Funkční bloky systému REX
Referenční příručka

REX Controls s.r.o.

Verze 2.50.1
7.11.2016
Plzeň

Obsah

Poznámka: U bloků označených * je k dispozici pouze částečná dokumentace. Kompletní dokumentace může být dostupná v ostatních jazykových mutacích manuálu.

Kapitola 1

Úvod

Příručka „Funkční bloky systému REX“ je, jak už její název napovídá, referenční příručkou knihovny RexLib funkčních bloků řídicího systému REX. Kromě referenčního popisu jednotlivých tříd, popisuje (referenčním způsobem) všechny subsystémy řídicího systému REX.

1.1 Jak číst tuto příručku

Standardně dodávaná rozsáhlá knihovna funkčních bloků RexLib řídicího systému REX je rozdělena do menších skupin logicky příbuzných bloků, tzv. *kategorií* (podknihoven). Každá kategorie je popisována v samostatné kapitole, obsahující nejprve obecné vlastnosti celé kategorie a jejích funkčních bloků, následované postupně popisem všech funkčních bloků dané kategorie.

Jednotlivé kapitoly příručky obsahují:

1 **Úvod**
Tato úvodní kapitola, seznamující s uspořádáním příručky a uvádějící formát (konvenci) popisu jednotlivých funkčních bloků.

2 **EXEC – Konfigurace exekutivy reálného času**
Kapitola popisuje zejména bloky sloužící pro konfiguraci struktury a časování jednotlivých objektů zařazovaných do systému reálného času řídicího systému REX (programu RexCore). Tyto funkční bloky se nepoužívají při simulaci v Simulinku. Kromě toho jsou zde obsaženy ještě dva speciální bloky `LPBRK` a `SLEEP` důležité právě pro exekuci v systému Simulink.

3 **INOUT – Bloky vstupů a výstupů systému REX**
Tato podknihovna vstupně-výstupních bloků opět obsahuje převážně bloky určené jen pro systém REX a zprostředkovávající hlavně vazbu mezi řídicími úlohami a vstupně-výstupními ovladači.

4 **MATH – Matematické bloky**
Podknihovna popisuje většinou jednoduché bloky pro matematické operace a základní matematické funkce. Obdobné bloky lze najít i ve standardně dodávaných knihovnách systému Simulink, takže mají význam v aplikacích, které budou cílově provozovány pod systémem REX.

5 **ANALOG – Zpracování analogových signálů**
Mezi bloky pro zpracování analogových signálů patří integrátor, derivátor, dopravní zpoždění, vlečný průměr, komparátory a selektory, filtry. Velmi zajímavým blokem je rozběhová jednotka `AVS`.

9 **GEN – Generátory signálů**
Kapitola popisuje bloky generující analogové i logické testovací signály.

7 **REG – Bloky pro regulaci**
Bloky pro regulaci tvoří nejrozsáhlejší podknihovnu knihovny RexLib a zahrnují bloky od jednoduchých dynamických kompenzátorů, přes bloky pro přepínání regulačních struktur, bloky pro přizpůsobení výstupů akčním členům (krokové regulátory, šířková modulace) až po několik verzí PID (P, I, PI, PD a PID) regulátorů. Mezi regulátory jsou např. blok `PIDGS`, umožňující za běhu přepínat několik sad parametrů (tzv. *gain scheduling*), `PIDMA` s vestavěným *momentovým autotunerem*, blok `PIDAT` s vestavěným reléovým autotunerem nebo blok fuzzy regulátoru `FLCU`, a další.

8 **LOGIC – Logické řízení**
Kapitola popisuje bloky pro kombinační i sekvenční logické řízení od jednoduchých logických operací (negace, součet, součin), až po sekvenční logický automat `ATMT`, implementující standard SCF (Sequential Function Charts, dříve Grafcet).

10 **ARC – Archivace dat**
Mezi bloky pro archivaci dat v systému REX patří bloky pro generování alarmů a bloky pro záznam trendů přímo na cílovém zařízení. Tyto bloky nemají žádnou analogii v systému Simulink.

12 **PARAM – Práce s parametry**
Bloky této podknihovny umožňují pracovat s parametry konfigurace systému REX zejména ukládat a nahrávat parametry nebo je vzdáleně modifikovat.

13 **MODEL – Modely dynamických systémů**
Systém REX může být využit i pro tvorbu matematických modelů dynamických systémů běžících v reálném čase. Bloky této podknihovny byly vyvinuty právě pro takové účely.

14 **MATRIX – Práce s maticovými a vektorovými daty**
Tato podknihovna obsahuje bloky pro práci s vektorovými a maticovými signály v systému REX.

?? **MC_SINGLE – Řízení pohybu v jedné ose**
Bloky této podknihovny byly vyvinuty dle normy PLCopen Motion Control pro řízení pohybu v jedné ose.

?? **MC_MULTI – Řízení pohybu ve více osách**
Bloky této podknihovny byly vyvinuty dle normy PLCopen Motion Control pro řízení pohybu ve více osách.

?? **MC_COORD – Koordinované řízení pohybu**
Bloky této podknihovny byly vyvinuty dle normy PLCopen Motion Control pro koordinované řízení pohybu.

15 **SPEC – Speciální bloky**
Do skupiny speciálních bloků patří v současné době dva zajímavé bloky. Prvním je blok `REXLANG`, umožňující překlad a interpretaci uživatelských algoritmů vytvořených v jazyce velmi podobném jazyku C (syntaxe většiny příkazů jazyka `REXLANG` je totožná se syntaxí jazyka C). Druhým blokem je blok `RDC`, umožňující v reálném čase komunikaci mezi dvěma Simulinky (i na různých počítačích), mezi dvěma systémy REX nebo mezi systémem REX a Simulinkem. Bloky `RDC` mohou navíc předávat data OPC serveru pro Matlab.

Jednotlivé kapitoly příručky na sebe navazují jen volně, a proto mohou být čteny téměř v libovolném pořadí, dokonce může být čtena vždy jen nezbytně nutná informace potřebná k pochopení funkce konkrétního funkčního bloku. Pro tento účel je vhodná zejména elektronická podoba příručky (ve formátu `.pdf`), vybavená hypertextovými záložkami a obsahem, které usnadňují rychlé nalezení příslušných bloků.

Přesto lze ještě doporučit přečtení následující podkapitoly, která popisuje konvence užívané při popisu bloků ve zbytku příručky.

1.2 Formát popisu funkčních bloků

Popis každého funkčního bloku se skládá z několika sekcí (v uvedeném pořadí):

Symbol bloku – graficky zobrazuje symbolickou značku bloku

Popis funkce – stručně popisuje funkci daného bloku, aniž by byly uváděny příliš detailní informace.

Vstupy – detailně popisuje všechny vstupy daného bloku

Výstupy – detailně popisuje všechny výstupy daného bloku

Parametry – detailně popisuje všechny parametry daného bloku

Příklad – graficky znázorňuje na jednoduchém příkladu použití daného bloku v kontextu ostatních bloků a často uvádí i obrázek s průběhem vstupních a výstupních signálů tak, aby chování bloku bylo přiblíženo co nejnázorněji.

Pokud je funkce bloku zřejmá, nemusí být sekce Příklad uvedena. V případě, že blok nemá žádný vstup nebo výstup nebo parametr, není ani příslušná sekce v popisu obsažena.

Vstupy, výstupy a parametry jsou popisovány v tabulkové formě:

`<jmeno>` [*jm*] Podrobný popis vstupu (výstupu, parametru) `<jmeno>`. `<typ>` Matematický symbol *jm* na pravé straně prvního sloupce je používán ve vzorcích v sekci Popis funkce a bude uváděn, pokud se od jména vstupu liší víc než jen typograficky. Pokud daná proměnná nabývá pouze několika vyjmenovaných hodnot, je význam těchto hodnot uveden v tomto sloupci. [⊙`<def>`] [↓`<min>`] [↑`<max>`]

Význam jednotlivých sloupců je celkem zřejmý. Ve třetím sloupci je vždy uveden pouze `<typ>`. Řídicí systém REX podporuje typy uvedené v tabulce 1.1. Standardní funkční bloky však nejčastěji používají pro logické proměnné typ `bool`, pro celočíselné proměnné typ `long` a pro reálné proměnné (v pohyblivé řádové čárce) typ `double`.

Každá takto popsaná proměnná (vstup, výstup či parametr) má v řídicím systému REX konkrétní implicitní (default) hodnotu `<def>`, uvozenou symbolem ⊙ a podobně i minimální příp. maximální přípustou hodnotu, uvozenou symbolem ↓, příp. ↑. Všechny tyto tři hodnoty mohou být uvedeny ve druhém sloupci, ale nejsou povinné (jsou umístěny v [ ]). Pokud není uvedena hodnota ⊙`<def>`, je vždy tato hodnota nulová. Není-li uvedena hodnota ↓`<min>` příp. ↑`<max>`, nabývá minimální příp. maximální hodnoty příslušného typu (viz tabulku 1.1)

Typ	Význam	Minimum	Maximum
`bool`	logická hodnota `0` nebo `1`	0	1
`byte`	8 bit. celé číslo bez znaménka	0	255
`short`	16 bit. celé číslo se znaménkem	-32768	32767
`long`	32 bit. celé číslo se znaménkem	-2147483648	2147483647
`large`	64 bit. celé číslo se znaménkem	-9223372036854775808	9223372036854775807
`word`	16 bit. celé číslo bez znaménka	0	65535
`dword`	32 bit. celé číslo bez znaménka	0	4294967295
`float`	32 bit. číslo v pohyblivé ř. čárce	< -3.4E+38	>3.4E+38
`double`	64 bit. číslo v pohyblivé ř. čárce	< -1.7E+308	>1.7E+308
`string`	znakový řetězec		

Tabulka 1.1: Typy proměnných řídicího systému REX.

1.3 Konvence pojmenování proměnných, bloků a subsystémů

Pro usnadnění práce s řídicím systémem REX se používá několik konvencí. V předchozí podkapitole byly zavedeny všechny používané typy proměnných. Pod pojmem proměnná

budeme mít v této podkapitole na mysli vstupy, výstupy a parametry bloků. Ve velké většině bloků se používají pouze tyto tři typy:

`bool` – pro dvouhodnotové logické proměnné, např. zapnuto/vypnuto, ano/ne, pravda/nepravda, true/false, on/off, apod. V této příručce budeme hodnoty logické jedničky (ano, pravda, true, on) zapisovat jako `1` a hodnoty logické nuly (ne, nepravda, false, off) jako `0`, přestože v některých nástrojích mohou být jejich hodnoty zobrazovány (kvůli požadované kompatibilitě se systémem Matlab-Simulink) jako `on` pro `1` a `off` pro `0`. Názvy logických proměnných používají velká písmena, např. `RUN`, `YCN`, `R1`, `UP`.

`long` – pro celočíselné hodnoty, např. číslo sady parametrů, délka trendového bufferu, typ generovaného signálu, chybový kód, výstup čítače, apod. Názvy celočíselných proměnných jsou obvykle psány malými písmeny a počáteční písmeno (vždy malé) je nejčastěji jedno z písmen {`i`, `k`, `l`, `m`, `n`, `o`}, např. `ips`, `l`, `isig`, `iE`, apod. Existuje však několik výjimek z tohoto pravidla, např. `cnt` v bloku `COUNT`, `btype`, `ptype1`, `pfac` a `afac` v bloku `TRND`, apod.

`double` – pro čísla v pohyblivé řádové čárce (reálná), např. zesílení, saturační meze, výsledky většiny matematických funkcí, parametry PID regulátorů, délky časových intervalů v sekundách, apod. Názvy proměnných v pohyblivé řádové čárce používají pouze malá písmena, např. `k`, `hilim`, `y`, `ti`, `tt`.

Typy funkčních bloků v řídicím systému jsou pojmenovávány velkými písmeny, uvnitř jména se mohou vyskytovat číslice a znak `'_'` (podtržítko). Při vytváření uživatelských instancí bloků doporučujeme na začátku ponechat název typu bloku a doplnit jej o uživatelský název, kde doporučujeme používat všechny uvedené typy znaků a navíc malá písmena.

Výslovně se nedoporučuje používat v uživatelských názvech bloků a vytvořených subsystémů znaky s diakritikou a speciální znaky jako jsou mezery, znaky konce řádků, interpunkční znaménka, operátory, apod. Použití těchto znaků omezuje přenositelnost vytvořených algoritmů na různé platformy a může vést k velké nesrozumitelnosti. Jména jsou kontrolována překladačem RexComp a pokud obsahují některý z nevhodných znaků je hlášeno varování.

