FAKULTA ELEKTROTECHNICKÁ ZÁPADOČESKÉ UNIVERZITY V PLZNI

RICE

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Návrh software a automatické generování kódů v Matlab Simulink

Část 2., uživatelské funkce

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. 4

1) Globální proměnná v Simulinku (Data Store)

- Globální proměnná typu Data Store
- Nastavení priorit bloků

2) Uživatelské funkce Simulinku

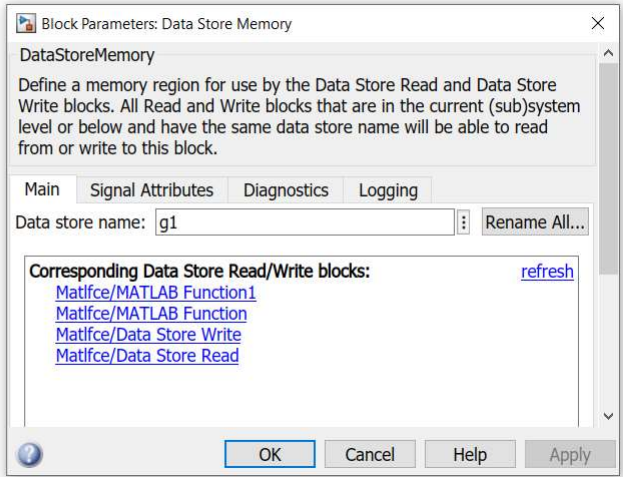
- Simulink function
- Matlab function
- S-function
- Stateflow

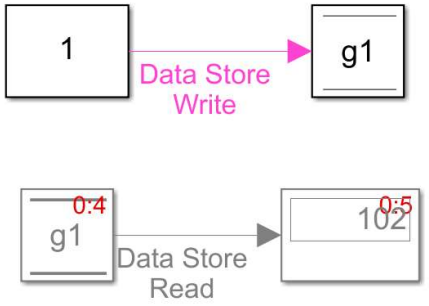
Globální proměnná v Simulinku

Globální proměnná v Simulinku

Globální proměnná v Simulinku (Data Store Memory)

- Globální proměnná se deklaruje a inicializuje pomocí bloku Data Store Memory
- Pro práci s globálníma proměnnýma slouží bloky Data Store Read/Write
- Globální proměnné mohou být přístupné i např. z Matlab funkce

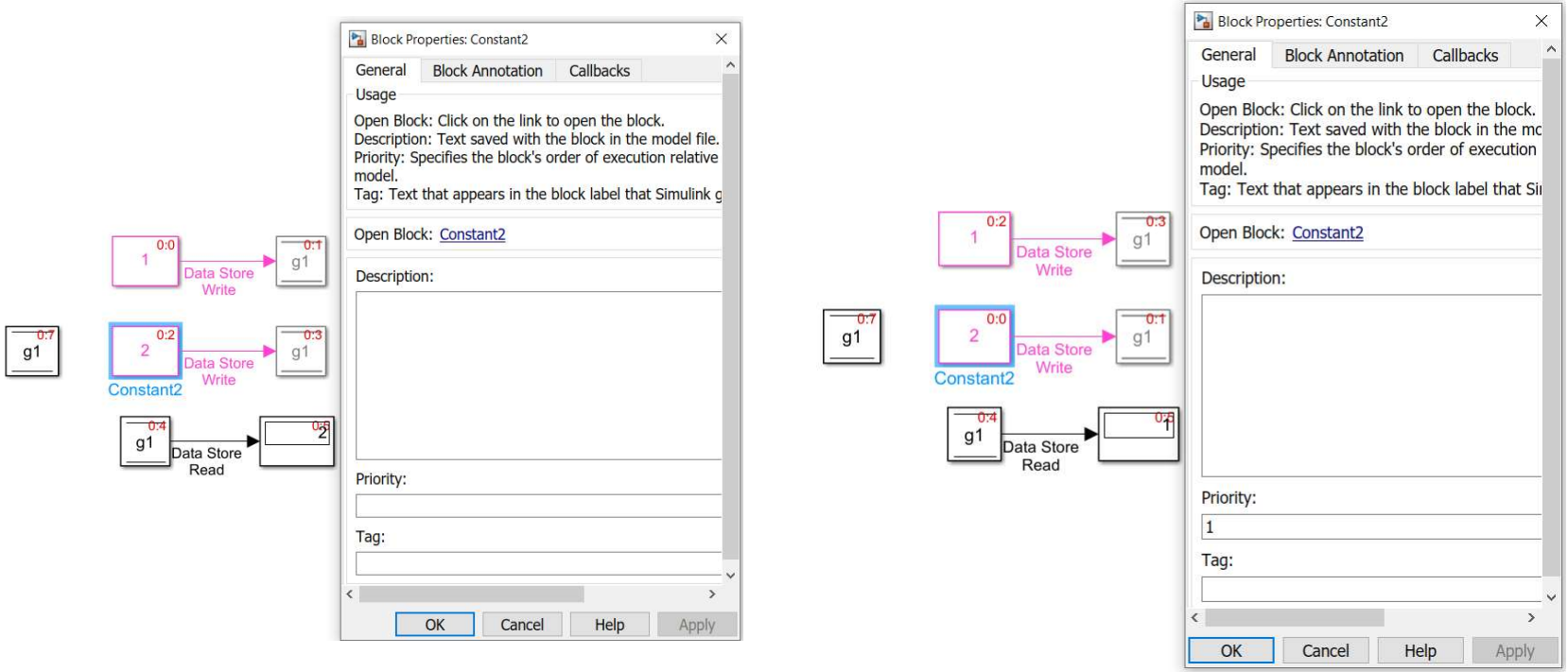




Globální proměnná v Simulinku

Nastavení priorit provádění bloků

- Pořadí provádění bloků lze ovlivnit nastavením priority pomocí záložky Block Properties (vyšší číslo vyšší priorita)
- To může mít veliké dopady speciálně při práci s globálními proměnnými



Tvorba vlastních uživatelských funkcí v Simulinku

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI
OP Výzkum a vývoj pro inovace

Matlab + Simulink

Tvorba vlastních uživatelských funkcí v Simulinku s podporou automatického generování kódu:

- **1) Pomocí základních Simulinkovských bloků**
- **2) Matlab function (pomocí jazyka Matlab)**
- **3) S-function (pomocí C/C++)**
- **4) Stateflow (pomocí grafického prostředí stavových automatů)**

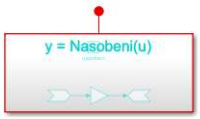
1) Pomocí základních Simulinkovských bloků (Funkce v Simulinku)

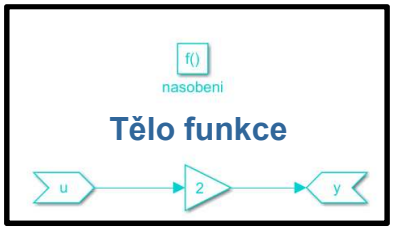
Simulink function

Simulink function (Simulinkovská funkce)

- Simulink function – umožňuje vytvořit funkci ze základních bloků v Simulinku
- Umožňuje automatické generování kódu (C/C++,VHDL)

