the default API

CMSIS-RTOS Priority Levels

osPriorityIdle
osPriorityLow
osPriorityBelowNormal
osPriorityNormal
osPriorityAboveNormal
osPriorityHigh
osPriorityRealTime
osPriorityError

RTOS 쓰레드

- **Each thread has its own stack for saving its data during a context switch. The thread control block is used by the kernel to manage the active thread**



RTOS 쓰레드

- Main()함수가 실행될 때 기본적으로 CMSIS-RTOS 스케쥴러가 동작
  - main() 함수는 첫번째 active thread임
  - osKernelInitialize() 함수 호출에 의해서 스케쥴러 task를 switching 할 수 있음
  - Init_Thread() 함수에서 추가적인 thread 생성 및 RTOS object 생성
  - osKernelStart() 함수를 실행함으로써 RTOS scheduler를 재시작

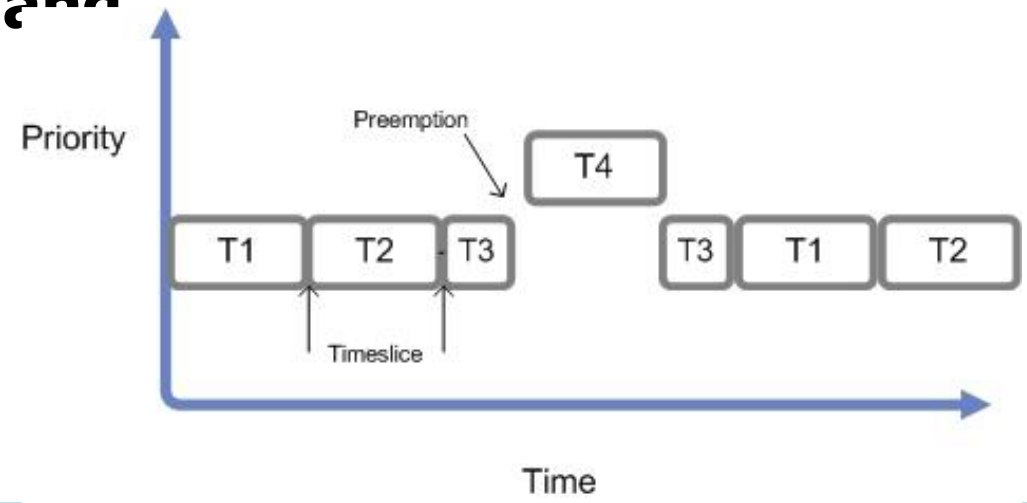
상태	설명
Running	실행중인 스레드 상태
Ready	실행이 준비된 스레드 상태
Wait	특정 Event를 대기중인 Blocked Threads 상태

RTOS 쓰레드

```
17  int main (void) {
18
19    SystemInit(); //system initialization
20
21    GPIO_SetDir(1, 0xB0000000, 1); //Set 28,29,31 of GPIO1 as output
22    GPIO_SetDir(2, 0x0000007C, 1); //Set 2,3,4,5,6 of GPIO2 as output
23
24    GPIO_ClearValue(1, 0xB0000000); //Set 28,29,31 of GPIO1 off : LED1,2,3 off
25    GPIO_ClearValue(2, 0x0000007C); //Set 2,3,4,5,6 of GPIO2 off : LED4,5,6,7,8 off
26
27    osKernelInitialize ();                    // initialize CMSIS-RTOS
28    Init_Thread ();
29    osKernelStart ();                         // start thread execution
30
31  }
```

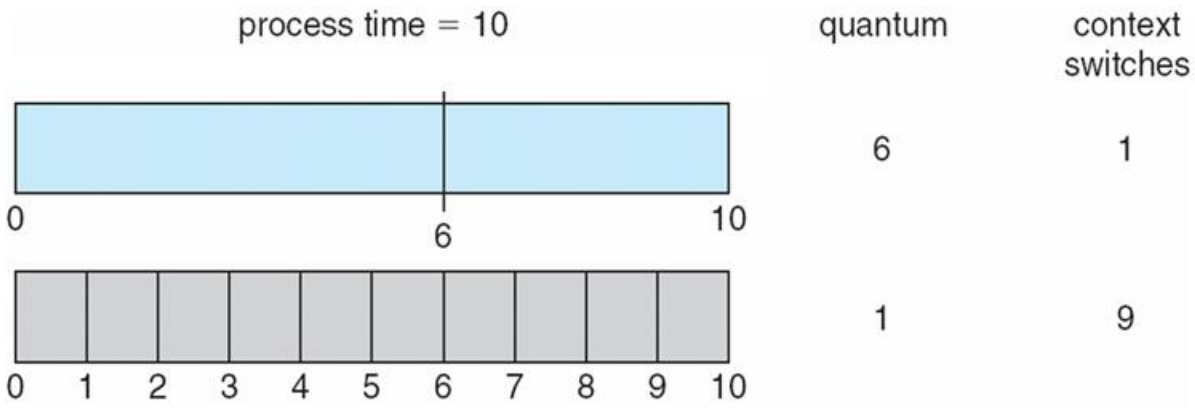
RTOS 쓰레드

- 쓰레드가 생성될 때 **default priority** 를 할당
- **Ready** 쓰레드가 여러 개이고, 모두 같은 **priority**를 가질 경우, 이들은 **round-robin fashion**으로 실행됨
- **Threads of equal priority will be scheduled in a round-robin fashion. High priority tasks will pre-empt low priority tasks and enter the running state 'on dema**



