

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Základy generování aplikačního SW ze Simulinku

Mikroprocesorové řízení pohonů 2

Jakub Talla

- 1) Úvod do generování kódů ze Simulinku
- 2) Automatické generování aplikačního SW ze Simuinku
- 3) Testování vygenerovaného SW pomocí SIL/PIL

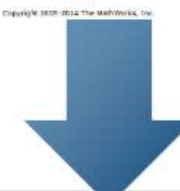
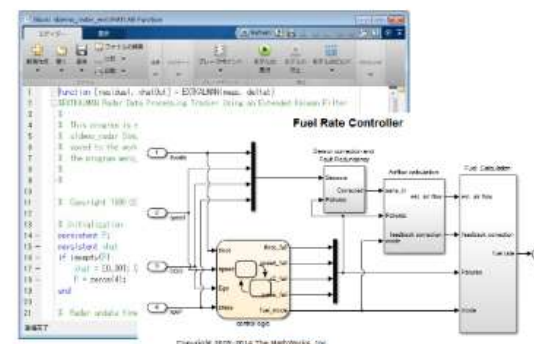
Motivace automatického generování C kódu ze Simulinku

Modelování a Simulace

- **Grafické modelování**
čitelnost, srozumitelnost, přenositelnost
- **Brzká verifikace a validace**
(průběžná validace modelu, kódu SIL... HIL apod.)

Automatické generování kódů

- Redukce programátorského času
- Redukce chyb při transformaci algoritmu do kódu
- Propojení modelu a kódu
- Automatizace testů



```

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

```



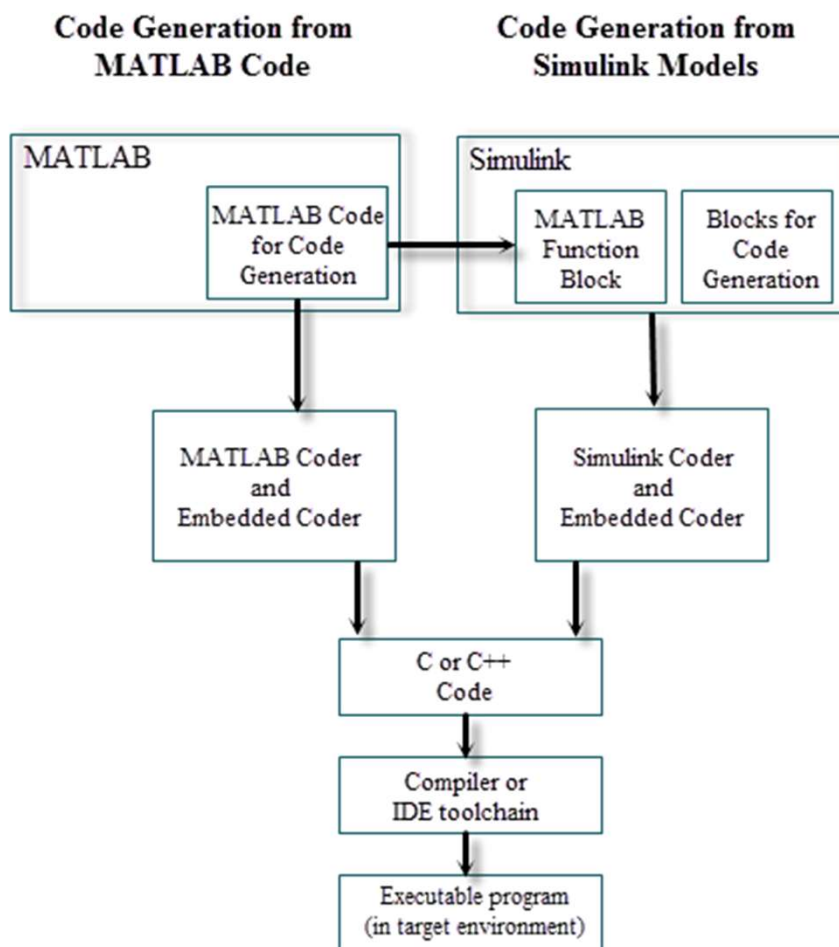
Úvod do generování kódů ze Simulinku

Příklad automatického generování na SW ECU

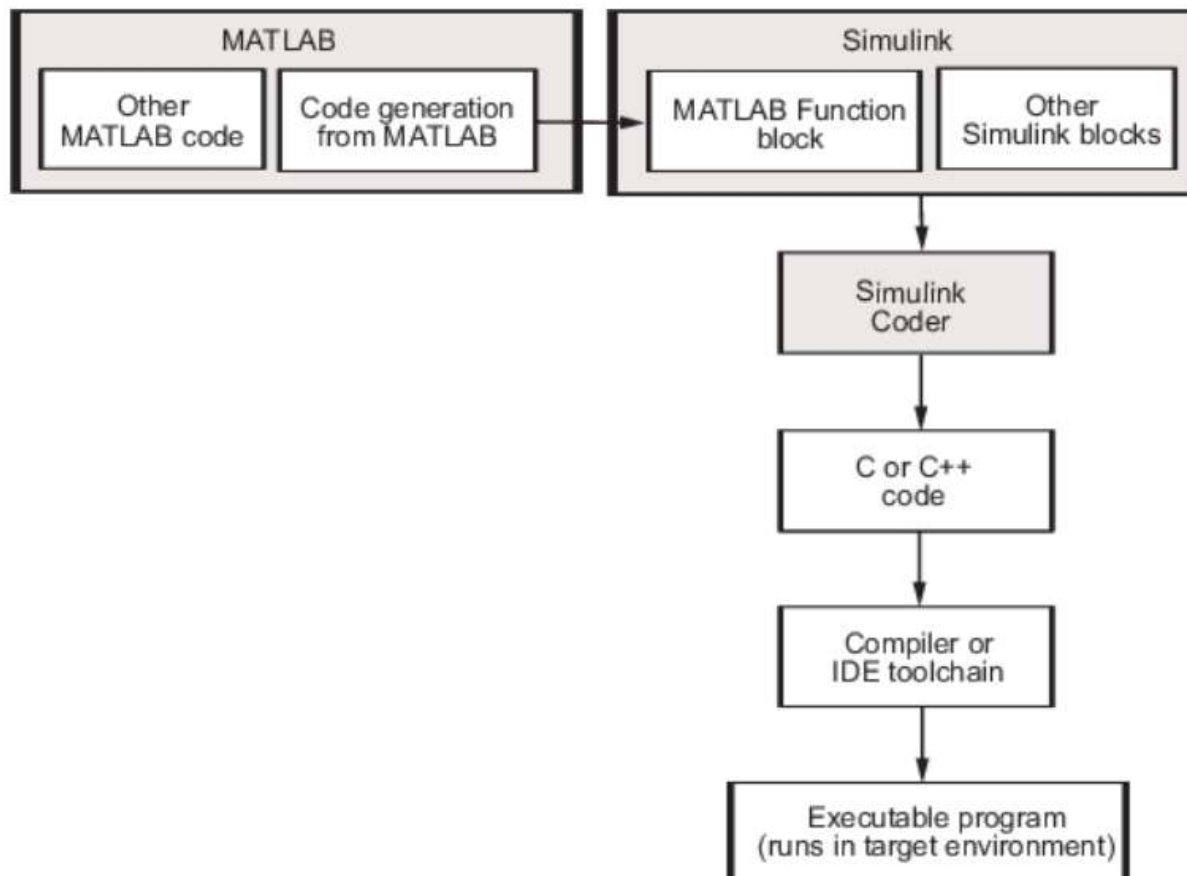
	Hand Code	Auto-code	Percent Change
Calibration ROM	9464	9464	0%
Code ROM	2952	2900	-1.76%
RAM	240	238	-0.83%

More
Efficient
Than Hand

Úvod do generování kódů ze Simulinku



Automatické generování C kódu ze Simulinku



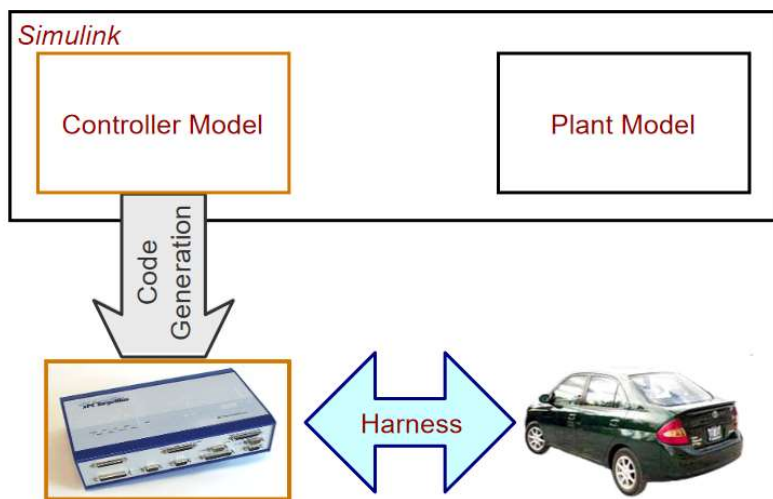
Úvod do generování kódů ze Simulinku

Pro jaké účely se hodí generování kódu ze Simulinku a Stateflow?