Tělo funkce





Volání funkce

Volání funkce

Tělo funkce

```
void funkce_Nasobeni(real_T rtu_u, real_T *rty_y)
{
  /* SignalConversion: '<S2>/TmpSignal ConversionAtyInport1' incorporates:
   *  Gain: '<S2>/Gain'
   *  SignalConversion: '<S2>/TmpSignal ConversionAtuOutport1'
   */
  *rty_y = 2.0 * rtu_u;
}

/* Model step function */
void funkce_step(void)
{
  real_T rtb_FunctionCaller1;

  /* FunctionCaller: '<S1>/Function Caller' incorporates:
   *  Inport: '<Root>/In1'
   */
  funkce_Nasobeni(funkce_U.In1, &rtb_FunctionCaller1);

  /* FunctionCaller: '<S1>/Function Caller1' */
  funkce_Nasobeni(rtb_FunctionCaller1, &rtb_FunctionCaller1);

  /* Outport: '<Root>/Out1' */
  funkce_Y.Out1 = rtb_FunctionCaller1;
}
```

Volání funkce

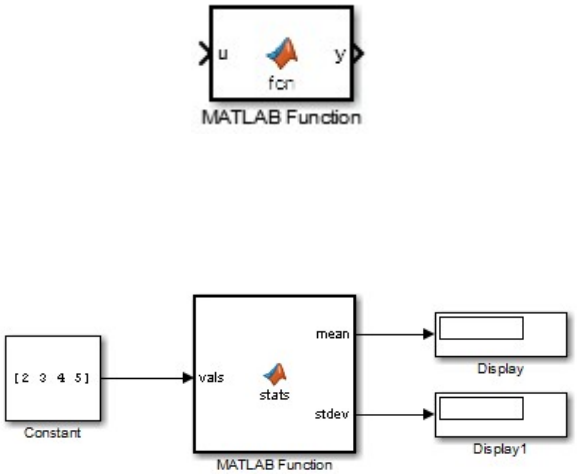
Volání funkce

2) Matlab function

Matlab function

Matlab function (Matlabovská funkce)

- Matlab function – umožňuje implementaci Matlab kódu v Simulinku
- Umožňuje automatické generování kódu (C/C++,VHDL)



Vstupní entry fce

Výstupy fce Název fce Vstupy fce

```
1   function [mean,stdev] = stats(vals)
2   % #codegen
3
4   % calculates a statistical mean and a standard
5   % deviation for the values in vals.
6
7 -  len = length(vals);
8 -  mean = avg(vals,len);
9 -  stdev = sqrt(sum(((vals-avg(vals,len)).^2))/len);
10 -  coder.extrinsic('plot');
11 -  plot(vals,'-+');
12
13   function mean = avg(array,size)
14 -  mean = sum(array)/size;
```

Matlab fce

Lokální fce

Matlab function

Matlab function (a Matlab) a typy proměnných

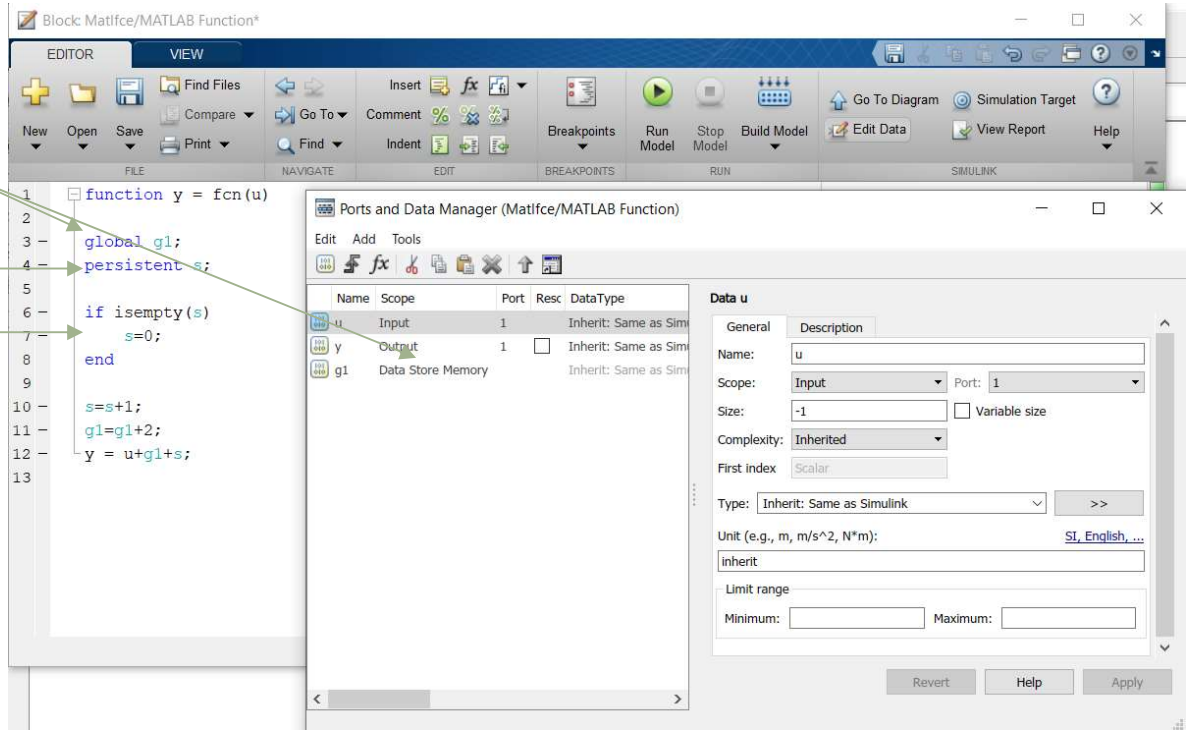
- global – globální proměnná, musí být v Ports and Data jako Data Store Memory a deklarovaná a inicializovaná pomocí bloku Data Store Memory v Simulinku
- persistent – statická lokální proměnná, musí být inicializována pomocí funkce isempty()