1.4 Kvalita signálu používaná v OPC

Každý signál (vstup, výstup, parametr) v řídicím systému REX má kromě své hodnoty některého z typů uvedených v tab. 1.1 ještě tzv. *příznaky kvality*. Příznaky kvality používané v řídicím systému REX jsou shodné s příznaky kvality používanými specifikacemi OPC (OLE for Process Control), viz [1] a obsahují jednobajtovou informaci, jejíž struktura je uvedena v tabulce 1.2

Základní druh kvality určují příznaky QQ v nejvyšších dvou bitech. Podle jejich kombinací uvedených v tabulce rozlišujeme kvalitu dobrou (`GOOD`), nejistou (`UNCERTAIN`) a špatnou (`BAD`). Jemnější rozlišení, tzv. substatus poskytují čtyři bity SSSS. Tyto bity

Číslo bitu	7	6	5	4	3	2	1	0
Váha bitu	128	64	32	16	8	4	2	1
Bitová pole	**Kvalita**		**Substatus**				**Omezení**	
	Q	Q	S	S	S	S	L	L
Špatná (`BAD`)	0	0	S	S	S	S	L	L
Nejistá (`UNCERTAIN`)	0	1	S	S	S	S	L	L
(Nevyužito v OPC)	1	0	S	S	S	S	L	L
Dobrá (`GOOD`)	1	1	S	S	S	S	L	L

Tabulka 1.2: Struktura příznaků kvality

mají různý význam pro různou základní kvalitu. Nejnižší dva bity LL informují o tom, zda daná veličina překročila své meze nebo zda má konstantní hodnotu. Podrobnosti a význam ostatních bitů lze nalézt v kap. 6.8 specifikace [1].

Kapitola 2

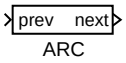
EXEC – Konfigurace exekutivy reálného času

Obsah

ARC – Archiv systému REX

Symbol bloku

Licence: STANDARD



Popis funkce

Blok ARC slouží v systému REX pro konfiguraci archivů, sloužících pro průběžné zaznamenávání alarmů, událostí a historických trendů přímo na cílovém zařízení. Vstup prev prvního z archivů se propojí s výstupem Archives bloku EXEC. Další archivy se přidávají propojováním vstupu prev s výstupem next předchozího archivu. Na každý výstup next smí být připojen nejvýše jeden vstup prev následujícího archivu, u posledního archivu zůstává výstup next nepřipojen. Vzniklá posloupnost určuje pořadí alokace a inicializace jednotlivých archivů v řídicím systému REX a také určuje index archivu, používaný v parametru arc archivačních bloků (viz kap. 10). Archivy jsou číslovány od 1 a jejich maximální počet je omezen na 15 (archiv č. 0 je interní systémový log).

Typ archivu z hlediska zachování dat i po restartu cílového zařízení je určen parametrem atype. Přípustné volby závisejí na možnostech cílového zařízení a lze je po úspěšném připojení k danému zařízení zjistit v záložce Target programu RexView.

Archivy jsou na cílovém zařízení tvořeny posloupností úložek proměnné délky (optimalizace paměti a disku), z nichž každá obsahuje časovou značku. Proto dalšími parametry archivu jsou celková velikost v bytech asize a maximální počet časových značek nmarks pro urychlení sekvenčního vyhledávání v archivu.

Vstup

prev	Vstup sloužící pro připojení prvního archivu na výstup Archives bloku EXEC nebo k připojení na výstup next předchozího archivu	long

Výstup

next	Výstup sloužící pro zřetězování archivů připojením na vstup prev následujícího archivu	long

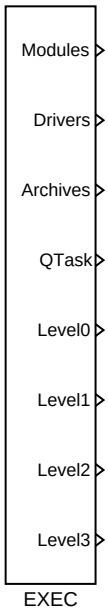
Parametry

`atype`	Typ archivu `1` ..... archiv je alokován v paměti RAM (po restartu cílového zařízení je nenávratně ztracen) `2` ..... archiv je alokován v zálohované paměti, např. CMOS (po restartu cílového zařízení zůstává zachován) `3` ..... archiv je alokován na disku (zůstává zachován v souboru i po restartu)	⊙`1`	`long`
`asize`	Velikost archivu (v bytech)	↓`256` ⊙`102400`	`long`
`nmarks`	Počet časových značek pro urychlení sekvenčního vyhledávání v archivu	↓`2` ⊙`720`	`long`
`ldaymax`	Maximální velikost archivu za den [byte]	↓`1000` ↑`2147480000` ⊙`1048576`	`large`
`period`	Perioda zapisování dat na disk [s]	⊙`60.0`	`double`

EXEC – Exekutiva reálného času

Symbol bloku

Licence: STANDARD



Popis funkce

Blok `EXEC` tvoří základ tzv. *hlavního souboru projektu* ve formátu `.mdl`, kterým se konfigurují jednotlivé subsystémy řídicího systému REX, a který nemá analogii v systému Matlab-Simulink. Konfigurace bloku `EXEC` a na něj navázané bloky nerealizují žádný výpočetní algoritmus, ale jsou zpracovány překladačem RexComp pro sestavení celé aplikace řídicího systému REX.

Konfigurace systému REX se skládá z modulů (`Modules`), vstupně-výstupních ovladačů (`Drivers`), archivačního subsystému (`Archives`) a subsystému reálného času, obsahujícího rychlou výpočetní úlohu (blíže viz blok `QTASK`) a čtyři prioritní úrovně (`Level0` až `Level3`) pro zařazování výpočetních úloh (blíže viz blok `TASK`).

Parametr `tick` určuje základní (nejkratší) periodu, se kterou bude možno spouštět jednotlivé úlohy. Zadaná hodnota je kontrolována překladačem RexComp podle zvoleného cílového zařízení. Obecně lze říci, že čím menší hodnota je zadána, tím je větší režie jádra řídicího systému REX.

Periody jednotlivých výpočetních úrovní `Level0` až `Level3` jsou určeny násobky parametrů `ntick0` až `ntick3` a základní periody `tick`. Parametry `pri0` až `pri3` jsou logickými prioritami odpovídajících výpočetních úrovní v systému REX. Poznamenejme, že systém REX používá 32 logických priorit, kterým jsou interně přiřazeny priority zá-

vislé na operačním systému cílového zařízení. Nejvyšší logická priorita systému REX je 0, nejnižšší má hodnotu 31, přičemž platí, že pokud mají běžet dvě úlohy s různými prioritami, bude úloha s nižší prioritou (vyšší hodnotou) přerušena úlohou s vyšší prioritou (nižší hodnotou). Řídicí systém REX vychází z obecně přijímané myšlenky, že „rychlé“ úlohy (s krátkou periodou vzorkování) je vhodné spouštět s vyšší prioritou než úlohy „pomalé“ (tzv. *Rate monotonic scheduling*). Proto přednastavené hodnoty priorit pri0 až pri3 není ve většině případů třeba měnit; neuvážená změna může vést k těžko předvídatelným důsledkům!

Výstupy

Modules	Výstup pro připojování rozšiřujících modulů systému REX, viz blok MODULE	long
Drivers	Výstup pro připojování vstupně výstupních ovladačů systému REX, viz bloky IODRV a TIODRV	long
Archives	Výstup pro konfiguraci archivů, viz blok ARC	long
QTask	Výstup pro připojení rychlé úlohy (tzv. quick task) s nejvyšší prioritou a s nejkratší periodou, viz blok QTASK	long
Level0	Výpočetní úroveň pro zařazování úloh (viz blok TASK) s vysokou prioritou pri0 a krátkou periodou určenou parametrem ntick0	long
Level1	Výpočetní úroveň pro zařazování úloh se střední prioritou pri1 a středně dlouhou periodou určenou parametrem ntick1	long
Level2	Výpočetní úroveň pro zařazování úloh s nízkou prioritou pri2 a dlouhou periodou určenou parametrem ntick2	long
Level3	Výpočetní úroveň pro zařazování úloh s nejnižší prioritou pri3 a nejdelší periodou určenou parametrem ntick3	long

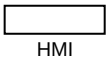
Parametry

target	Cílové zařízení Obecné cílové zařízení	⊙PC - Windows	string
tick	Základní perioda (tik) jádra řídicího systému REX a současně též perioda rychlé úlohy QTASK (zadávaná ve vteřinách)	⊙0.05	double
ntick0	Určuje základní periodu úloh zařazených do úrovně Level0 podle vztahu tick*ntick0	↓1 ⊙10	long
ntick1	Určuje základní periodu úloh zařazených do úrovně Level0 podle vztahu tick*ntick1	↓ntick0+1 ⊙50	long
ntick2	Určuje základní periodu úloh zařazených do úrovně Level0 podle vztahu tick*ntick2	↓ntick1+1 ⊙100	long
ntick3	Určuje základní periodu úloh zařazených do úrovně Level0 podle vztahu tick*ntick3	↓ntick2+1 ⊙1200	long
pri0	Priorita všech úloh zařazených do úrovně Level0	↓3 ↑31 ⊙5	long
pri1	Priorita všech úloh zařazených do úrovně Level1	↓pri0+1 ↑31 ⊙9	long
pri2	Priorita všech úloh zařazených do úrovně Level2	↓pri1+1 ↑31 ⊙13	long
pri3	Priorita všech úloh zařazených do úrovně Level3	↓pri2+1 ↑31 ⊙18	long

HMI – * Konfigurace visualizace

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

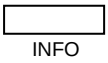
Parametry

`IncludeHMI`	Zahrnout soubory HMI do projektu	⊙`on`	`bool`
`HmiDir`	Výstupní adresář pro soubory vizualizace (HMI)	⊙`hmi`	`string`
`SourceDir`	Zdrojový adresář	⊙`hmisrc`	`string`
`GenerateWebWatch`	Vygenerovat WebWatch vizualizaci z MDL souborů	⊙`on`	`bool`
`GenerateRexHMI`	Při překladu projektu vygenerovat HMI ze SVG a JS souborů	⊙`on`	`bool`
`RedirectToHMI`	Webserver bude automaticky přesměrovávat na stránku s HMI	⊙`on`	`bool`
`Compression`	Aktivovat kompresi dat		`bool`

INFO – * Dodatečné informace o projektu

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

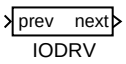
Parametry

`Title`	Název projektu	`string`
`Author`	Autor projektu	`string`
`Description`	Stručný popis projektu	`string`
`Customer`	Informace o zákazníkovi	`string`

IODRV – Vstupně-výstupní ovladač systému REX

Symbol bloku

Licence: STANDARD



Popis funkce

Vstupně-výstupní ovladače jsou v systému REX implementovány jako rozšiřující moduly (viz blok MODULE). Modul může obsahovat několik ovladačů, které se do konfigurace systému přidávají pomocí bloků IODRV. Vstup `prev` prvního z ovladačů se propojí s výstupem `Drivers` bloku EXEC. Další ovladače se přidávají propojováním vstupu `prev` s výstupem `next` předchozího ovladače. Na každý výstup `next` smí být připojen nejvýše jeden vstup `prev` následujícího ovladače, u posledního ovladače zůstává výstup `next` nepřipojen. Vzniklá posloupnost určuje pořadí inicializace jednotlivých ovladačů do řídicího systému REX (pořadí zavádění jednotlivých ovladačů je určeno pořadím modulů, v nichž jsou obsaženy, viz popis bloku MODULE).

Každý ovladač je v systému REX identifikován svým jménem, které se zadává v parametru `classname`. Pozor, parametr `classname` rozlišuje velká a malá písmena! Pokud se jméno ovladače liší od jména modulu, obsahujícího daný ovladač, musí se zadat i jméno modulu `module`, jinak se ponechá prázdné. Přesné nastavení těchto dvou parametrů je popsáno v příručce pro každý ovladač systému REX.

Většina ovladačů má svá vlastní konfigurační data uložena v souborech s příponou `.rio` (REX Input/Output), jejichž jméno určuje parametr `cfgname`. Soubory `.rio` se vytvářejí na stejném adresáři jako hlavní soubor projektu s příponou `.mdl` v němž je použit tento blok. Konfigurační data ovladačů (např. názvy vstupních/výstupních signálů, jejich připojení na konkrétní fyzické vstupy/výstupy, parametry komunikace se vstupně-výstupním zařízením, apod.) se zadávají ve vestavěných editorech poskytovaných přímo ovladači. V programu RexDraw systému REX se editory volají stisknutím tlačítka `Configure` v parametrickém dialogu bloku, v systému Simulink je pro stejnou funkci nutno zaškrtnout pomocné políčko `"Tick this checkbox to call IOdrv EDIT dialog"`.

Zbylé parametry bloku určují chování ovladače při běhu řídicího systému REX a mají význam jen tehdy, pokud ovladač implementuje vlastní úlohu (viz příručku k odpovídajícímu ovladači). Parametr `factor` je násobkem základní periody `tick` bloku EXEC, určujícím periodu spouštění této úlohy (`factor*tick`). Parametr `stack` udává velikost zásobníku v bytech (není-li v příručce k ovladači napsáno jinak, není jej třeba měnit). Poslední parametr `pri` určuje logickou prioritu úlohy ovladače. Nevhodná hodnota priority může kriticky ovlivnit výkonnost celého řídicího systému, proto doporučujeme konzultovat příručku k ovladači a poté si ověřit zatížení řídicího systému (ovladačů, výpočetních

úrovní a úloh) v programu RexView.

