Informační a řídicí systémy I.
Jiné systémy vs. IEC 61131-3

Pavel Balda
ZČU v Plzni, FAV, KKY

Osnova přednášky

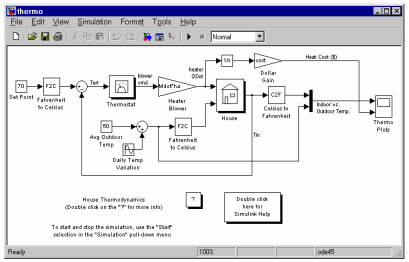
- Srovnání IEC 61131-3 a Matlab
  - FBD vs. Simulink
  - SFC vs. Stateflow
  - Real Time Workshop
- Srovnání IEC 61131-3 a ŘS REX
  - SFC vs. blok ATMT
  - ST vs. blok REXLANG

Srovnání: IEC 61131-3 vs. Matlab

- Účel:
  - IEC 61131-3 je od svého vzniku normou pro standardizaci průmyslových aplikací řízení v reálném čase
  - Matlab a jeho toolboxy byly původně nástroje pro technické výpočty offline.
    - Simulink a Stateflow pracují v simulačním čase.
    - V poslední době se masivně rozvíjí podpora řízení v reálném čase
    - Real-Time Workshop (RTW) – generuje (a též optimalizuje) kód v jazyku C, který lze přeložit pro daný cílový hardware (samostatnou řídicí stanici, desku se signálovými procesory, apod.)
    - RTW je vhodný pro testování řídicích algoritmů (regulátorů) na prototypových zařízeních, tzv. technologie Hardware-in-the-loop
- Návrh:
  - V IEC 61131-3 kladen důraz na průmyslové požadavky, např. ošetření mezních stavů a selhání výpočtu funkcí a FB pomocí vstupů **EN** a výstupů **ENO**
  - Algoritmy FB v Simulinku mají převážně akademický charakter, i když mohou být pokročilé, neřeší průmyslové požadavky a uživatel si je musí ošetřit velkým úsilím sám (např. vytvářením vlastních subsystémů s doplněnou logikou pro ošetření mezních situací)

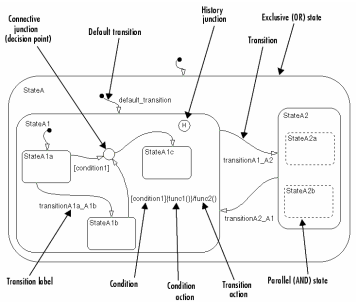
FBD vs. Simulink

- Analogií k FBD je v Matlabu Simulink
  - Na rozdíl od normy umí simulovat i spojité systémy. Nevýhodou je dlouhá a nezaručená doba odezvy při zjemňování integrace.
  - Pro připojení vstupů a výstupů a konfiguraci časování na konkrétních řídicích hw (pomocí RTW) používá speciální sady bloků (blocksets) dodané výrobci zařízení
  - Z formálního hlediska umožňuje i jiný tvar funkčních bloků než jen obdélník, spojovací čáry jsou doplněny hranami pro vyznačení orientace, bloky se dají otáčet a překlápět, ...



SFC vs. Stateflow

- Analogií k SFC je v Matlabu Stateflow, více v Stateflow User's Guide, The Mathworks, Inc., 1994-2005
- Logika propojení Simulinku a Stateflow je obrácená oproti IEC 61131-3. Bloky Stateflow se zařazují do schémat Simulinku



Co je Stateflow?

- Grafický návrhový systém pro simulaci konečných automatů
- Stateflow je vhodný pro logické řízení a ovládání modelů vytvářených v Simulinku
- Stateflow je integrován do Simulinku a umožňuje modelovat, simulovat a analyzovat daný systém
- Pro popis komplexních systémů používá kombinaci:
  - Teorie konečných automatů (finite state machine theory)
  - Vyjadřovacích prostředků vývojových diagramů
  - Stavové diagramy