Globální proměnná

Lokální statická proměnná

Inicializace statické proměnné

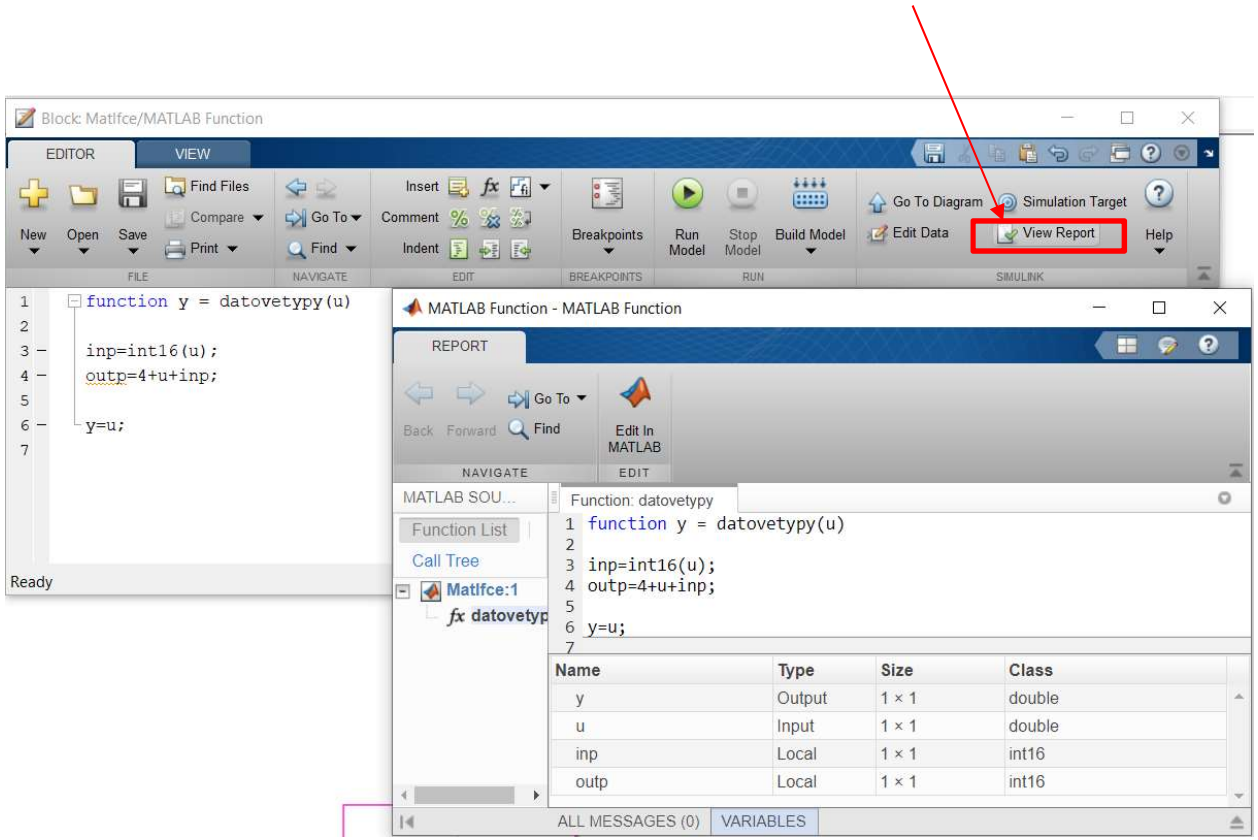
Data Store Memory



Matlab function

Matlab function (a Matlab) a datové typy

- Implicitní datový typ pro proměnné i konstanty je double
- Pro analýzu datových typů je vhodné zapnout View Report

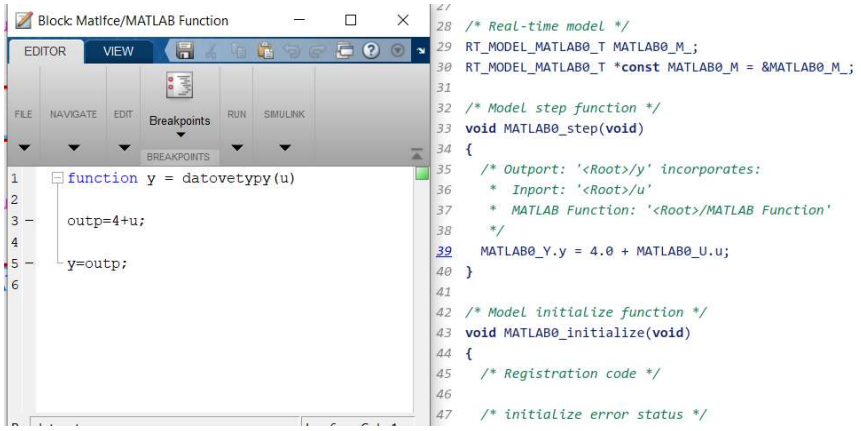


Matlab function

Matlab function (a Matlab) a datové typy

- Použité datové typy v Matlab function mají dopady do vygenerovaného kódu

Vst. proměnná *u* double, součet v doublech



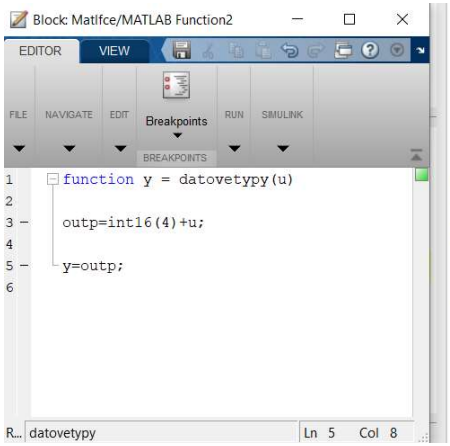

```c
/* Real-time model */
RT_MODEL_MATLAB0_T MATLAB0_M_;
RT_MODEL_MATLAB0_T *const MATLAB0_M = &MATLAB0_M_;

/* Model step function */
void MATLAB0_step(void)
{
  /* Outport: '<Root>/y' incorporates:
   *  Inport: '<Root>/u'
   *  MATLAB Function: '<Root>/MATLAB Function'
   */
  MATLAB0_Y.y = 4.0 + MATLAB0_U.u;
}

/* Model initialize function */
void MATLAB0_initialize(void)
{
  /* Registration code */

  /* initialize error status */
```

Vst. proměnná *u* int16, součet v integerech (zapnuté saturate on integer overflow)




```c
RT_MODEL_MATLAB0_T *const MATLAB0_M = &MATLAB0_M_;

/* Model step function */
void MATLAB0_step(void)
{
  /* MATLAB Function: '<Root>/MATLAB Function2' incorporates:
   *  Inport: '<Root>/u'
   */
  if (MATLAB0_U.u > 32763) {
    /* Outport: '<Root>/y' */
    MATLAB0_Y.y = MAX_int16_T;
  } else {
    /* Outport: '<Root>/y' */
    MATLAB0_Y.y = 4 + MATLAB0_U.u;
  }

  /* End of MATLAB Function: '<Root>/MATLAB Function2' */
}

/* Model initialize function */
void MATLAB0_initialize(void)
```