Vstup

`prev`	Vstup sloužící pro k připojení prvního ovladače na výstup `Drivers` bloku `EXEC` nebo k připojení na výstup `next` předchozího ovladače	`long`

Výstup

`next`	Výstup sloužící pro zřetězování ovladačů připojením na vstup `prev` následujícího ovladače	`long`

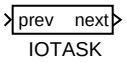
Parametry

`module`	Jméno modulu, ve kterém je daný vstupně výstupní ovladač obsažen (nemusí se zadávat, je-li shodné s `classname`)		`string`
`classname`	Jméno třídy ovladače, rozlišuje malá a velká písmena!	⊙`DrvClass`	`string`
`cfgname`	Jméno konfiguračního souboru ovladače	⊙`iodrv.rio`	`string`
`factor`	Násobek parametru `tick` bloku `EXEC` určující periodu spouštění úlohy ovladače	↓1 ⊙10	`long`
`stack`	Velikost zásobníku úlohy ovladače v bytech	↓1024 ⊙10240	`long`
`pri`	Priorita úlohy ovladače	↓1 ↑31 ⊙3	`long`
`timer`	Ovladač je zdrojem pro časování		`bool`

IOTASK – Úloha řídicího systému REX spouštěná ovladačem

Symbol bloku

Licence: STANDARD



Popis funkce

Standardní úlohy řídicího systému REX jsou do konfigurace zařazovány pomocí bloku TASK nebo QTASK. Takové úlohy jsou spouštěny systémovým časovačem, jehož tik (tick) se konfiguruje v bloku EXEC.

V některých případech však využití systémového časovače nevyhovyje, např. z důvodu příliš dlouhé nejkratší periody spouštění nebo pokud má být úloha spouštěna od externí události (přerušení od vstupního signálu) apod. V takovém případě může úlohu IOTASK spouštět přímo vstupně-výstupní ovladač zkonfigurovaný pomocí bloku TIODRV. Zda je uvedený způsob spouštění úloh v konkrétním ovladači implementován a za jakých podmínek, lze najít v uživatelské příručce daného ovladače.

Vstup

prev	Vstup sloužící pro k připojení první úlohy na výstup Tasks bloku TIODRV nebo k připojení na výstup next předchozí úlohy	long

Výstup

next	Výstup sloužící pro zřetězování úloh připojením na vstup prev následující úlohy	long

Parametry

factor	Parametr, který může být využit ovladačem pro určení periody úlohy, viz. uživatelská příručka daného ovladače	⊙1	long
stack	Velikost zásobníku (v bytech)	⊙10240	long
filename	Jméno souboru s příponou .mdl obsahující algoritmus úlohy; není-li jméno zadáno, je jméno souboru určeno jménem tohoto bloku (v hlavním souboru projektu) doplněném příponou .mdl		string

LPBRK – Rozpojení zpětné vazby

Symbol bloku

Licence: STANDARD

Popis funkce

Blok LPBRK je pomocným blokem často používaným v řídicích schématech složených z bloků systému REX. Blok se obvykle umisťuje do všech zpětných vazeb ve schématu. Jeho chování je však v systémech Simulink a REX odlišné.

V systému Simulink funguje blok LPBRK jako zpoždění signálu o jeden krok. Kdyby nebyl tento blok vložen do každé zpětné vazby, vyhodnotil by systém Simulink (od verze Matlab 6.1), že schéma obsahuje tzv. „rychlou smyčku“ a simulace by po čase selhala.

V systému REX je při překladu schématu programem RexComp tento blok vypuštěn, avšak ještě před tím způsobí přerušení zpětnovazební smyčky v místě svého výskytu. Pokud po vypuštění všech bloků LPBRK ještě v řídicím schématu zbývá nějaká smyčka, vypíše překladač RexComp varovnou zprávu a zpětnou vazbu rozpojí v místě, které si sám určí. Pro dosažení co nejvyšší kompatibility mezi systémy REX a Simulink se doporučuje používat blok LPBRK i v konfiguraci řídicího systému REX.

Vstup

u	Vstupní signál	double

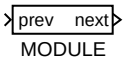
Výstup

y	Výstupní signál	double

MODULE – Rozšiřující modul systému REX

Symbol bloku

Licence: STANDARD



Popis funkce

Řídicí systém REX má otevřenou architekturu, jeho zabudované funkce lze tedy dále rozšiřovat a doplňovat. Toto rozšiřování je realizováno právě pomocí modulů. Každý modul je určen svým jménem (umístěným pod symbolem bloku). První rozšiřující modul se zařadí do konfigurace systému REX tím, že se jeho vstup `prev` propojí s výstupem `Modules` bloku EXEC. Další moduly se přidávají propojováním vstupu `prev` s výstupem `next` předchozího modulu. Na každý výstup `next` smí být připojen nejvýše jeden vstup `prev` následujícího modulu, u posledního modulu zůstává výstup `next` nepřipojen. Vzniklá posloupnost určuje pořadí zavádění jednotlivých modulů do řídicího systému REX a též pořadí jejich inicializace.

Každý modul je dodáván ve dvou verzích: ve verzi pro vývojové prostředí (`Host`) a pro cílové prostředí (`Target`). V operačních systémech Windows a Windows CE jsou moduly realizovány jako DLL knihovny se jmény `<modname>_H.dll` (pro vývojové prostředí) a `<modname>_T.dll` (pro cílové prostředí), kde `<modname>` je jméno modulu.

Vstup

`prev`	Vstup sloužící pro připojení prvního modulu na výstup `Modules` bloku EXEC nebo k připojení na výstup `next` předchozího modulu	`long`

Výstup

`next`	Výstup sloužící pro zřetězování modulů připojením na vstup `prev` následujícího modulu	`long`

PROJECT – * Další nastavení projektu

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

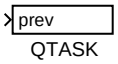
Parametry

`CompileParams`	Parametry příkazového řádku RexComp	`string`
`TargetURL`	Cílová URL adresa	`string`

QTASK – Rychlá úloha řídicího systému REX

Symbol bloku

Licence: STANDARD



Popis funkce

Blok QTASK slouží pro zařazení tzv. rychlé úlohy (quick task) s vysokou prioritou do exekutivy řídicího systému REX. Použití této úlohy je opodstatněné v případech, kdy je nutná co nejrychlejší zpracování vstupních signálů, např. pro číslicovou filtraci vstupních signálů zatížených šumem, nebo pro rychlou odezvu na stisk tlačítek připojených přes logické vstupy. Úloha se zařadí do exekutivy reálného času propojením vstupu prev s výstupem QTask bloku EXEC. Rychlá úloha se inicializuje před inicializací výpočetní úrovně Level0 (viz blok TASK).

Zkonfigurovaná úloha QTASK běží s logickou prioritou č. 2 a může být v systému REX nejvýše jedna. Algoritmus této úlohy se konfiguruje stejným způsobem jako algoritmus standardní úlohy TASK v samostatném souboru s příponou .mdl.

Úloha běží s periodou danou součinem parametru factor tohoto bloku a parametru tick exekutivy EXEC. Pro hodnotu factor=1 bude úloha spouštěna s nejkratší periodou tick a také zatížení systému bude největší. Pozor, v každé periodě se musí úloha QTASK stihnout za dobu kratší než tick, v opačném případě dojde k fatální chybě běhu exekutivy reálného času a vykonávání všech úloh se ukončí! Proto by úloha QTASK by měla být používána uvážlivě! Naštěstí lze dobu její exekuce zjistit v programu RexView.

Vstup

prev	Vstup, sloužící pro k připojení k výstupu QTask bloku EXEC	long

Parametry

factor	Násobek času tick bloku EXEC určující periodu úlohy (factor∗tick)	⊙1	long
stack	Velikost zásobníku (v bytech)	⊙10240	long
filename	Jméno souboru s příponou .mdl obsahující algoritmus úlohy; není-li jméno zadáno, je jméno souboru určeno jménem tohoto bloku (v hlavním souboru projektu) doplněným příponou .mdl		string

SLEEP – Časovací blok pro Simulink

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SLEEP slouží k zajištění co nejpřesnější periody spouštění algoritmu. V řídicím systému REX je časování výpočetních úloh zajištěno systémovými prostředky (viz blok EXEC), a proto je blok SLEEP ignorován. V systému Matlab/Simulink se pracuje se simulačním časem, který může běžet rychleji nebo pomaleji než reálný čas (podle výkonu počítače a složitosti algoritmu).

Má-li simulace běžet v reálném čase, stačí do simulačního algoritmu zařadit blok SLEEP, který jej v každém kroku pozastaví na tak dlouho, aby byl jeho algoritmus volán s periodou danou parametrem ts. Mechanismus samozřejmě funguje jen v případě, že simulace běží rychleji než ve skutečnosti.

V současné době je blok SLEEP implementován pro systém Matlab/Simulink ve verzi pro operační systémy Windows. Vzhledem k tomu, že ve Windows běží obvykle ještě jiné úlohy, které přerušují simulaci, je vhodné nepoužívat příliš krátké periody v řádu milisekund, doporučená hodnota je od 100 ms. Pro správnou funkci je nutné v parametrech simulace `Solver options` nastavit parametr `Type` na `fixed-step`, `discrete (no continuous states)` a parametr `Fixed step size` na stejnou hodnotu, jako parametr `ts` bloku SLEEP. Blok SLEEP by měl být nejvýše jeden v celém simulačním schématu (počítáno včetně subsystémů).

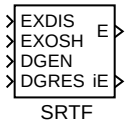
Parametr

`ts`	Perioda spouštění simulační úlohy v sekundách	⊙`0.1`	`double`

SRTF – Blok pro nastavování příznaků běhu

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok SRTF (Set Run-Time Flags) slouží pro nastavování příznaků určujících běh úloh, sekvencí (subsystémů) a bloků řídicího systému REX. Tento blok není určen pro Matlab-Simulink. V popisu tohoto bloku bude termín objekt označovat konkrétní objekt řídicího systému REX spouštěný v reálném čase, tj. vstupně-výstupní ovladač, některou z úloh (viz níže), výpočetní sekvenci (subsystém) nebo obyčejný blok systému REX.

Všechny níže uvedené operace jsou prováděny s objektem, jehož úplná cesta je uvedena v parametru bname. Není-li tento parametr zadán (prázdný řetězec), provádí se operace s nejbližším vlastníkem daného bloku, tj. pokud je blok obsažen v sekvenci (subsystému) pak s nejbližší nadřazenou sekvencí, jinak přímo s úlohou obsahující daný blok.

Příznaky bloku umožňují:

- **Zakázat spouštění** daného objektu vstupem EXDIS = on. Spouštění lze opětovně povolit (EXDIS = off). Vstup EXDIS nastavuje stejný příznak běhu jako tlačítko Halt/Run v pravém horním rohu záložky pracovního prostoru bloku (Workspace) v programu RexView.

- **Jednorázově spustit** daný objekt. Pokud je spouštění objektu zakázáno příznakem EXDIS = on nebo je zakázáno z programu RexView), lze vstupem EXOSH = on (One Shot Execution) spustit daný objekt právě jednou.

- **Povolit zjišťování diagnostických informací** pro objekt vstupem DGEN = on. Příznak je shodný s příznakem Enable nastavovaným z programu RexView z diagnostických záložek pro jednotlivé objekty (I/O Driver, Level, Quick Task, Task, I/O Task, Sequence).

- **Vynulovat diagnostické informace** pro daný objekt vstupem DGRES = on. Příznak je rovněž nastaven z programu RexView stisknutím tlačítka Reset v diagnostické záložce příslušného objektu. Po vynulování informací je v řídicím systému REX příznak automaticky shozen.

Následující tabulka ukazuje, jaké příznaky lze nastavovat pro různé druhy objektů řídicího systému REX.

Druh objektu	EXDIS	EXOSH	DGEN	DGRES
Vstupně výstupní ovladač (I/O Driver)	√	√	√	√
Výpočetní úroveň (Level)	√	×	√	√
Výpočetní úloha (Task)	√	√	√	√
Rychlá úloha (Quick Task)	√	√	√	√
Úloha vstupně-výstupního ovladače (I/O Task)	√	√	√	√
Výpočetní sekvence (Sequence, subsystém)	√	×	√	√
Obyčejný blok (Block)	√	×	×	×

Vstupy

`EXDIS`	Zakázání spouštění daného objektu	`bool`
`EXOSH`	Jednorázové spuštění daného objektu	`bool`
`DGEN`	Povolení shromažďování diagnostických informací o daném objektu	`bool`
`DGRES`	Vynulování diagnostických údajů o objektu	`bool`
`DGLOG`	Povolení rozšířené logování o objektu	`bool`

Výstupy

`E`	Příznak chyby `off` ... bez chyby `on` .... nastala chyba	`bool`
`iE`	Kód chyby (při `E` = `on`) `0` ..... bez chyby `1` ..... objekt nebyl nalezen, neplatný parametr `bname` `2` ..... interní chyba systému REX (nesprávné ukazatele) `3` ..... příznak se nepodařilo nastavit (timeout)	`long`

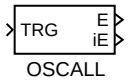
Parametr

`bname`	Úplná cesta k bloku (objektu), rozlišuje malá a velká písmena. Jednotlivé vrstvy jsou oddělovány tečkami, názvy objektů kromě úloh (`TASK`, `QTASK`) začínají jedním z následujících speciálních znaků:
	`^` ..... výpočetní úroveň (Level), např. `^0` pro `Level0`
	`&` ..... vstupně-výstupní ovladač (I/O Driver), např. `&WcnDrv`
	Jméno úlohy spouštěné vstupně-výstupním ovladačem (`IOTASK`) se zadává ve tvaru `&<jmeno_ovladace>.<jmeno_ulohy>`
		`string`

OSCALL – Volání funkcí operačního systému

Symbol bloku

Licence: STANDARD



Popis funkce

Blok OSCALL je určen pro volání funkcí operačního systému ze systému REX. Zvolená operace je spuštěna vzestupnou hranou (off→on) na vstupu TRG. Na jednotlivých platformách však nemusí být podporovány všechny funkce. Výsledek operace a případný chybový kód jsou indikovány pomocí výstupů E a iE.