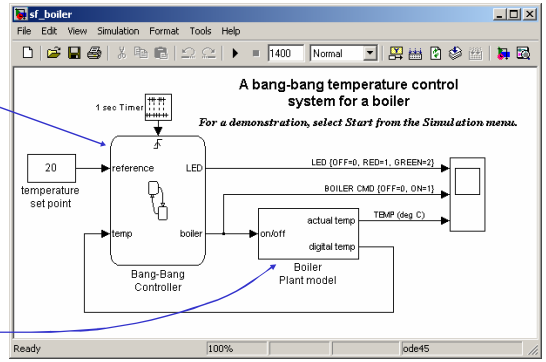
Stateflow: Příklad řízení teploty (1/3)

Dvoustavové řízení – zapnuto/vypnuto (on/off)

Blok Stateflow v modelu v Simulinku

Po dvojkliku se otevře okno v následujícím snímku

Simulovaný boiler prostředky Simulinku



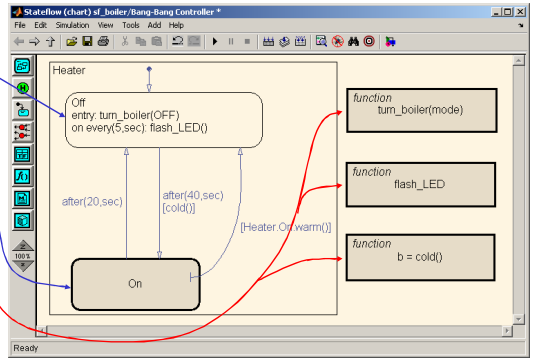
Stateflow: Příklad řízení teploty (2/3)

Blok Bang-Bang controller ve Stateflow

Stavy

stav **On** obsahuje další stavy – tzv. superstav (Superstate)

Grafické funkce

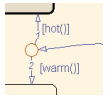


Stateflow: Příklad řízení teploty (3/3)

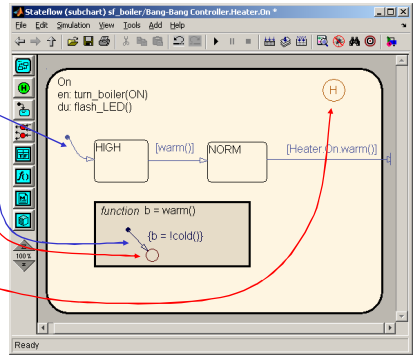
Vnitřek superstavu **On**

Implicitní přechod
(default transition)

Větvení/spojení (junction)



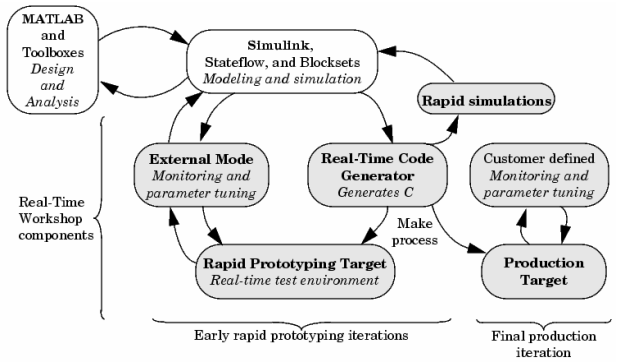
History junction – při vstupu do superstavu aktivuje poslední stav, ve kterém se nacházel před opuštěním



