FAKULTA
ELEKTROTECHNICKÁ
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

RICE

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Základy generování cílového SW ze Simulinku

Mikroprocesorové řízení pohonů 2

Jakub Talla

www.rice.zcu.cz

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj pro inovace

OSNOVA PŘEDNÁŠKY č. 10

1) Podpora cílových platforem

2) Simulink a HW periferie

3) Scheduling

4) Code replacement optimalizace (HW specifická)

Podpora cílových platforem

Podpora cílových platforem

Podpora cílových platforem

- Umožňuje konfigurovat HW periferie
- Umožňuje nastavení HW interruptů (přerušení)
- Umožňuje specifickou optimalizaci kódu
- Umožňuje spolupráci s vývojovým prostředím mikrokontroléru (např. CCS)

Podporované cílové platformy

Podporované cílové platformy dle firem:

- 3D Robotics
- Advantech
- Altium
- Analog Devices
- Android
- Apple
- Arduino
- ARM
- Atmel
- Avnet
- B&R
- BeagleBoard
- Data Translation
- dSPACE
- Eclipse
- Ettus Research
- Freescale
- Green Hills Software
- Humusoft
- Infineon
- Intel
- Keithley Instruments
- LEGO
- Linux
- Measurement Computing
- Microchip
- Microsoft
- National Instruments
- NVIDIA
- NXP
- Parrot
- Quanser
- Raspberry Pi
- Renesas
- Robot Operating System
- RTI
- Speedgoat
- STMicroelectronics
- **Texas Instruments**
- TTi
- Vector
- Vex Robotics
- Xilinx

Podporované cílové platformy TI

Podporované cílové platformy firmy Texas Instrument:

Microcontrollers	ARM Based Processors	Digital Signal Processors
MSP430™	Tiva™ C Series – M4 Stellaris® Cortex™-M3	C6000™ Power Optimized
C2000™ • Delfino™ • Piccolo™	TMS570 Cortex™-R4	C5000™ Ultra Low Power
	Sitara™	KeyStone Multicore
	Cortex™-A8 and ARM9®	KeyStone Multicore DSP+ARM
	KeyStone™	C6000™ Multicore
	OMAP Smart Multicore Processors with Cortex-A8, A9, and A15	

Podporované cílové platformy TI

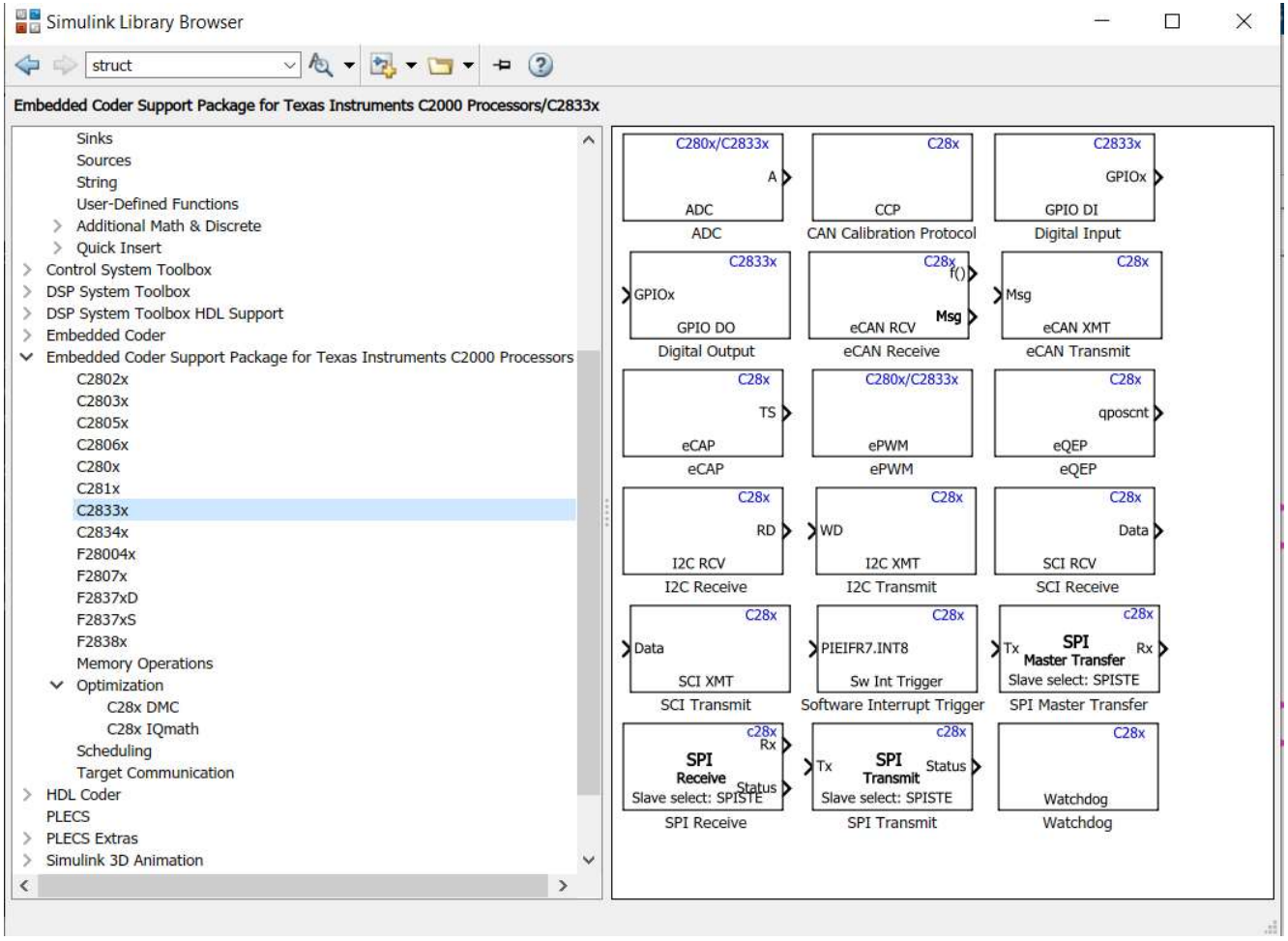
Podporované cílové mikrokontroléry série TMS320C2000 firmy Texas Instrument:

Processor Family	Processors
TI Delfino F28377S LaunchPad	F28377S
TI Delfino F2837xS	F28379S, F28377S, F28376S, F28375S, and F28374S
TI Delfino F28379D LaunchPad	F28379D
TI Delfino F2837xD	F28379D, F28377D, F28376D, F28375D, and F28374D
TI Delfino F2833x	F28335, F28334, and F28332
TI Delfino C2834x	C28346, C28345, C28344, C28343, C28342, and C28341
TI Piccolo F280049C LaunchPad	F280049C
TI Piccolo F28004x	F280049M, F280049C, F280049, F280048C, F280048, F280045, F280041C, F280041, F280040C, and F280040
TI Piccolo F2807x	F28075 and F28074
TI Piccolo F2806x	F28069M, F28069, F28068, F28067, F28066, F28065, F28064, F28063, and F28062
TI Piccolo F28069M LaunchPad	F28069M
TI Piccolo F2805x	F28055, F28054, F28053, F28052, F28051, and F28050
TI Piccolo F2803x	F28035, F28034, F28033, F28032, F28031, and F28030
TI Piccolo F2802x	F28027, F28026, F28023, F28022, F28021, F28020, and F280200
TI Piccolo F28027/F28027F LaunchPad	F28027
TI F280x	F2809, F2808, F2806, F2802, F2801, F28016, and F28015
TI F28044	F28044
TI F281x	F2812, F2811, and F2810
TI F2838x	F28388D, F28388S, F28386D, F28386S, F28384D, and F28384S