Pro volání externích programů je možno též využít blok EPC.

Vstup

TRG	Spuštění zvolené akce	bool

Výstupy

E	Příznak chyby	bool
iE	Kód chyby	long
	i ..... obecná chyba systému REX	

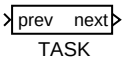
Parametr

action	Systémová funkce	⊙1	long
	1 ..... restartovat systém		
	2 ..... vypnout systém		
	3 ..... zastavit systém (HALT)		
	4 ..... synchronizace diskových jednotek		
	5 ..... zamknout systémovou partition		
	6 ..... odemknout systémovou partition		
	7 ..... povolit interní webserver		
	8 ..... zakázat interní webserver		

TASK – Standardní úloha řídicího systému REX

Symbol bloku

Licence: STANDARD



Popis funkce

Algoritmy řídicích úloh (task) jsou do systému REX zařazovány pomocí bloků typu TASK. Aplikace řídicího systému může obsahovat několik úloh, které se v konfiguraci systému zařazují do jednotlivých výpočetních úrovní připojením na výstupy Level0 až Level3 bloku EXEC. Vstup prev první úlohy dané úrovně <i> se propojí s výstupem Level<i> bloku EXEC. Další úlohy této úrovně se přidávají propojováním vstupu prev s výstupem next předchozí úlohy. Na každý výstup next smí být připojen nejvýše jeden vstup prev následující úlohy stejné úrovně, u poslední úlohy zůstává výstup next nepřipojen. Vzniklá posloupnost úloh dané úrovně určuje pořadí inicializace a spouštění úloh této úrovně v řídicím systému REX. Jednotlivé úrovně se inicializují v pořadí od Level0 do Level3 (rychlá úloha QTASK se inicializuje před úrovní Level0).

Všechny úlohy dané úrovně se spouštějí se shodnou prioritou danou parametrem pri<i> bloku EXEC a periodou rovnou násobku parametru factor a základní periody dané úrovně ntick<i>*tick v bloku EXEC. Pro svou exekuci má daná úloha vymezen čas od tiku č. start do tiku č. stop, přičemž parametry start a stop musí splňovat podmínku $0 \leq$ start $<$ stop $\leq$ ntick<i>. Navíc musí být splněna podmínka postupného spouštění úloh kontrolovaná překladačem RexComp říkající, že parametr stop předchozí úlohy nesmí být větší než parametr start úlohy následující (intervaly vymezené pro jednotlivé úlohy se nesmějí překrývat). V případě nesprávné volby časování jednotlivých úloh dané úrovně (jsou přerušovány úlohami vyšších úrovní a dalšími úlohami s vyšší prioritou), nedojde k ukončení činnosti systému (narozdíl od rychlé úlohy QTASK), ale vykonávání následujících úloh se odsouvá. Programem RexView (záložky Level a Task) lze zjistit, zda došlo k časovému posunutí pouze jednorázově nebo dochází k trvalému sklouzávání plánovaných časů.

Vstup

prev	Vstup sloužící pro k připojení první úlohy na některý z výstupů Level0 až Level3 bloku EXEC nebo k připojení na výstup next předchozí úlohy dané úrovně	long

Výstup

next	Výstup sloužící pro zřetězování úloh dané úrovně připojením na vstup prev následující úlohy téže úrovně	long

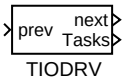
Parametry

`factor`	Faktor spouštění, násobek periody `tick*ntick<i>` bloku `i`-té výpočetní úrovně bloku EXEC určující periodu úlohy (`factor * tick* ntick<i>`) ⊙`1`	`long`
`start`	Číslo tiku periody dané výpočetní úrovně, na kterém má být úloha spuštěna ↓`0` ↑`ntick<i>`	`long`
`stop`	Číslo tiku periody dané výpočetní úrovně, do kterého má být úloha dokončena ↓`start+1` ↑`ntick<i>`	`long`
`stack`	Velikost zásobníku (v bytech) ⊙`10240`	`long`
`filename`	Jméno souboru s příponou `.mdl` obsahující algoritmus úlohy. Není-li jméno zadáno, je jméno souboru určeno jménem tohoto bloku (v hlavním souboru projektu) doplněným příponou `.mdl`.	`string`

TIODRV – Vstupně-výstupní ovladač systému REX s úlohami

Symbol bloku

Licence: STANDARD



Popis funkce

Blok TIODRV slouží pro konfiguraci speciálních ovladačů řídicího systému REX, které jsou samy schopny spouštět úlohy konfigurované bloky IOTASK, viz. uživatelská příručka konkrétního ovladače. První z úloh IOTASK se připojí svým vstupem prev na výstup Tasks bloku TIODRV. Pokud daný ovladač umožňuje spouštět více než jednu úlohu, připojí se další úloha svým vstupem prev na výstup next předchozí úlohy IOTASK, atd. Počet připojených úloh a jejich pořadí nekontroluje překladač RexComp (jako v případě bloků TASK), ale přímo vstupně-výstupní ovladač.

Pokud ovladač nemůže pro některou z úloh zajistit periodické spouštění (např. úloha spouštěná od externí události), nastaví pro tuto úlohu odpovídající příznak. Taková úloha nesmí obsahovat bloky, vyžadující konstantní periodu vzorkování (např. většina regulátorů). V případě, že nějaký ze zakázaných bloků je přesto použit, zahlásí exekutiva chybu běhu úlohy, kterou lze zjistit v programu RexView.

Vstup

prev	Vstup sloužící pro k připojení prvního ovladače na výstup Drivers bloku EXEC nebo k připojení na výstup next předchozího ovladače	long

Výstupy

next	Výstup pro řetězení ovladačů (s úlohami)	long
Tasks	Výstup sloužící pro zřetězování ovladačů připojením na vstup prev následujícího ovladače	long

Parametry

module	Jméno modulu, ve kterém je daný vstupně výstupní ovladač obsažen (nemusí se zadávat, je-li shodné s classname)		string
classname	Jméno třídy ovladače; rozlišuje malá a velká písmena!	⊙DrvClass	string
cfgname	Jméno konfiguračního souboru ovladače	⊙iodrv.rio	string
factor	Násobek parametru tick bloku EXEC určující periodu spouštění úlohy ovladače	↓1 ⊙10	long
stack	Velikost zásobníku úlohy ovladače v bytech	↓1024 ⊙10240	long
pri	Priorita úlohy ovladače	↓1 ↑31 ⊙3	long

`timer`	Ovladač je zdrojem pro časování	`bool`

WWW – * Obsah pro interní webserver

Symbol bloku Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Parametry

`Source`	Zdrojový adresář	`string`
`Target`	Cílový adresář	`string`
`Compression`	Aktivovat kompresi dat	`bool`

Kapitola 3

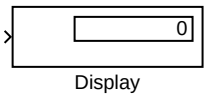
INOUT – Bloky vstupů a výstupů systému REX

Obsah

Display – * Zobrazení vstupní hodnoty

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstup

`u`	Vstupní signál	`unknown`

Parametry

`Format`	Formát zobrazované hodnoty
	`short`
	`long` ..
	`short_e`
	`long_e`
	`bank` ..
	`hex` ...
	`bin` ...
	`det` ...
		⊙1	`long`

`Decimation`	Po kolika periodách je hodnota zobrazována	↓1 ↑100000 ⊙1	`long`
`Suffix`	Přípona		`string`
`DispValue`	Zobrazená hodnota		`string`

From, INSTD – Připojení signálu nebo vstupní signál

Symboly bloků Licence: STANDARD

Popis funkce

Bloky From (připojení signálu) a INSTD (standardní vstup) mají stejný symbol a slouží k připojení vstupního signálu do řídicího algoritmu. Blok From se používá jak v řídicím systému REX tak i simulačním systému Matlab-Simulink, blok INSTD existuje však pouze v řídicím systému REX.

O tom, zda daný symbol bloku bude považován za blok From nebo INSTD rozhoduje překladač RexComp podle řetězcového parametru GotoTag následovně:

- Obsahuje-li parametr GotoTag oddělovač __ (za sebou dva znaky '_'), jedná se o blok INSTD. Část parametru (substring) před tímto oddělovačem (na výše uvedeném obrázku DRV) je považována za jméno bloku typu IODRV obsaženého v hlavním souboru projektu. Pokud takový ovladač není v hlavním souboru projektu obsažen, hlásí program RexComp chybu. V případě, že takový ovladač v projektu existuje, je druhá část parametru GotoTag (za oddělovačem, zde A) považována za jméno vstupního signálu v nalezeném ovladači. Toto jméno je daným ovladačem zkontrolováno a v případě, že ovladač zná vstupní signál s uvedeným jménem, je vytvořena instance bloku INSTD, která bude za běhu v reálném čase získávat hodnotu daného vstupního signálu a přivádět ji při každém spuštění dané úlohy do řídicího algoritmu.

- Pokud parametr GotoTag oddělovač "__" neobsahuje, je daný blok považován za blok From. Při překladu programem RexComp se hledá odpovídající blok Goto se stejným parametrem GotoTag a požadovanou viditelností danou parametrem TagVisibility (viz popis bloku Goto). V případě, že není nalezen, oznámí překladač RexComp varovnou zprávu a blok From odstraní. V opačném případě se propojí odpovídající bloky From a Goto, jako by byly propojeny „neviditelným“ vodičem. Blok From se i v tomto případě odstraní a proto nebude obsažen ve výsledné konfiguraci řídicího systému.

V systému Matlab-Simulink neexistuje blok INSTD, a proto i bloky, jejichž parametr GotoTag obsahuje znaky __, jsou bloky From. Této vlastnosti lze s výhodou využít pro simulaci řídicího systému včetně modelu. Model lze připojit k řídicímu systému pomocí bloků From a Goto, jejichž parametry GotoTag obsahují oddělovač __. Navíc lze dále využít vlastnost překladače RexComp, který záměrně ignoruje (vypouští) všechny subsystémy, jejichž jméno začíná řetězcem Simulation. Pokud je simulační model včetně

připojení svých vstupů a výstupů „schován“ do takového subsystému, lze přecházet od simulace k řízení v reálném čase systémem REX bez jakýchkoliv úprav souboru `.mdl`. Podrobněji viz [2].

Výstup

`value`	Signál z I/O ovladače nebo bloku `Goto`. Typ výstupu je určen typem signálu, který je na vlajku přiveden.	`unknown`

Parametr

`GotoTag`	Odkaz na parametr `GotoTag` bloku `Goto`, se kterým má být blok `From` propojen nebo odkaz na vstupní signál ovladače systému REX, který má být přiveden na výstup bloku.	`string`

Goto, OUTSTD – Zdroj signálu nebo výstupní signál

Symboly bloků Licence: STANDARD

Popis funkce

Bloky Goto (zdroj signálu) a OUTSTD (standardní výstup) mají stejný symbol a slouží k připojení výstupního signálu z řídicího algoritmu. Blok Goto se používá jak v řídicím systému REX tak i simulačním systému Matlab-Simulink, blok OUTSTD existuje však pouze v řídicím systému REX.

O tom, zda daný symbol bloku bude považován za blok Goto nebo OUTSTD, rozhoduje překladač RexComp podle řetězcového parametru GotoTag následovně:

- Obsahuje-li parametr GotoTag oddělovač __ (za sebou dva znaky '_'), jedná se o blok OUTSTD. Část parametru (substring) před tímto oddělovačem (na výše uvedeném obrázku DRV) je považována za jméno bloku typu IODRV obsaženého v hlavním souboru projektu. Pokud takový ovladač není v hlavním souboru projektu obsažen, hlásí program RexComp chybu. V případě, že takový ovladač v projektu existuje, je druhá část parametru GotoTag (za oddělovačem, zde A) považována za jméno výstupního signálu v nalezeném ovladači. Toto jméno je daným ovladačem zkontrolováno a v případě, že ovladač zná výstupní signál s uvedeným jménem, je vytvořena instance bloku OUTSTD, která bude při každém spuštění dané úlohy v reálném čase nastavovat hodnotu daného výstupního signálu z řídicího algoritmu do ovladače.

- Pokud parametr GotoTag oddělovač __ neobsahuje, je daný blok považován za blok Goto. Při překladu programem RexComp se hledá odpovídající blok From se stejným parametrem GotoTag, pro který je tento blok Goto viditelný (dosažitelný), viz dále. V případě, že není nalezen, oznámí překladač RexComp varovnou zprávu a blok Goto odstraní. V opačném případě se propojí odpovídající bloky Goto a From, jako by byly propojeny „neviditelným“ vodičem. Blok Goto se i v tomto případě odstraní a proto nebude obsažen ve výsledné konfiguraci řídicího systému.