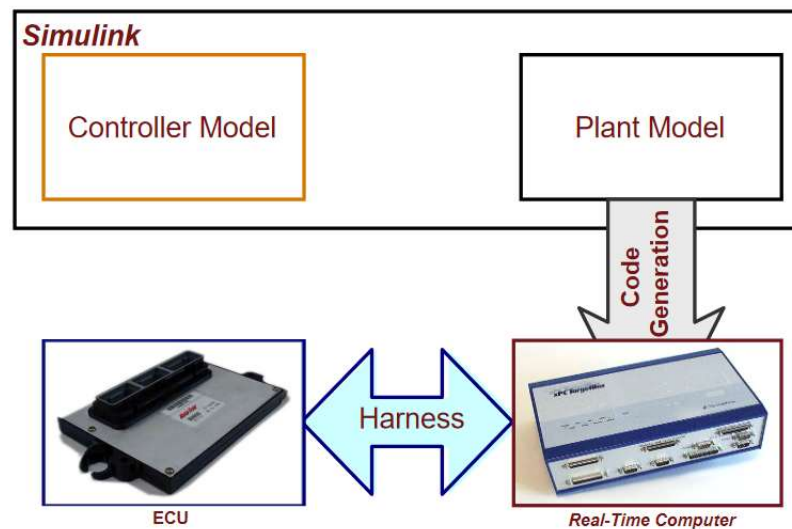
- Akcelerace Simulací (mód rapid accelerator)
- Rapid prototyping (prototypování aplikací)
- Hardware In the Loop (HIL) testování
- Vývoj aplikačního embedded SW

Úvod do generování kódů ze Simulinku

Rapid Prototyping



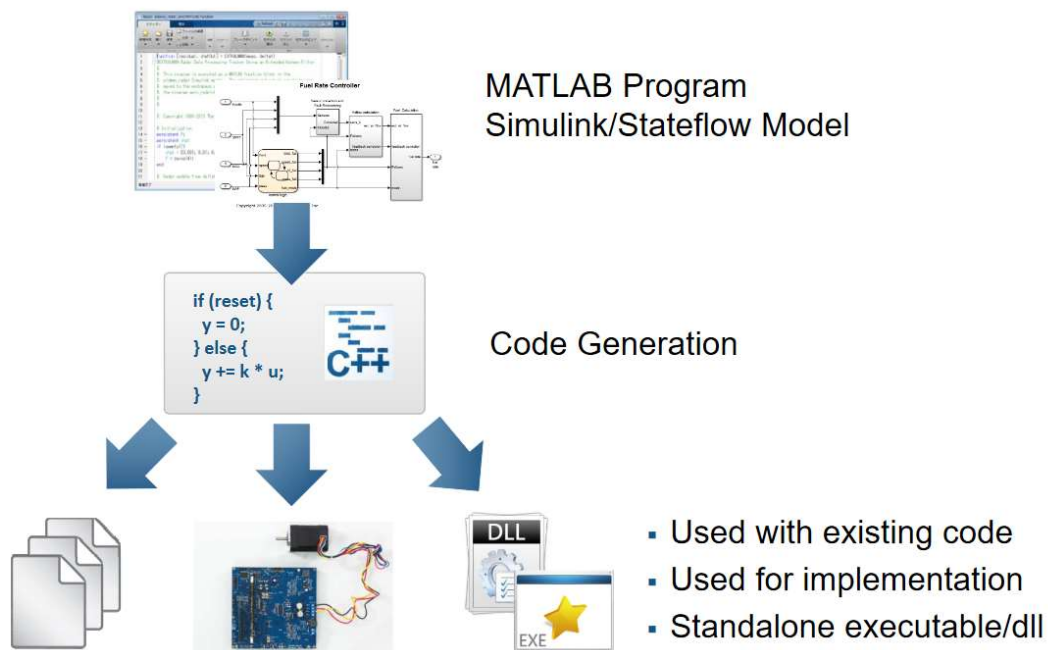
Hardware in Loop (HIL)



Úvod do generování kódů ze Simulinku

Kód může být vygenerován v různých podobách:

- 1) Aplikační SW takřka nezávislý na HW - nutná vlastní integrace do projektu
- 2) Projekt s nastavením základních HW komponent (interrupty apod.), další HW funkce se ručně importují do vygenerovaného projektu
- 3) Kompletní projekt (není nutné SW ručně upravovat)