Simulink a HW periferie

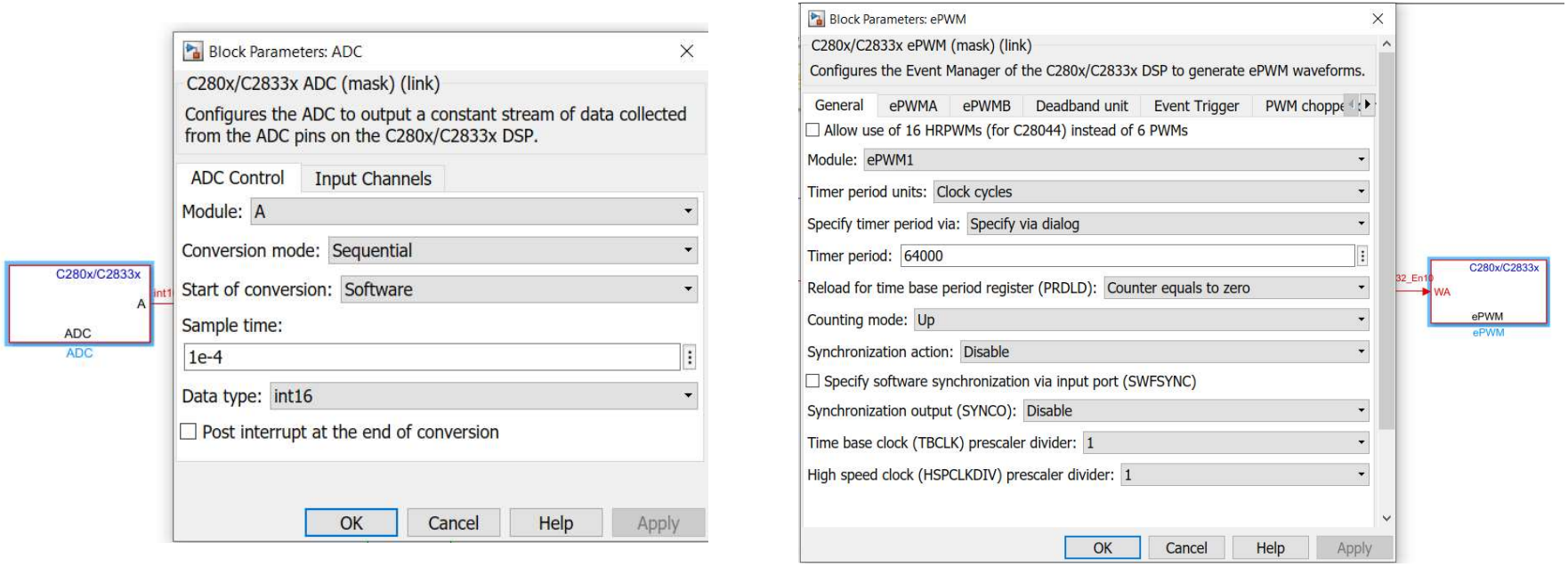
HW periferie

Bloky HW periferií



HW periferie

Bloky HW periferií - konfigurace

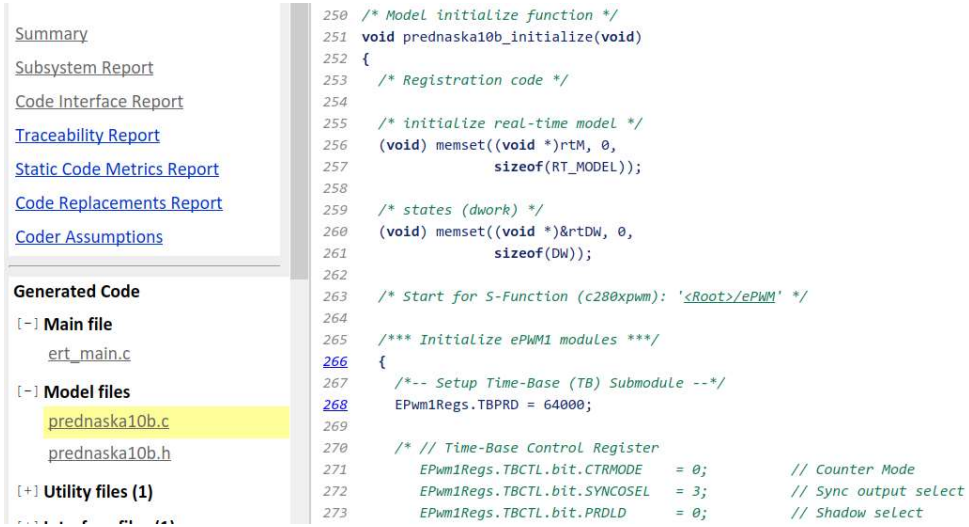


Tvorba vlastního driveru:

https://www.mathworks.com/help/supportpkg/raspberrypi/device-driver-blocks.html?s_tid=CRUX_lftnav

HW periferie

Konfigurace HW periferií



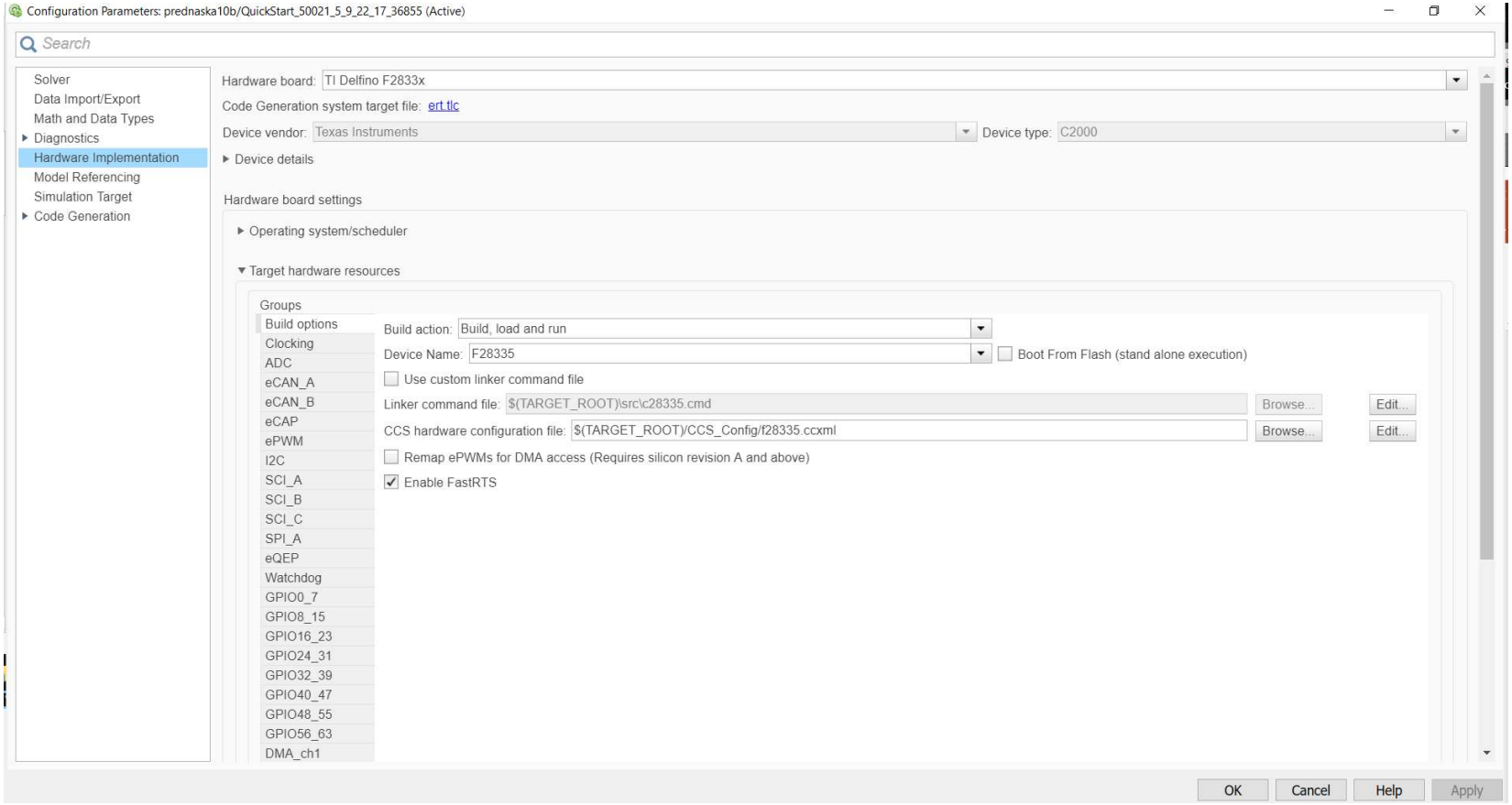

```
250 /* Model initialize function */
251 void prednaska10b_initialize(void)
252 {
253   /* Registration code */
254
255   /* initialize real-time model */
256   (void) memset((void *)rtM, 0,
257                 sizeof(RT_MODEL));
258
259   /* states (dwork) */
260   (void) memset((void *)&rtDW, 0,
261                 sizeof(DW));
262
263   /* Start for S-Function (c280xpwm): '<Root>/ePWM' */
264
265   /*** Initialize ePWM1 modules ***/
266   {
267     /*-- Setup Time-Base (TB) Submodule --*/
268     EPwm1Regs.TBPRD = 64000;
269
270     /* // Time-Base Control Register
271        EPwm1Regs.TBCTL.bit.CTRMODE    = 0;          // Counter Mode
272        EPwm1Regs.TBCTL.bit.SYNCOSEL   = 3;          // Sync output select
273        EPwm1Regs.TBCTL.bit.PRDLD      = 0;          // Shadow select
```

```
355     EPwm1Regs.TZCTL.all = (EPwm1Regs.TZCTL.all & ~0xF) | 0xF;
356
357     /* // Trip-Zone Enable Interrupt Register
358        EPwm1Regs.TZEINT.bit.OST       = 0;          // Trip Zones On
359        EPwm1Regs.TZEINT.bit.CBC       = 0;          // Trip Zones Cy
360      */
361     EPwm1Regs.TZEINT.all = (EPwm1Regs.TZEINT.all & ~0x6) | 0x0;
362     EDIS;
363   }
364
365   /* Start for S-Function (c280xadc): '<Root>/ADC' */
366   if (adcInitFlag == 0) {
367     InitAdc();
368     adcInitFlag = 1;
369   }
370
371   config_ADC_A (0U, 0U, 0U, 0U, 0U);
372 }
373
```