Druhý parametr TagVisibility bloku Goto určuje viditelnost daného bloku uvnitř souboru .mdl. Může nabývat hodnot local, global a scoped, jejichž význam je vysvětlen v tabulce parametrů níže. V případě, že je daný blok přeložen jako blok OUTSTD je tento parametr ignorován.

V systému Matlab-Simulink neexistuje blok OUTSTD, a proto i bloky, jejichž parametr GotoTag obsahuje znaky __, jsou bloky Goto. Této vlastnosti lze s výhodou využít pro simulaci řídicího systému včetně modelu. Model lze připojit k řídicímu systému pomocí

bloků `Goto` a `From`, jejichž parametry `GotoTag` obsahují oddělovač `__`. Navíc lze dále využít vlastnost překladače RexComp, který záměrně ignoruje (vypouští) všechny subsystémy, jejichž jméno začíná řetězcem `Simulation`. Pokud je simulační model včetně připojení svých vstupů a výstupů „schován“ do takového subsystému, lze přecházet od simulace k řízení v reálném čase systémem REX bez jakýchkoliv úprav `.mdl` souboru. Podrobněji viz [2].

Vstup

`value`	Signál odesílaný do I/O ovladače nebo bloku `From`. V případě napojení na I/O ovladač systému REX, je typ vstupu určen ovladačem z parametru `GotoTag`.	`unknown`

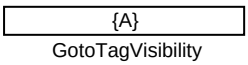
Parametry

`GotoTag`	Odkaz na parametr `GotoTag` bloku `From`, se kterým má být blok `Goto` propojen, nebo odkaz na výstupní signál ovladače systému REX, jehož hodnota je pak určena vstupem bloku.	`string`
`TagVisibility`	Viditelnost (dostupnost) daného bloku uvnitř `.mdl` souboru. Určuje podmínky pro umístění bloku `Goto` a k němu odpovídajícímu bloku `From` tak, aby byly vzájemně dostupné: ⊙`local` `local` oba bloky se musí nacházet ve stejném subsystému `global` bloky mohou být umístěny kdekoliv v daném `.mdl` souboru `scoped` bloky musí být umístěny ve stejném subsystému nebo v jakékoliv hierarchické úrovni pod umístěním bloku `GotoTagVisibility` se stejným parametrem `GotoTag`	`string`

`GotoTagVisibility` – Viditelnost zdroje signálu

Symbol bloku

Licence: STANDARD



Popis funkce

Bloky `GotoTagVisibility` upřesňují dostupnost (viditelnost) bloků `Goto` s viditelností `scoped`. Symbol (tag) specifikovaný v bloku `Goto` parametrem `GotoTag` je dostupný ze všech bloků `From` ze subsystému, který obsahuje odpovídající blok `GotoTagVisibility` a též ze všech subsystémů v hierarchii níže.

Blok `GotoTagVisibility` je požadován jen pro takové bloky `Goto`, jejichž parametr `TagVisibility` má hodnotu `scoped`. Pokud má parametr `TagVisibility` hodnoty `local` nebo `global`, není blok `GotoTagVisibility` třeba.

Blok `GotoTagVisibility` se používá jen při překladu konfigurace překladačem RexComp a ve výsledné konfiguraci není obsažen, protože v reálném čase nevykonává žádnou činnost.

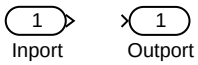
Parametr

`GotoTag`	Odkaz na parametr `GotoTag` bloku `Goto`, jehož viditelnost je dána umístěním tohoto bloku `GotoTagVisibility`	`string`

`Inport`, `Outport` – Vstupní a výstupní port

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky typů vstupní port (`Inport`) a výstupní port (`Outport`) slouží k propojování signálů mezi jednotlivými úrovněmi hierarchie. V řídicím systému REX se používají dvěma způsoby:

1. K připojení vstupů a výstupů subsystému. Bloky realizují přechod mezi symbolickou značkou subsystému a jeho vnitřkem (posloupností bloků skrytých v subsystému). Vlastní značka bloku `Inport` nebo `Outport` je obsažena uvnitř subsystému, jméno daného portu je znázorněno v symbolické značce subsystému v nadřazené hierarchické úrovni.

2. K propojení mezi výpočetními úlohami. V tomto případě jsou bloky obsaženy v nejvyšší hierarchické úrovni dané úlohy (souboru `.mdl`). Propojení vzájemně si jménem odpovídajících bloků `Inport` a `Outport` mezi různými úlohami zkontroluje a vytvoří překladač RexComp.

V obou případech je pořadí propojovaných vstupních a výstupních signálů určeno parametrem `Port` daného bloku. Číslování vstupních a výstupních portů je navzájem nezávislé, začíná od 1 a v obou případech se provádí automaticky jak v programu RexDraw, tak i v grafickém editoru systému Matlab-Simulink. Čísla portů musí být navíc jednoznačná v dané hierarchické úrovni, a proto v případě ruční změny čísla portu jsou ostatní porty automaticky přečíslovány. Pozor, pokud jsou přečíslovány porty již připojeného subsystému, dojde v důsledku změny pořadí vstupů (nebo výstupů) k změně připojení signálů v nadřazené úrovni subsystému!

V systému Matlab-Simulink mohou mít vstupní a výstupní porty ještě další funkce, které však v systému REX nejsou využívány. Podrobnou dokumentaci uvedených bloků pro Matlab-Simulink lze nalézt v [3].

Vstup

`value`	Hodnota odcházející na výstupní připojení nebo do bloku `Inport`	`unknown`

Výstup

`value`	Hodnota přicházející ze vstupního připojení nebo bloku `Outport`	`unknown`

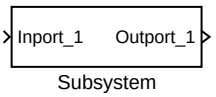
Parametr

`Port`	Číslo portu bloku `Inport` nebo `Outport`	`long`

SubSystem – Subsystém

Symbol bloku

Licence: STANDARD



Popis funkce

Blok **Subsystem** je prostředkem pro budování hierarchických řídicích (a simulačních) algoritmů tím, že umožňuje vkládat subsystém do jiného systému (subsystému). Subsystém se skládá z jednotlivých funkčních bloků, jejich vzájemných propojení a případně z dalších subsystémů. Při běhu řídicího systému REX se subsystém vykonává jako seřazená posloupnost bloků, proto je někdy nazýván výpočetní posloupností (anglicky *sequence*). Mezi bloky této posloupnosti není vykonán žádný blok z okolí subsystému, takže řídicí systém REX podporuje pouze subsystémy nazývané v terminologii systému Matlab-Simulink atomickými subsystémy (atomic subsystems), viz [3].

Subsystém může být vytvořen jak v programu RexDraw, tak i Matlab-Simulink, dvěma způsoby (dále je popsán postup v programu RexDraw):

- Zkopírováním bloku **Subsystem** z knihovny **INOUT** do daného schématu (soubor **.mdl**). Po otevření vytvořeného subsystému mohou být do něj přidávány bloky, včetně vstupních portů **Inport** a výstupních portů **Outport**.
- Označením skupiny bloků a volbou příkazu **Vytvoř subsystém** (**Create subsystem** z menu **Edit**. Vybrané bloky jsou nahrazeny subsystémem, po jehož otevření je možné vidět původní bloky a bloky **Inport** a **Outport**, zprostředkující spojení s bloky v nadřazené (původní) úrovni.

Vstupy

Počet a jména vstupů subsystému jsou dány počtem a jmény bloků **Inport** použitých uvnitř subsystému.

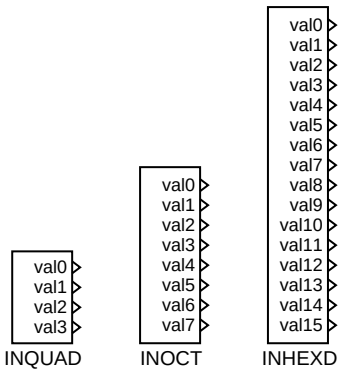
Výstupy

Počet a jména výstupů subsystému jsou dány počtem a jmény bloků **Outport** použitých uvnitř subsystému.

INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů

Symboly bloků

Licence: STANDARD



Popis funkce

Řídicí systém REX umožňuje kromě čtení každého jednotlivého vstupu z řízeného procesu také současné čtení několika signálů jedním blokem (například vstupů celého modulu nebo zásuvné desky). Pro popsaný způsob získávání vstupů slouží bloky `INQUAD`, `INOCT` a `INHEXD`, které se od sebe liší pouze maximálním počtem současně získaných signálů (po řadě 4, 8 a 16). Tyto bloky nemají obdobu v knihovně RexLib pro systém Matlab-Simulink.

Symbol ovladače `<DRV>` a název signálu `<signal>` z daného ovladače je kódován přímo do jména každé instance některého z uvedených bloků ve tvaru:

```
<DRV>__<signal>
```

Kódování jména bloku umístěného přímo pod symbolem bloku v řídicím algoritmu (a tedy na první pohled viditelného ze schématu) dodržuje stejná pravidla jako kódování parametru `GotoTag` bloků INSTD a OUTSTD.

Použití těchto bloků vícenásobných vstupů minimalizuje režii potřebnou k získání signálů prostřednictvím vstupně-výstupních ovladačů, což je významné zejména v případě velmi rychlých řídicích algoritmů s periodou vzorkování do 1 ms a navíc čte všechny uvedené vstupy buď současně nebo po sobě nejrychleji, jak je to možné. Informace, zda je možno pro konkrétní ovladač uvedené bloky používat a jakým způsobem jsou na jejich výstupech vyvedeny vstupy řídicího systému, lze nalézt v uživatelské příručce daného ovladače.

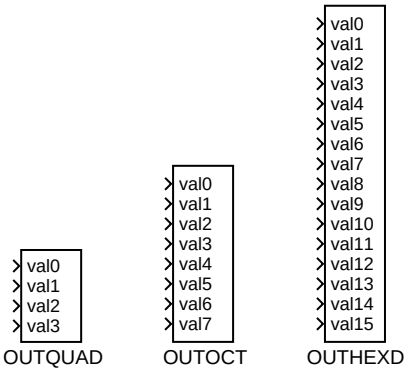
Výstupy

`val`i	Vstupní signály z procesu přivedené prostřednictvím I/O ovladačů do řídicího algoritmu. Typ a umístění jednotlivých signálů je popsáno v uživatelské příručce příslušného ovladače.	`unknown`

OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů

Symboly bloků

Licence: STANDARD



Popis funkce

Řídicí systém REX umožňuje kromě zápisu každého jednotlivého výstupu z řízeného procesu také současný zápis několika signálů jedním blokem (například výstupů celého modulu nebo zásuvné desky). Pro popsaný způsob nastavování výstupů slouží bloky OUTQUAD, OUTOCT a OUTHEXD, které se od sebe liší pouze maximálním počtem současně zapisovaných signálů (po řadě 4, 8 a 16). Tyto bloky nemají obdobu v knihovně RexLib pro systém Matlab-Simulink.

Symbol ovladače <DRV> a název signálu <signal> z daného ovladače je kódován přímo do jména každé instance některého z uvedených bloků ve tvaru:

```
<DRV>__<signal>
```

Kódování jména bloku umístěného přímo pod symbolem bloku v řídicím algoritmu (a tedy na první pohled viditelného ze schématu) dodržuje stejná pravidla jako kódování parametru GotoTag bloků INSTD a OUTSTD.

Použití těchto bloků vícenásobných výstupů minimalizuje režii potřebnou k nastavení signálů prostřednictvím vstupně-výstupních ovladačů, což je významné zejména v případě velmi rychlých řídicích algoritmů s periodou vzorkování do 1 ms a navíc zapisuje všechny uvedené vstupy buď současně nebo po sobě nejrychleji, jak je to možné. Informace, zda je možno pro konkrétní ovladač uvedené bloky používat a jakým způsobem se na jejich vstupy připojují výstupy řídicího systému lze nalézt v uživatelské příručce daného ovladače.

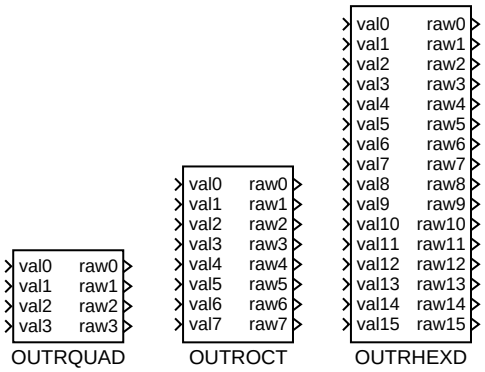
Vstupy

`val`*i*	Výstupní signály řídicího algoritmu do procesu nastavované `unknown` prostřednictvím I/O ovladačů. Typ a umístění jednotlivých signálů je popsáno v uživatelské příručce příslušného ovladače.

OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací

Symboly bloků

Licence: ADVANCED



Popis funkce

Bloky `OUTRQUAD`, `OUTROCT` a `OUTRHEXD` se používají pro nastavování několika výstupů najednou podobně jako bloky OUTQUAD, OUTOCT a OUTHEXD. Navíc však umožňují získat pro každý i-tý výstup ovladače přivedený na vstup `val`i zpětnou informaci o výsledku zápisu na odpovídajícím výstupu `raw`i daného bloku.

