RTX přepínací strategie (Scheduling Options)

- **Pre-emptive scheduling**
  Každý task má **různou prioritu** a běží až ho přeruší task s vyšší prioritou nebo zablokuje volání funkce OS.
- **Round-Robin scheduling**
  Každý task má **stejnou prioritu** a běží fixní periodu nebo ho zablokuje volaní funkce OS.
- **Co-operative multi-tasking**
  Každý task má **stejnou prioritu** a Round-Robin je zakázán. Task zablokuje volaní OS nebo funkce **os_tsk_pass()**.

Pre-emptive Scheduling

Tasky s vyšší prioritou suspendují tasky s nižší prioritou. Potom RTX přepne v následujících případech :

*-task scheduler je volán funkcí OS **tick timer interrupt**. Task scheduler aktivuje **delays** funkce .Jestlí-že delay funkce vyprší ,potom task s vyšší prioritou startuje, místo aktuálního tasku.*

*-událost (vlajka, příznak) **event** je nastavena na task s vyšší prioritou než aktuálně běžící task, task je suspendován a vysokoprioritní task jde do stavu run.*

*-token je vrácen OS funkci **semaphore** a vysokoprioritní task čeká na token před semaforem. Aktuálně běžící task je suspendován a vysokoprioritní task jde do stavu run. Token je pčedán aktuálnímu tasku nebo interrupt service programu.*

*- **mutex** je uvolněn and a a vysokoprioritní task čeká na token před mutexem. Aktuálně běžící task je suspendován a vysokoprioritní task jde do stavu run.*

*-zpráva byla zaslána do **mailbox** a vysokoprioritní task čeká na mailbox zprávu. Aktuálně běžící task je suspendován a vysokoprioritní task jde do stavu run. Zprávu může vyslat aktuálně ě běžící task nebo interrupt service routine.*

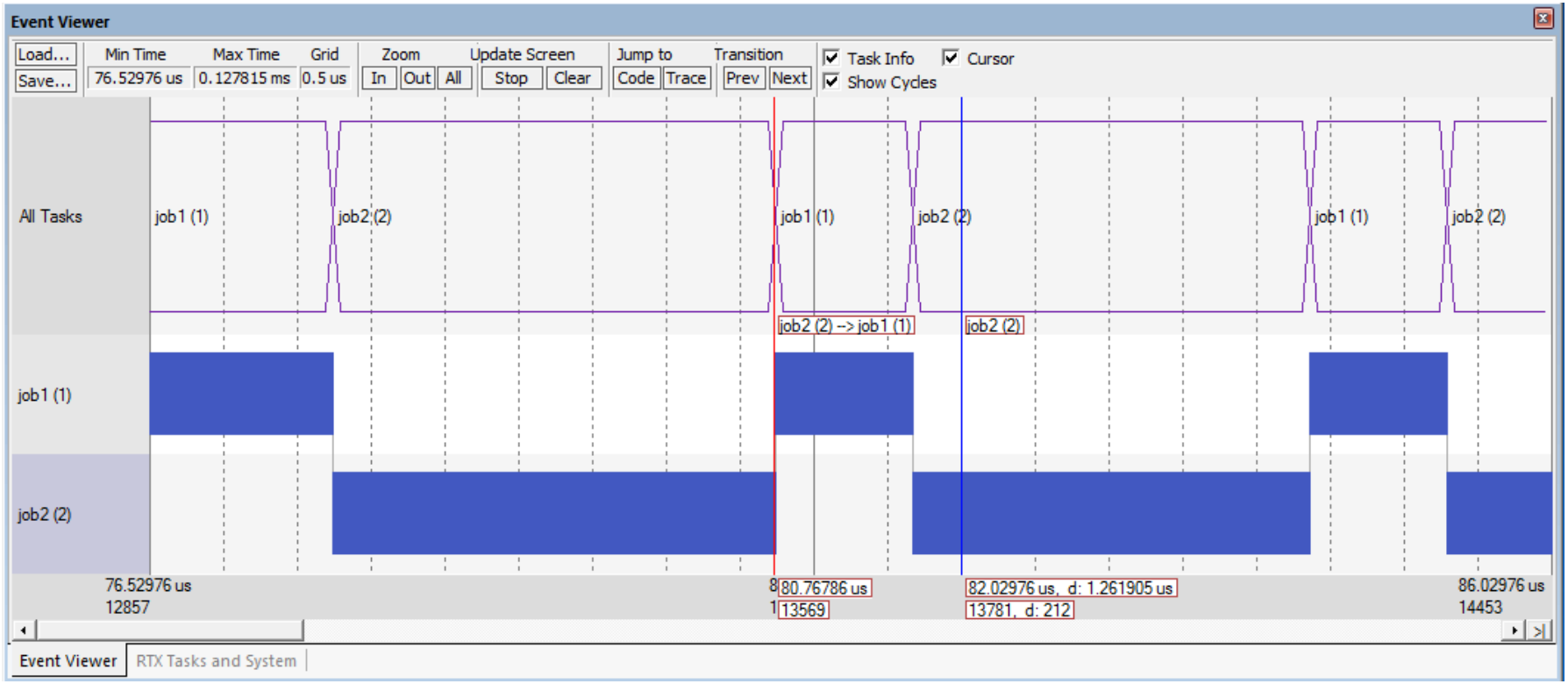
*- **priorita** aktuálně běžícího tasku je redukována. Jestliže další task je ve stavu ready a má vyšší prioritu, aktuálně běžící task je suspendován a vysokoprioritní task jde do stavu run.*