V periodicky volané funkci

```
/*-- Update CMPA value for ePWM1 --*/
{
  _iq10 InPortValue = rtDW.IQNxIQN;
  EPwm1Regs.CMPA.half.CMPA = (uint16_T)((_IQ10mpyI32int(InPortValue,
    EPwm1Regs.TBPRD) * 0.01));
}

/* S-Function (c280xadc): '<Root>/ADC' */
{
  AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;  /* Reset SEQ1 module*/
  AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;/*clear INT sequencer*/
  AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1;  /* Software Trigger*/
  while (AdcRegs.ADCST.bit.INT_SEQ1 == 0) {
  }                                  /*Wait for Sequencer INT bit to clear */

  asm(" RPT #11 || NOP");
  rtDW.ADC = (AdcRegs.ADCRESULT0) >> 4;
}
```

Další konfigurace projektu a HW periferií



Scheduling

(harmonogram časování)

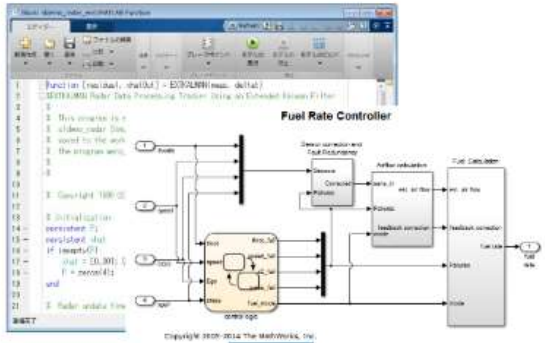
Scheduling

Simulink umožňuje tři typy událostí:

BASE RATE Synchronní událost

- **V Simulinku jako časové smyčky, HW řešen většinou jako interrupt časovače**

Základní krok simulace nastaví interrupt časovače a v něm SW řeší jednotlivé časové smyčky podle priorit od nejmenšího kroku do největšího



HW INTERRUPT Obecně asynchronní událost

- **V Simulinku řešen jako volání funkce**

V rámci bloku HW interruptu lze nastavit priority, přerušitelnost apod. Řešen jako asynchronní funkce

IDLE TASK Asynchronní s nízkou prioritou

- **V Simulinku řešen jako volání funkce**

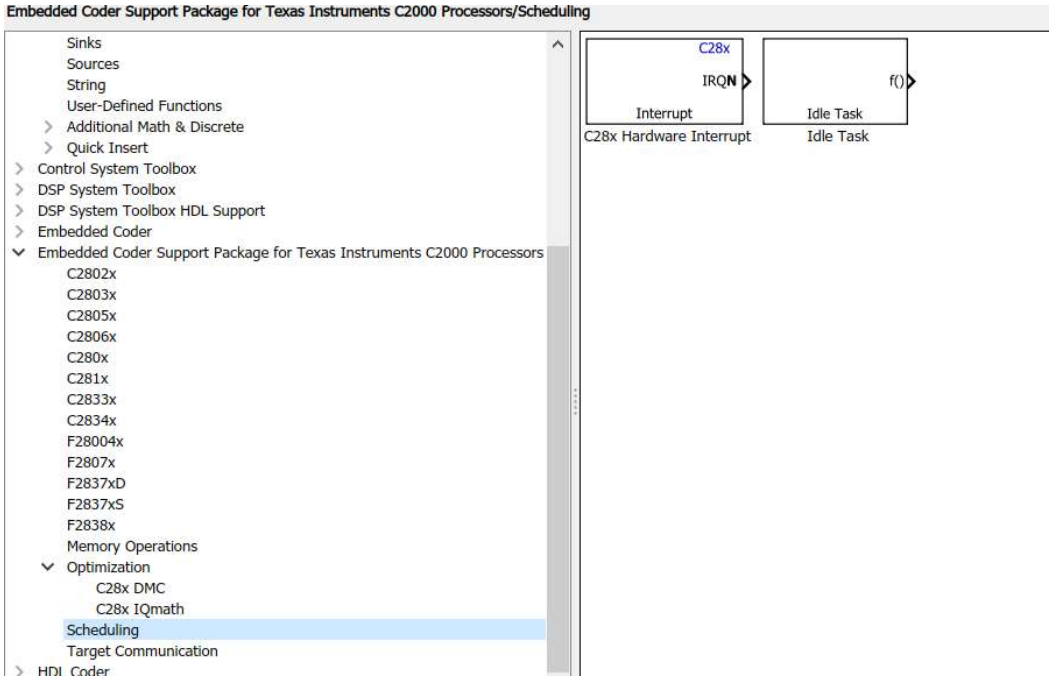
Úloha volaná mimo HW interrupt v nekonečné while smyčce. Priority určují pořadí volání funkcí.

```
if (reset) {
  y = 0;
} else {
  y += k * u;
}
```