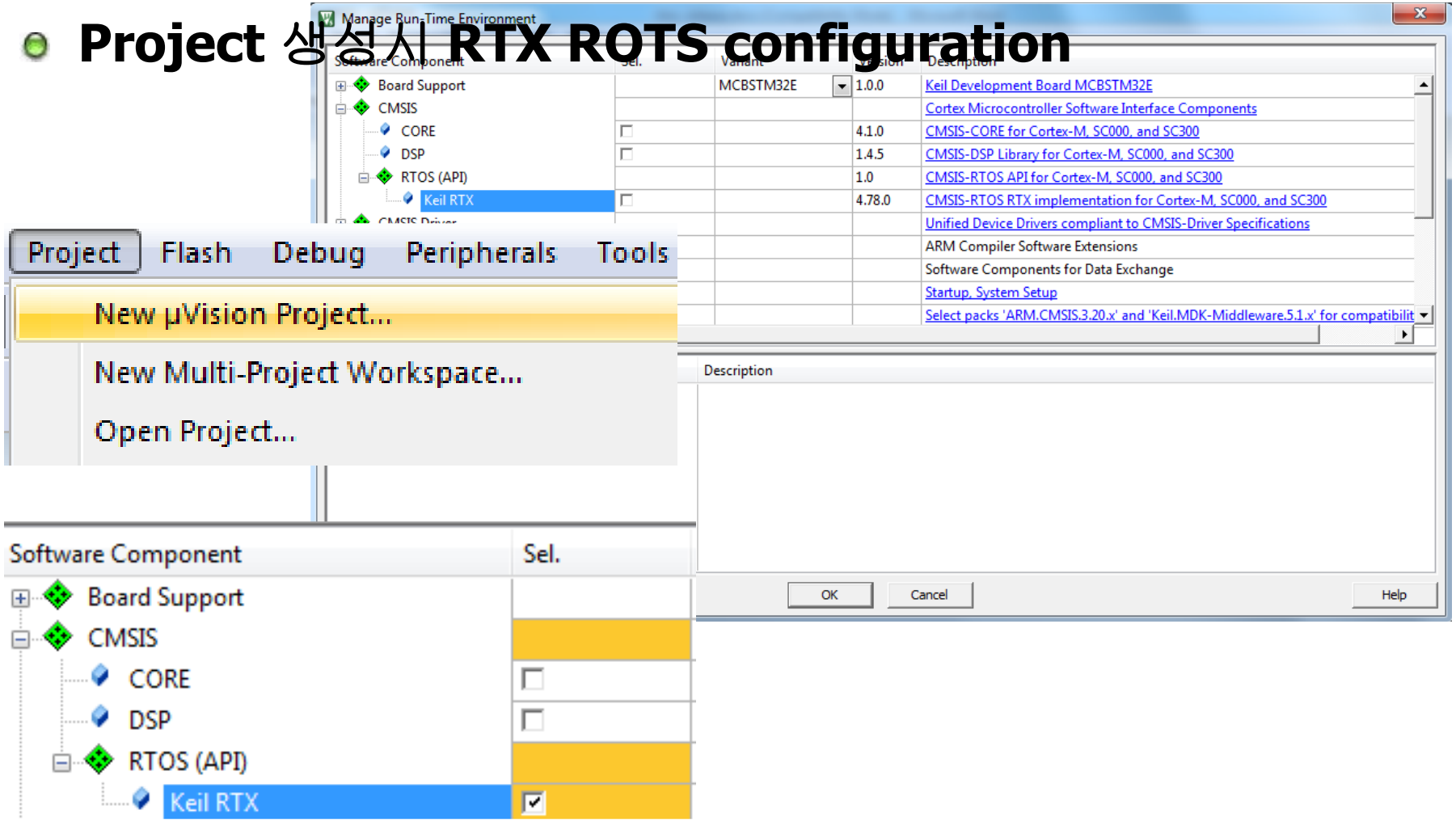
Round Robin

- **Round Robin 스케쥴링**
  - 각 프로세스가 작은 단위 CPU 시간을 가지고, 이 시간동안 프로세스를 실행함, 이러한 단위 CPU 시간은 time quantum이라 불리우고, 약 10~100milliseconds (시스템에 따라 다름)
  - 이 시간이 경과하면 실행중인 프로세스가 preemption 되고 ready queue의 가장 끝에 추가됨
  - Ready queue에 n개 proces가 존재하고 time quantum이 q라면, 이 프로세스가 다음번 스케쥴될때까지 최대 대기 시간은 (n-1)q 이내임