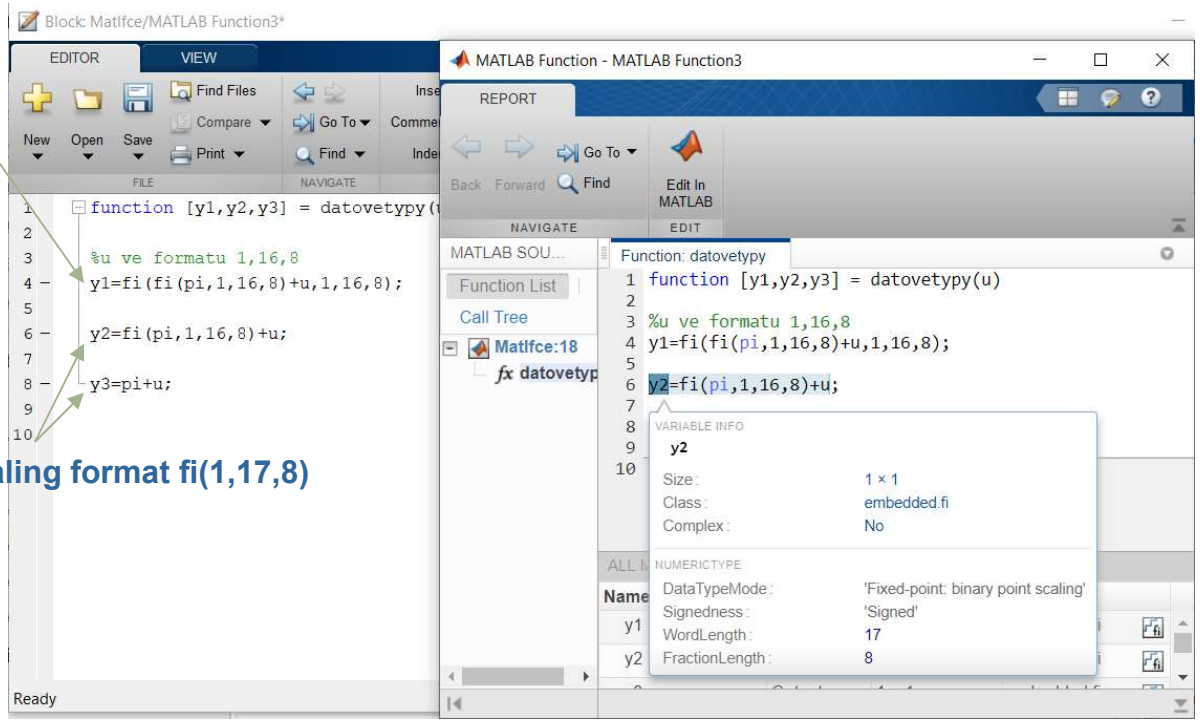
Matlab function

Matlab function (a Matlab) a pevná datová čárka

- Formáty výpočtů (součtů, násobení apod.) se nastavují implicitně fixed point aritmetikou

Binary point scaling format fi(1,16,8)

Binary point scaling format fi(1,17,8)

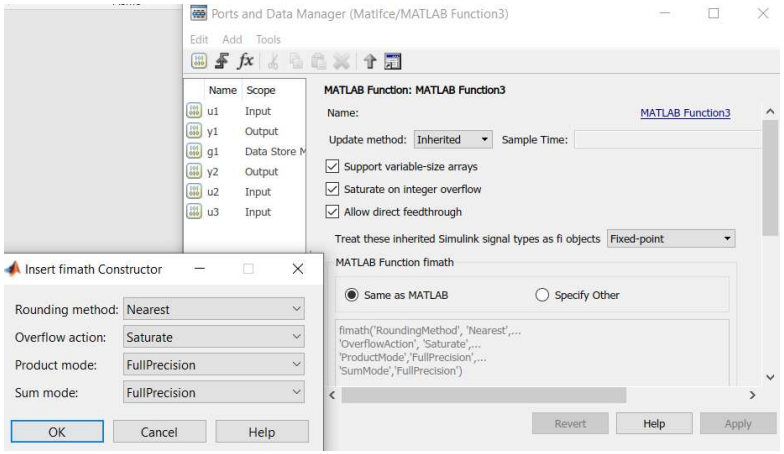


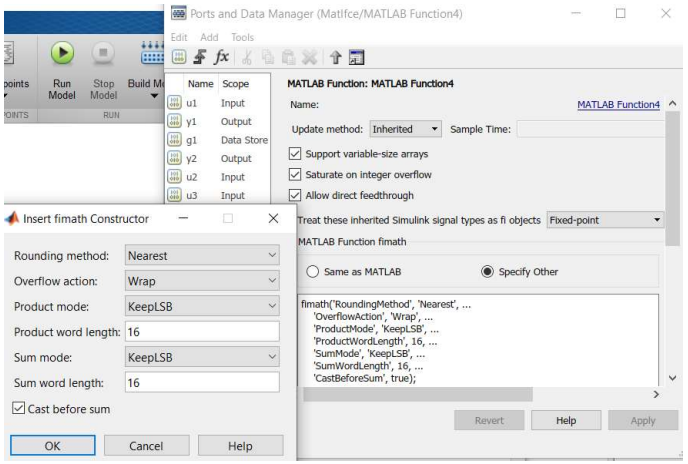
Proměnné ve vygenerovaném kódu pro TI C2000

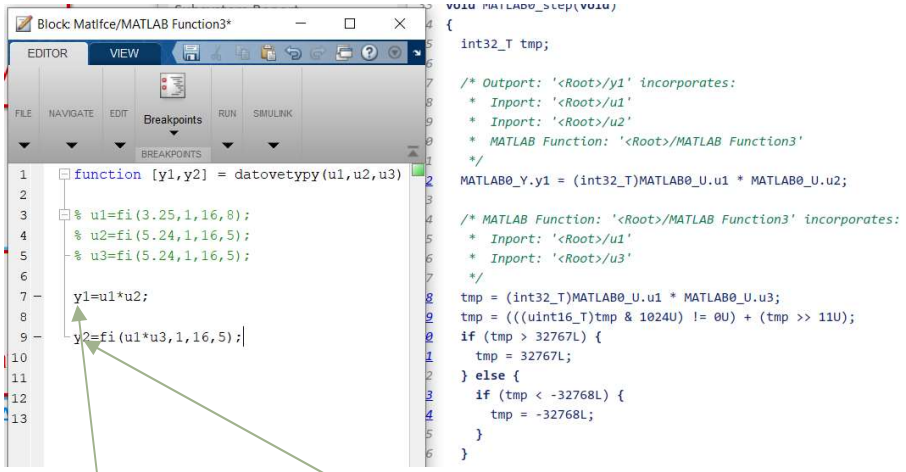
```
typedef struct {
  int16_T y1;
  int32_T y2;
  int32_T y3;
} ExtY_MATLAB0_T;
```

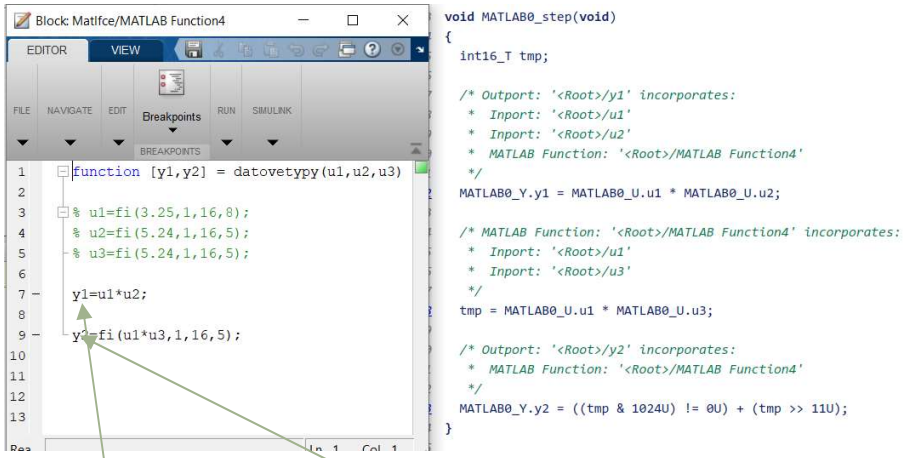
Matlab function

Matlab function (a Matlab) a pevná datová čárka – nastavení fixp. aritmetiky









Binary point scaling format fi(1,32,16)