C++

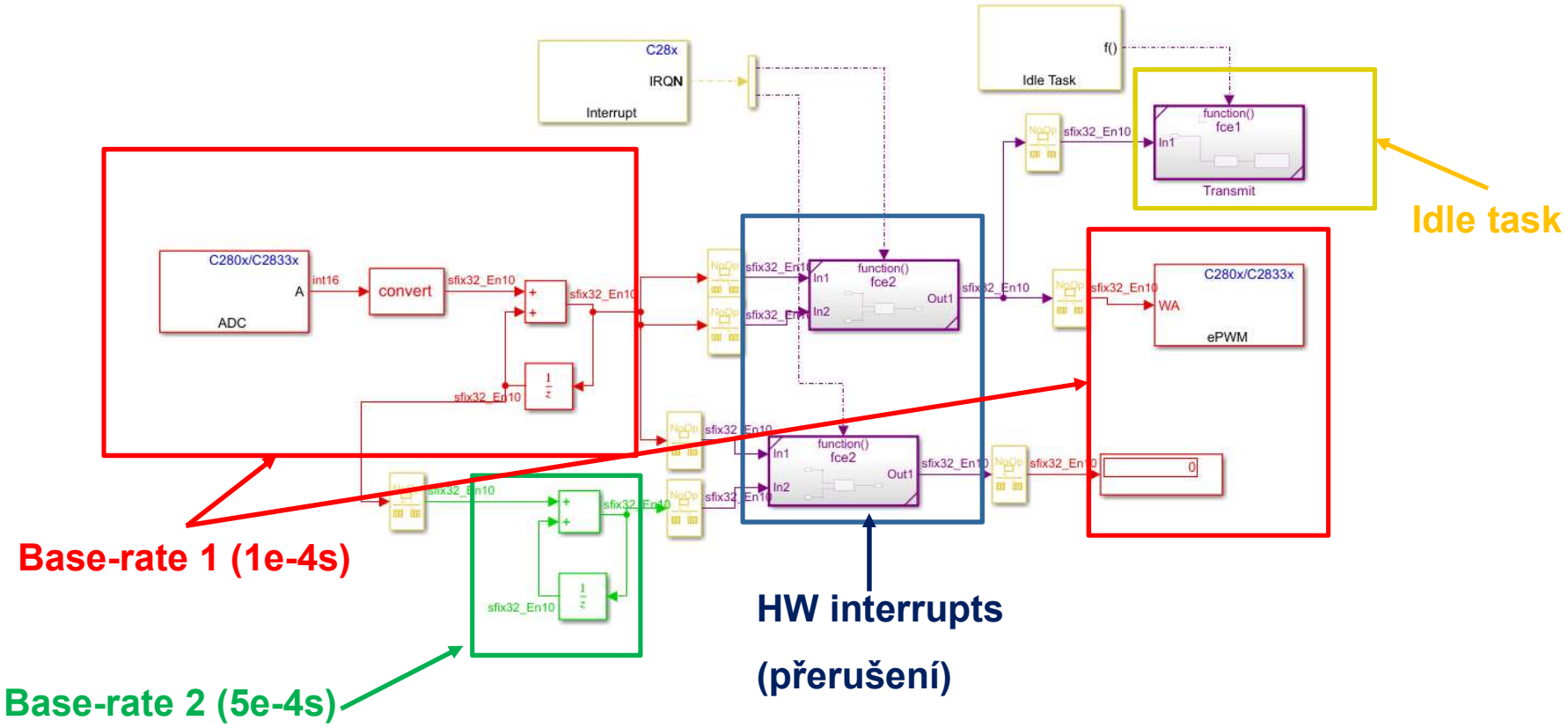
Scheduling

Scheduling = řízení událostí (HW přerušení apod.)



Scheduling

Simulink umožňuje mix všech typů časování



Scheduling Base-rate

Soubor Main

```
24 volatile int IsrOverrun = 0;
25 static boolean_T OverrunFlag = 0;
26 void rt_OneStep(void)
27 {
28
41 }
45 int main(void)
46 {
47   float modelBaseRate = 0.0001;
48   float systemClock = 150;
49
50   /* Initialize variables */
51   stopRequested = false;
52   runModel = false;
53   c2000_flash_init();
54   init_board();
64   prednaska10b_initialize();
65   configureTimer0(modelBaseRate, systemClock);
66   runModel =
67     rtmGetErrorStatus(rtM) == (NULL);
68   enableTimer0Interrupt();
69   enable_interrupts();
70   globalInterruptEnable();
71   while (runModel) {
72     stopRequested = !(
73                       rtmGetErrorStatus(rtM) == (NULL));
74   }
75
76   /* Disable rt_OneStep() here */
77   globalInterruptDisable();
78   return 0;
79 }
```

rt_OneStep - Funkce Base-rate základní časové smyčky (volaná přerušením Timeru či ADC)

main

Inicializace HW a modelu (software)

Nekonečná while smyčka v ní může být volání idle funkce

Scheduling Base-rate

rt_OneStep funkce řídící multitasking základních base-rate časových smyček

```
29  void rt_OneStep(void)
30  {
31    boolean_T eventFlags[2];
32
33    /* Check base rate for overrun */
34    if (isRateRunning[0]++) {
35      IsrOverrun = 1;
36      isRateRunning[0]--;                /* c
37      return;
38    }
39
49    enableTimer0Interrupt();
50    prednaska10b_step0();
51
52    /* Get model outputs here */
53    disableTimer0Interrupt();
70      enableTimer0Interrupt();
71
72      /* Step the model for subrate "1" */
73      switch (1)
74      {
75       case 1 :
76        prednaska10b_step1();
77
78        /* Get model outputs here */
79        break;
80
81       default :
82        break;
83      }
84
85      disableTimer0Interrupt();
```

```
static void rate_monotonic_scheduler(void)
{
  /* Compute which subrates run during the next base time step.  Subrates
   * are an integer multiple of the base rate counter.  Therefore, the subtask
   * counter is reset when it reaches its limit (zero means run).
   */
  (rtM->Timing.TaskCounters.TID[1])++;
  if ((rtM->Timing.TaskCounters.TID[1]) > 4) {/* Sample time: [0.0005s, 0.0s] */
    rtM->Timing.TaskCounters.TID[1] = 0;
  }
}

/* Model step function for TID0 */
void prednaska10b_step0(void)          /* Sample time: [0.0001s, 0.0s] */
{
  {                                    /* Sample time: [0.0001s, 0.0s] */
    rate_monotonic_scheduler();
  }

  /* UnitDelay: '<Root>/Unit Delay' */
  rtDW.UnitDelay = rtDW.UnitDelay_DSTATE;

  /* S-Function (c280xpwm): '<Root>/ePWM' */

  /*-- Update CMPA value for ePWM1 --*/
  {
    _iq10 InPortValue = rtDW.IQNxIQN;
    EPwm1Regs.CMPA.half.CMPA = (uint16_T)((_IQ10mpyI32int(InPortValue,
      EPwm1Regs.TBPRD) * 0.01));
```

Scheduling Base-rate

Base-rate HW přerušení od Timeru konfigurace v souboru c2833xSchedulerTimer0.c

- Model files
  - prednaska10b.c
  - prednaska10b.h
- Utility files (1)
- Interface files (1)
- Other files
  - DSP2833x_ADC_cal.asm
  - DSP2833x_CodeStartBranc
  - DSP2833x_CpuTimers.c
  - DSP2833x_DMA.c
  - DSP2833x_DefaultIsr.c
  - DSP2833x_GlobalVariableD
  - DSP2833x_MemCopy.c
  - DSP2833x_PieCtrl.c
  - DSP2833x_PieVect.c
  - DSP2833x_SysCtrl.c
  - DSP2833x_usDelay.asm
  - MW_c28xx_board.c
  - MW_c28xx_csl.c
  - MW_c28xx_pie.h