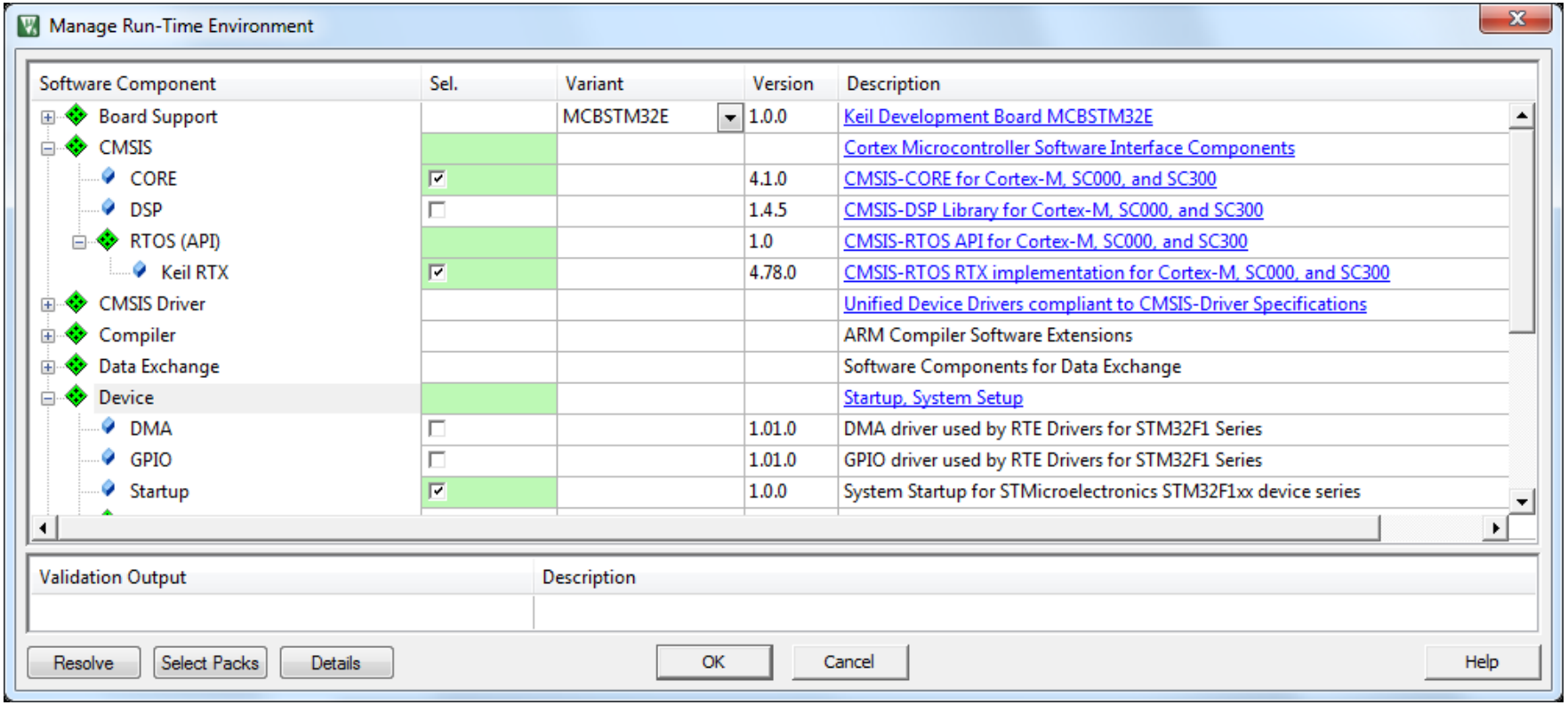
RTOS 프로젝트 설정

- **Project 생성시 RTX ROTS configuration**



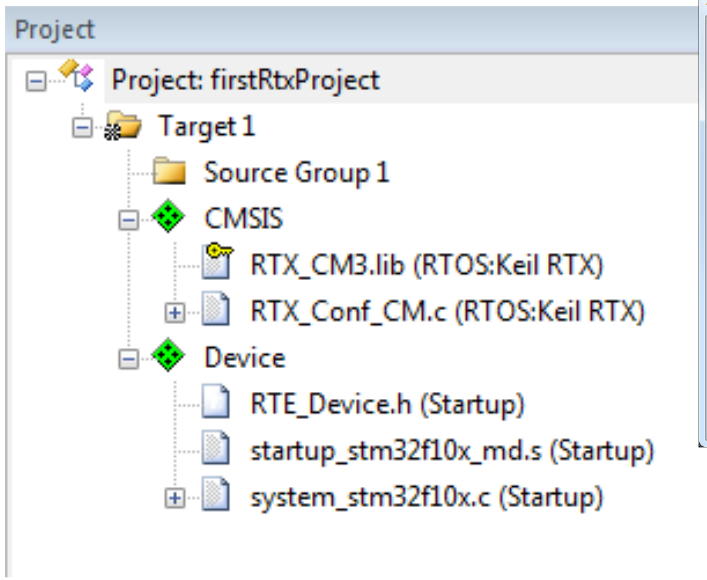
RTOS 프로젝트 설정

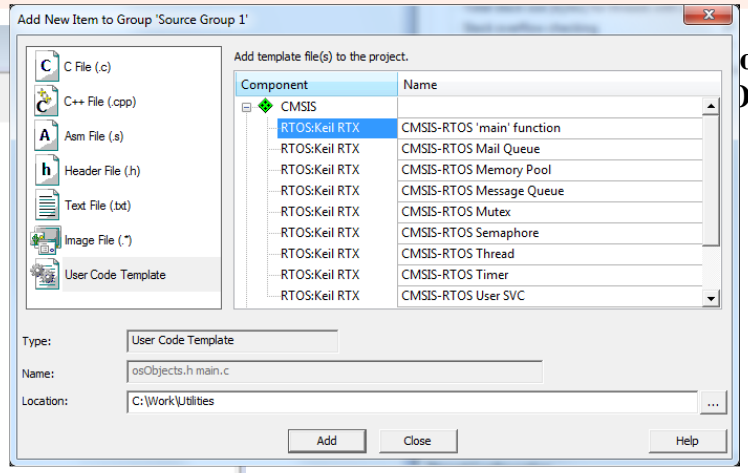
- To add the missing components you can press the "Resolve" button in the bottom left hand corner of the RTE



RTOS 프로젝트 설정

파일	설명
Startup_STM32F10x_md.s	벡터테이블을 포함한 어셈블리 코드
System_STM32F10x.c	시스템 및 주변장치 초기화 코드, Clock Tree, PLL, 외부 메모리 인터페이스 등에 관한 설정
RTE_Device.h	Configures the pin multiplex
RTX_Conf_CM.c	Configures Keil RTX





ode template' Icon and
)S 'main' function'

RTOS 다중 쓰레드(busy waiting)