Binary point scaling format fi(1,16,5)

Binary point scaling format fi(1,16,16)

Binary point scaling format fi(1,16,5)

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
EVROPSKÁ UNIE EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ INVESTICE DO VAŠÍ BUDOUCNOSTI
OP Výzkum a vývoj pro inovace

3) S-function

S-function

S-function (Systémová-funkce)

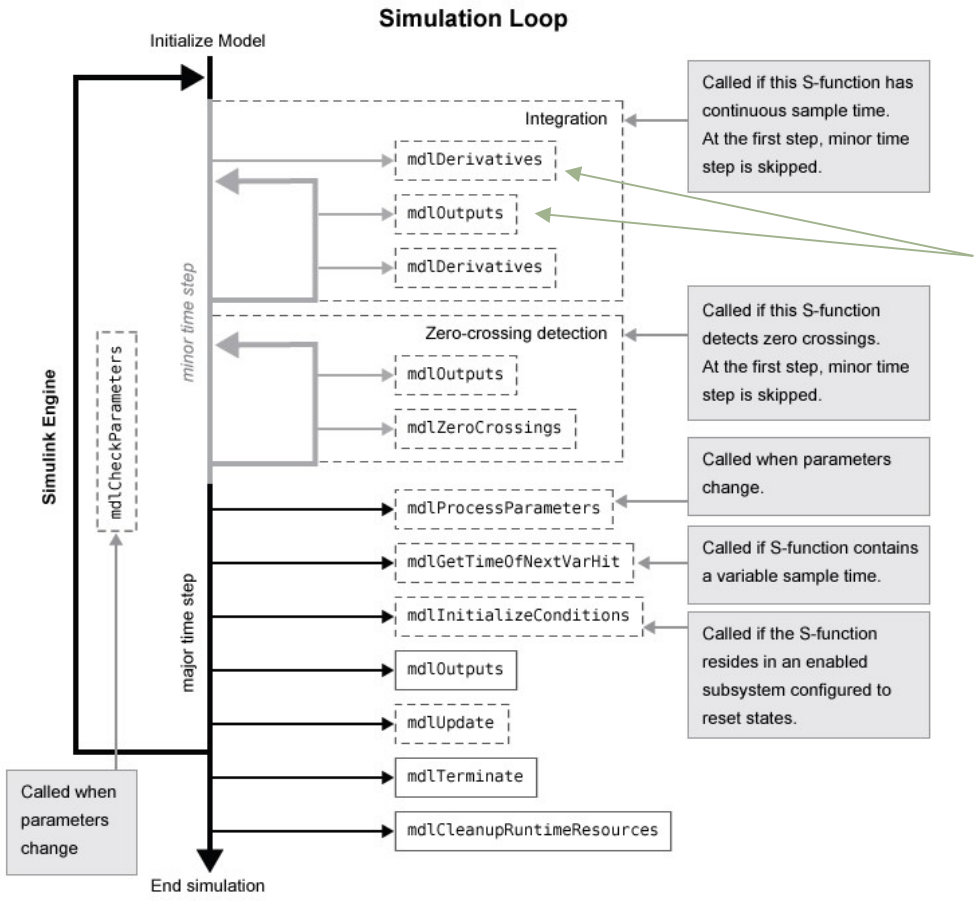
S-funkce je způsob jak vytvořit blok v Simulinku pomocí Matlabu, C, C++, nebo Fortranu

Varianty S-funkcí:

- **Level-2 Matlab S-function** (blok slouží k manuálnímu importu napsané S-funkce)
- **C MEX S-Function** (slouží k importu mex S-funkce vytvořené v Matlabu z vlastního C)
- **The S-Function Builder** (GUI pro jednoduchou tvorbu S-funkce z vlastního C)
- **The Legacy Code Tool** (tool se sadou příkazů na jednoduchou tvorbu S-fce v Matlabu)

S-function

Základní průběh S-funkce (a C-funkce)



Průběh výpočtu S-funkce odpovídá základnímu průběhu simulace Simulinku.

Jednotlivé funkce v jazyce C popisující danou S-funkci

Typická S-funkce má stovky řádek C kódu proto je výhodné využívat např. S-function builder

S-funkce umožňuje pracovat s diskrétními i spojitými stavy a i detekci zero-crossing pro spojité stavy

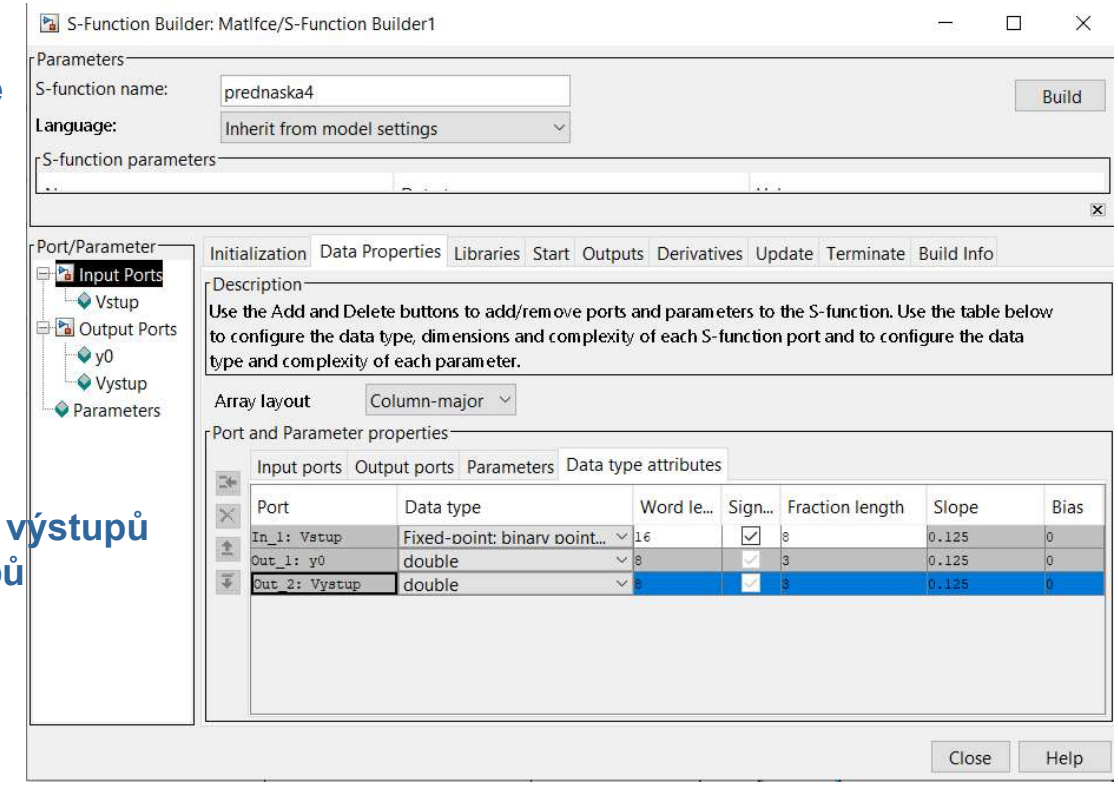
S-Function Builder

S-Function Builder – prostředí, vstupy, výstupy

S-Function umožňuje práci s reálnými, komplexními proměnnými, skaláry, vektory, maticemi, a veškerými datovými typy (int, double, fixed point atd.)