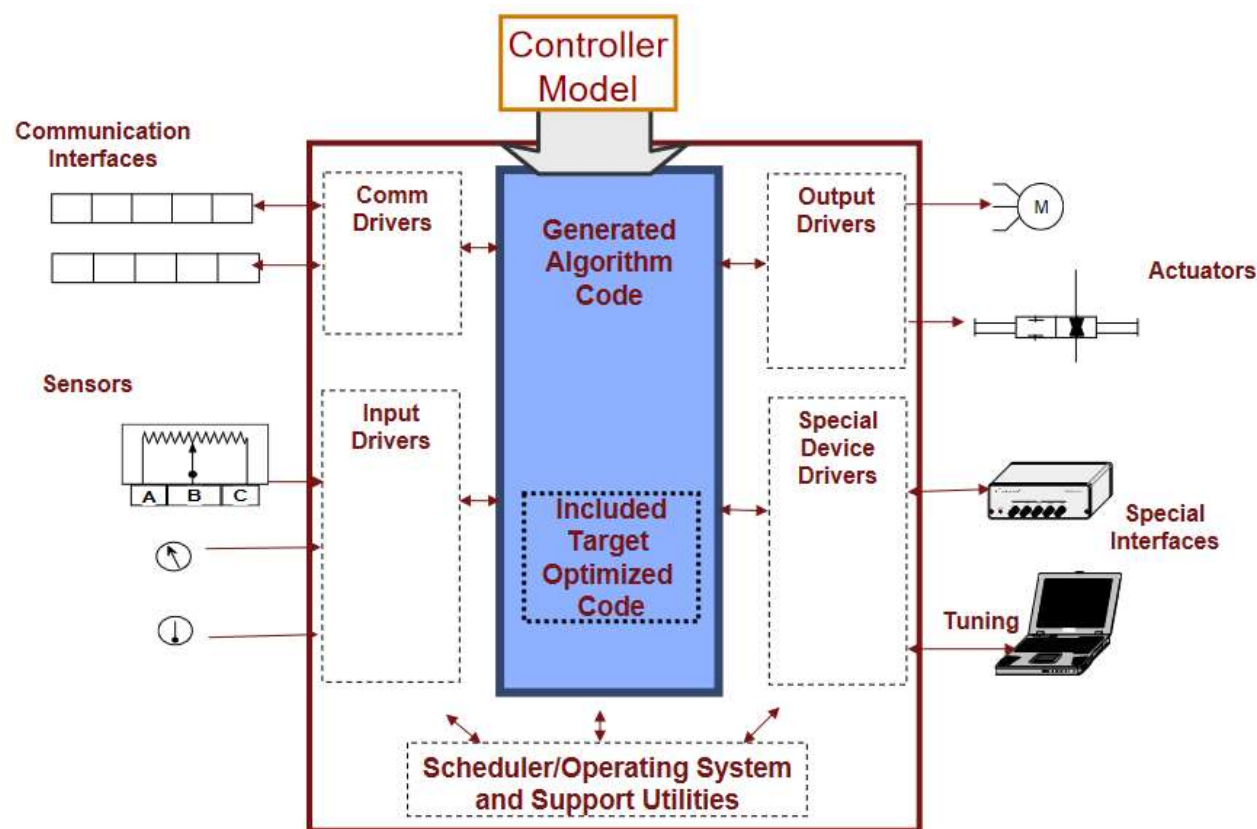


Vývoj a integrace aplikačního SW pomocí Simulinku

Generování aplikačního SW ze Simulinku

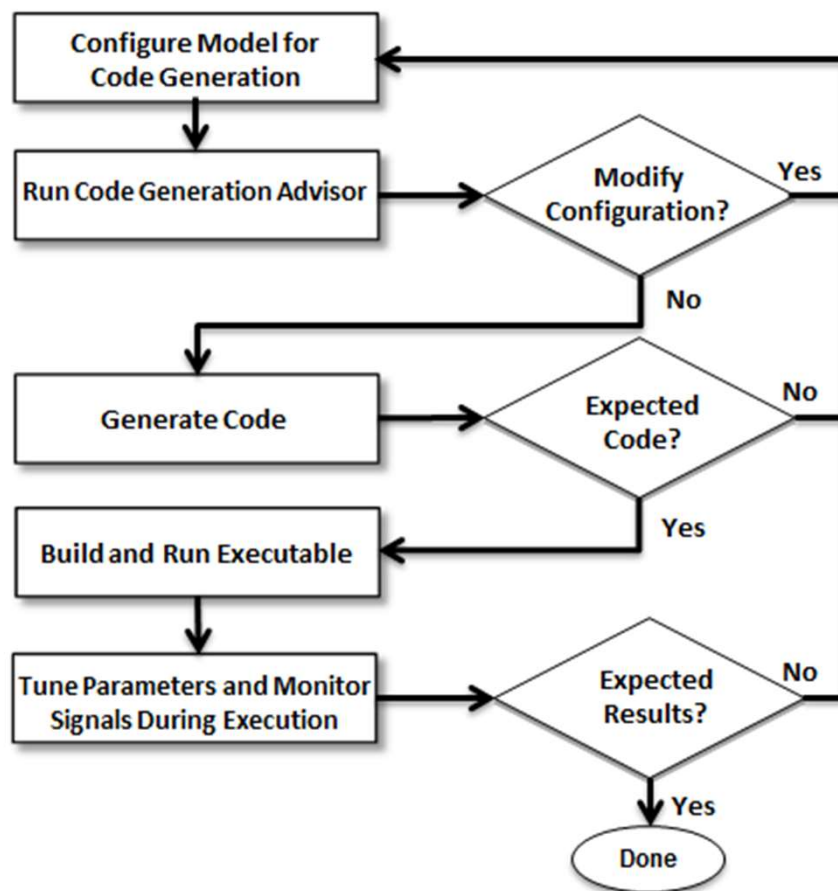
Generujeme pouze funkce nezávislé na HW a provádíme jejich otestování a integraci v IDE (např. CCS)

Tj. dnes se nebudeme bavit o nastavení HW, interruptů a schedulingu



Generování aplikačního SW ze Simulinku

Průběh generování kódu ze Simulinku



Postup při návrhu:

1) Úprava modelu pro generování

přechod ze spojité oblasti do diskrétní, double -> fixed point, optimalizace, pojmenování signálů apod.

2) Konfigurace modelu pro generování kódu

Konfigurece v model settings – nastavení záložek Solver, HW Implementation, Code Generation

3) Výběr co chci generovat

Model reference, subsystem, function

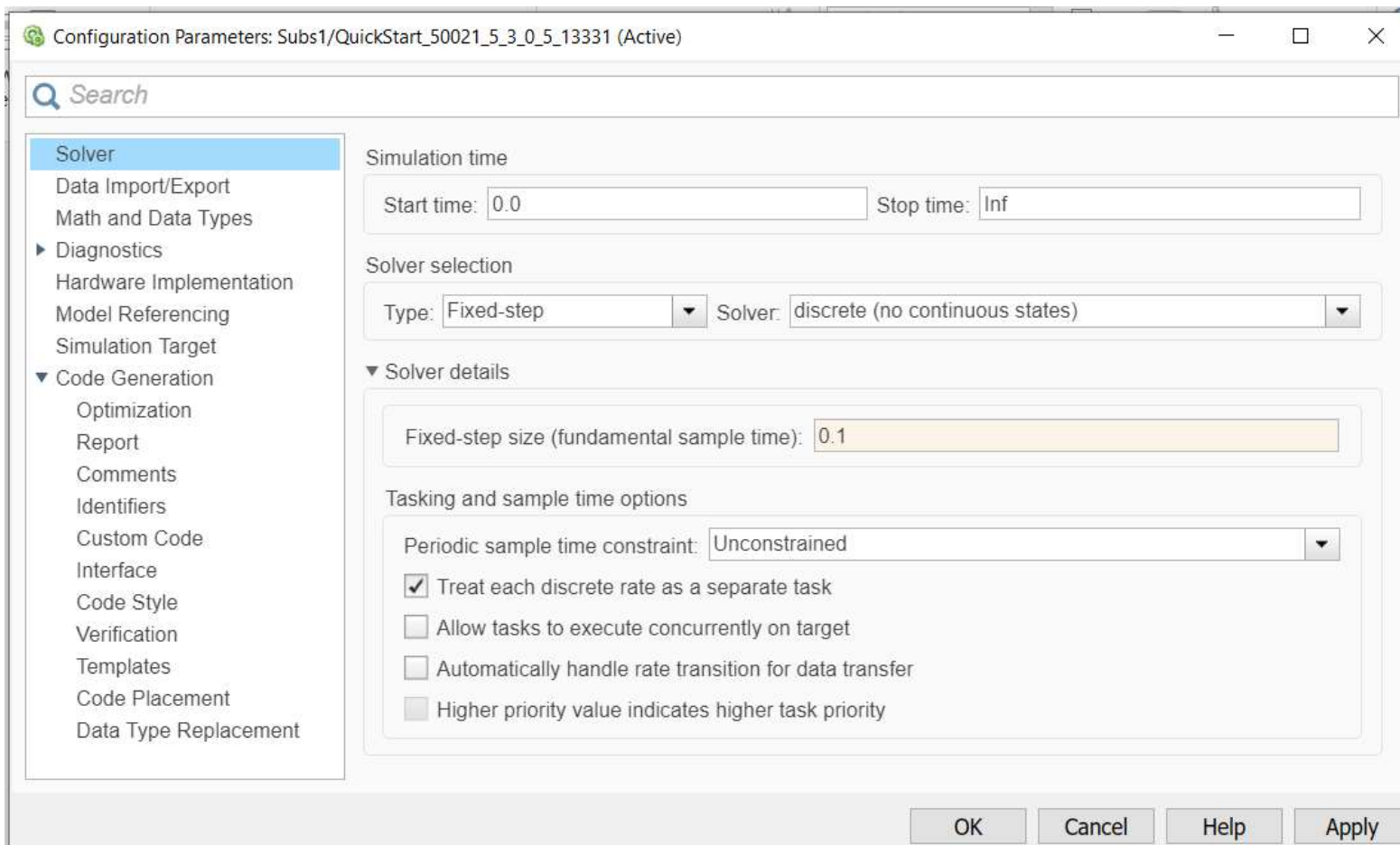
4) Otestování

S-function builder, automatic SIL block...

5) Integrace

„Ruční“ integrace, testy integrace....

Konfigurace modelu pro generování kódu Solver



Konfigurace modelu pro generování kódu Hardware Implementation

Configuration Parameters: Subs1/QuickStart_50021_5_3_0_5_13331 (Active)

Search

- Solver
- Data Import/Export
- Math and Data Types
- ▶ Diagnostics
- Hardware Implementation**
- Model Referencing
- Simulation Target
- ▼ Code Generation
 - Optimization
 - Report
 - Comments
 - Identifiers
 - Custom Code
 - Interface
 - Code Style
 - Verification
 - Templates
 - Code Placement
 - Data Type Replacement

Hardware board: TI Delfino F2833x

Code Generation system target file: [ert.tlc](#)

Device vendor: Texas Instruments Device type: C2000

▼ Device details

Number of bits				Largest atomic size	
char:	16	short:	16	int:	16
long:	32	long long:	64	float:	32
double:	64	native:	16	pointer:	32
size_t:	32	ptrdiff_t:	32	integer:	Int
				floating-point:	None