Real-Time Workshop

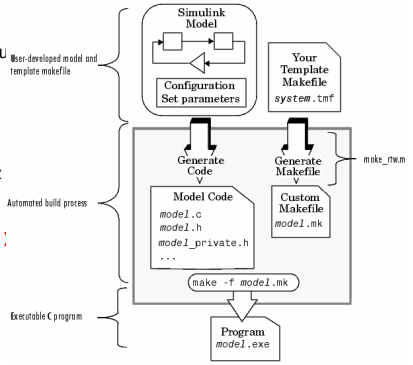
- Real-Time Workshop (RTW) je rozšíření systému Matlab-Simulink které automaticky generuje a překládá zdrojový kód z diagramů programu Simulink pro testování prototypů i finální řízení na cílových řídicích platformách
- Spolu s dalšími nástroji firmy The MathWorks poskytuje
  - Automatické generování kódu „ušitého na míru“ množství cílových platforem
  - Rychlou a přímou cestu od návrhu k implementaci
  - Přímou integraci s Matlabem a Simulinkem
  - Jednoduché uživatelské rozhraní
  - Otevřenou architekturu a rozšiřitelný proces vytváření aplikací (make process)
- Základní komponenty a vlastnosti RTW jsou
  - Simulink Code Generator – automaticky generuje kód v jazyku C z modelu
  - Make Process – modifikovatelný překlad a sestavení kódu pro rychlou tvorbu prototypů
  - Simulink External Mode – umožňuje komunikaci mezi Simulinkem a modelem běžícím v reálném čase na testovacím zařízení. Umožňuje ladit parametry a zaznamenávat data
  - Targeting Support – podpora cílových prostředí dodávaných s RTW, např. Real-Time Windows Target, xPC Target, apod.
  - Rapid Simulations – zrychluje simulace 5 až 20 krát, vysoce optimalizovaný kód.
  - Large-Scale Modeling – možnost postupného generování kódu pro hierarchii komponent

RTW – Role při návrhu softwaru



RTW – automatické sestavování programu

- `make_rtw.m` – makefile vygenerovaný RTW pro většinu cílových prostředí
  - Lze jej doplňovat o příkazy z pole Make commands v sekci Build process konfiguračních parametrů
- Aplikace se sestavuje jedním z příkazů
  - `rtwbuild jmenomodelu`
  - `rtwbuild('jmenomodelu')`



Prototypová zařízení firmy dSpace

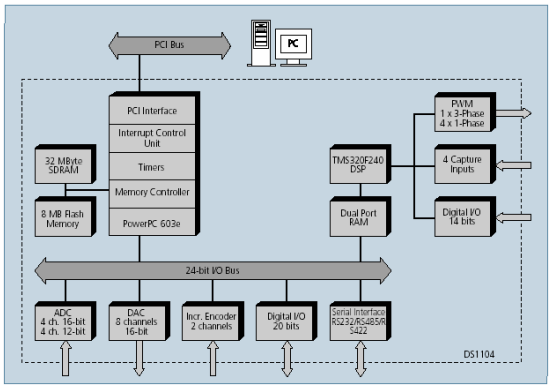
- dSpace GmbH je špičková firma zabývající se vývojem HW i SW řídicích jednotek pro mechatronické systémy
- Oblasti aplikací: ▪ Automobilový průmysl ▪ Letectví ▪ Pohony
- Hardware pro řídicí systémy

Single-Board Hardware  Kompaktní prototypové systémy s rychlými procesory a pokročilými vstupy/výstupy	RapidPro Hardware  Přizpůsobení signálů a výkonné pohony pro funkční prototypy
Modular Hardware  Výkonná platforma pro kompletní vývoj prototypů a jejich testování	Calibration Hardware  Modulární řešení úloh kalibrace a měření
MicroAutoBox Hardware  Kompaktní hardware pro testování pohonů a automobilů	Příslušenství  Konektory, panely, rozšiřující moduly pro testovací účely

Jednotka dSpace DS1104

- DS1104 R&D Controller Board
  - R&D = Research and Development, do PCI slotu
  - Plně grafické programování ze Simulinku/Stateflow
  - Založeno na technologii Power PC (PPC 603E)
- Vstupy a výstupy
  - Časovače: 4 obecné, 1 pro periodu vzorkování, 1 časová základna
  - A/D převodníky: 4 16 bitové multiplexované AI (2 us), 4 12 bitové paralelní AI (800 ns)
  - D/A převodníky: 8 16 bitových AO
  - Digitální I/O: 20 paralelních konfigurovatelných I/O
  - Inkrementální převodníky: 2 kanály, 24 bit
  - Šířková modulace pomocí signálového procesoru (DSP)
- Zařazení do Matlab-Simulink
  - Pomocí funkčních bloků z knihovny Real-Time Interface firmy dSpace