Název S-funkce

Nastavení vstupů, výstupů a datových typů



S-Function Builder

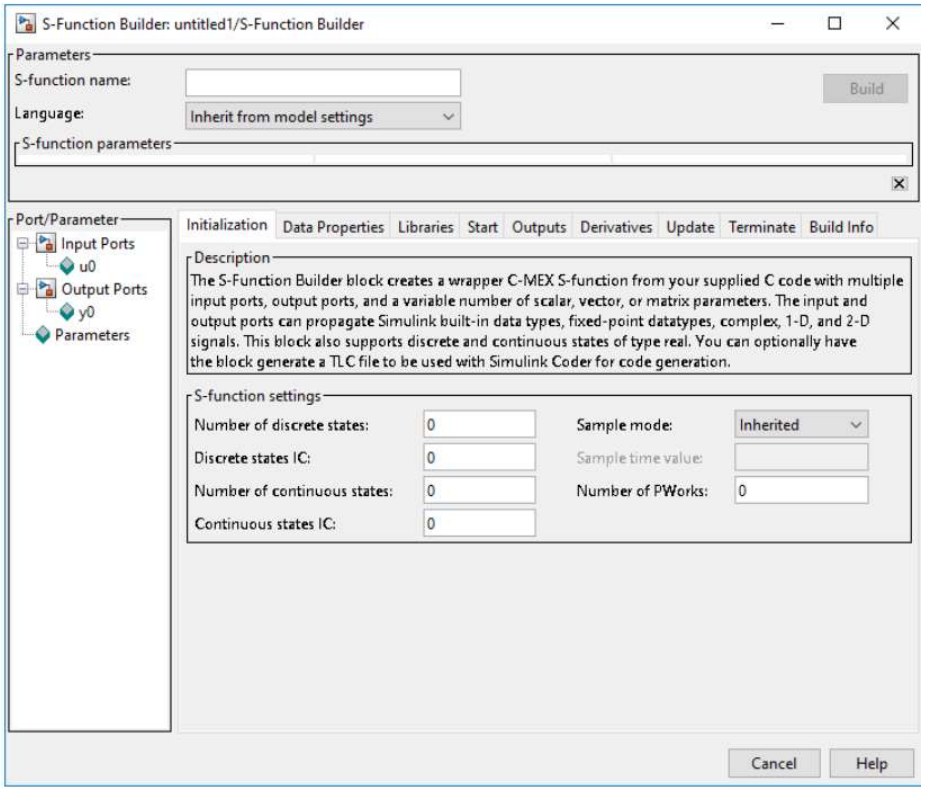
S-Function Builder

Vygenerované soubory:

- **sfun.c** (zdrojový C kód S-fce, obsahuje volání wrappovací/obalové funkce)
- **sfun_wrapper.c** (tzv. obalová funkce, obsahuje veškerý C kód napsaný v S-function builderu)
- **sfun.tlc** (slouží ke správnému vygenerování kódu S-funkce a jeho zařazení v rámci vygenerovaného kódu)

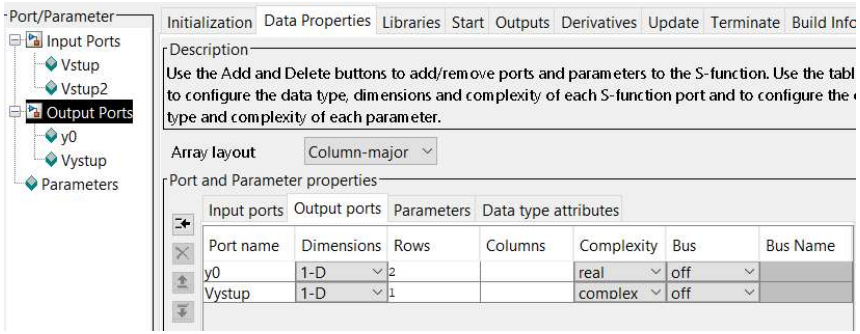
Zkomplilovaná S-funkce:

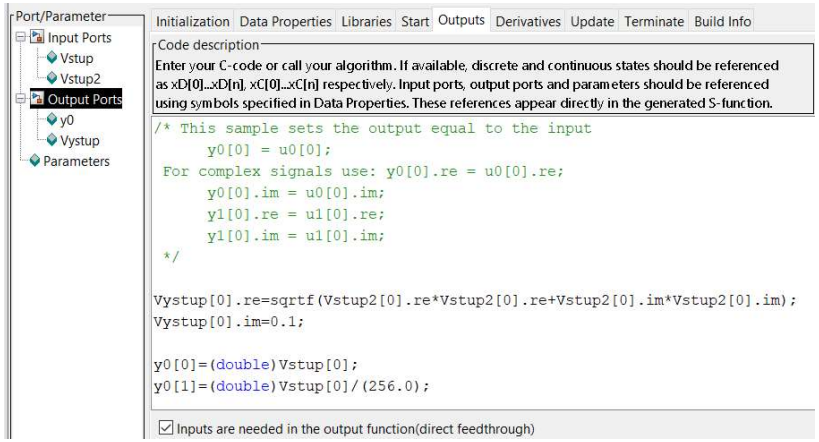
- **sfun.mexw64** (zkompilovaná a spustitelná S-funkce pomocí 64 bitového compileru pro Windows)

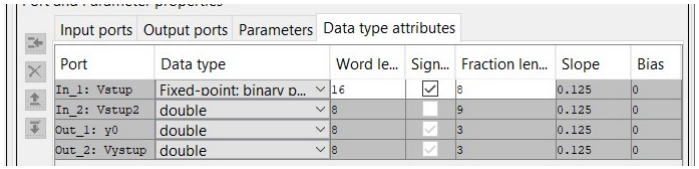


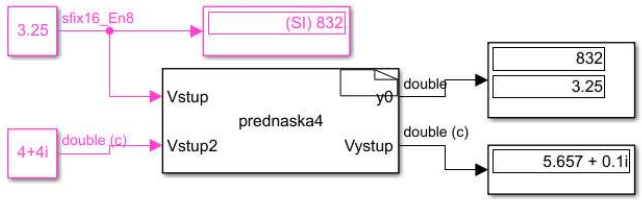
S-Function Builder – práce se Simulinkovskými datovými typy

Základní kód se píše do Outputs. Z něj je poté vytvořena wrappovací funkce (obalová funkce mezi prostředím S-funkce a vlastním uživatelským kódem).