Byte ordering: Little Endian Signed integer division rounds to: Zero

Shift right on a signed integer as arithmetic shift

Support long long

Hardware board settings

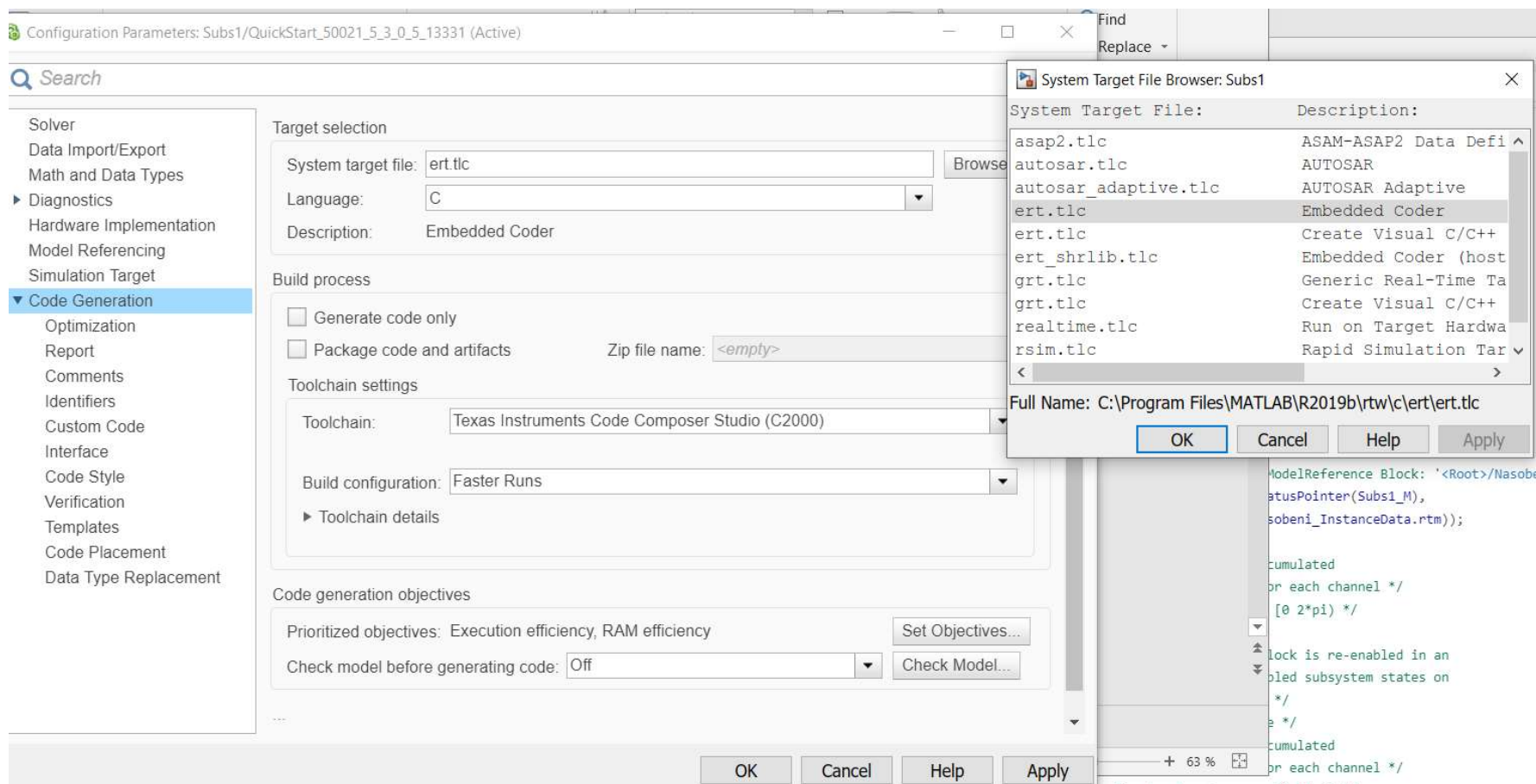
▼ Operating system/scheduler

Base rate trigger: Timer 0

▶ Target hardware resources

OK Cancel Help Apply

Konfigurace modelu pro generování kódu Code Generation (mnoho dalších nastavení)



The screenshot shows the 'Configuration Parameters' dialog for a Simulink model, specifically the 'Code Generation' section. A 'System Target File Browser' dialog is overlaid on top, showing a list of target files. The 'ert.tlc' file is selected, and its full path is shown at the bottom of the browser dialog.

System Target File Browser: Subs1

System Target File:	Description:
asap2.tlc	ASAM-ASAP2 Data Defi
autosar.tlc	AUTOSAR
autosar_adaptive.tlc	AUTOSAR Adaptive
ert.tlc	Embedded Coder
ert.tlc	Create Visual C/C++
ert_shrllib.tlc	Embedded Coder (host
grt.tlc	Generic Real-Time Ta
grt.tlc	Create Visual C/C++
realtime.tlc	Run on Target Hardwa
rsim.tlc	Rapid Simulation Tar

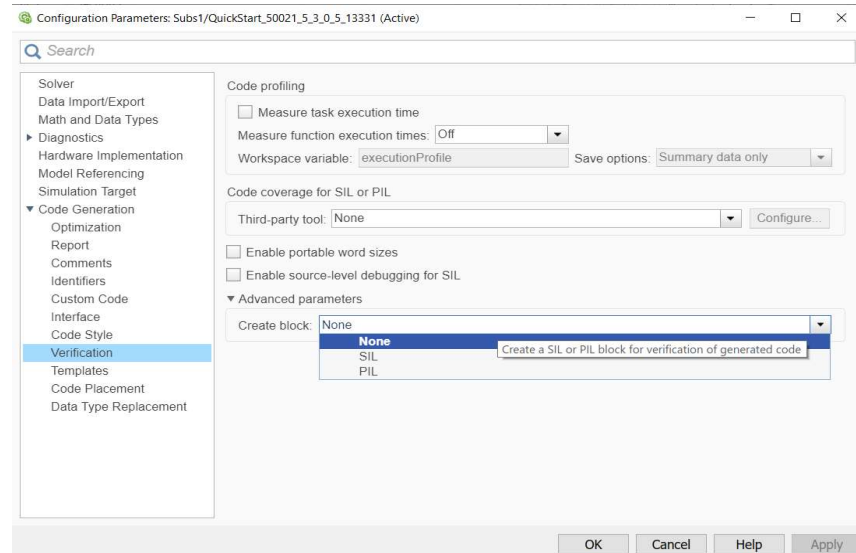
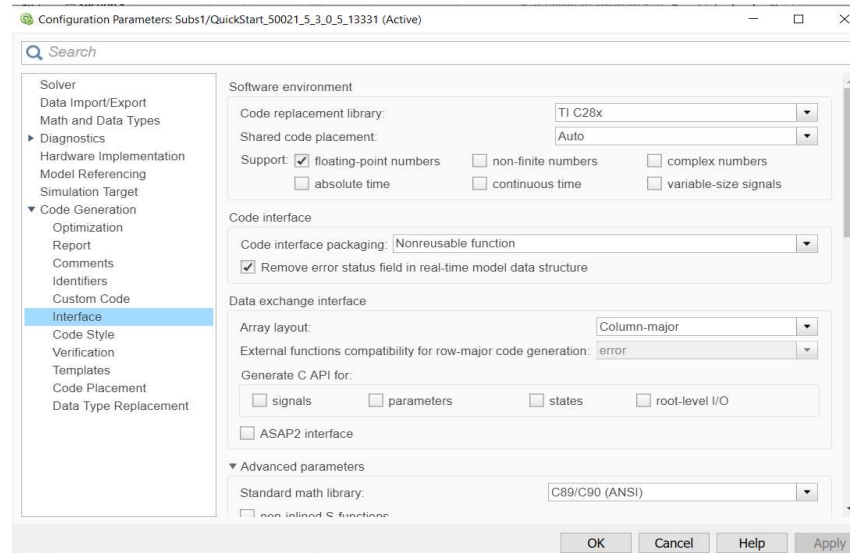
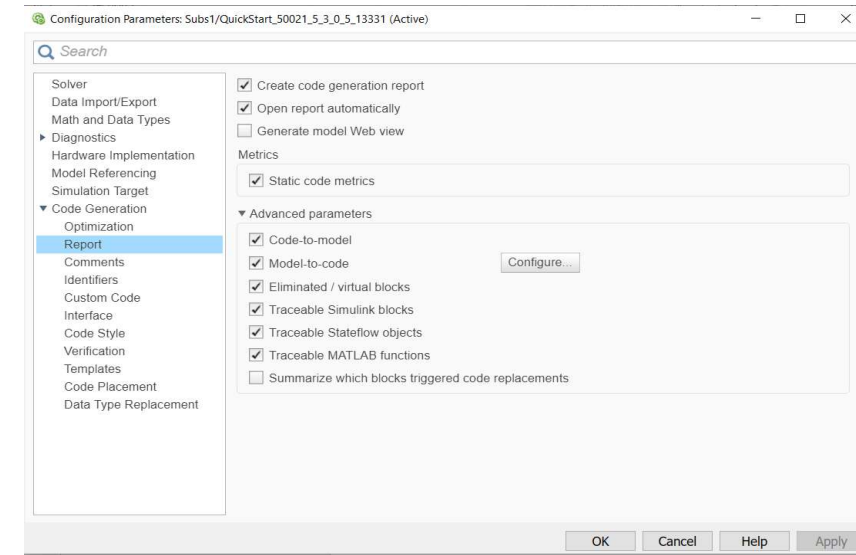
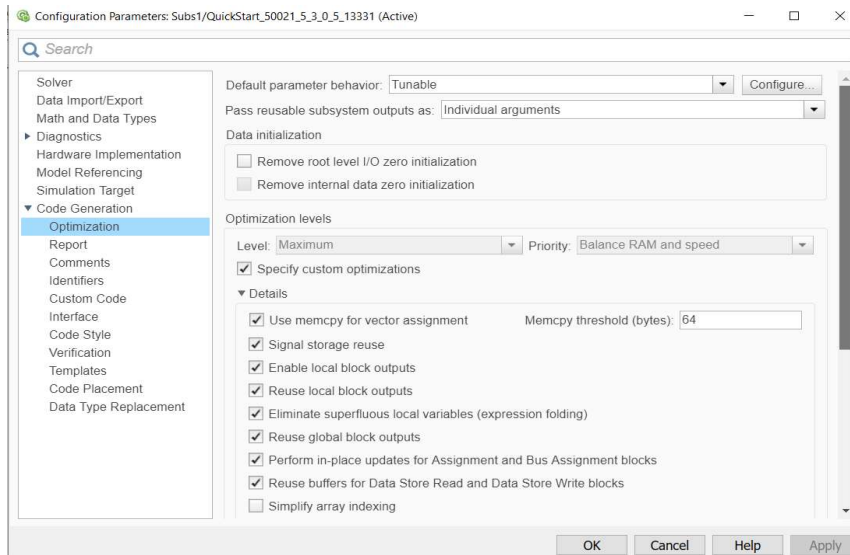
Full Name: C:\Program Files\MATLAB\R2019b\rtw\c\ert\ert.tlc