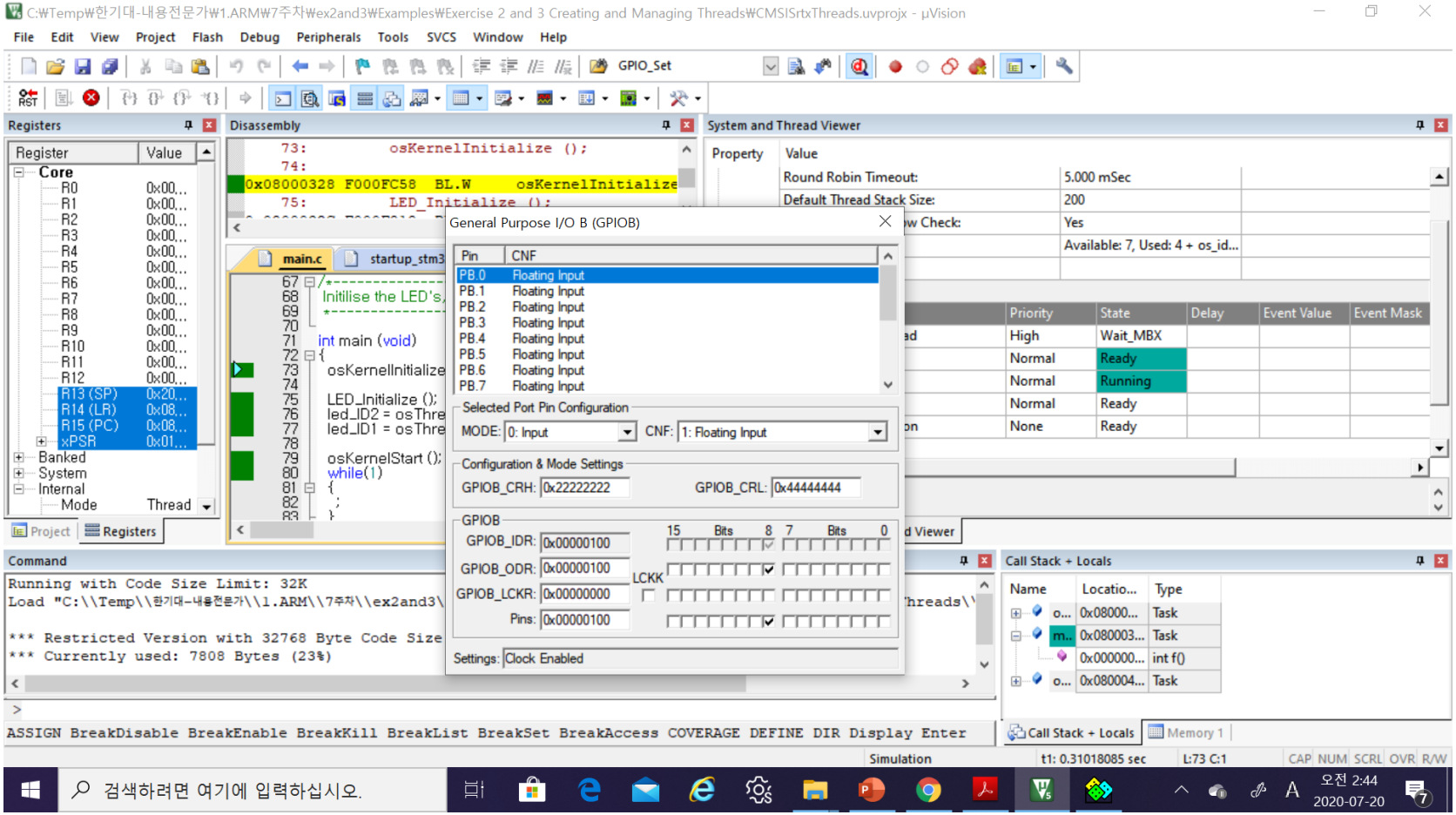
```
63  osThreadId main_ID,led_ID1,led_ID2;
64  osThreadDef(led_thread2, osPriorityNormal, 1, 0);
65  osThreadDef(led_thread1, osPriorityNormal, 1, 0);
```

```
71  int main (void)
72  {
73    osKernelInitialize ();                  // initialize CMSIS-RTOS
74
75    LED_Initialize ();
76    led_ID2 = osThreadCreate(osThread(led_thread2), NULL);
77    led_ID1 = osThreadCreate(osThread(led_thread1), NULL);
78
79    osKernelStart ();                       // start thread execution
80    while(1)
81    {
82      ;
83    }
84  }
```

```
33  void led_thread1 (void const *argument)
34  {
35
36    for (;;)
37    {
38      LED_On(0);
39      delay(500);
40      LED_Off(0);
41      delay(500);
42    }
43  }
44
45  void led_thread2 (void const *argument)
46  {
47    for (;;)
48    {
49      LED_On(2);
50      delay(500);
51      LED_Off(2);
52      delay(500);
53    }
54  }
```