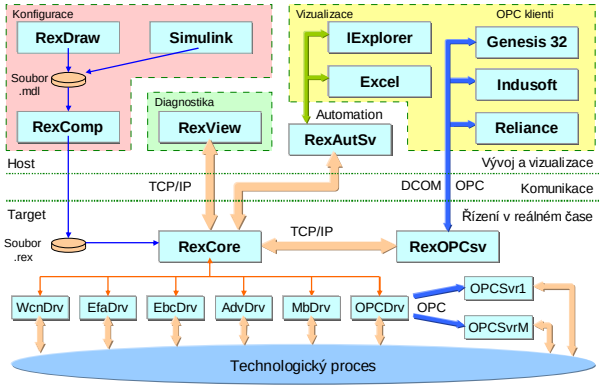
DS1104 – Blokové schéma



Řídicí systém REX

- Cíle návrhu
  - Kompatibilita s globálně rozšířeným programovým systémem Matlab-Simulink
  - Kvalitní knihovna funkčních bloků (blockset) splňující požadavky na využití v průmyslu (spolehlivost algoritmů, ošetření mezních stavů ...)
  - Standardní komunikace s s nadřazeným systémem prostřednictvím OPC (OLE for Process Control)
  - Přenositelnost na různé HW i SW platformy (různá cílová zařízení a operační systémy).
  - Definované rozhraní pro ovladače vstupně-výstupních zařízení
  - Snadná změna řídicích algoritmů změnou konfiguračního souboru (bez nutnosti překladu a sestavování celé aplikace jako v RTW)

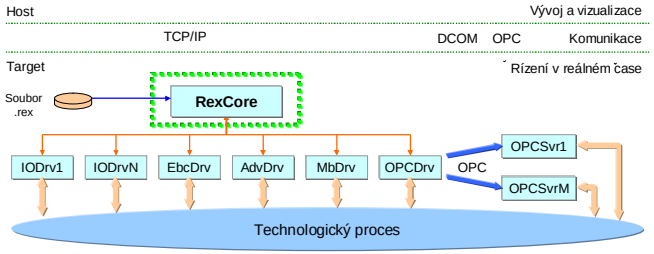
Struktura ŘS REX



Provozování ŘS REX, diagnostika

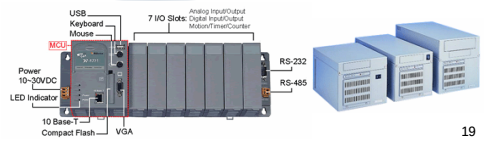
- Jádro systému REX – program **RexCore**
  - Subsystémy jádra
  - Subsystém reálného času
  - Vstupně-výstupní subsystém
- Diagnostický program **RexView**
  - Běh úloh subsystému reálného času
  - Pracovní prostor bloků
  - Trendy
- Spouštění konfigurací – programy **RexRun** a **DDDShell**

Jádro systému REX – program RexCore



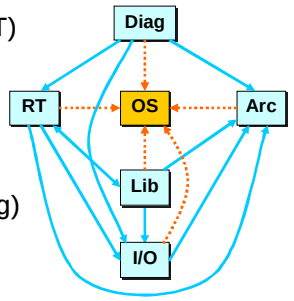
Cílové platformy:

- WinCon
  - Windows CE .NET
- Průmyslová PC
  - Phar Lap ETS



Subsystémy jádra RexCore

- Subsystém reálného času (RT)
- Vstupně-výstupní subsystém (I/O)
- Algoritmický subsystém (knihovna FB) (Lib)
- Diagnostický subsystém (Diag)
- Archivační subsystém (Arc)

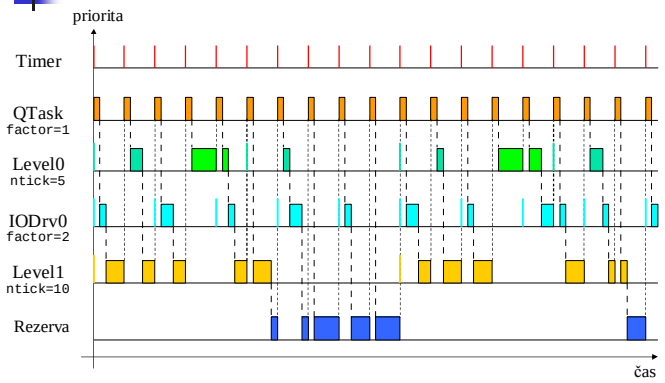


Subsystém reálného času

- Určuje periodické spouštění jednotlivých úloh operačního systému reálného času
- Úlohy OS:
  - Výpočetní úlohy (Quick task, Tasks)
  - Úlohy některých vstupně-výstupních ovladačů
- Deterministický plánovač (Scheduler)
  - Technika „Rate Monotonic Scheduling“
  - 32 uživatelsky přiřaditelných logických priorit
  - Nejvyšší priorita 0, nejnižší 31
  - Priority jsou přednastaveny, změna jen po důkladném uvážení ! Špatné nastavení může zhoršit chování systému