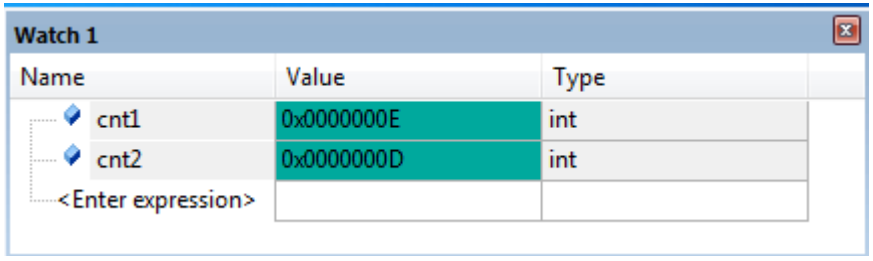
Příklad pro pre-emptive strategií.

Dva tasky jsou nekonečné smyčky. RTX startuje task 1 s vyšší prioritou, jehož funkce se jmenuje **job1**. Tato funkce startuje další task zvaný **job2**.Když **job1** začne čekat na příznak 1 **os_evt_wait_or**, RTX přepne kontext na **job2**. když **job2** inkrementuje počítadlo 2, pošle příznak 1 pro task 1 **os_evt_set** , RTX přepne kontext na **job1**. Tyto činnosti se opakují neustále.