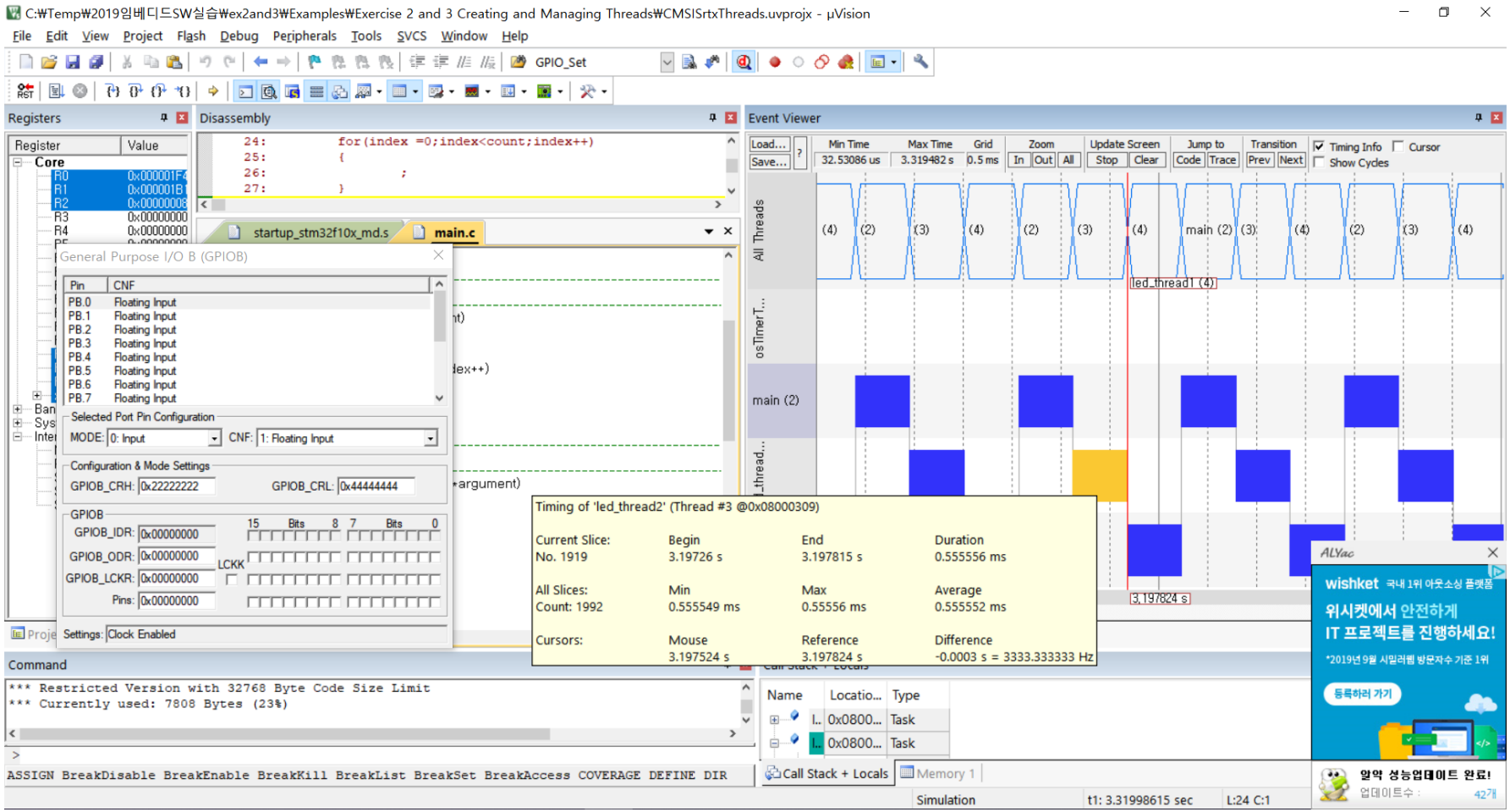
RTOS 다중 쓰레드

- 실행 결과



RTOS 다중 쓰레드

- 실행 결과

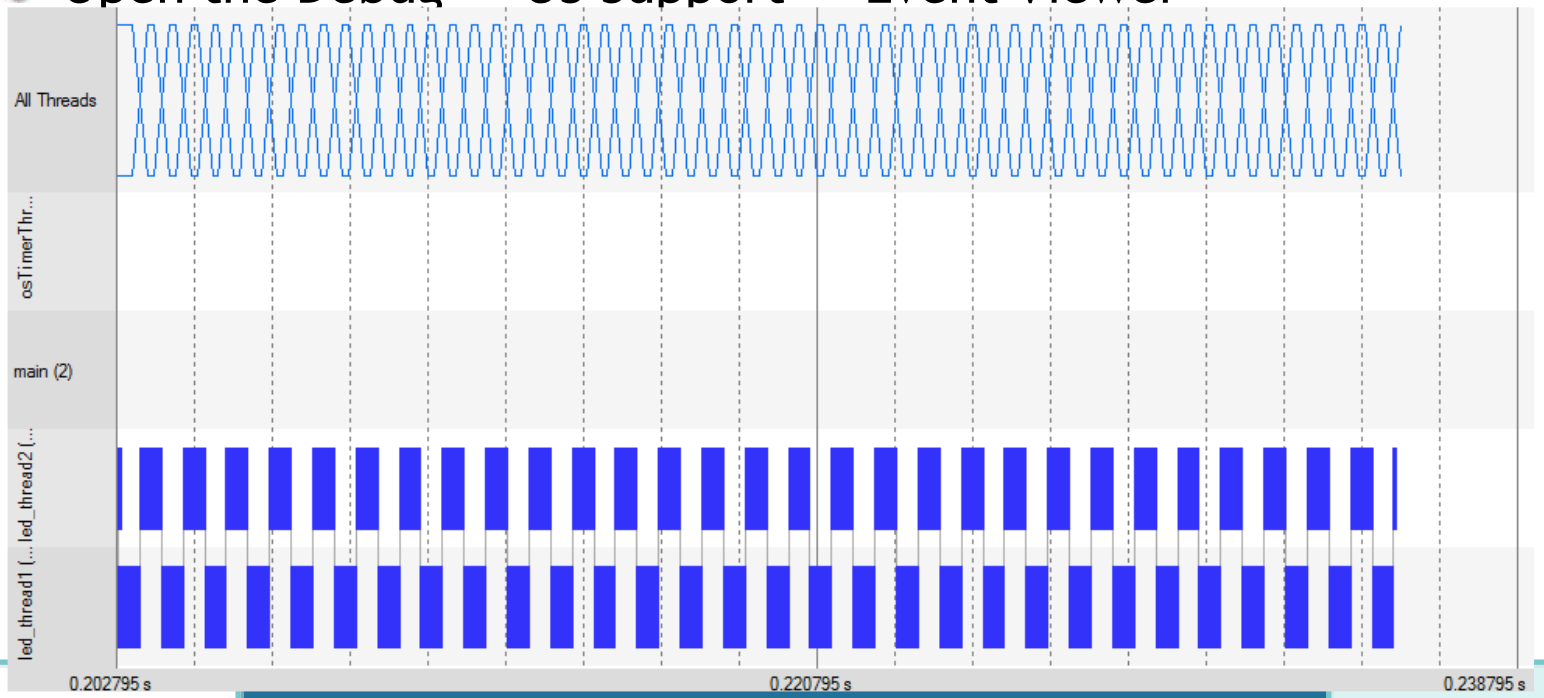


RTOS 다중 쓰레드

- Debug -> OS Support -> System and Thread Viewer

ID	Name	Priority	State	Delay	Event Value	Event Mask	Stack Usage
1	osTimerThread	High	Wait_MBX				32%
3	led_thread2	Normal	Running				0%
4	led_thread1	Normal	Ready				32%
255	os_idle_demon	None	Ready				

- Open the Debug -> OS Support -> Event Viewer



RTOS 다중 쓰레드

- **Go back to the project "Ex 2 and 3 Threads" Change the priority of LED Thread 2 to Above Normal**
  - osThreadDef(led_thread2, osPriorityAboveNormal, 1, 0);
  - osThreadDef(led_thread1, osPriorityNormal, 1, 0);

- **Led_thread1 is running at normal priority and led_thread2 is running at a higher priority so has pre-empted led_thread1.**
  - To make it even worse led_thread2 never blocks so it will run forever preventing the lower priority thread from ever running

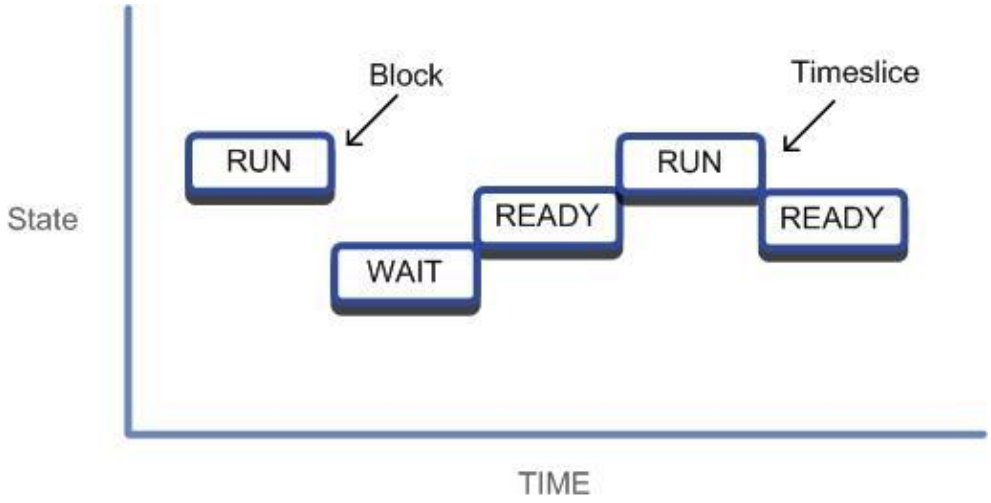
RTOS 다중 쓰레드 (blocked wait)