21

Plánovač (Scheduler) – Příklad



22

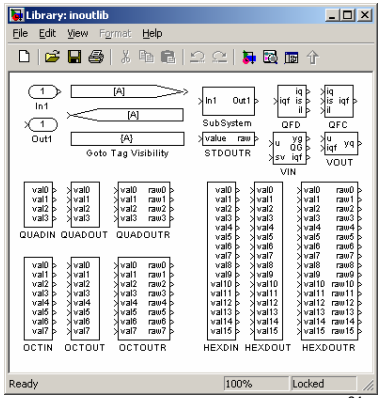
FB v IEC 61131-3 a v REXu

- FB v REXu inspirovány normou IEC 61131-3
  - Důraz kladen na ošetření mezních stavů, např. ošetření unášení integrační složky u regulátorů
  - FB, které mohou selhat mají chybový výstup, který je v případě selhání nastaven a lze jej testovat následnými bloky ve schématu
  - Bloky pracují pouze v diskrétním čase, spojité bloky jsou diskretizovány
- Na rozdíl od IEC 61131-3 jsou v systému REX implementována následující rozšíření:
  - Knihovna funkčních bloků RexLib ŘS REX obsahuje celou řadu velmi pokročilých bloků
  - Výstupy schématu se nenastavují až na konci programu, ale v závislosti na použitém vstupně-výstupním ovladači mohou být poslány do technologie okamžitě po vygenerování, čímž se minimalizuje parazitní zpoždění způsobené exekucí v PLC

23

Knihovna vstupně-výstupních bloků

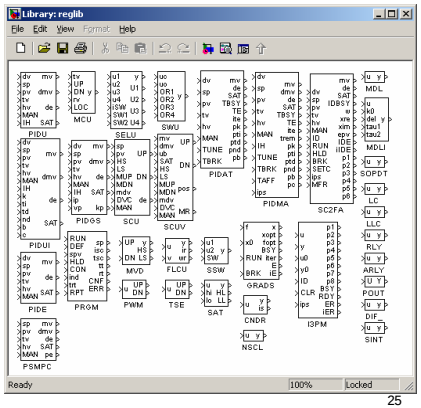
- Knihovna **InOutLib**
  - Bloky komunikující se vstupně-výstupními ovladači
  - Propojovací bloky pro přenos signálů mezi úlohami a „neviditelnými“ spojovacími čarami
  - Bloky pro zpracování kvality vstupních a výstupních signálů



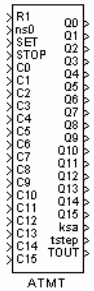
24

Knihovna bloků pro regulaci

- Knihovna **RegLib**
  - Regulátory, včetně pokročilých regulátorů s automatickým nastavováním parametrů
  - Modely procesů,
  - Zadávací jednotka
  - Dynamické kompenzátory
  - Bloky pro zpracování kvality vstupních a výstupních signálů
  - Bloky pro přepínání regulační struktury
  - A mnoho dalších ...