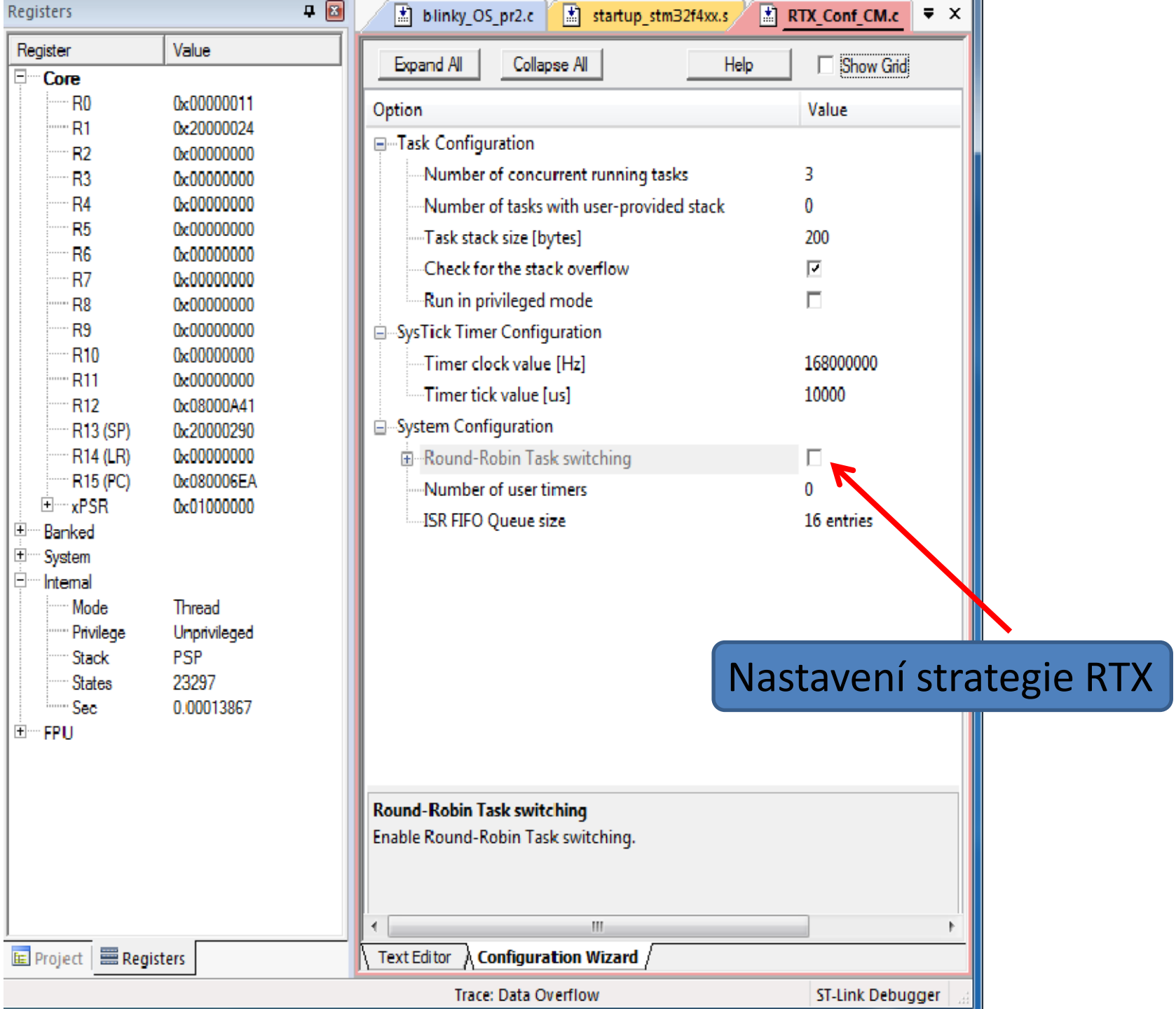
```
OS_TID tsk1,tsk2;
int    cnt1,cnt2;
__task void job1 (void);
__task void job2 (void);

__task void job1 (void) {
      os_tsk_prio(tsk1,2);
      tsk1 = os_tsk_self ();              /* vytvoří task 1 */
      os_tsk_create (job2, 1);            /* vytvoří task 2 */
  while (1) {
       os_evt_wait_or (0x0001, 0xffff);    // čeká na příznak 1 od task 2

       cnt1++;                            /* update počítadlo 1 */
  }}
__task void job2 (void) {
      while (1) {
          os_evt_set (0x0001, tsk1);       // nastaví příznak 1 pro task 1
          cnt2++;                          /* update počítadlo 2 */
  }}

void main (void) {
      os_sys_init (job1);
      while (1);}
```