Buttons: OK, Cancel, Help, Apply

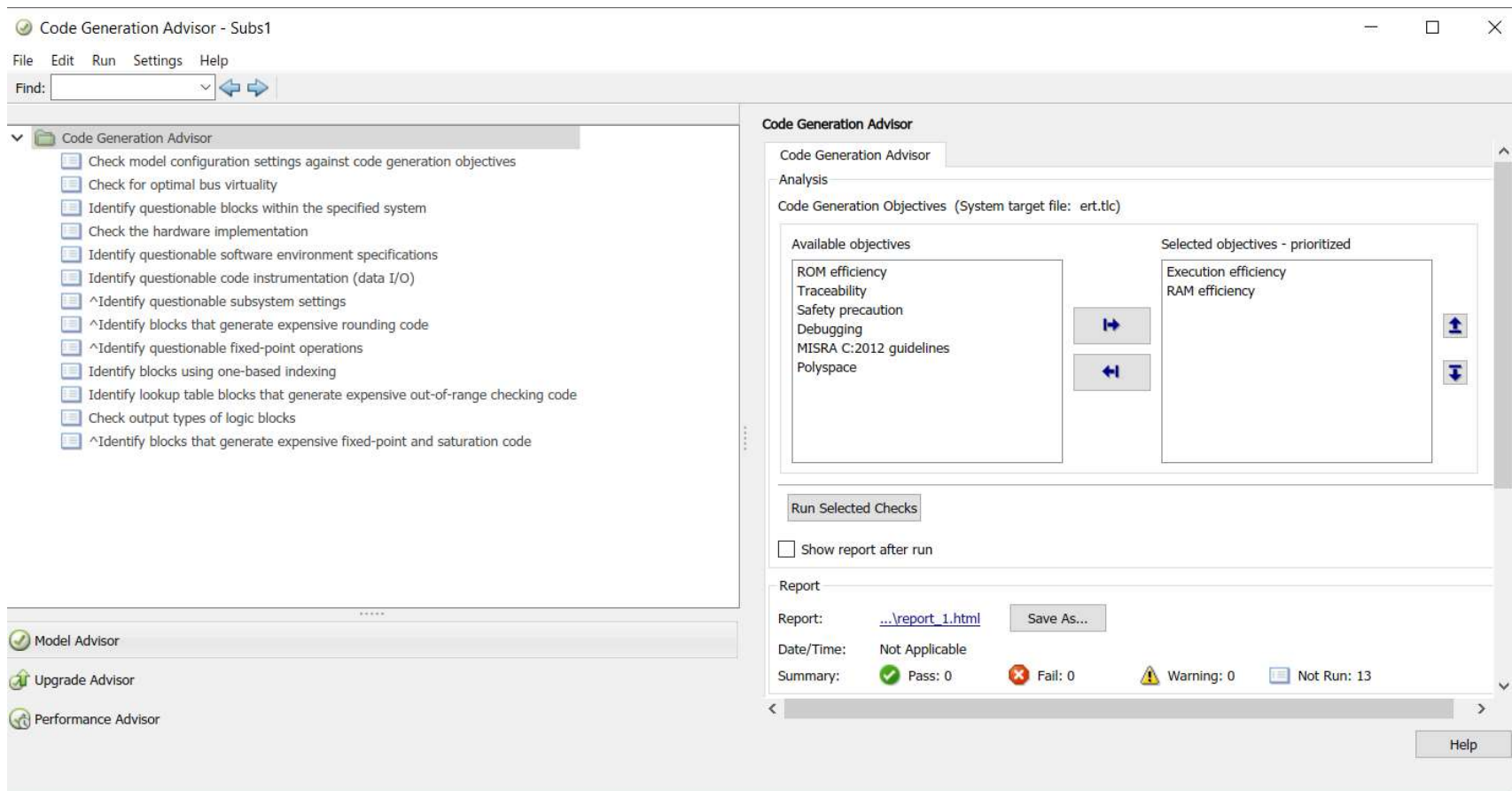
Main Configuration Parameters Dialog (Code Generation section):

- Target selection:** System target file: ert.tlc, Language: C, Description: Embedded Coder
- Build process:**
 - Generate code only
 - Package code and artifacts (Zip file name: <empty>)
- Toolchain settings:**
 - Toolchain: Texas Instruments Code Composer Studio (C2000)
 - Build configuration: Faster Runs
 - Toolchain details: [expanded]
- Code generation objectives:**
 - Prioritized objectives: Execution efficiency, RAM efficiency (Set Objectives...)
 - Check model before generating code: Off (Check Model...)

Generování aplikačního SW ze Simulinku



Konfigurace modelu pro generování kódu – Code generation advisor



Code Generation Advisor - Subs1

File Edit Run Settings Help

Find:

Code Generation Advisor

- Check model configuration settings against code generation objectives
- Check for optimal bus virtuality
- Identify questionable blocks within the specified system
- Check the hardware implementation
- Identify questionable software environment specifications
- Identify questionable code instrumentation (data I/O)
- ^Identify questionable subsystem settings
- ^Identify blocks that generate expensive rounding code
- ^Identify questionable fixed-point operations
- Identify blocks using one-based indexing
- Identify lookup table blocks that generate expensive out-of-range checking code
- Check output types of logic blocks
- ^Identify blocks that generate expensive fixed-point and saturation code

Code Generation Advisor

Code Generation Advisor

Analysis

Code Generation Objectives (System target file: ert.tlc)

Available objectives

- ROM efficiency
- Traceability
- Safety precaution
- Debugging
- MISRA C:2012 guidelines
- Polyspace

Selected objectives - prioritized

- Execution efficiency
- RAM efficiency

Run Selected Checks

Show report after run

Report

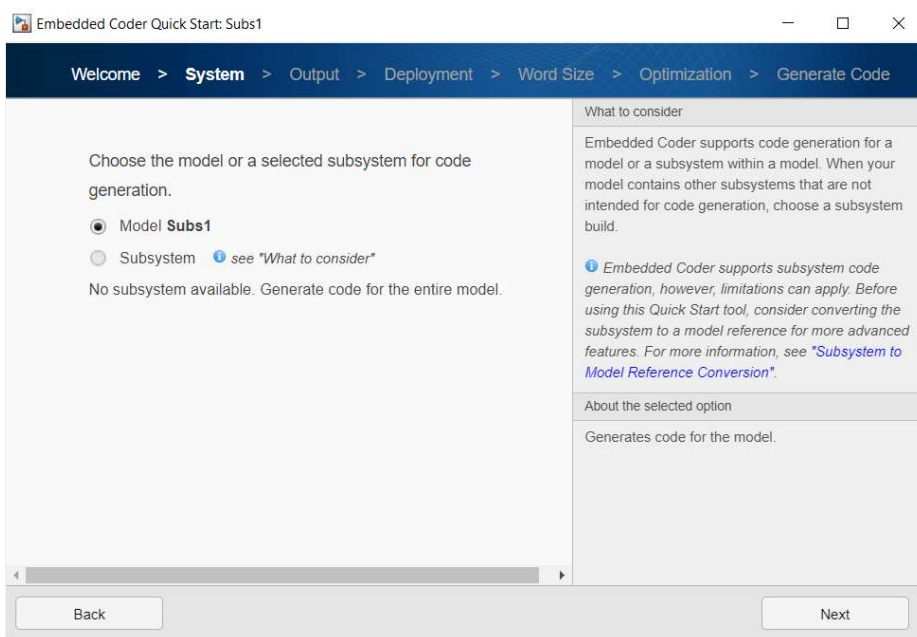
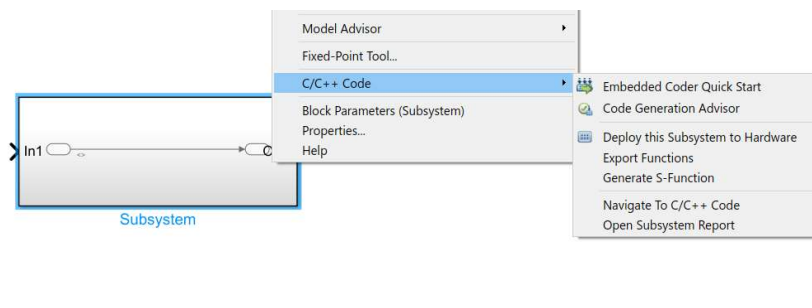
Report: [...report_1.html](#) Save As...