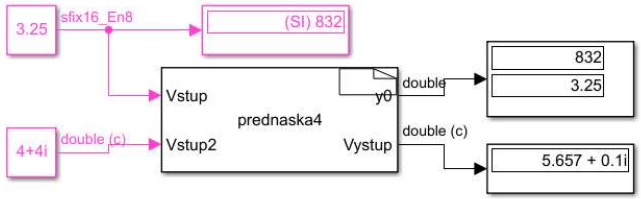




S-Function Builder

S-Function Builder – vygenerovaný kód z S-funkce

Vygenerovaný kód obsahuje volání obalové funkce a v samostatném modulu obalovou funkci



Code Replacements Report
Coder Assumptions

Generated Code

[-] **Main file**
- ert_main.c

[-] **Model files**
- Subsystem.c
- Subsystem.h
- Subsystem_private.h
- Subsystem_types.h

[+] **Utility files (1)**

[-] **Interface files**
- prednaska4_wrapper.c

```
30 void Subsystem_step(void)
31 {
32   /* S-Function (prednaska4): '<S1>/S-Function Builder1' incorporc
33    *  Inport: '<Root>/In1'
34    *  Inport: '<Root>/In2'
35    *  Outport: '<Root>/Out1'
36    *  Outport: '<Root>/Out2'
37    */
38   prednaska4_Outputs_wrapper(&Subsystem_U.In1, &Subsystem_U.In2,
39     &Subsystem_Y.Out1[0], &Subsystem_Y.Out2);
40 }
41
42 /* Model initialize function */
43 void Subsystem_initialize(void)
44 {
45   /* Registration code */
46
47   /* initialize error status */
```

Function: *prednaska4_Outputs_wrapper*

Code Interface Report
Traceability Report
Static Code Metrics Report
Code Replacements Report
Coder Assumptions

Generated Code

[-] **Main file**
- ert_main.c

[-] **Model files**
- Subsystem.c
- Subsystem.h
- Subsystem_private.h
- Subsystem_types.h

[+] **Utility files (1)**

[-] **Interface files**
- prednaska4_wrapper.c

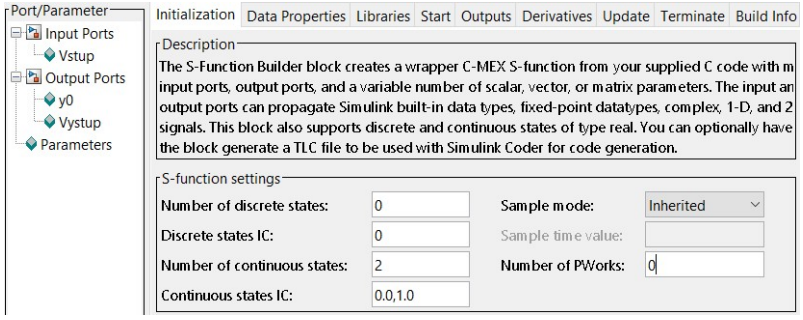
```
32  */
33 void prednaska4_Outputs_wrapper(const int16_T *Vstup,
34                                  const creal_T *Vstup2,
35                                  real_T *y0,
36                                  creal_T *Vystup)
37 {
38 /* %%%-SFUNWIZ_wrapper_Outputs_Changes_BEGIN --- EDIT HERE TO _END */
39 /* This sample sets the output equal to the input
40       y0[0] = u0[0];
41  For complex signals use: y0[0].re = u0[0].re;
42       y0[0].im = u0[0].im;
43       y1[0].re = u1[0].re;
44       y1[0].im = u1[0].im;
45  */
46
47 Vystup[0].re=sqrtf(Vstup2[0].re*Vstup2[0].re+Vstup2[0].im*Vstup2[0].im);
48 Vystup[0].im=0.1;
49
50 y0[0]=(double)Vstup[0];
51 y0[1]=(double)Vstup[0]/(256.0);
52 /* %%%-SFUNWIZ_wrapper_Outputs_Changes_END --- EDIT HERE TO _BEGIN */
53 }
54
55
```

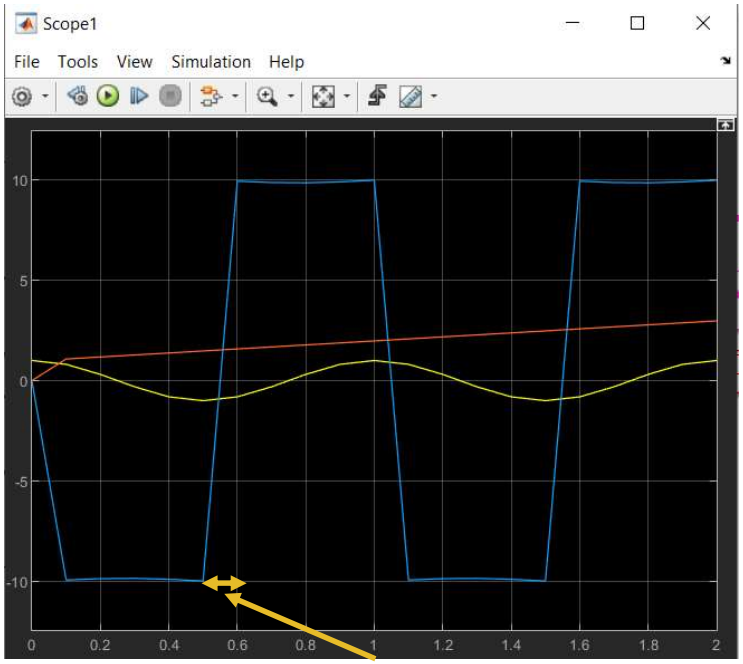
Function: *prednaska4_Outputs_wrapper*

S-Function Builder

S-Function Builder použití Derivatives

S-funkce Derivatives umožňuje popsat v S-function builderu diferenciální rovnici a nechat solver Simulinku jí vyřešit (s proměnným variable i diskrétním krokem). Může obsahovat detekci zero-crossing pro spojité stavy. Podobně záložka update se dá použít pro diferenční rovnici.