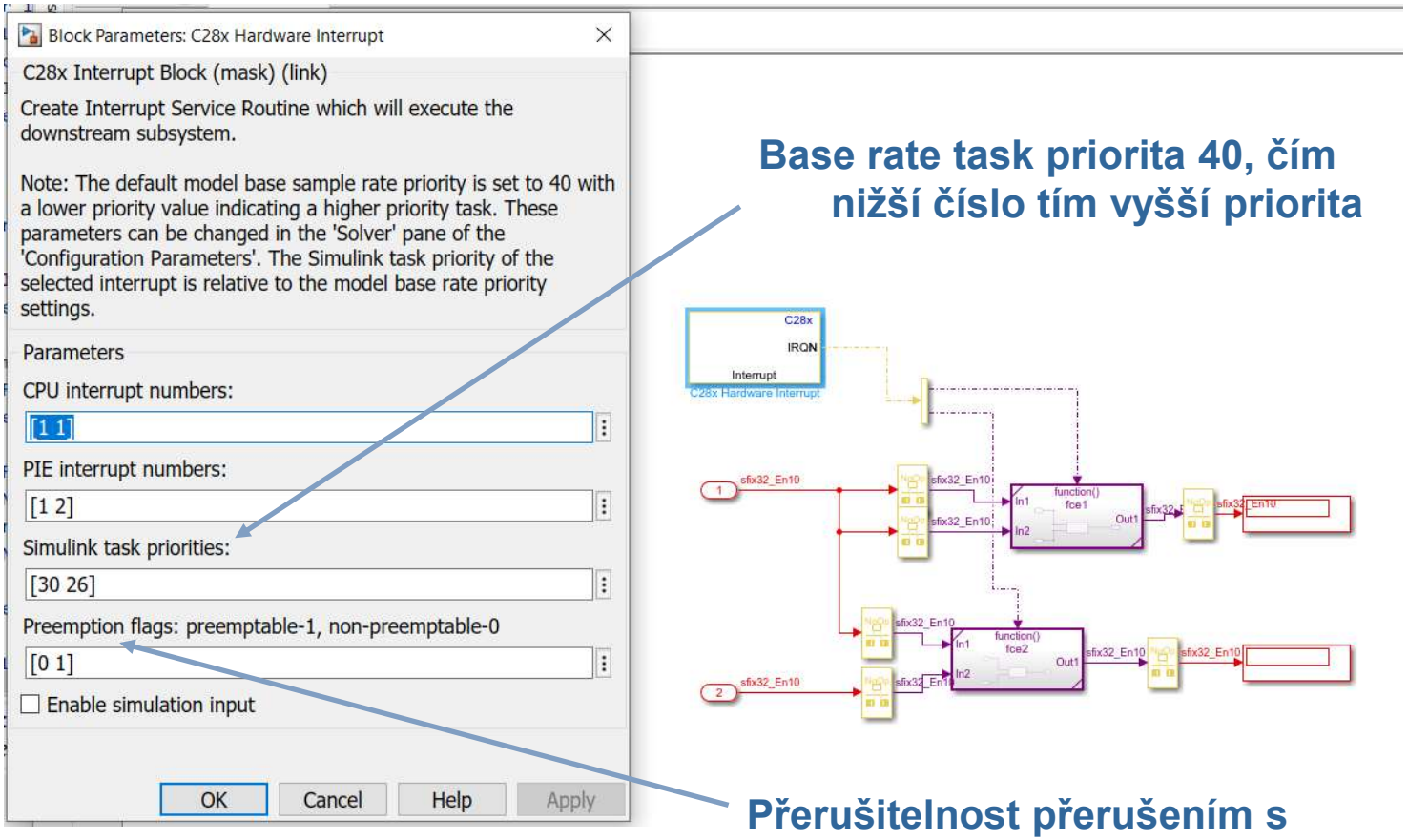
File: c2833xSchedulerTimer0.c

```
#include <stdbool.h>

#include "DSP2833x_Device.h"
#include "DSP2833x_CpuTimers.h"
#include "MW_c28xx_pie.h"

extern void rt_OneStep(void);

interrupt void TINT0_isr(void)
{
      #ifdef PIEMASK0
        volatile unsigned int PIEIER1_stack_save = PieCtrlRegs.PIEIER1.all;
      #endif



      asm(" RPT #5 || NOP");              /* wait 5 cycles */
      IFR &= ~IFRMASK;                          /* eventually disable lower/equal
      PieCtrlRegs.PIEACK.all = IFRMASK;         /* ACK to allow other interrupts fr
      IER |= 1;
      EINT;                               /* global interrupt enable */
      rt_OneStep();
      DINT;                               /* disable global interrupts during cont
```

```
void configureTimer0(float baseRate, float systemClock)
{
    /* Initialize CPU Timers here only if Profiling is disabled.
     * Because if profiling is enabled, CPU Timers are initialized in
     * MW_c28xx_board.c in the generated code.
     */
        #ifndef MW_EXEC_PROFILER_ON
        InitCpuTimers();
        #endif

        /* Configure CPU-Timer 0 to interrupt every 0.0025 sec. */
        /* Parameters:  Timer Pointer, CPU Freq in MHz, Period in usec. */
        ConfigCpuTimer(&CpuTimer0, systemClock, baseRate * 1000000);
        StartCpuTimer0();

        EALLOW;
        PieVectTable.TINT0 = &TINT0_isr;     /* Hook interrupt to the ISR*/
        EDIS;

        PieCtrlRegs.PIEIER1.bit.INTx7 = 1;   /* Enable interrupt TINT0 */
        IER |= M_INT1;
}
```

Scheduling HW interrupt

Hardware Interrupt (HW přerušení)



Base rate task priorita 40, čím nižší číslo tím vyšší priorita

Přerušitelnost přerušením s vyšší prioritou (0 ne 1 ano)

Scheduling HW interrupt

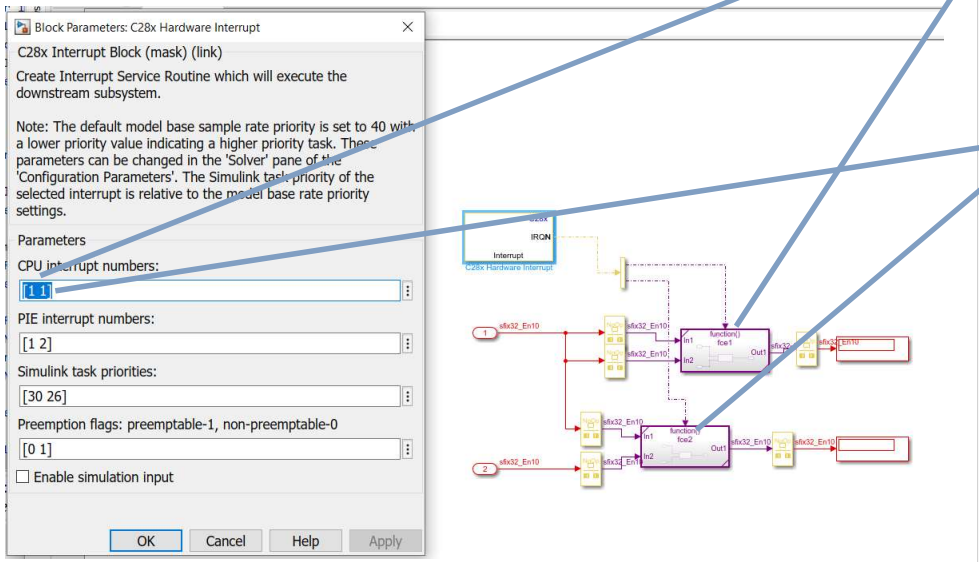
Tabulka přerušení (součást help)

PIE and CPU Interrupt Numbers for F280x, F2802x, F2803x, F2805x, F2806x, F2833x, F28M35x, and F28M36x Processors