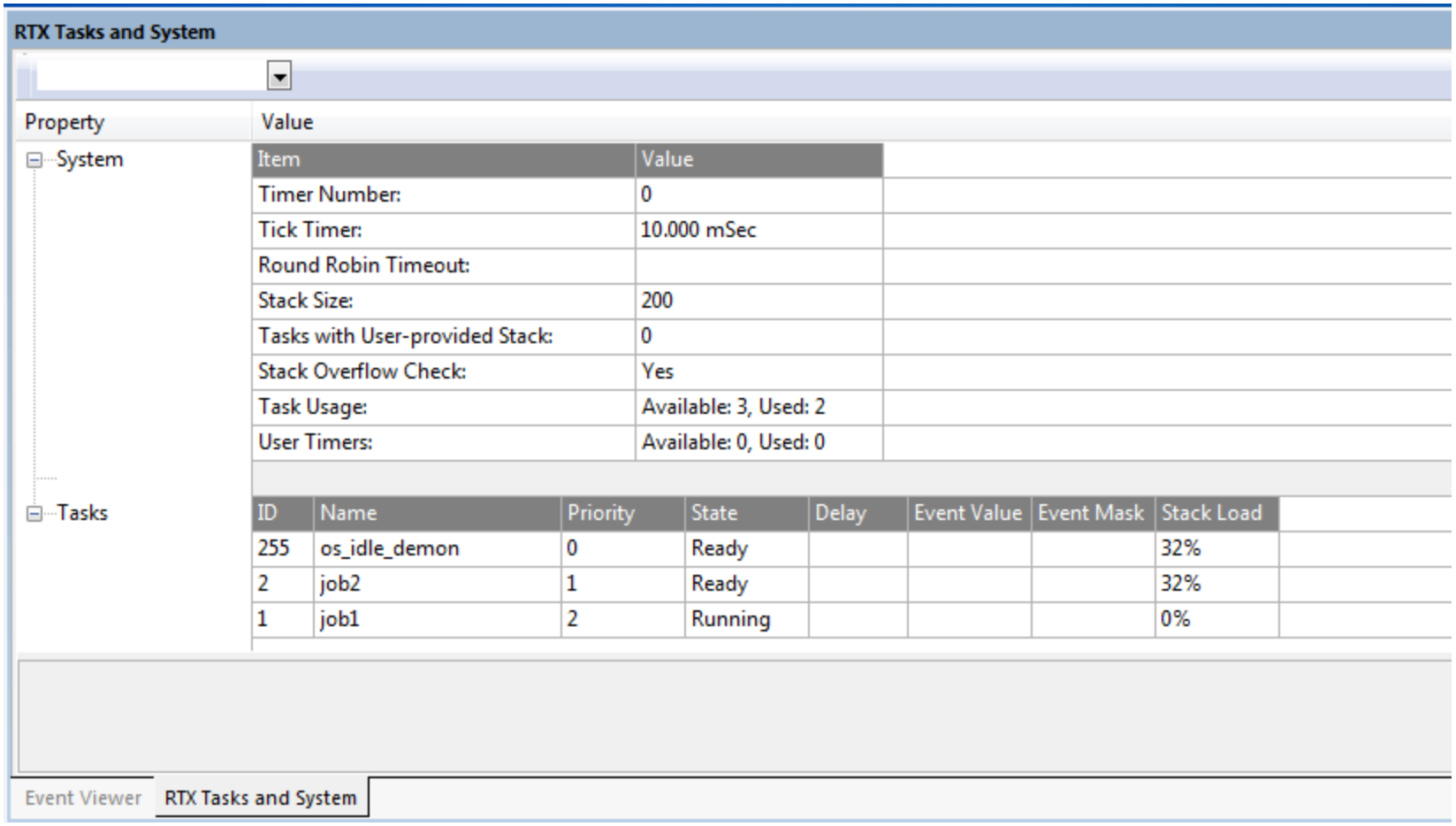
Sledování události na časové ose

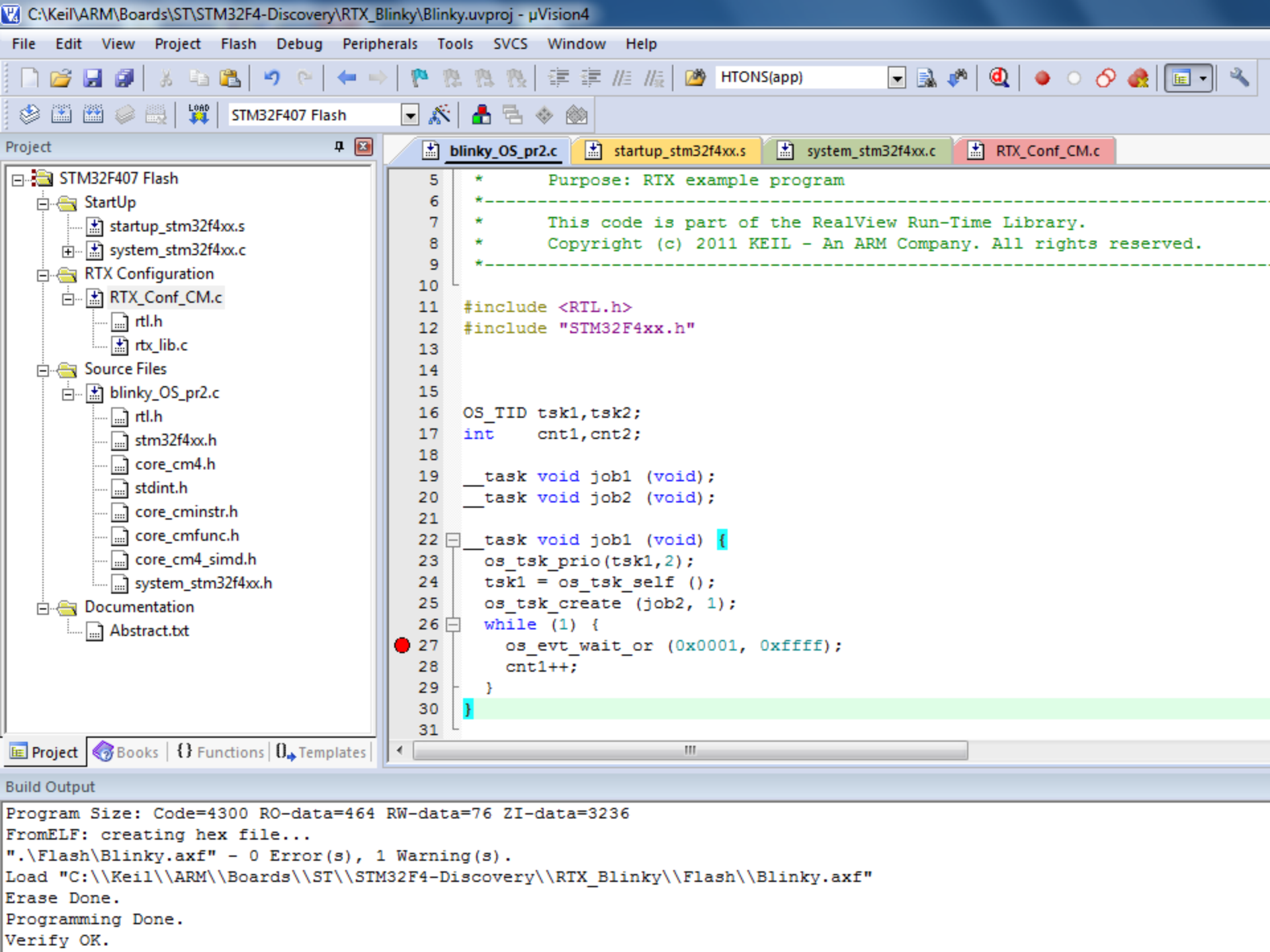




Trasování proměnných v obou procesech


Registers
Register
Value
Core
R0 0x00000011
R1 0x20000024
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x08000A41
R13 (SP) 0x20000290
R14 (LR) 0x00000000
R15 (PC) 0x080006EA
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Privilege Unprivileged
Stack PSP
States 23297
Sec 0.00013867
FPU
Project
Registers
blinky_OS_pr2.c
startup_stm32f4xx.s
RTX_Conf_CM.c
Expand All
Collapse All
Help
Show Grid
Option
Value
Task Configuration
Number of concurrent running tasks 3
Number of tasks with user-provided stack 0
Task stack size [bytes] 200
Check for the stack overflow
Run in privileged mode
SysTick Timer Configuration
Timer clock value [Hz] 168000000
Timer tick value [us] 10000
System Configuration
Round-Robin Task switching
Number of user timers 0
ISR FIFO Queue size 16 entries
Round-Robin Task switching
Enable Round-Robin Task switching.
Text Editor
Configuration Wizard
Trace: Data Overflow
ST-Link Debugger
Nastavení strategie RTX


RTX Tasks and System
Property
Value
System
Item
Value
Timer Number:
0
Tick Timer:
10.000 mSec
Round Robin Timeout:
Stack Size:
200
Tasks with User-provided Stack:
0
Stack Overflow Check:
Yes
Task Usage:
Available: 3, Used: 2
User Timers:
Available: 0, Used: 0
Tasks
ID
Name
Priority
State
Delay
Event Value
Event Mask