Date/Time: Not Applicable

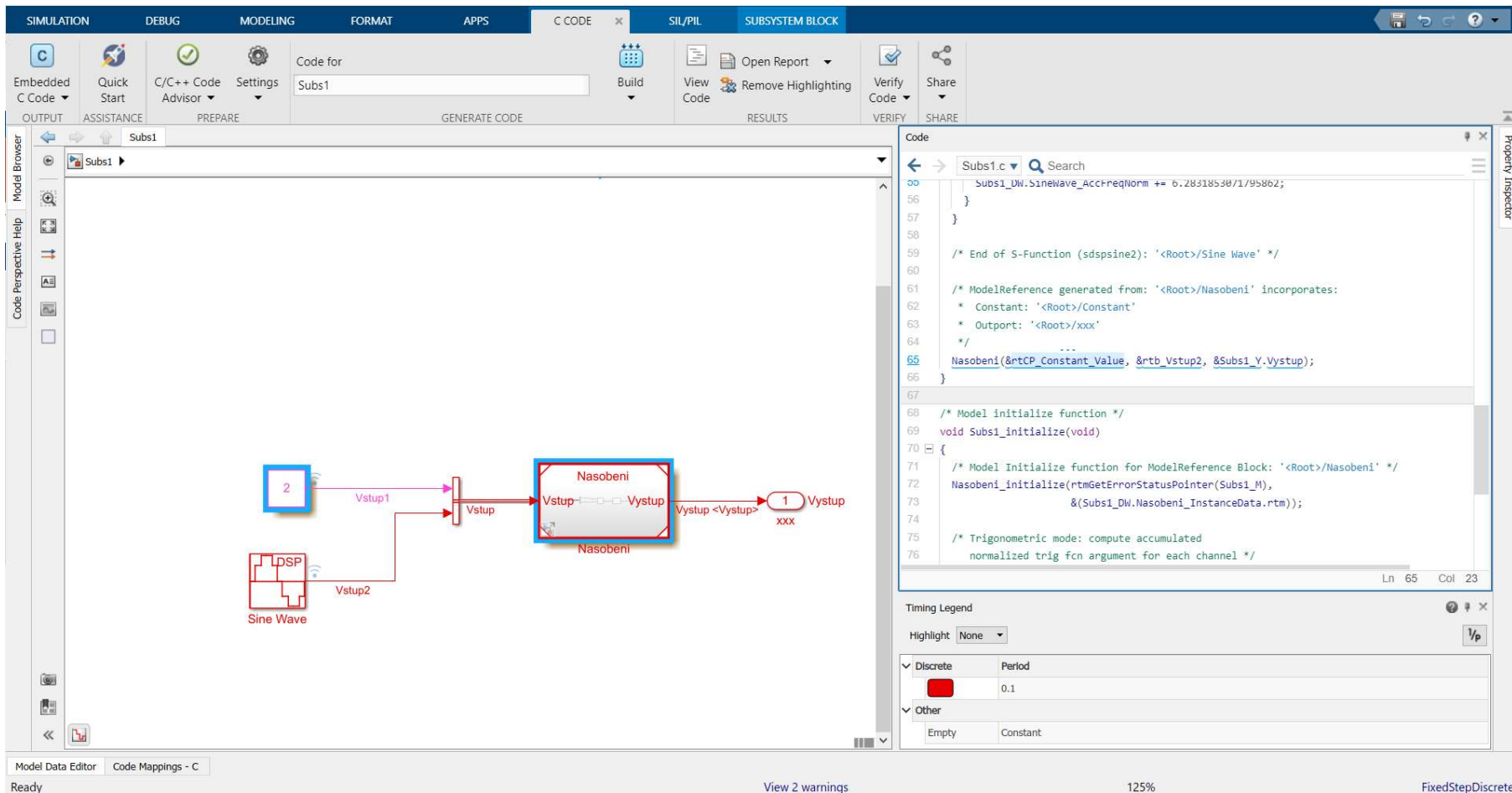
Summary: ✔ Pass: 0 ✘ Fail: 0 ⚠ Warning: 0 📄 Not Run: 13

Help

Generování kódu = pro jednoduché nastavení Embedded Coder Quick Start



Práce se Simulinkem a vygenerovaným kódem



The screenshot displays the MATLAB/Simulink environment. The main workspace shows a Simulink model with two input ports, 'Vstup1' and 'Vstup2', feeding into a block labeled 'Nasobeni'. The output of the 'Nasobeni' block is 'Vystup', which is connected to a scope '1'.

The right-hand pane shows the generated C code for the 'Nasobeni' block. The code includes comments and function definitions:

```

Code
Subs1.c
56     Subs1_DW.SineWave_AccFreqNorm += 6.2831853071795862;
57   }
58 }
59 /* End of S-Function (sdsp sine2): '<Root>/Sine Wave' */
60
61 /* ModelReference generated from: '<Root>/Nasobeni' incorporates:
62  * Constant: '<Root>/Constant'
63  * Outport: '<Root>/xxx'
64  */
65 Nasobeni(&rtCP_Constant_Value, &rtb_Vstup2, &Subs1_Y.Vystup);
66 }
67
68 /* Model initialize function */
69 void Subs1_initialize(void)
70 {
71 /* Model Initialize function for ModelReference Block: '<Root>/Nasobeni' */
72 Nasobeni_initialize(rtmGetErrorStatusPointer(Subs1_M),
73                   &(Subs1_DW.Nasobeni_InstanceData.rtm));
74
75 /* Trigonometric mode: compute accumulated
76  normalized trig fcn argument for each channel */

```

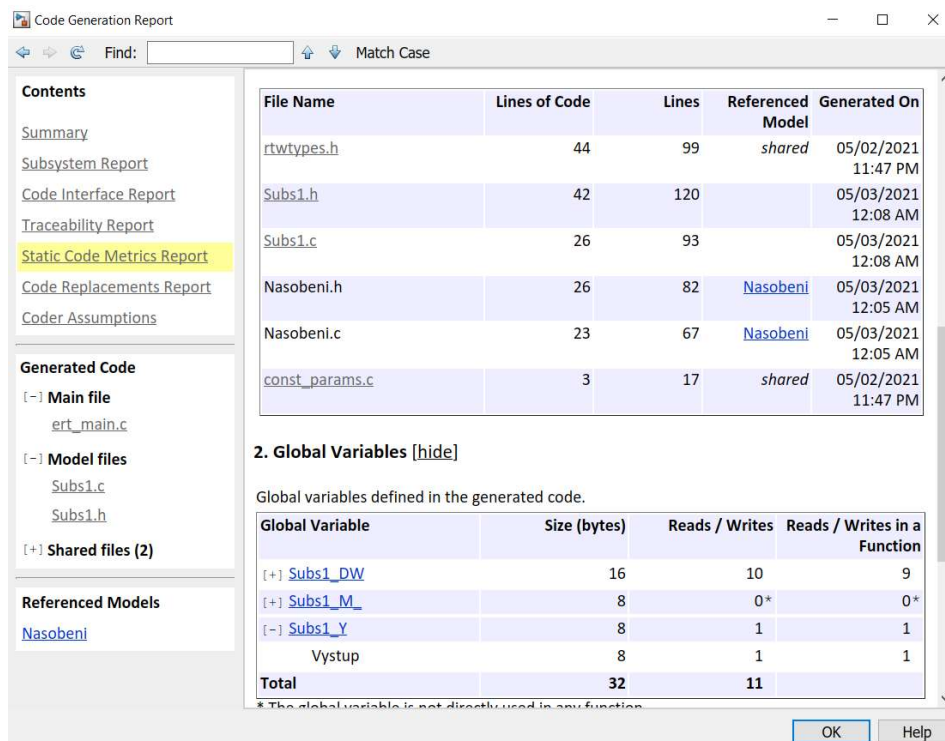
At the bottom of the code editor, there is a 'Timing Legend' section:

Discrete	Period
■	0.1
Empty	Constant

Vygenerování HTML reportu

Report je ve formátu html provázaný s modelem pomocí odkazů :

Model to Code a Code to Model



Code Generation Report

Find: Match Case

Contents

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report
- Coder Assumptions

Generated Code

- [-] Main file
 - ert_main.c
- [-] Model files
 - Subs1.c
 - Subs1.h
- [+] Shared files (2)

Referenced Models

- Nasobeni

File Name	Lines of Code	Lines	Referenced Model	Generated On
rtwtypes.h	44	99	shared	05/02/2021 11:47 PM
Subs1.h	42	120		05/03/2021 12:08 AM
Subs1.c	26	93		05/03/2021 12:08 AM
Nasobeni.h	26	82	Nasobeni	05/03/2021 12:05 AM
Nasobeni.c	23	67	Nasobeni	05/03/2021 12:05 AM
const_params.c	3	17	shared	05/02/2021 11:47 PM

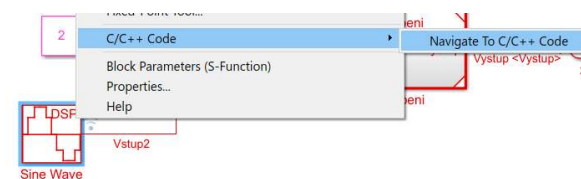
2. Global Variables [hide]

Global variables defined in the generated code.