Výstupy `raw`i mohou být použity k informování řídicího algoritmu o výsledku zápisu dvojím způsobem:

- Hodnotou tohoto výstupu, který může např. u analogového výstupu při překročení maximálního rozsahu A/D převodníku vracet skutečně zapsanou bitovou hodnotu (odtud je v názvu text `raw`).
- Prozkoumáním příznaků kvality tohoto signálu, které lze od signálu oddělit blokem VIN a dále zpracovat blokem QFD.

Hodnota odpovídající danému zápisu se na výstupech `raw`i nemusí objevit ihned po spuštění daného bloku, ale může mít určité zpoždění dané vlastnostmi použitého ovladače, např. zpožděním komunikace s cílovým zařízením.

Tyto bloky nemají obdobu v knihovně RexLib pro systém Matlab-Simulink.

Vstupy

`val`i	výstupní signály řídicího algoritmu do procesu nastavované `unknown` prostřednictvím I/O ovladačů. Typ a umístění jednotlivých signálů je popsáno v uživatelské příručce příslušného ovladače.

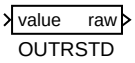
Výstupy

`raw`*i*	Zpětná informace od ovladače o výsledku nastavení odpovídajícího výstupu. Typ a význam signálů je popsán v uživatelské příručce příslušného ovladače.	`unknown`

OUTRSTD – Výstupní signál s verifikací hodnoty

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok OUTRSTD se používá pro nastavování výstupu z řídicího algoritmu podobně jako blok OUTSTD. Navíc však umožňuje získat zpětnou informaci o výsledku zápisu na výstupu raw daného bloku.

Výstup raw může být použit k informování řidicího algoritmu o výsledku zápisu dvojím způsobem:

- Hodnotou tohoto výstupu, která může např. u analogového výstupu při překročení maximálního rozsahu A/D převodníku vracet skutečně zapsanou bitovou hodnotu (odtud je název raw).
- Prozkoumáním příznaků kvality tohoto signálu, které lze od signálu oddělit blokem VIN a dále zpracovat blokem QFD.

Hodnota odpovídající danému zápisu se na výstupu raw nemusí objevit ihned po spuštění daného bloku, ale může mít určité zpoždění dané vlastnostmi použitého ovladače, např. zpožděním komunikace s cílovým zařízením.

Tento blok nemá obdobu v knihovně RexLib pro systém Matlab-Simulink.

Vstup

value	Výstupní signál řídicího algoritmu nastavovaný prostřednictvím ovladače do procesu. Typ a pojmenování signálu je popsáno v uživatelské příručce příslušného ovladače.	unknown

Výstup

raw	Zpětná informace od ovladače o výsledku nastavení výstupu. Typ a význam signálu je popsán v uživatelské příručce příslušného ovladače.	unknown

QFC – Kódování příznaků kvality signálu

Symbol bloku

Licence: ADVANCED

Popis funkce

Blok QFC vytváří kombinací tří složek iq, is a il výsledný 8 bitový kód iqf příznaků kvality signálu. Příznaky kvality jsou součástí každého vstupního i výstupního signálu v řídicím systému REX. Bližší informace o jejich využití jsou uvedeny v kapitole 1.4 této příručky. Knihovna RexLib pro Matlab-Simulink příznaky kvality nepoužívá.

Blok QFC lze využít v kombinaci s blokem VOUT pro nastavení potřebných příznaků kvality danému signálu. Obrácenou funkci k bloku QFC provádí blok QFD.

Vstupy

iq	Základní příznaky kvality, viz tab. 1.2, str. 14	long
is	Doplňující příznaky kvality, viz [1]	long
il	Příznaky dosažení mezních úrovní, viz [1]	long

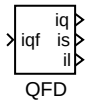
Výstup

iqf	Bitová kombinace vstupních signálů iq, is a il	long

QFD – Dekódování příznaků kvality signálu

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok QFD rozkládá 8 bitové příznaky kvality iqf na jednotlivé složky iq, is a il. Příznaky kvality jsou součástí každého vstupného i výstupního signálu v řídicím systému REX. Bližší informace o jejich využití jsou uvedeny v kapitole 1.4 této příručky. Knihovna RexLib pro Matlab-Simulink příznaky kvality nepoužívá.

Blok QFD lze využít v kombinaci s blokem VIN pro detailní zpracování příznaků kvality vstupního signálu u bloku VIN v řídicím algoritmu. Obrácenou funkci k bloku QFD provádí blok QFC.

Vstup

iqf	Příznaky kvality, které mají být dekomponovány na složky iq, is a il	long

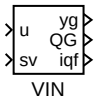
Výstupy

iq	Příznaky základního typu kvality, viz tabulku 1.2, str. 14	long
is	Doplňující příznaky kvality, viz [1]	long
il	Příznaky dosažení mezních úrovní, viz [1]	long

VIN – Ověření kvality vstupního signálu

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok VIN slouží pro ověření kvality vstupního signálu u v řídicím systému REX. Bližší informace o využití příznaků kvality jsou uvedeny v kapitole 1.4 této příručky. V prostředí Matlab-Simulink nemá tento blok význam.

Blok průběžně odděluje příznaky kvality vstupu u a nastavuje je na výstup iqf. Na základě těchto příznaků a parametru GU(Good if Uncertain) jsou vstupní signály v bloku VIN dále zpracovány následujícím způsobem:

- Pro GU = off je hodnota výstupu QG nastavena na on, pouze pokud je kvalita vstupu dobrá (GOOD). V případě špatné (BAD) nebo nejisté (UNCERTAIN) kvality je nastaveno QG = off.
- Pro GU = on je hodnota výstupu QG nastavena na on, pokud je kvalita vstupu dobrá (GOOD) nebo nejistá (UNCERTAIN). V případě špatné (BAD) kvality je nastaveno QG = off.

Je-li vstupní signál u vyhodnocen jako kvalitní (QG = on, je přiveden na výstup yg. V případě problémů s kvalitou signálu je pro výstup použit náhradní signál ze vstupu sv (substitution variable).

Vstupy

u	Vstupní signál, jehož kvalita se vyhodnocuje. Typ signálu je určen podle typu připojené hodnoty.	unknown
sv	Náhradní hodnota pro případ chyby	unknown

Výstupy

yg	Validní výstupní signál (u pro QG = on nebo sv pro QG = off)	unknown
QG	Indikátor platnosti vstupního signálu	bool
iqf	Úplné příznaky kvality oddělené od vstupu u	long

Parametr

`GU`	Přípustnost kvality UNCERTAIN `off` ... kvalita UNCERTAIN je nepřípustná `on` .... kvalita UNCERTAIN je přípustná	`bool`

VOUT – Nastavení kvality výstupního signálu

Symbol bloku

Licence: ADVANCED

Popis funkce

Blok VOUT umožňuje signálu u nastavit (vnutit) příznaky kvality ze vstupu iqf. Bližší informace o využití příznaků kvality jsou uvedeny v kapitole 1.4 této příručky. Blok je určen pro řídicí systém REX, v prostředí Matlab-Simulink nemá význam.

Vstupy

u	Vstup, jehož příznaky kvality mají být nahrazeny. Typ tohoto vstupu je určen podle připojeného signálu.	unknown
iqf	Požadované příznaky kvality	long

Výstup

yq	Výsledný signál sestavený z hodnoty vstupu u a příznaků kvality daných hodnotou vstupu iqf. Typ výstupu je určen podle připojeného vstupního signálu u.	unknown

Kapitola 4

MATH – Matematické bloky

Obsah

ABS_ – Absolutní hodnota

Symbol bloku

Licence: STANDARD

Popis funkce

Blok ABS_ počítá absolutní hodnotu analogového vstupního signálu u. Na výstupu y je absolutní hodnota vstupu $y = |u|$ a výstup sgn určuje znaménko vstupu,

$$sgn = \begin{cases} -1, & \text{pro } u < 0, \\ 0, & \text{pro } u = 0, \\ 1, & \text{pro } u > 0. \end{cases}$$

Vstup

u	Analogový vstupní signál	double

Výstupy

y	Absolutní hodnota vstupního signálu	double
sgn	Indikátor znaménka vstupního signálu	long

ADD – Součet dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok ADD počítá součet dvou vstupních analogových signálů, výstup je dán vztahem

$$y = u1 + u2.$$

Pro sčítání a odečítání více signálů můžete použít blok ADDOCT.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double

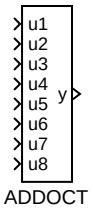
Výstup

y	Součet vstupních signálů	double

ADDOCT – Součet osmi signálů

Symbol bloku

Licence: STANDARD



Popis funkce

Blok ADDOCT sečte více (až 8) vstupních signálů. Parametr nl udává seznam vstupů, které se místo přičtení odečítají. Pokud je tento parametr prázdný, tak blok provádí funkci y = u1 + u2 + u3 + u4 + u5 + u6 + u7 + u8. Pokud bude například nl=2,5,7, tak bude realizována funkce y = u1 − u2 + u3 + u4 − u5 + u6 − u7 + u8.

Pro jednoduché operace sčítání a odečítání můžete použít bloky ADD a SUB.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double
u3	Třetí analogový vstup bloku	double
u4	Čtvrtý analogový vstup bloku	double
u5	Pátý analogový vstup bloku	double
u6	Šestý analogový vstup bloku	double
u7	Sedmý analogový vstup bloku	double
u8	Osmý analogový vstup bloku	double

Výstup

y	Součet vstupních signálů	double

Parametr

nl	Seznam signálů, které se místo přičítání odečítají. Zadává se ve tvaru např. 1,3..5,8. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	long

CNB – Booleovská (logická) konstanta

Symbol bloku

Licence: STANDARD

Popis funkce

Blok CNB slouží pro zadání Booleovské (logické) konstanty.

Výstup

Y	Logický výstupní signál	bool

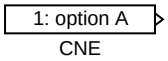
Parametr

YCN	Booleovská (logická) konstanta off ... zakázáno on .... povoleno	⊙on	bool

CNE – Předdefinovaná konstanta

Symbol bloku

Licence: STANDARD



Popis funkce

Blok CNE umožňuje výběr celočíselné konstanty z předem připraveného seznamu. Rozbalovací seznam konstant je definován řetězcem pupstr, jehož syntaxe je zřejmá z počáteční hodnoty uvedené níže. Na výstupu bloku je celočíselná hodnota odpovídající číslu ze začátku vybrané položky. V případě, že formát řetězce pupstr není správný, je na výstupu bloku 0.

V Simulinku je připravena knihovna CNEs, ve které jsou připraveny bloky CNE s nejčastěji používanými seznamy konstant.

Parametry

yenum	Konstanta ze seznamu	⊙1: option A	string
pupstr	Definice seznamu konstant	⊙1: option A|2: option B|3: option C	string

Výstup

iy	Celočíselný výstupní signál	long

CNI – Celočíselná konstanta

Symbol bloku

Licence: STANDARD

Popis funkce

Blok CNI slouží pro zadání celočíselné konstanty.

Výstup

iy	Celočíselný výstupní signál		long

Parametr

icn	Celočíselná konstanta	⊙1	long

CNR – Reálná konstanta

Symbol bloku

Licence: STANDARD

Popis funkce

Blok CNR slouží pro zadání reálné konstanty.

Výstup

y	Analogový výstupní signál		double

Parametr

ycn	Reálná konstanta	⊙1.0	double

DIF_ – Blok diference

Symbol bloku

Licence: STANDARD

Popis funkce

Blok DIF_ počítá diferenci vstupního signálu u podle vztahu

$$y_k = u_k - u_{k-1},$$

kde $u = u_k$, $y = y_k$ a u_{k-1} je vstup u zpožděný o jeden krok (o periodu T_S, s níž je blok spouštěn).