PIE ⇒ / CPU ⇓	1	2	3	4	5	6	7	8
1	SEQ1INT (ADC) / ADCINT1	SEQ2INT (ADC) / ADCINT2	Reserved	XINT1	XINT2	ADCINT / ADCINT9	TINT0 (TIMER 0)	WAKEINT (LPM/WD)
2	EPWM1_TZINT	EPWM2_TZINT	EPWM3_TZINT	EPWM4_TZINT	EPWM5_TZINT	EPWM6_ TZINT	EPWM7_TZINT	EPWM8_TZINT
3	EPWM1_INT	EPWM2_INT	EPWM3_INT	EPWM4_INT	EPWM5_INT	EPWM6_ INT	EPWM7_INT	EPWM8_INT
4	ECAP1_INT	ECAP2_INT	ECAP3_INT	ECAP4_INT	ECAP5_INT	ECAP6_INT	EPWM10_TZINT / HRCAP1_INT	EPWM9_TZINT / HRCAP2_INT
5	EQEP1_INT	EQEP2_INT	EQEP3_INT	HRCAP3_INT	HRCAP4_INT	Reserved	EPWM10_INT	EPWM9_INT
6	SPIRXINTA (SPI-A)	SPITXINTA (SPI-A)	SPIRXINTB (SPIB_RX) / MRINTB (McBSP-B)	SPITXINTB (SPIB_TX) / MXINTB (McBSP-B)	SPIRXINTC (SPI-C) / MRINTA (McBSP-A_RX)	SPITXINTC (SPI-C) / MXINTA (McBSP-A_TX)	SPIRXINTD (SPI-D) / EPWM12_TZINT	SPITXINTD (SPI-D) / EPWM11_TZINT
7	DINTCH1 (DMA1)	DINTCH2 (DMA2)	DINTCH3 (DMA3)	DINTCH4 (DMA4)	DINTCH5 (DMA5)	DINTCH6 (DMA6)	EPWM12_INT	EPWM11_INT
8	I2CINT1A	I2CINT2A	Reserved	Reserved	SCIRXINTC (SCI-C)	SCITXINTC (SCI-C)	Reserved	Reserved
9	SCIRXINTA (SCIA_RX)	SCITXINTA (SCIA_TX)	SCIRXINTB (SCIB_RX) / LINA_INT0	SCITXINTB (SCIB_TX) / LINA_INT1	ECAN0INTA (CANA_1)	ECAN1INTA (CANA_2)	ECAN0INTB (CANB_1)	ECAN1INTB (CANB_2)
10	EPWM9_TZINT / ADCINT1	EPWM10_TZINT / ADCINT2	EPWM11_TZINT / ADCINT3	EPWM12_TZINT / ADCINT4	EPWM13_TZINT / ADCINT5	EPWM14_TZINT / ADCINT6	EPWM15_TZINT / ADCINT7	EPWM16_TZINT / ADCINT8
11	CLA1_INT1 / EPWM9_INT7 / MTOCIPCINT1	CLA1_INT2 / EPWM10_INT / MTOCIPCINT2	CLA1_INT3 / EPWM11_INT / MTOCIPCINT3	CLA1_INT4 / EPWM12_INT / MTOCIPCINT4 /	CLA1_INT5 / EPWM13_INT	CLA1_INT6 / EPWM14_INT	CLA1_INT7 / EPWM15_INT	CLA1_INT8 / EPWM16_INT
12	XINT3	XINT4 / C28FLSINGERR	XINT5	XINT6 / C28RAMSINGERR	XINT7 / C28RAMACCVIOL	Reserved	LVF	LUF

Scheduling HW interrupt

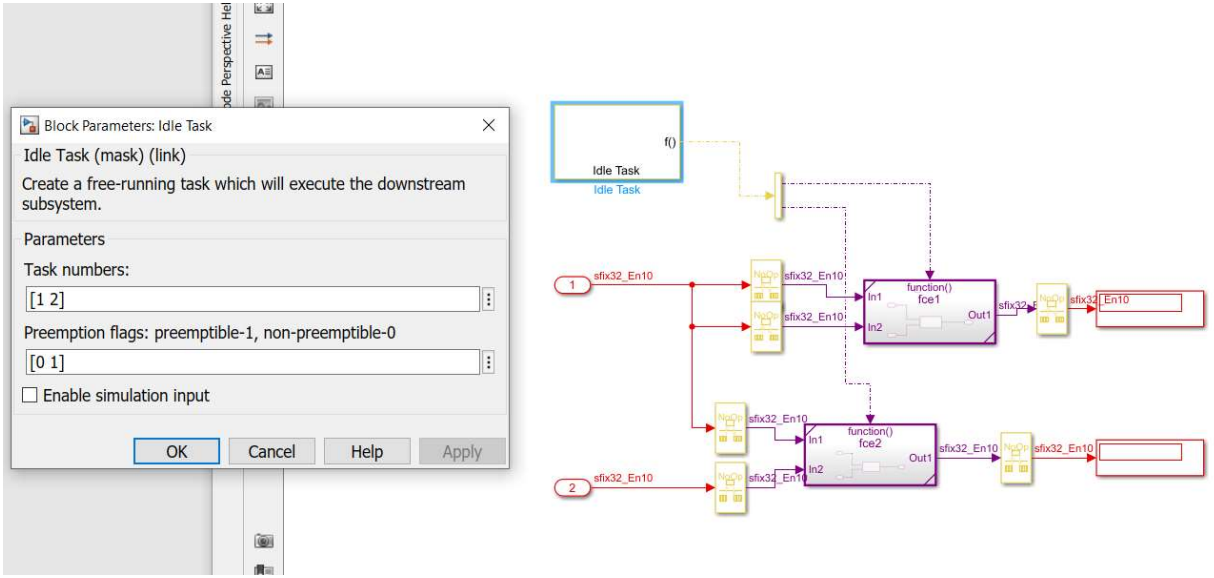
Hardware Interrupt (HW přerušení)




```
17 interrupt void SEQ1INT_isr(void)
18 {
19   isr_int1pie1_task_fcn();
20   EALLOW;
21   AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
22   EDIS;
23   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
24                                   /* Acknowledge to receive more interrupts*/
25 }
26
27 interrupt void SEQ2INT_isr(void)
28 {
29   volatile unsigned int PIEIER1_stack_save = PieCtrlRegs.PIEIER1.all;
30   PieCtrlRegs.PIEIER1.all &= ~67;
31                               /*disable group1 lower/equal priority interrupts*/
32   asm(" RPT #5 || NOP");              /*wait 5 cycles        */
33   IFR &= ~1;      /*eventually disable lower/equal priority pending interrupts*/
34   PieCtrlRegs.PIEACK.all = 1;
35                  /*ACK to allow other interrupts from the same group to fire*/
36   IER |= 1;
37   EINT;                               /*global interrupt enable*/
38   isr_int1pie2_task_fcn();
39   DINT;
40   /* disable global interrupts during context switch, CPU will enable global interrupts af
41   PieCtrlRegs.PIEIER1.all = PIEIER1_stack_save;
42                                  /*restore PIEIER register that was modified*/
43   EALLOW;
44   AdcRegs.ADCST.bit.INT_SEQ2_CLR = 1;
45   EDIS;
46   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
47                                   /* Acknowledge to receive more interrupts*/
48 }
49
```