Stack Load
255
os_idle_demon
0
Ready
32%
2
job2
1
Ready
32%
1
job1
2
Running
0%
Event Viewer
RTX Tasks and System


C:\Keil\ARM\Boards\ST\STM32F4-Discovery\RTX_Blinky\Blinky.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HTONS(app)
STM32F407 Flash
Project
STM32F407 Flash
StartUp
startup_stm32f4xx.s
system_stm32f4xx.c
RTX Configuration
RTX_Conf_CM.c
rtl.h
rtx_lib.c
Source Files
blinky_OS_pr2.c
rtl.h
stm32f4xx.h
core_cm4.h
stdint.h
core_cminstr.h
core_cmfunc.h
core_cm4_simd.h
system_stm32f4xx.h
Documentation
Abstract.txt
Project Books Functions Templates
blinky_OS_pr2.c
startup_stm32f4xx.s
system_stm32f4xx.c
RTX_Conf_CM.c
5 * Purpose: RTX example program
6 *----------
7 * This code is part of the RealView Run-Time Library.
8 * Copyright (c) 2011 KEIL - An ARM Company. All rights reserved.
9 *----------
10
11 #include <RTL.h>
12 #include "STM32F4xx.h"
13
14
15
16 OS_TID tsk1,tsk2;
17 int cnt1,cnt2;
18
19 __task void job1 (void);
20 __task void job2 (void);
21
22 __task void job1 (void) {
23 os_tsk_prio(tsk1,2);
24 tsk1 = os_tsk_self ();
25 os_tsk_create (job2, 1);
26 while (1) {
27 os_evt_wait_or (0x0001, 0xffff);
28 cnt1++;
29 }
30 }
31
Build Output
Program Size: Code=4300 RO-data=464 RW-data=76 ZI-data=3236
FromELF: creating hex file...
".\Flash\Blinky.axf" - 0 Error(s), 1 Warning(s).
Load "C:\\Keil\\ARM\\Boards\\ST\\STM32F4-Discovery\\RTX_Blinky\\Flash\\Blinky.axf"
Erase Done.
Programming Done.
Verify OK.