Vstup

u	Analogový vstupní signál	double

Výstup

y	Diference vstupního signálu	double

Parametr

ISSF	Nulový výstup při spuštění off ... v prvním cyklu bude na výstupu y = u on .... v prvním cyklu bude výstup y = 0	bool

DIV – Dělení dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok DIV dělí dva vstupní analogové signály y = u1/u2. V případě, že je u2 = 0, nastaví se výstup E = on a na výstup y je dána náhradní hodnota y = yerr.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double

Výstupy

y	Podíl vstupních signálů	double
E	Indikátor chyby – dělení nulou	bool

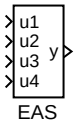
Parametr

yerr	Náhradní hodnota pro případ chyby	⊙1.0	double

EAS – Rozšířené sčítání a odečítání

Symbol bloku

Licence: STANDARD



Popis funkce

Blok EAS sčítá vstupní analogové signály u1, u2, u3 a u4 s příslušnými vahami a, b, c a d. Výstup y je pak dán vztahem

$$y = a * u1 + b * u2 + c * u3 + d * u4 + y0.$$

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double
u3	Třetí analogový vstup bloku	double
u4	Čtvrtý analogový vstup bloku	double

Výstup

y	Analogový výstupní signál	double

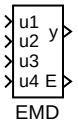
Parametry

a	Váhový koeficient pro vstup u1	⊙1.0	double
b	Váhový koeficient pro vstup u2	⊙1.0	double
c	Váhový koeficient pro vstup u3	⊙1.0	double
d	Váhový koeficient pro vstup u4	⊙1.0	double
y0	Aditivní konstanta (bias)		double

EMD – Rozšířené násobení a dělení

Symbol bloku

Licence: STANDARD



Popis funkce

Blok EMD slouží k násobení a dělení vstupních analogových signálů u1, u2, u3 a u4 s příslušnými vahami a, b, c a d. Výstup y je pak dán vztahem

$$y = \frac{(a * u1 + a0)(b * u2 + b0)}{(c * u3 + c0)(d * u4 + d0)}. \tag{4.1}$$

V případě, že jmenovatel vztahu (4.1) je roven 0, nastaví se výstup E = on a na výstup y je dána náhradní hodnota y = yerr.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double
u3	Třetí analogový vstup bloku	double
u4	Čtvrtý analogový vstup bloku	double

Výstupy

y	Analogový výstupní signál	double
E	Indikátor chyby - dělení nulou	bool

Parametry

a	Váhový koeficient pro vstup u1	⊙1.0	double
a0	Aditivní konstanta pro vstup u1		double
b	Váhový koeficient pro vstup u2	⊙1.0	double
b0	Aditivní konstanta pro vstup u2		double
c	Váhový koeficient pro vstup u3	⊙1.0	double
c0	Aditivní konstanta pro vstup u3		double
d	Váhový koeficient pro vstup u4	⊙1.0	double
d0	Aditivní konstanta pro vstup u4		double
yerr	Náhradní hodnota pro případ chyby	⊙1.0	double

FNX – Výpočet hodnoty funkce jedné proměnné

Symbol bloku

Licence: STANDARD

Popis funkce

Blok FNX počítá hodnotu základních matematických funkcí jedné proměnné. Seznam dostupných funkcí s příslušnými omezeními je v níže uvedené tabulce. Vybraná funkce ze seznamu je určená parametrem ifn.
Tabulka funkcí bloku FNX:

ifn: zkratka	funkce	omezení u
1: acos	arcus cosinus	$u \in < -1.0, 1.0 >$
2: asin	arcus sinus	$u \in < -1.0, 1.0 >$
3: atan	arcus tangens	–
4: ceil	zaokrouhlení na nejbližší vyšší celé číslo	–
5: cos	cosinus	–
6: cosh	cosinus hyperbolický	–
7: exp	exponenciální křivka e^u	–
8: exp10	exponenciální křivka 10^u	–
9: fabs	absolutní hodnota	–
10: floor	zaokrouhlení na nejbližší nižší celé číslo	–
11: log	logaritmus	$u > 0$
12: log10	dekadický logaritmus	$u > 0$
13: random	náhodné číslo $z < 0, 1 >$ (nezávisí na u)	–
14: sin	sinus	–
15: sinh	sinus hyperbolický	–
16: sqr	druhá mocnina	–
17: sqrt	druhá odmocnina	$u > 0$
18: srand	mění násadu pro funkci random na u	$u \in \mathbb{N}$
19: tan	tangens	–
20: tanh	tangens hyperbolický	-

V případě, že vstup u je mimo povolený rozsah nebo nastala chyba při výpočtu funkční hodnoty zvolené funkce (závisí na implementaci), např. výpočet odmocniny záporného čísla, je aktivován chybový výstup E = on a na výstup y je nastavena náhradní hodnota y = yerr.

Vstup

`u`	Analogový vstupní signál		`double`

Výstupy

`y`	Výsledek vybrané funkce		`double`
`E`	Příznak chyby		`bool`

Parametry

`ifn`	Typ funkce (viz tabulka výše)	⊙1	`long`
`yerr`	Náhradní hodnota pro případ chyby		`double`

FNXY – Výpočet hodnoty funkce dvou proměnných

Symbol bloku

Licence: STANDARD

Popis funkce

Blok FNXY počítá hodnotu základních matematických funkcí dvou proměnných. Seznam dostupných funkcí s příslušnými omezeními je v níže uvedené tabulce. Vybraná funkce ze seznamu je určená parametrem ifn.
Tabulka funkcí bloku FNXY:

ifn: zkratka	funkce	omezení u1, u2
1: atan2	arcus tangens u1/u2	–
2: fmod	zbytek po dělení u1/u2	u2 $\neq 0.0$
3: pow	výpočet mocniny y = u1^{u2}	viz níže

Funkce atan2 vrací funkční hodnotu v intervalu $\langle -\pi, \pi \rangle$. Pro určení správného kvadrantu se využívá znamének obou vstupů u1 a u2.

Funkce fmod počítá zbytek po dělení u1/u2 tak, že platí u1 $= i *$ u2 + y, kde i je celé číslo, výstup y má stejné znaménko jako vstup u1 a pro absolutní hodnotu výstupu y platí: |y| < |u2|.

Výpočet mocniny funkcí pow se řídí následujícími pravidly:

- Nepracuje se vstupními hodnotami u1 a u2 většími než 2^{64},
- u1$^0 = 1$ pro libovolné u1 (i u1 $= 0$),
- 0^{u2} vrací chybu pro u2 < 0.

V případě, že vstup u2 nesplňuje omezení nebo nastala chyba při výpočtu funkční hodnoty zvolené funkce (závisí na implementaci), je aktivován chybový výstup E = on a na výstup y je nastavena náhradní hodnota y = yerr.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double

Výstupy

y	Výsledek vybrané funkce	double

`E`	Příznak chyby `off` ... bez chyby `on` .... nastala chyba		`bool`

Parametry

`ifn`	Typ funkce (viz tabulka výše) `1` ..... atan2 `2` ..... fmod `3` ..... pow	⊙`1`	`long`
`yerr`	Náhradní hodnota pro případ chyby		`double`

GAIN – Násobení konstantou

Symbol bloku

Licence: STANDARD

Popis funkce

Blok GAIN násobí analogový vstup u reálnou konstantou k. Výstup je pak

$$y = ku.$$

Vstup

u	Analogový vstupní signál		double

Výstup

y	Analogový výstupní signál		double

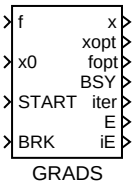
Parametr

k	Zesílení	⊙1.0	double

GRADS – Gradientní optimalizace

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok GRADS umožňuje provádět jednodimenzionální minimalizaci funkce $f(\mathtt{x}, v)$ gradientní metodou, kde $\mathtt{x} \in \langle \mathtt{xmin}, \mathtt{xmax} \rangle$ je optimalizační proměnná a y je libovolná vektorová proměnná. Předpokládá se, že pro daný výstup x v kroku k je hodnota funkce $f(\mathtt{x}, v)$ vyčíslena na vstupu f v kroku $(k + \mathtt{n})$. To značí, že jednotlivé iterace gradientní metody jsou prováděny s periodou $\mathtt{n} * T_S$, kde T_S je perioda spouštění bloku GRADS. Délka kroku gradientní metody je určována podle vztahu

$$\begin{aligned} grad &= (\mathtt{f}_i - \mathtt{f}_{i-1}) * (dx)_{i-1} \\ (dx)_i &= -\mathtt{gamma} * grad, \end{aligned}$$

kde k značí číslo iterace. Je-li krok $((dx)_i < \mathtt{dmin})$ nebo $((dx)_i > \mathtt{dmax})$, potom je příslušně omezen.

Vstupy

f	Hodnota minimalizované funkce $f(.)$ v bodě x	double
x0	Startovní bod optimalizace	double
START	Spouštěcí signál (reaguje na náběžnou hranu)	bool
BRK	Signál pro předčasné přerušení	bool

Výstupy

x	Aktuální hodnota optimalizované proměnné x	double
xopt	Výsledná optimální hodnota proměnné x	double
fopt	Výsledná optimální hodnota funkce $f(\mathtt{x}, v)$	double
BSY	Indikátor probíhající optimalizace	bool
iter	Číslo aktuální iterace	long
E	Příznak chyby	bool
iE	Kód chyby	long
	1 ..... $\mathtt{x} \notin < \mathtt{xmin}, \mathtt{xmax} >$	
	2 ..... $\mathtt{x} = \mathtt{xmin}$ nebo $\mathtt{x} = \mathtt{xmax}$	

Parametry

`xmin`	Dolní mez přípustného intervalu optimální proměnné `x`		`double`
`xmax`	Horní mez přípustného intervalu optimální proměnné `x`	⊙`10.0`	`double`
`gamma`	Koeficient gradientní metody určující velikost kroku	⊙`0.3`	`double`
`d0`	Počáteční krok gradientní metody	⊙`0.05`	`double`
`dmin`	Minimální krok gradientní metody	⊙`0.01`	`double`
`dmax`	Maximální krok gradientní metody	⊙`1.0`	`double`
`n`	Perioda jedné iterace (v periodách vzorkování bloku T_S)	⊙`100`	`long`
`itermax`	Maximální počet iterací před ukončením	⊙`20`	`long`

IADD – Celočíselné sčítání

Symbol bloku

Licence: STANDARD

Popis funkce

Blok IADD sečte dva vstupní celočíselné signály n = i1 + i2. V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem vtype. Pokud se součet vejde do rozsahu proměnné, je výsledkem normální součet. V opačném případě výsledek závisí na hodnotě parametru SAT.

Pro SAT = off se přetečení rozsahu nekontroluje, tj. nastaví se výstup E = off a výstup n tak, jak počítá procesor. Například pro typ Short, který má rozsah -32768..+32767, dostaneme 30000 + 2770 = -32766).

Pro SAT = on se při přetečení rozsahu nastaví výstup E = on a na výstup n je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme 30000 + 2770 = 32767).

Vstupy

i1	První celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long
i2	Druhý celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long

Výstupy

n	Celočíselný součet vstupních signálů	long
E	Příznak chyby – přetečení rozsahu off ... bez chyby on .... nastala chyba	bool

Parametry

vtype	Typ hodnoty, může nabývat hodnot: 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙4	long

`SAT`	Kontrola přetečení `off` ... přetečení se nekontroluje `on` .... přetečení se kontroluje	`bool`

ISUB – Celočíselné odčítání

Symbol bloku

Licence: STANDARD

Popis funkce

Blok ISUB sečte dva vstupní celočíselné signály n = i1 − i2. V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem vtype. Pokud se rozdíl vejde do rozsahu proměnné, je výsledkem normální rozdíl. V opačném případě výsledek závisí na hodnotě parametru SAT.

Pro SAT = off se přetečení rozsahu nekontroluje, tj. nastaví se výstup E = off a výstup n tak jak počítá procesor (například pro typ Short, který má rozsah -32768..+32767 dostaneme 30000 - -2770 = -32766).

Pro SAT = on se při přetečení rozsahu nastaví výstup E = on a na výstup n je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme 30000 - -2770 = 32767).

Vstupy

i1	První celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long
i2	Druhý celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long

Výstupy

n	Celočíselný rozdíl vstupních signálů	long
E	Příznak chyby – přetečení rozsahu off ... bez chyby on .... nastala chyba	bool

Parametry

vtype	Číselný typ 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙4	long

`SAT`	Kontrola přetečení `off` ... přetečení se nekontroluje `on` .... přetečení se kontroluje	`bool`

IMUL – Celočíselné násobení

Symbol bloku

Licence: STANDARD

Popis funkce

Blok IMUL vynásobí dva vstupní celočíselné signály n = i1 * i2. V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem vtype. Pokud se součin vejde do rozsahu proměnné, je výsledkem normální součin. V opačném případě výsledek závisí na hodnotě parametru SAT.

Pro SAT = off se přetečení rozsahu nekontroluje, tj. nastaví se výstup E = off a výstup n tak jak počítá procesor (například pro typ Short, který má rozsah -32768..+32767 dostaneme 2000 * 20 = -25536).

Pro SAT = on se při přetečení rozsahu nastaví výstup E = on a na výstup n je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme 2000 * 20 = 32767).

Vstupy

i1	První celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long
i2	Druhý celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long

Výstupy

n	Celočíselný součin vstupních signálů	long
E	Příznak chyby – přetečení rozsahu off ... bez chyby on .... nastala chyba	bool

Parametry

vtype	Číselný typ 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙4	long

`SAT`	Kontrola přetečení `off` ... přetečení se nekontroluje `on` .... přetečení se kontroluje	`bool`

IDIV – Celočíselné dělení

Symbol bloku

Licence: STANDARD

Popis funkce

Blok IDIV dělí dva vstupní celočíselné signály n = i1 ÷ i2, kde ÷ označuje operátor celočíselného dělení. Pokud je obyčejný (neceločíselný, normální) podíl obou operandů celé číslo je tato hodnota i výsledkem celočíselného dělení. V opačném případě je výsledkem hodnota, která vznikne „odříznutím“ desetinné části normálního podílu k nejbližšímu celému číslu směrem blíže k nule. V případě, že i2 = 0, nastaví se výstup E = on a na výstup n je dána náhradní hodnota n = nerr.