- **This is our original led flasher program but the simple delay function has been replaced by the osDelay API call. LED2 is toggled every 100mS and LED1 is toggled every 500mS**

- void **osDelay** (uint32_t millisec)
  - This call will place the calling thread into the WAIT_DELAY state for the specified number of milliseconds. The scheduler will pass execution to the next thread in the READY state

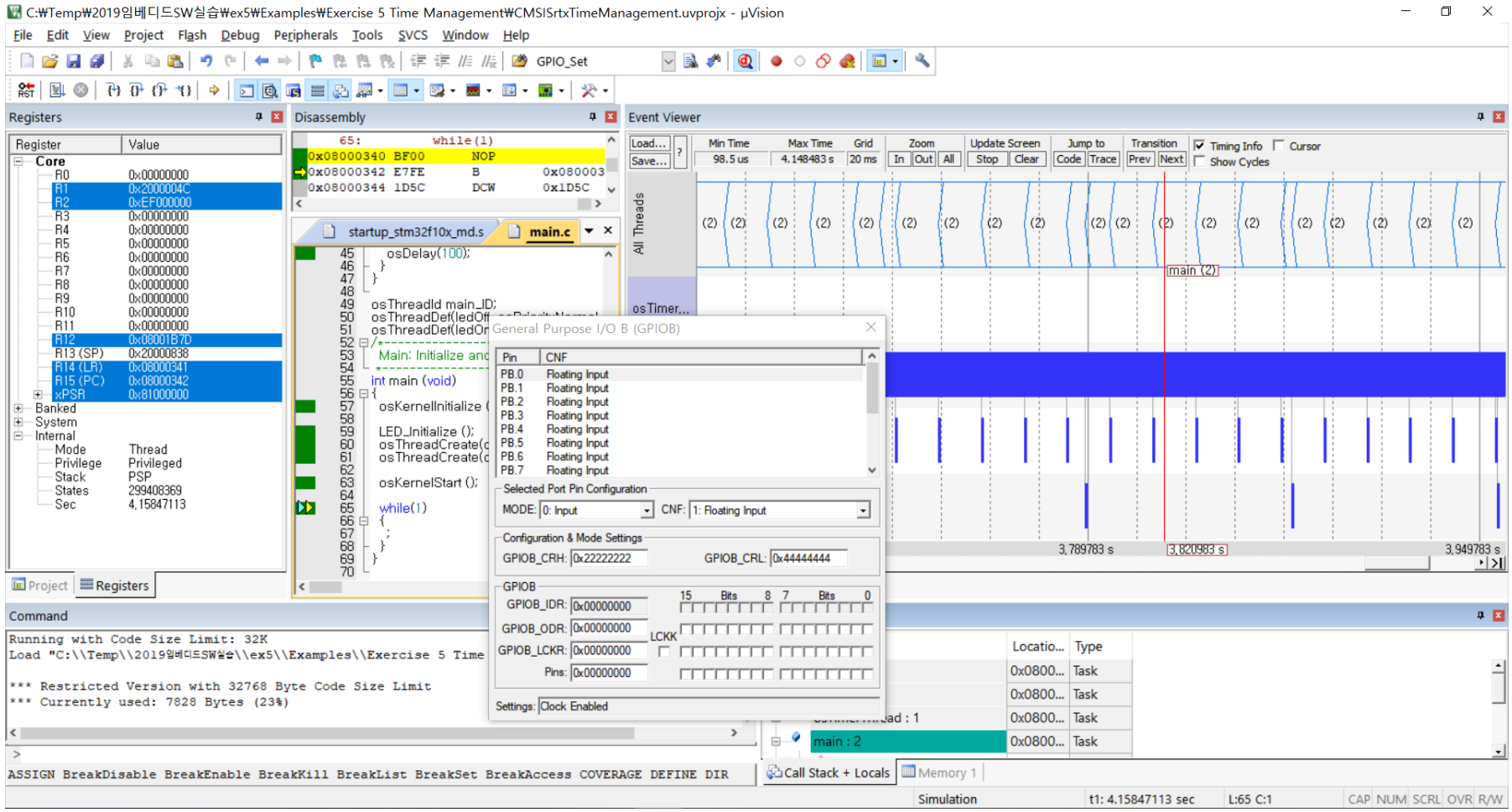
RTOS 다중쓰레드(blocked wait)

- **During their lifetime threads move through many states. Here a running thread is blocked by an osDelay call so it enters a wait state. When the delay expires, it moves to ready. The scheduler will place it in the run state. If its timeslice expires, it will move back to ready.**