Global Variable	Size (bytes)	Reads / Writes	Reads / Writes in a Function
[+] Subs1_DW	16	10	9
[+] Subs1_M_	8	0*	0*
[-] Subs1_Y	8	1	1
Vystup	8	1	1
Total	32	11	

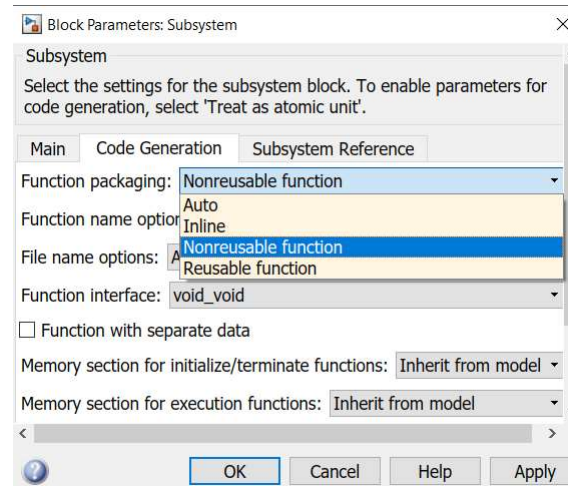
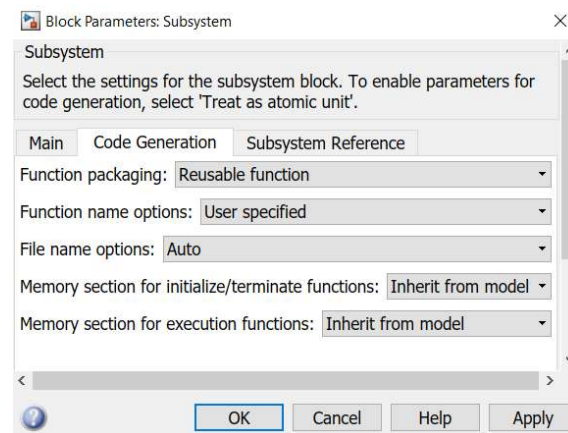
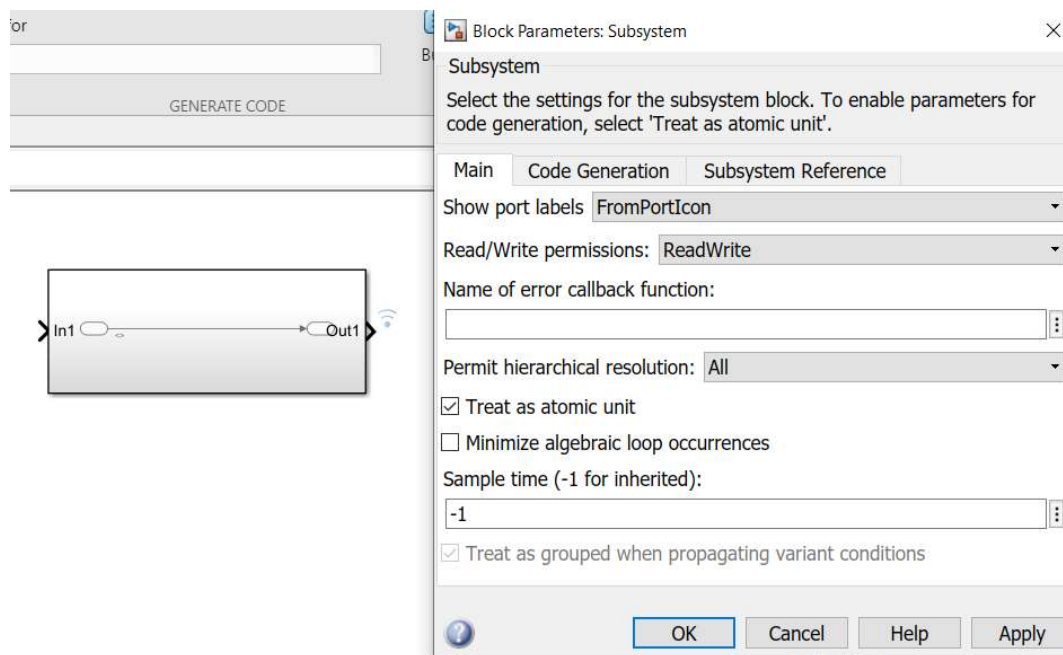
* The global variable is not directly used in any function.

OK Help



Parametry generování subsystému lze nastavit v Block parameters

Reusable function – pokud subsystém jako knihovna



Generování s referenčním modelem

Reference model = atomický uzavřený kód, který se snadno integruje, např. subsystém není implicitně atomický

Generated Code

[-] Main file
ert_main.c

[-] Model files
Subs1.c
Subs1.h

[-] Shared files
const_params.c
rtwtypes.h

Referenced Models

[Nasobeni](#)

```

28 # define Subs1_COMMON_INC
29 #include "rtwtypes.h"
30 #endif
31
32 /* Child system includes
33 #include "Nasobeni.h"
34
35 /* Macros for accessing r
36 #ifndef rtmGetErrorStatus
37 # define rtmGetErrorStatu:
38 #endif
39
40 #ifndef rtmSetErrorStatus
41 # define rtmSetErrorStatu:
42 #endif
43
44 #ifndef rtmGetErrorStatus1
45 # define rtmGetErrorStatu:

```

[Coder Assumptions](#)

Generated Code

[-] Model files
Nasobeni.c
Nasobeni.h

[-] Shared files
rtwtypes.h

```

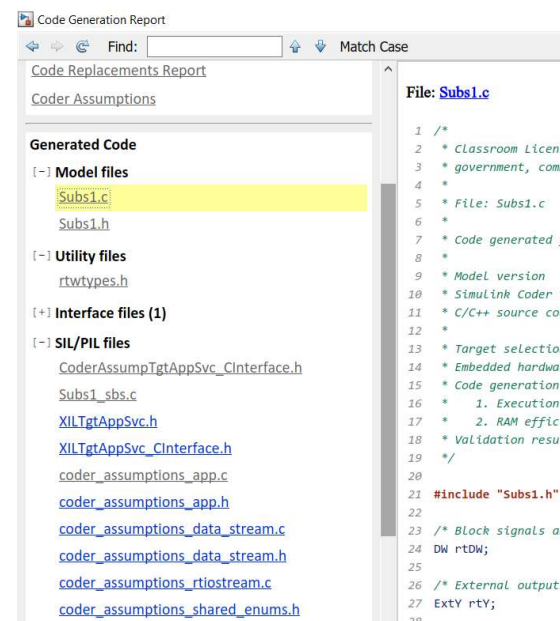
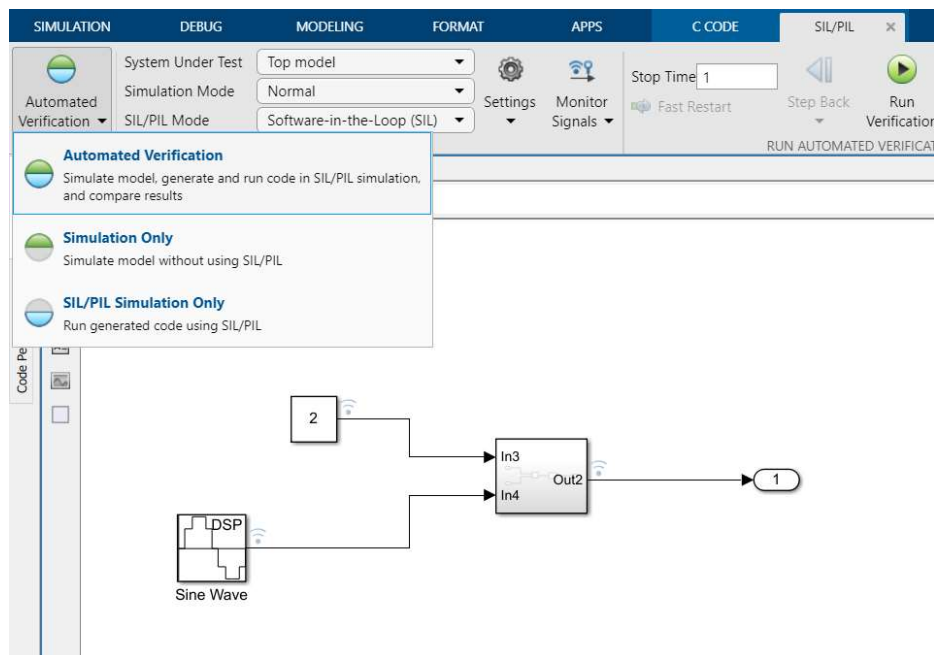
--
42 /* Output and update for referenced model: 'Nasobeni' */
43 void Nasobeni(const real_T *rtu_In1_Vstup1, const real_T *rtu_In1_Vstup2, real_T
44             *rty_Out3)
45 {
46     /* Bias: '<Root>/Bias' incorporates:
47      * Product: '<Root>/Product'
48      */
49     *rty_Out3 = *rtu_In1_Vstup1 * *rtu_In1_Vstup2 + 10.0;
50 }
51
52 /* Model initialize function */
53 void Nasobeni_initialize(const char_T **rt_errorStatus, RT_MODEL_Nasobeni_T *