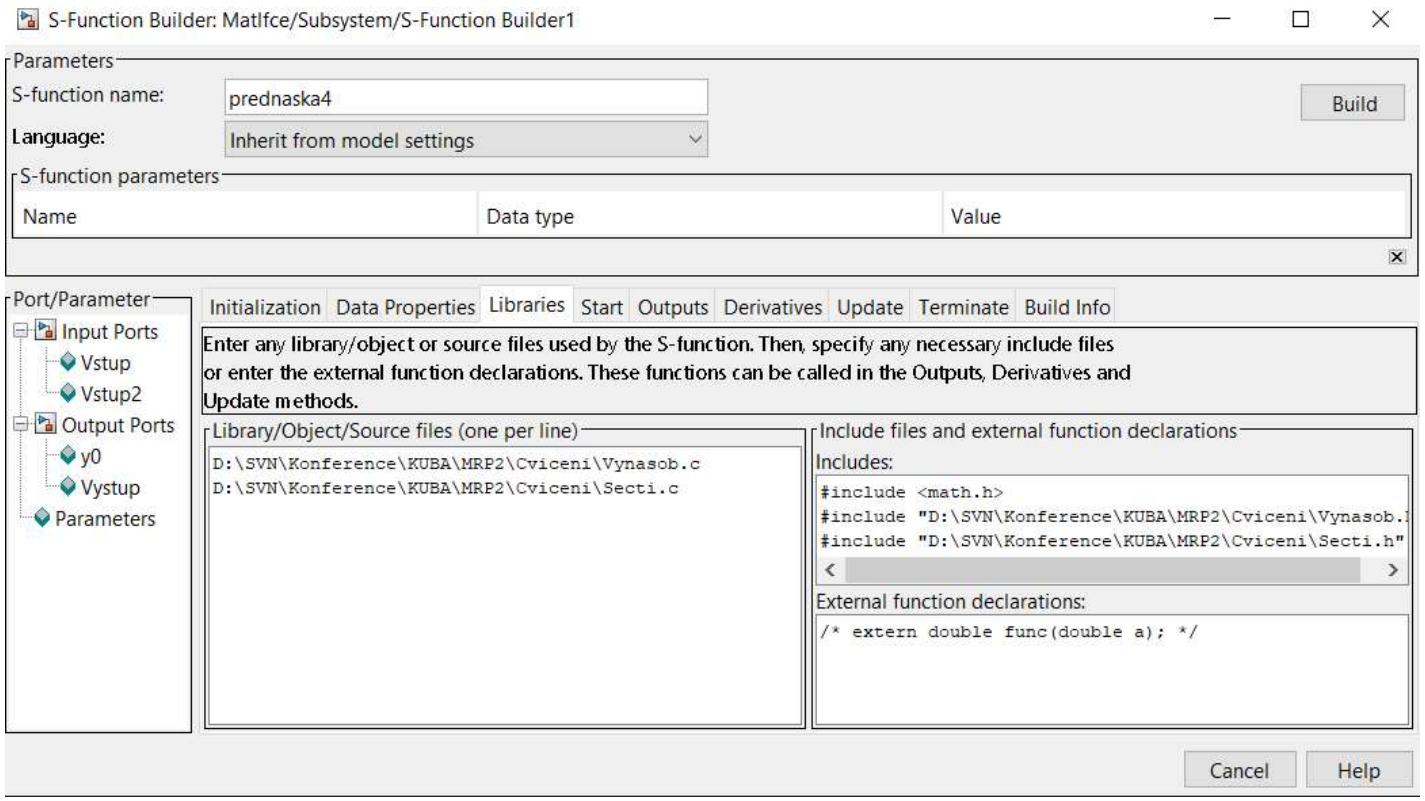


Velikost kroku solveru s proměnným krokem (vstupem cosinus 1 Hz)

S-Function Builder

S-Function Builder import souborů

Cesty k souborům projektu (.c a .h) se nastavují v položce Libraries.



S-Function Builder

S-Function Builder import souborů, volání funkcí a podmíněný překlad

Vynasob.h* Vynasob.c* Secti.h Secti.c

Ostatní soubory (Globální rozsah)

```
1
2   #include "d:\SVN\Konference\KUBA\MRP2\Cviceni\Secti.h"
3
4   double sect(double u, double v)
5   {
6       return u+v;
7
8   }
```

Vynasob.h* Vynasob.c* Secti.h Secti.c

Ostatní soubory (Globální rozsah)

```
1
2   extern double sect(double u, double v);
3
```

Vynasob.h* Vynasob.c* Secti.h Secti.c

Ostatní soubory (Globální rozsah)

```
1
2   #include "d:\SVN\Konference\KUBA\MRP2\Cviceni\Vynasob.h"
3
4   void vynas(int *u, int *v, int *vysl)
5   {
6       vysl[0]=u[0]*v[0];
7
8   }
9
10  #ifndef MATLAB_MEX_FILE
11
12  schvalne chyba
13  Toto nebude soucast S - funkce
14
15  #endif
16
17  int funkce2(int x, int y)
18  {
19      return x - y;
20  }
```

Port/Parameter

- Input Ports
  - Vstup
  - Vstup2
- Output Ports
  - y0
  - Vystup
- Parameters

Initialization | Data Properties | Libraries | Start | Outputs | Derivatives | Update | Terminate | Build Info

Code description

Enter your C-code or call your algorithm. If available, discrete and continuous states should be referenced as xD[0]...xD[n], xC[0]...xC[n] respectively. Input ports, output ports and parameters should be referenced using symbols specified in Data Properties. These references appear directly in the generated S-function.

```
        y1[0].im = u1[0].im;
 */
int i=2;
double x;

vynas(Vstup,&i,y0);

x=sect(Vstup2[0].re,Vstup2[0].im);

//Vystup[0].re=sqrtf(Vstup2[0].re*Vstup2[0].re+Vstup2[0].im*Vstup2[0].im);
//Vystup[0].im=0.1;
```

Vynasob.h* Vynasob.c* Secti.h Secti.c

Ostatní soubory (Globální rozsah)

```
1
2   extern void vynas(int *u, int *v, int *vysl);
3
4   #ifndef MATLAB_MEX_FILE
5
6       schvalne chyba
7       Toto nebude soucast S-funkce
8
9   #endif
10
```

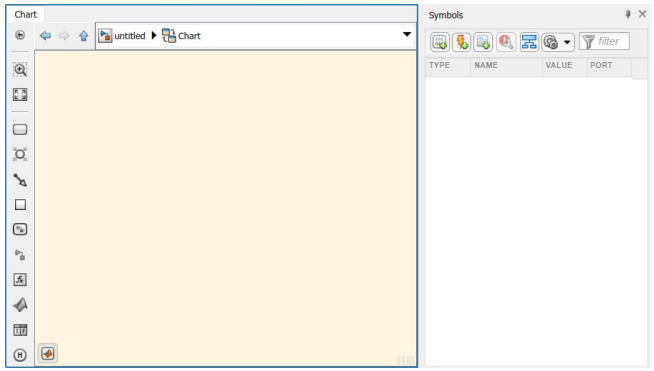

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
EVROPSKÁ UNIE EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ INVESTICE DO VAŠÍ BUDOUCNOSTI
OP Výzkum a vývoj pro inovace

4) Stateflow

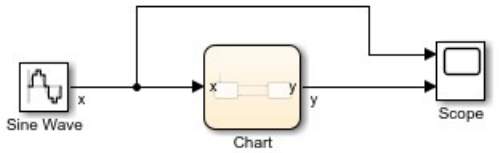
Stateflow

- Stateflow je grafické programovací prostředí pro modelování stavových automatů
- Umožňuje automatické generování kódu
- Bude mu věnována samostatná přednáška

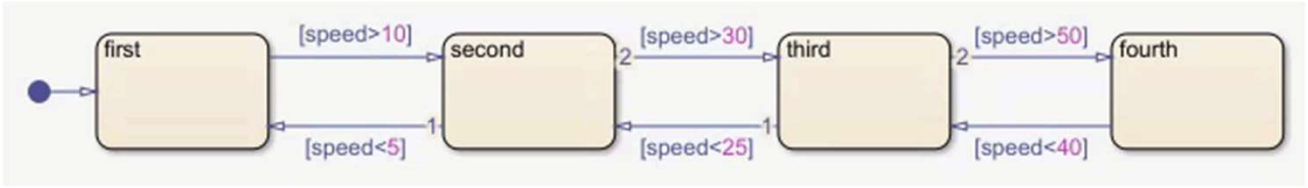
Prostředí Stateflow



Stateflow Chart v Simulinku



Stavový automat ve Stateflow



FAKULTA
ELEKTROTECHNICKÁ
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

RICE

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Příští přednáška

Profesionální návrh embedded software

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

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