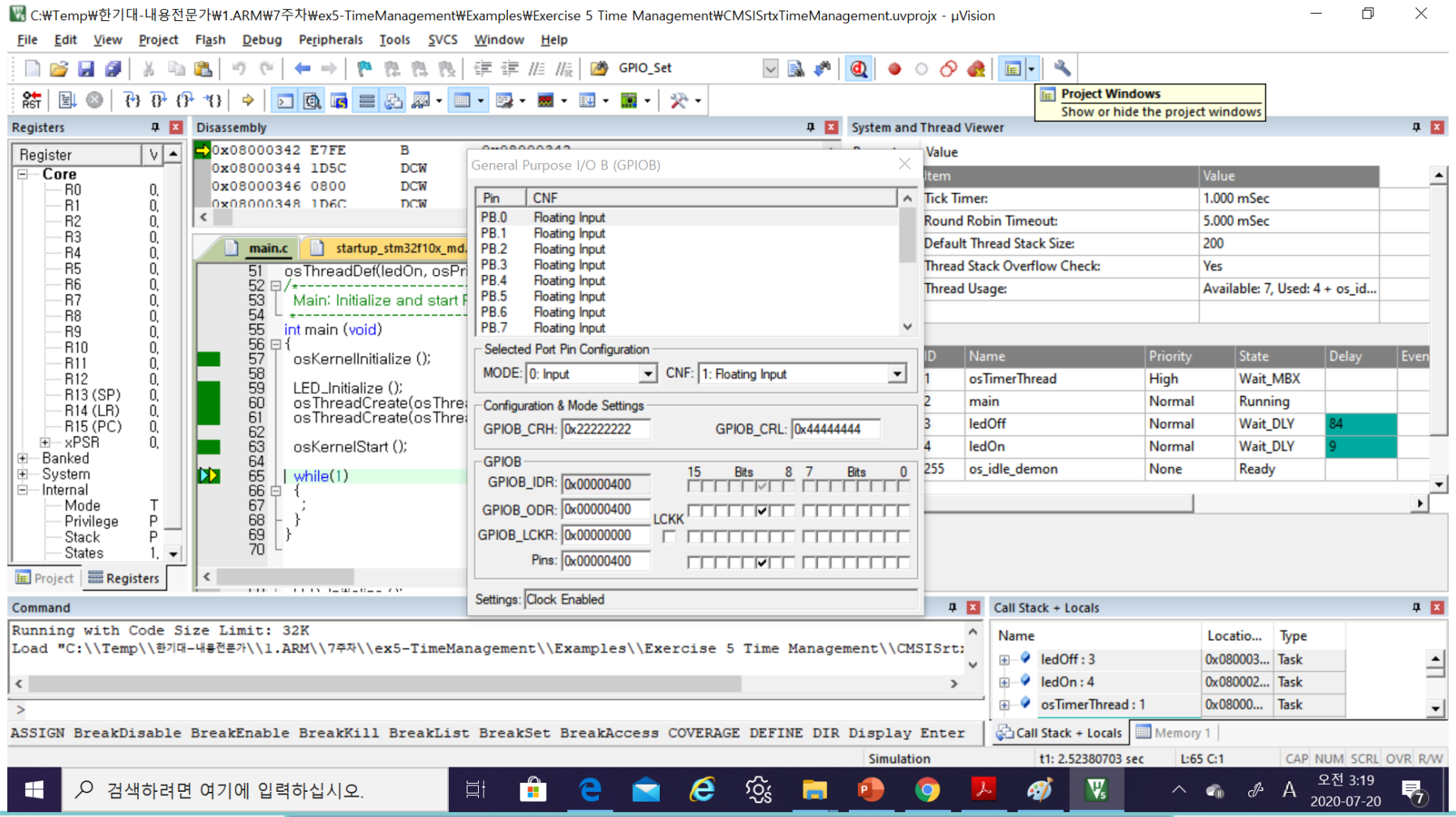
RTOS 다중 쓰레드 (blocked wait)

- 실행결과



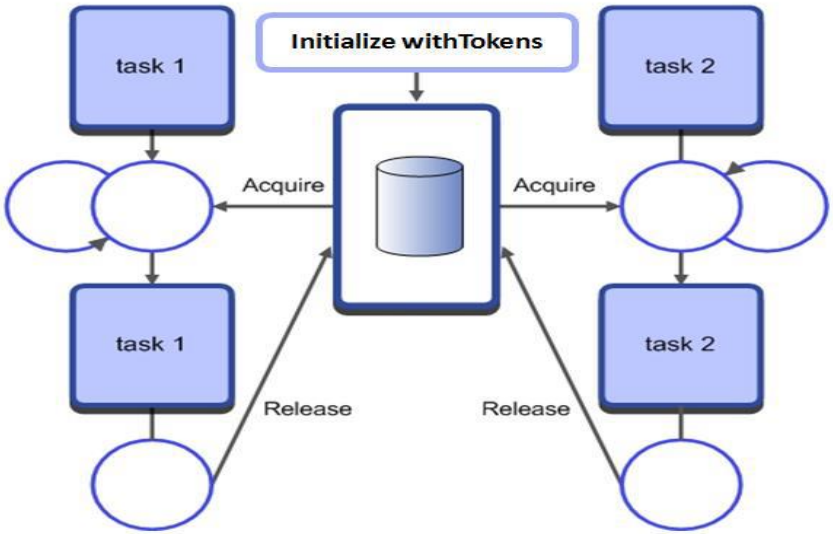
RTOS 다중쓰레드(blocked wait)