Scheduling Idle task

IDLE task



[-] **Main file**

ert_main.c

[-] **Model files**

prednaska10b.c

prednaska10b.h

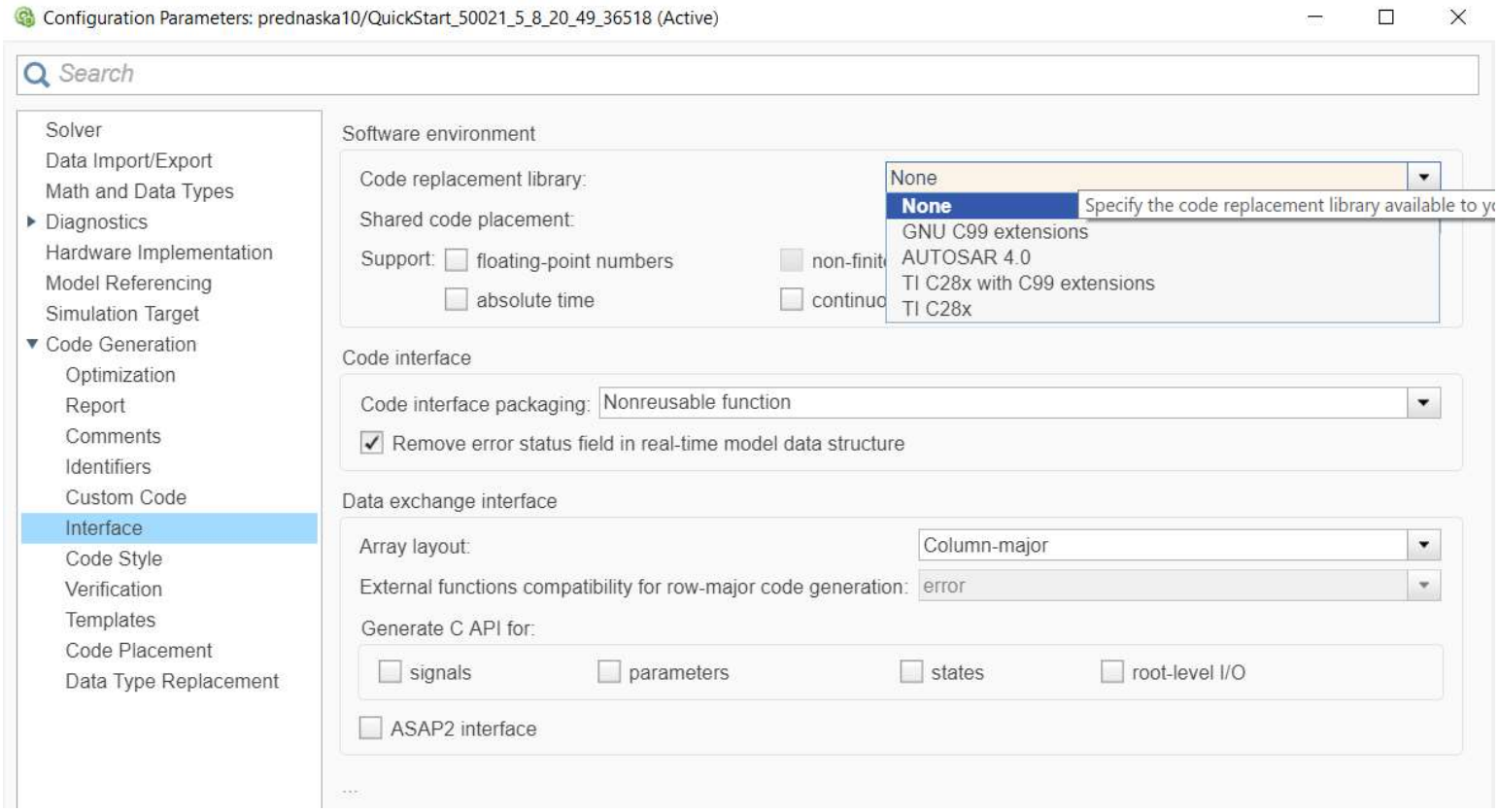
```
55   enableTimer0Interrupt();
56   globalInterruptEnable();
57   while (runModel) {
58     stopRequested = !(
59                       rtmGetErrorStatus(rtM) == (NULL));
60     idletask_num1();
61     idletask_num2();
62   }
```

```
17  void idletask_num1(void)
18  {
19    DINT;                          /* disable global interrupts*/
20    idle_num1_task_fcn();
21    EINT;                          /* reenable global interrupts*/
22  }
23
24  void idletask_num2(void)
25  {
26    idle_num2_task_fcn();
27  }
```

Optimalizace - Code replacement

Optimalizace - Code replacement

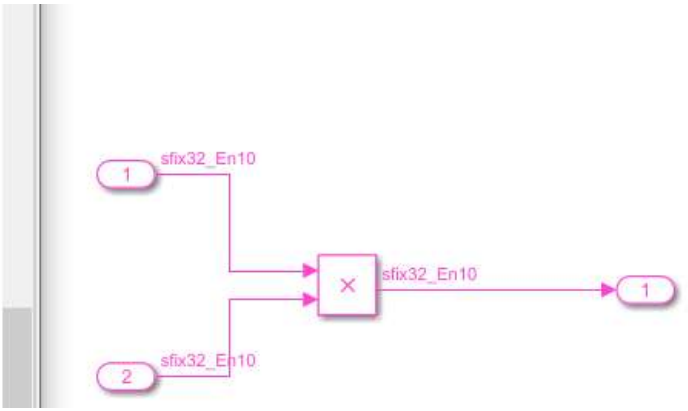
Code replacement umožňuje náhradu obecných funkcí specifickými (např. HW optimalizovanými)



Optimalizace - Code replacement

Code replacement lib. TI C28x

```
78  /* Model step function */
79  void Subsystem_step(void)
80  {
81    /* Outport: '<Root>/Out1' incorporates:
82     *  Inport: '<Root>/In1'
83     *  Inport: '<Root>/In2'
84     *  Product: '<S1>/Product'
85     */
86    rtY.Out1 = __IQmpy(rtU.In1, rtU.In2, 10);
87  }
88
```