```

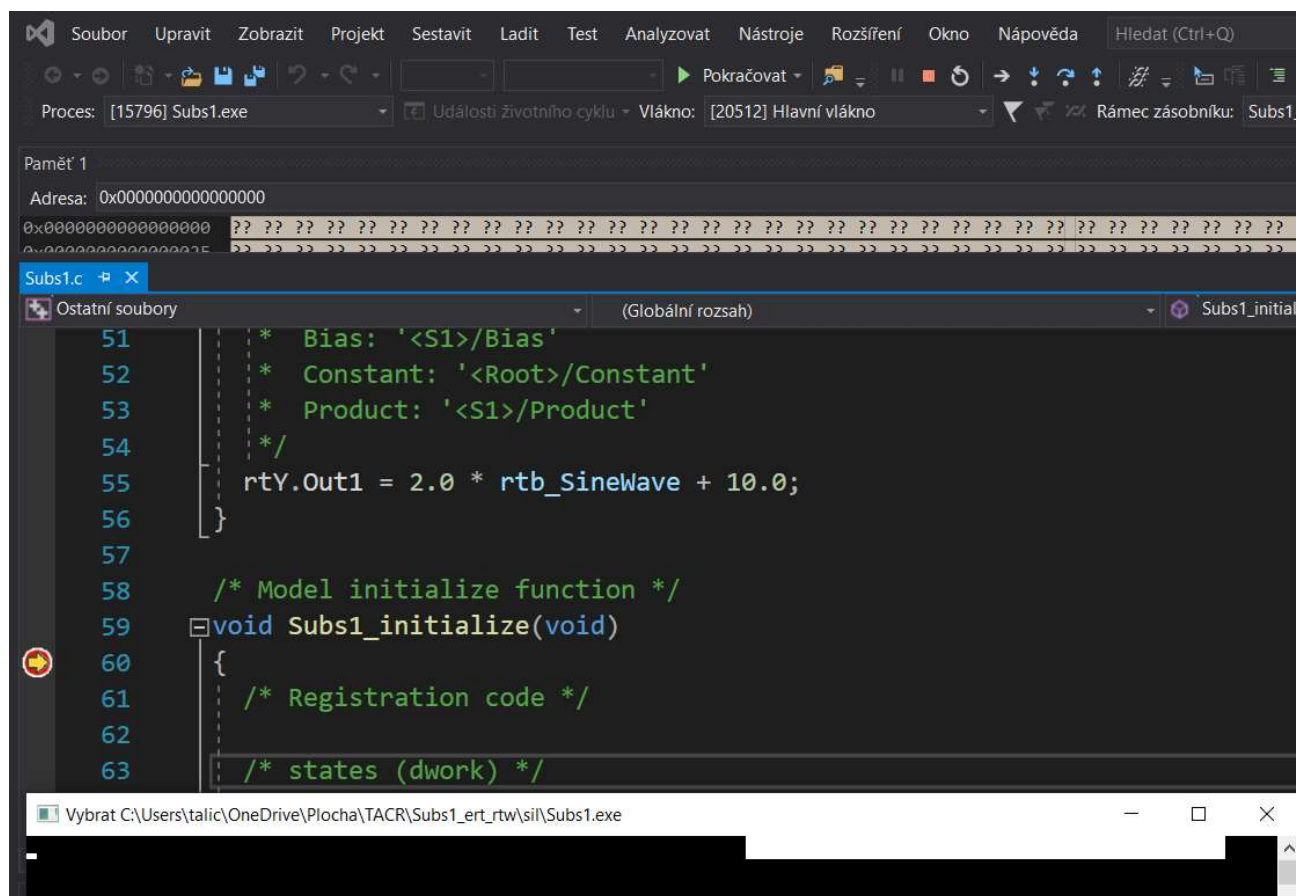
Testování SIL

Otestování SW pomocí automatizované verifikace, SIL a Data Inspector

Simulink automaticky vygeneruje interface s vygenerovaným kódem subsystému

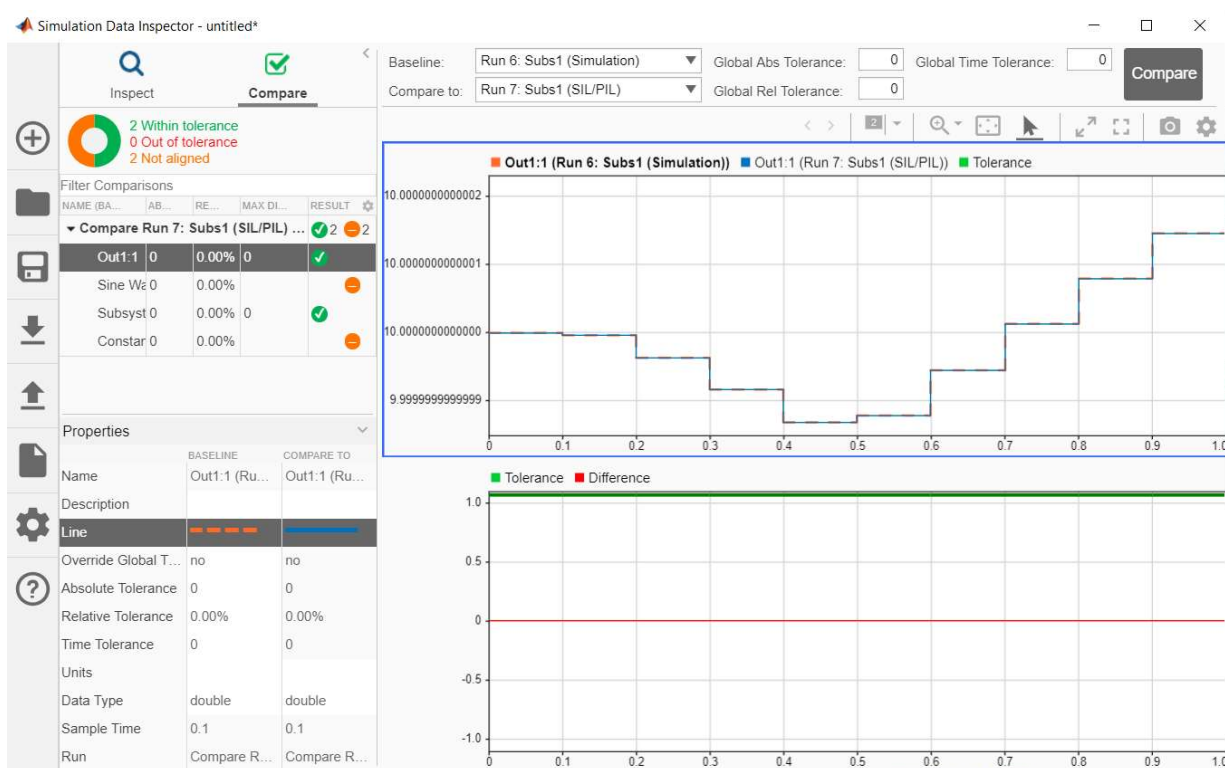


Je vygenerován projekt, vytvořen spustitelný soubor a automaticky laděn ve Visual studiu, krok po kroku, výsledky posílány zpět do Simulinku do Data Inspect



```
51      * Bias: '<S1>/Bias'  
52      * Constant: '<Root>/Constant'  
53      * Product: '<S1>/Product'  
54      */  
55      rtY.Out1 = 2.0 * rtb_SineWave + 10.0;  
56  }  
57  
58  /* Model initialize function */  
59  void Subs1_initialize(void)  
60  {  
61      /* Registration code */  
62  
63      /* states (dwork) */
```

Otestování SW pomocí automatizované verifikace, SIL a Data Inspector



Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Příští přednáška

Cílové automatické generování kódu ze Simulinku

Jakub Talla

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