SFC a blok ATMT (1/3)

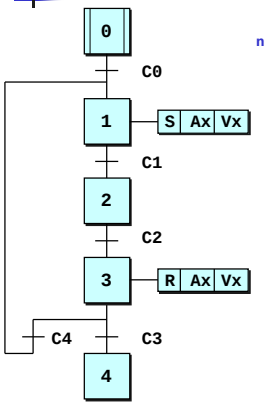


- Blok **ATMT** realizuje logický automat mající až 16 stavů S0 ... S15 indikovaných binárními výstupy **Q0** ... **Q15** a 16 podmínek přechodu **C0** ... **C15**
- Další vstupy:
  - **R1** – Reset převádí automat do stavu **0**, má přednost před vstupem **SET**
  - **ns0** – přednastavený stav, do kterého je automat převeden náběžnou hranou vstupu **SET**
  - **STOP** – hodnota **1** blokuje činnost automatu v daném stavu, výstup **tstep** nenarůstá
- Další výstupy:
  - **ksa** – celočíselná reprezentace stavu
  - **tstep** – čítač času v daném stavu. Při přechodu do jiného stavu se nuluje
  - **TOUT** – příznak překročení časového limitu pro aktuální stav

SFC a blok ATMT (2/3)

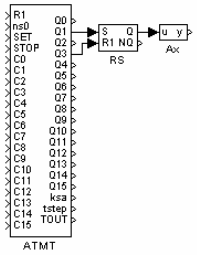
- Chování bloku **ATMT** je určeno tabulkou přechodů ve tvaru:
  - *S1 C1 NS1*
    S2 C2 NS2
    ...
    Sn Cn NSn
  - kde každý řádek tabulky vyjadřuje jednu podmínku přechodu
  - Např. prvý řádek *S1 C1 NS1* znamená: Je-li aktuální stav *S1* a současně je splněna podmínka *C1*, přejdi do nového stavu *NS1*
- Dalším parametrem je pole časů (timeoutů) v jednotlivých stavech
  - *TO1 TO2 ... TOm*
  - Setrvává-li automat ve stavu *Si* delší dobu než *TOi*, je nastaven výstup **TOUT** na **1** (při přechodu do nového stavu je **TOUT** nastaven na **0**)

SFC a blok ATMT (3/3)

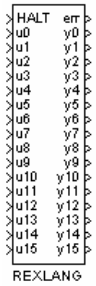


- Příklad: SFC na obr. Vlevo odpovídá následující tabulce přechodů bloku ATMT na obr. vpravo:

S	C	NS
0	0	1
1	1	2
2	2	3
3	3	4
3	4	1



ST a blok REXLANG (1/2)



- Blok **REXLANG** je volně programovatelný blok ŘS REX v jazyku téměř kompatibilním s jazykem C.
  - Uživatel vytvoří zdrojový soubor s algoritmem vlastního bloku a jméno souboru uvede jako parametr bloku
  - Systém REX pak soubor přeloží a pokud překlad skončí úspěšně, blok periodicky spouští v úloze, kam je zařazen
- Oproti jazyku C má zejména následující omezení
  - Navržený algoritmus pracuje pouze s jednoduchými typy **long** a **double** (**int**, **short** a **bool** se zpracovávají jako **long**, typ **float** se zpracovává jako **double**)
  - Nejsou implementovány ukazatele pomocí operátoru *, jsou však implementována jednorozměrná indexovatelná pole
  - Nejsou implementovány standardní knihovny, je však definována většina funkcí z **math.h**
  - Algoritmus se skládá z funkcí **init(void)**, **main(void)**, **exit(void)** a volitelně též **parchange(void)**

29

ST a blok REXLANG (2/2)

- Příklad jednoduchého číslicového filtru

```
double input(0) vstup;
//promenna 'vstup' odpovida u0
double output(0) vystup;
//promenna 'vystup' odpovida y0
double stav[20], param[20];
const long count=20;
int init(void)
{
  long i; const double a=0.95;
  param[0]=0.2;param[5]=0.2;
  param[10]=0.2;param[12]=0.2;
  param[15]=0.2;
  for(i=0;i<count;i++)
  {
    param[i]=param[i]+exp(-i*a)/a;
    Trace(1,param[i]);
  }
  return 0;
}
int main(void)
{
  long i;
  double soucet=0.0;
  for(i=0;i<count-1;i++)
    stav[i]=stav[i+1];
  if(fabs(vstup)>1)
    stav[count-1]=(vstup>0)? 1 : -1;
  else
    stav[count-1]=vstup;
  for(i=0;i<count;i++)
  {
    soucet+=stav[i]*param[count-1-i];
    Suspend(0.1);
  }
  vystup=soucet;
  return 0;
}
int exit(void){return 0;}
```

30