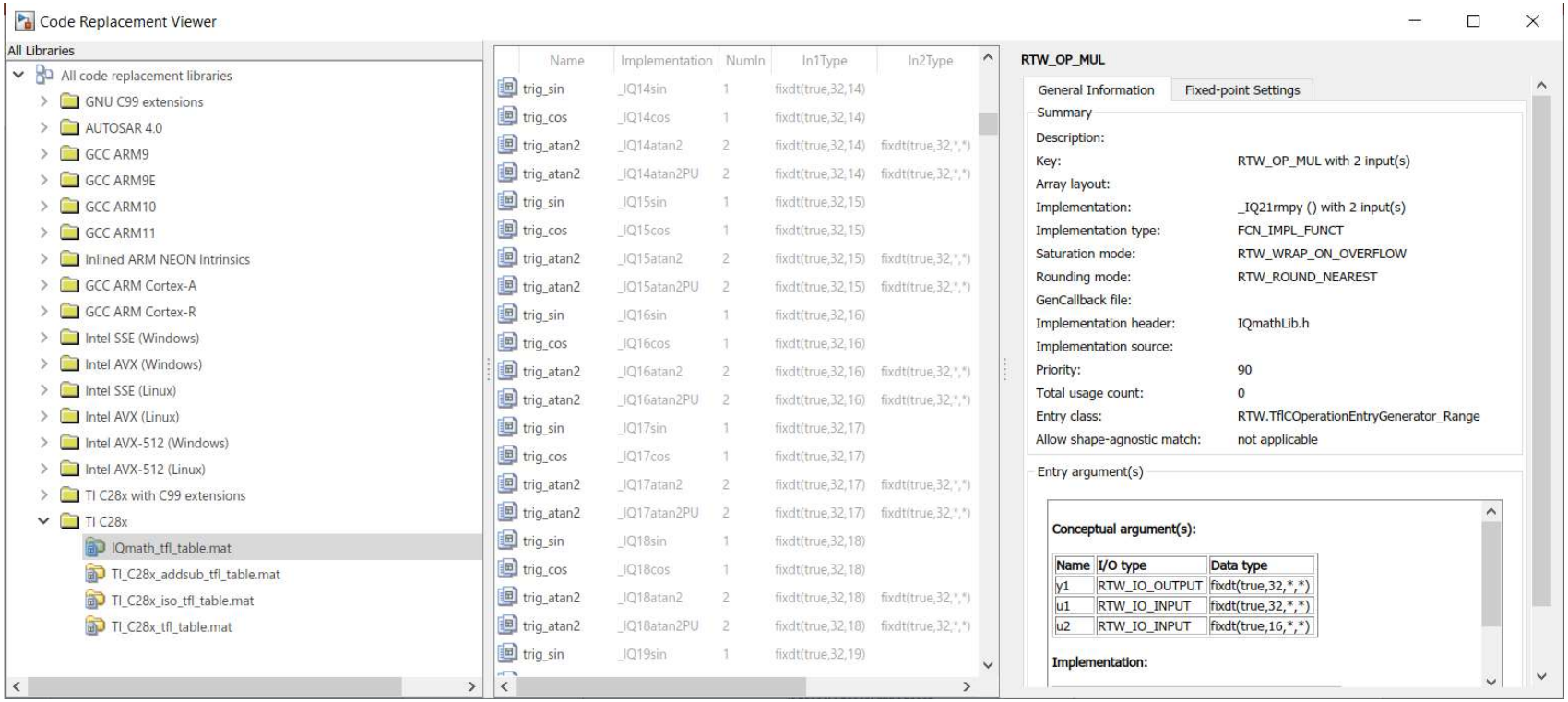


Code replacement lib. None

```
/* Model step function */
void Subsystem_step(void)
{
  /* Outport: '<Root>/Out1' incorporates:
   *  Inport: '<Root>/In1'
   *  Inport: '<Root>/In2'
   *  Product: '<S1>/Product'
   */
  rtY.Out1 = (int32_T)(((int64_T)rtU.In1 * rtU.In2) >> 10U);
}
```

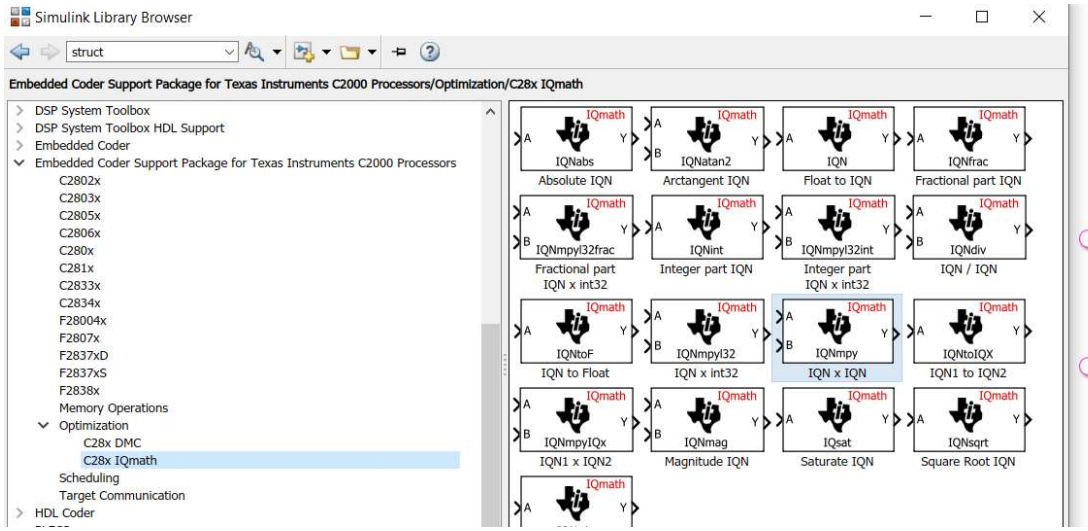
Optimalizace - Code replacement

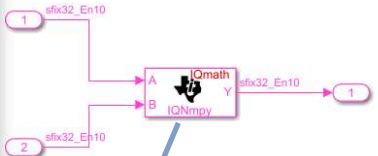
Code replacement viewer (příkaz crviewer, vlastní knihovna přikazem crtool)



Optimalizace - Code replacement

Použití specifických (optimalizovaných) bloků

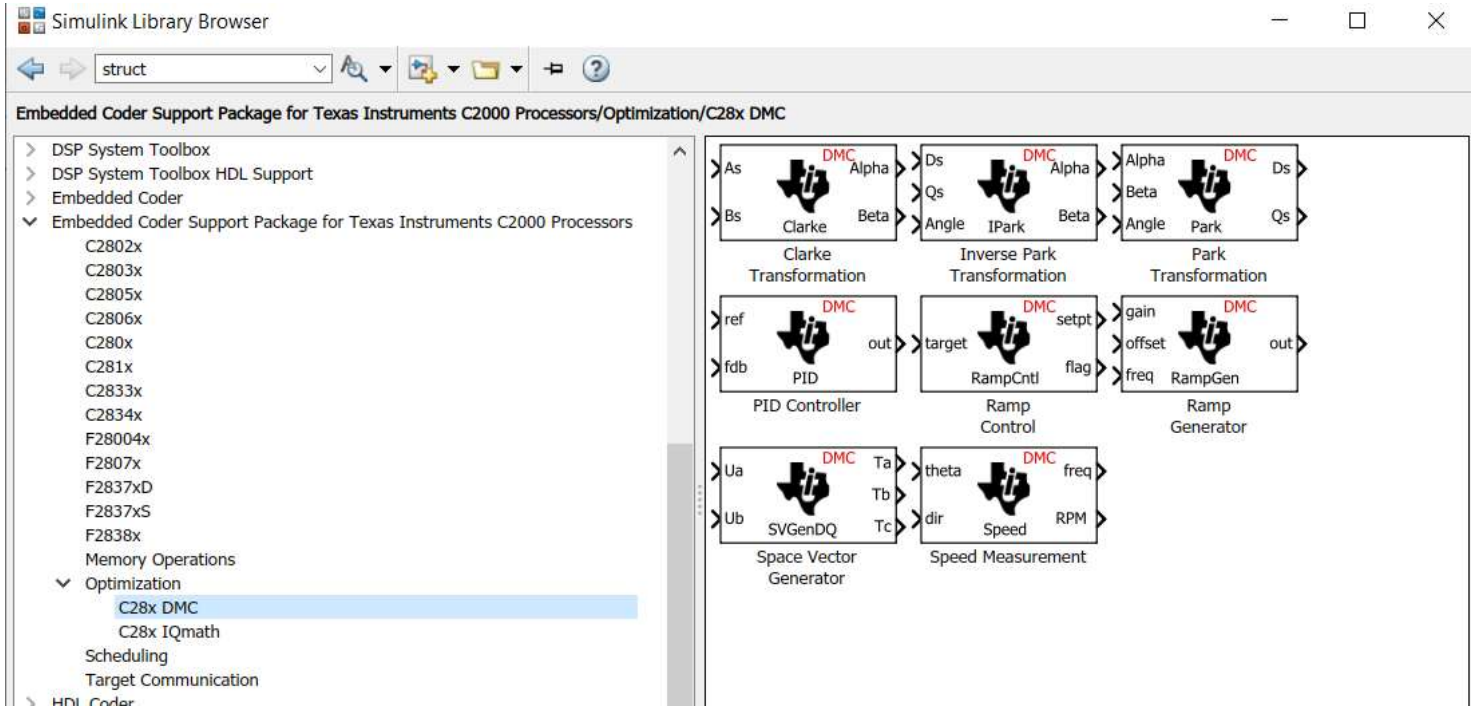





```
void Subsystem1_step(void)
{
  /* S-Function (stiiqmath_iqmpy): '<S1>/IQN x IQN' incorporates:
   *  Inport: '<Root>/In1'
   *  Inport: '<Root>/In2'
   *  Outport: '<Root>/Out1'
   */

  /* C28x IQmath Library (stiiqmath_iqmpy) - '<S1>/IQN x IQN' */
  {
    rtY.Out1 = _IQ10mpy (rtU.In1, rtU.In2);
  }
}
```

Optimalizace - Code replacement

Použití specifických (optimalizovaných) bloků DMC knihovna



FAKULTA
ELEKTROTECHNICKÁ
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

RICE

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Děkuji za pozornost!

Adresa:
Univerzitní 26
306 14 Plzeň
Česká republika

Tel: +420 377 634 443
Fax: +420 377 634 402

Email: talic@kev.zcu.cz

www.rice.zcu.cz

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj pro inovace