- 실행 결과



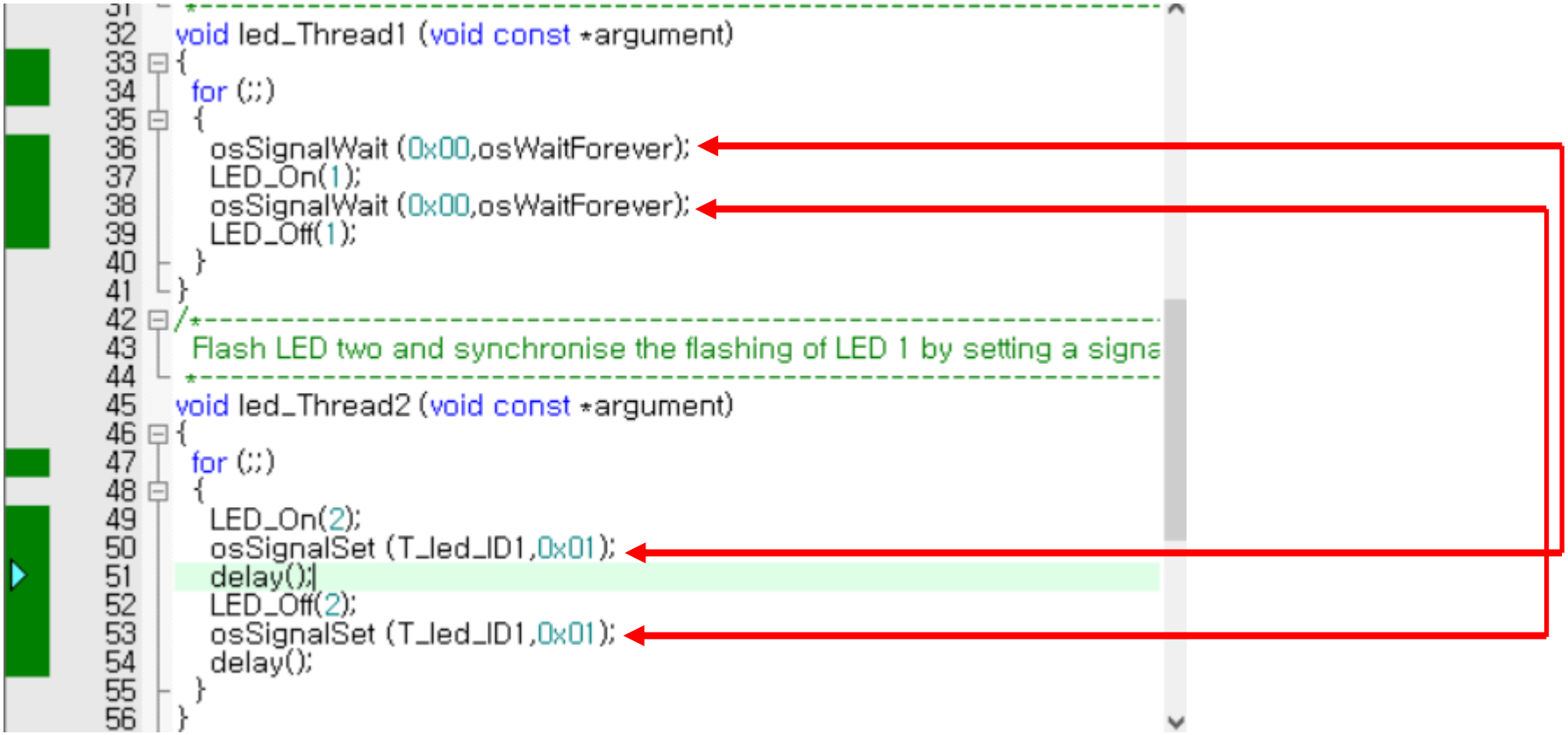
RTOS 세마포어

- semaphores are a method of synchronizing activity between two or more threads. Put simply, a semaphore is a container that holds a number of tokens.
  - Semaphores help to control access to program resources. Before a thread can access a resource, it must acquire a token. If none is available, it waits. When it is finished with the resource, it must return the token.



RTOS 쓰레드에서 동기화

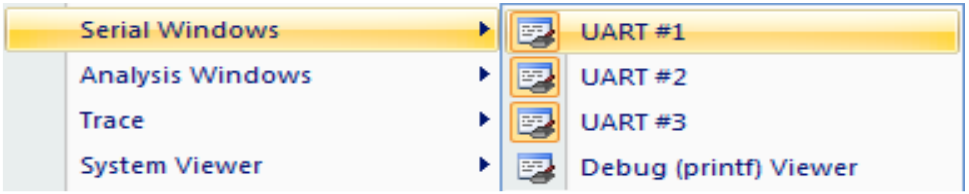
- **osSignalWait()** 호출을 통한 **mutex**

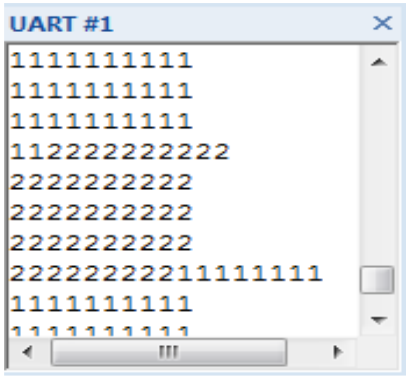


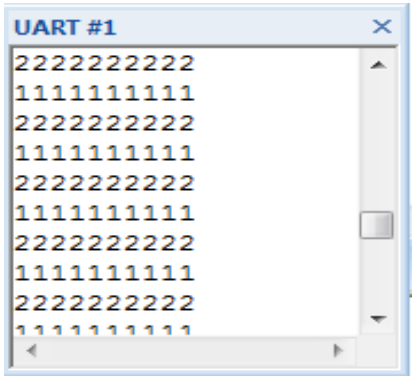
```
32  void led_Thread1 (void const *argument)
33  {
34    for (;;)
35    {
36      osSignalWait (0x00,osWaitForever);
37      LED_On(1);
38      osSignalWait (0x00,osWaitForever);
39      LED_Off(1);
40    }
41  }
42  /*----------------------------------------------------------
43    Flash LED two and synchronise the flashing of LED 1 by setting a signa
44   *----------------------------------------------------------*/
45  void led_Thread2 (void const *argument)
46  {
47    for (;;)
48    {
49      LED_On(2);
50      osSignalSet (T_led_ID1,0x01);
51      delay();
52      LED_Off(2);
53      osSignalSet (T_led_ID1,0x01);
54      delay();
55    }
56  }
```

RTOS 쓰레드에서 동기화

- **This project declares two threads which both write blocks of characters to the UART. Initially, the mutex is commented out.**







문제풀기

	문제	보기	정답	해설	난이도	관련 학습목표
1	다음은 ARM LPC1768 의 Timer 관련 레지스터에 해당하지 않는 것은?	1) PC 2) TC 3) TR 4) MR.	3)	TCR (Timer Control Register)는 있으나, TR 레지스터는 존재하지 않습니다	하	Timer
2	Prescale Counter Registe(PCR)의 값이 Prescale Register(PR)의 값과 동일해지면 어떠한 상황이 발생하는가?	1) 인터럽트가 발생한다 2) Timer가 정지한다. 3) Timer Counter 레지스터 값이 1증가 한다 4) 다음번 PCLK 클럭에서 PCR 값이 1 증가한다	3)	Timer Counter 레지스터 값이 1 증가합니다.	상	Timer
3	다음 중 Match Register output에 대한 설명이 잘못된 것은?	1) Set Low On Match : 타이머값 일치 시 Low 출력으로 설정 2) Set High On Match : 타이머값 일치 시 High 출력으로 설정 3) Toggle On Match : 타이머값 일치 시 출력값이 1로 설정 4) Do Nothing On Match : 타이머값 일치 시 출력 없음	3)	Toggle On Match 에서는 타이머값 일치시 출력값을 현재의 반대값으로 반전(toggle)합니다.	중	Timer

정리하기

- **ARM 프로세서 Timer**
  - 32비트 프리스케일러를 내장한 32비트 타이머/카운터
  - 32비트 Capture 채널 : 각 타이머별로 입력 신호가 변하는 순간의 타이머 값을 저장할 수 있는 레지스터(2 개).
  - 32비트 Match Register : 타이머 일치 동작을 설정, 모든 타이머 일치 동작에서 타이머 값 일치 시 인터럽트를 발생가능(4개)
  - Interrupt Register, Timer Control Register, Timer Counter, Prescale Register, Match Control Register 등
- **ARM 프로세서에서 활용가능한 실시간 운영체제**
  - VxWorks, FreeRTOS, uC-OS, CMSIS-RTOS 등
  - 쓰레드, busy-waiting, blocked waiting, mutex, semaphore 제공