Vstupy

i1	První celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long
i2	Druhý celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long

Výstupy

n	Celočíselný podíl vstupních signálů	long
E	Příznak chyby – dělení nulou	bool

Parametr

vtype	Typ hodnoty, může nabývat hodnot: 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙1	long
nerr	Náhradní hodnota pro případ chyby	⊙1	long

IMOD – Zbytek po celočíselném dělení

Symbol bloku

Licence: STANDARD

Popis funkce

Blok IMOD dělí dva vstupní celočíselné signály n = i1%i2, kde % označuje operátor zbytku celočíselného dělení (modulo). Pokud jsou obě čísla kladná a dělitel větší než jedna, je výsledek buď nula (pro soudělná čísla) nebo kladné číslo menší než dělitel. V případě, že je jedno z čísel záporné, má výsledek znaménko dělence, např. $15\%10 = 5$, $15\%(-10) = 5$, ale $(-15)\%10 = -5$. V případě, že i2 = 0, nastaví se výstup E = on a na výstup n je dána náhradní hodnota n = nerr.

Vstupy

i1	První celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long
i2	Druhý celočíselný vstup bloku	↓-9,22E+18 ↑9,22E+18	long

Výstupy

n	Zbytek po celočíselném dělení	long
E	Příznak chyby – dělení nulou	bool

Parametr

vtype	Typ hodnoty, může nabývat hodnot: 2 ..... Byte (rozsah 0 ... 255) 3 ..... Short (rozsah -32768 ... 32767) 4 ..... Long (rozsah -2147483648 ... 2147483647) 5 ..... Word (rozsah 0 ... 65536) 6 ..... DWord (rozsah 0 ... 4294967295) 10 .... Large (rozsah -9223372036854775808...9223372036854775807)	⊙1	long
nerr	Náhradní hodnota pro případ chyby	⊙1	long

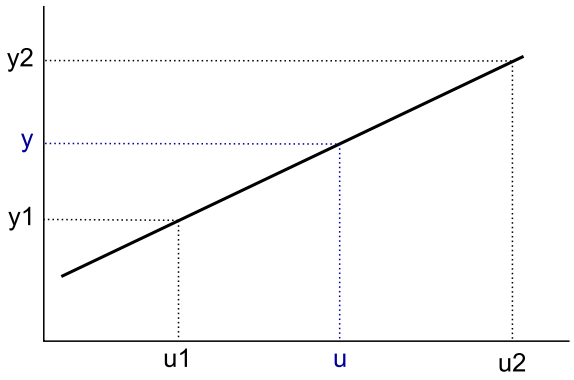
LIN – Lineární interpolace

Symbol bloku

Licence: STANDARD

Popis funkce

Blok LIN počítá lineární interpolaci. Následující obrázek ilustruje výpočet výstupu y ze vstupu u a ze zadaných bodů [u1, y1] a [u2, y2].



Vstup

u	Analogový vstupní signál		double

Výstup

y	Analogový výstupní signál		double

Parametry

u1	Souřadnice prvního bodu v ose x		double
y1	Souřadnice prvního bodu v ose y		double
u2	Souřadnice druhého bodu v ose x	⊙1.0	double
y2	Souřadnice druhého bodu v ose y	⊙1.0	double

MUL – Násobení dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok MUL násobí dva vstupní analogové signály $y = u1 \cdot u2$.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double

Výstup

y	Součin vstupních signálů	double

POL – Vyhodnocení polynomu

Symbol bloku

Licence: STANDARD

Popis funkce

Blok POL počítá hodnotu polynomiální funkce ve tvaru:

$$\mathtt{y} = \mathtt{a}_0 + \mathtt{a}_1\mathtt{u} + \mathtt{a}_2\mathtt{u}^2 + \mathtt{a}_3\mathtt{u}^3 + \mathtt{a}_4\mathtt{u}^4 + \mathtt{a}_5\mathtt{u}^5 + \mathtt{a}_6\mathtt{u}^6 + \mathtt{a}_7\mathtt{u}^7 + \mathtt{a}_8\mathtt{u}^8.$$

Pro zajištění numerické robustnosti je polynom interně vyhodnocen pomocí Hornerova schématu.

Vstup

u	Analogový vstupní signál	double

Parametry

ai	Koeficient i-té mocniny vstupu, $i = 0, 1, \ldots, 8$	double

Výstup

y	Analogový výstupní signál	double

REC – Převrácená hodnota

Symbol bloku

Licence: STANDARD

Popis funkce

Blok REC počítá převrácenou hodnotu vstupního signálu u. Výstup je pak

$$y = \frac{1}{u}.$$

Pokud je vstup u = 0, nastaví se chybový výstup E = on a na výstupu y je náhradní hodnota yerr.

Vstup

u	Analogový vstupní signál		double

Výstupy

y	Analogový výstupní signál		double
E	Indikátor chyby – dělení nulou		bool

Parametr

yerr	Náhradní hodnota pro případ chyby	⊙1.0	double

REL – Relační operace dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok REL vyhodnocuje binární relaci u1 ∘ u2 z hodnot vstupů a podle výsledku relace „∘“ je nastavována hodnota výstupu Y na hodnotu on (relace platí) nebo off (relace neplatí). Kód binární operace je uveden v parametru irel popsaném níže.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double

Výstup

Y	Logický výstupní signál indikující platnost relace	bool

Parametr

irel	Typ relace	⊙1	long
	1 ..... rovnost (==)		
	2 ..... nerovnost (!=)		
	3 ..... menší než (<)		
	4 ..... větší než (>)		
	5 ..... menší nebo rovno (<=)		
	6 ..... větší nebo rovno (>=)		

RTOI – Konverze reálného čísla na celé číslo

Symbol bloku

Licence: STANDARD

Popis funkce

Blok RTOI převádí reálné číslo r na celé číslo i se znaménkem. Výsledná zaokrouhlená hodnota je určena vztahem:

$$\mathtt{i} := \begin{cases} -2147483648 & \text{pro} \quad \mathtt{r} \leq -2147483648.0 \\ \text{round}(\mathtt{r}) & \text{pro} \quad -2147483648.0 < \mathtt{r} \leq 2147483647.0\,, \\ 2147483647 & \text{pro} \quad \mathtt{r} > 2147483647.0 \end{cases}$$

kde round(r) je zaokrouhlení na nejbližší celé číslo. Čísla ve tvaru $n + 0.5$ (n celé) zaokrouhluje k číslu s vyšší absolutní hodnotou, např. $\text{round}(1.5) = 2$, $\text{round}(-2.5) = -3$. Poznamenejme, že čísla -2147483648 a 2147483647 odpovídají po řadě nejmenšímu a největšímu číslu se znaménkem zobrazitelným ve formátu s 32 bity (v jazyku C zapsanými v šestnáctkové soustavě jako 0x7FFFFFFF a 0x80000000).

Vstup

r	Analogový vstupní signál	double

Výstup

i	Zaokrouhlený a zkonvertovaný vstupní signál	long

SQR – Druhá mocnina

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SQR počítá druhou mocninu analogového vstupu u. Výstup je pak

$$y = u^2.$$

Vstup

u	Analogový vstupní signál	double

Výstup

y	Druhá mocnina vstupního signálu	double

SQRT_ – Druhá odmocnina

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SQRT počítá druhou odmocninu analogového vstupu u. Výstup je pak

$$y = \sqrt{u}.$$

V případě $u < 0$ je aktivován chybový výstup $E = on$ a výstup y je nastaven na hodnotu parametru yerr.

Vstup

u	Analogový vstupní signál		double

Výstupy

y	Odmocnina ze vstupní hodnoty		double
E	Příznak chyby off ... bez chyby on .... odmocňování záporného čísla		bool

Parametr

yerr	Náhradní hodnota při odmocňování záporného čísla	⊙1.0	double

SUB – Odčítání dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SUB počítá rozdíl dvou vstupních analogových signálů, výstup je dán vztahem

$$y = u1 - u2.$$

Pro sčítání a odečítání více signálů můžete použít blok ADDOCT.

Vstupy

u1	Analogový vstupní signál	double
u2	Analogový vstupní signál	double

Výstup

y	Rozdíl mezi vstupními signály	double

Kapitola 5

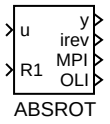
ANALOG – Zpracování analogových signálů

Obsah

ABSROT – Zpracování dat z absolutního snímače polohy

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok `ABSROT` se typicky používá v případech, kdy máme na nějaké hřídeli absolutní čidlo úhlu natočení v rozsahu např. 5° až 355° (popř. -175° až +175°), ale potřebujeme řídit pohyb o několik otáček. Blok předpokládá spojitý signál, takže při přechodu z například 355° na 5° předpokládá, že nastala další otáčka a úhel je ve skutečnosti 365°.

Protože v případě dlouhodobého otáčení jedním směrem by došlo k ztrátě přesnosti, je možné vstupem `R1` nastavit výstup `y` zpět do základního intervalu. Pokud je nastaven příznak `RESR` = `on`, dojde i k vynulování čítače otáček `irev`. V každém případě je však potřeba zároveň resetovat všechny související signály (např. signál `sp` připojeného regulátoru).

Výstup `MPI` (mid-point indicator) detekuje střední polohu čidla, což může být vhodný okamžik k resetování bloku. Výstup `OLI` (off-limits indicator) informuje o tom, že čidlo natočení je v tzv. mrtvém úhlu, kdy neposkytuje platná data.

Vstupy

`u`	Signál z absolutního snímače polohy	`double`
`R1`	Reset bloku (nastavení výstupu do základního intervalu)	`bool`

Výstupy

`y`	Vypočtená poloha	`double`
`irev`	Počet dokončených otáček (překročení hranic intervalu)	`long`
`MPI`	Indikátor středové polohy	`bool`
`OLI`	Indikátor polohy mimo rozsah senzoru	`bool`

Parametry

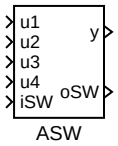
`lolim`	Dolní mez pro údaj z čidla	⊙`-3.141592654`	`double`
`hilim`	Horní mez pro údaj z čidla	⊙`3.141592654`	`double`
`tol`	Rozsah pro indikaci středové pozice	⊙`0.5`	`double`
`hys`	Hystereze pro indikaci středové pozice		`double`

`RESR`	Příznak pro resetování počítadla otáček
	`off` ... resetovat pouze vypočtenou polohu `y`
	`on` .... vynulovat i počítadlo otáček `irev`
		`bool`

ASW – Přepínač s automatickou volbou vstupu

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok ASW ukládá na výstup y hodnotu jednoho ze vstupů vstup u1, ..., u4 nebo jeden z parametrů p1, ..., p4. Pokud je na vstupu iSW jedna z hodnot $\{1, 2, 3, 4\}$, je na výstupu y hodnota příslušného vstupu. Pokud je na vstupu iSW jedna z hodnot $\{-1, -2, -3, -4\}$, je na výstupu y hodnota příslušného parametru (tj. pro $iSW = -1$ je na výstupu y hodnota p1, pro $iSW = 3$ je na výstupu y hodnota u3 atd.). Pokud je na vstupu iSW jiná hodnota (tj. $iSW = 0$ nebo $iSW < -4$ nebo $iSW > 4$), je na výstupu y hodnota toho ze vstupů u1, ..., u4 nebo parametrů p1, ..., p4, který se naposledy změnil. Pokud se změní více hodnot najednou, pak se použije hodnota podle následujícího pořadí p4, p3, p2, p1, u4, u3, u2, u1. Hodnota se považuje za změněnou, pokud se změnila o více než udává parametr delta od minulé detekce změny na příslušném vstupu resp. parametru (tj. změny se uvažují integrálně nikoliv diferenciálně od minulého vzorku). Ve všech režimech je na výstupu oSW číslo vstupu (resp. číslo parametru, pokud je hodnota záporná), který se použil pro generování výstupu y.

Blok ASW má dále tu speciální vlastnost, že nová hodnota y se kopíruje na parametry p1, ..., p4 (stejná vlastnost je i u bloků PARR, PARI, PARB). To má za následek, že všechny externí nástroje jako hodnotu všech těchto vstupů přečtou stejnou hodnotu y. To se hodí zejména v nadřízených systémech, které používají metodu nastav a sleduj (např. "potenciometr"v Iconics Genesis). Tato vlastnost není implementována ve verzi bloku ASW pro Simulink, protože tam není možnost používat externí programy pro čtení vstupu bloku.

POZOR! Pokud je blok zařazen ve schématu v nějaké smyčce, může se stát, že jeden ze vstupů u1, ..., u4 je o krok zpožděn, čímž se zdánlivě ignoruje priorita (výstup oSW pak zcela nepochopitelně signalizuje, že poslední změna nastala na tomto o krok zpožděném vstupu). Dalším důsledkem tohoto stavu je, že externí nástroje na zpožděném vstupu nezobrazují hodnotu y. Takovému chování lze zabránit vhodným použitím bloků LPBRK (např. za oba výstupy).

Vstupy

u1..u4	Analogové vstupní signály, ze kterých se vybírá ten aktivní	double
iSW	Volba aktivního signálu nebo parametru	long

Výstupy

`y`	Zvolený signál nebo parametr		`double`
`oSW`	Identifikátor použitého vstupu nebo parametru		`long`

Parametry

`delta`	Práh pro detekci změny	⊙`0.000001`	`double`
`p1..p4`	Parametry, ze kterých se vybírá ten aktivní		`double`

AVG – Filtr: vlečný průměr

Symbol bloku

Licence: STANDARD

Popis funkce

Blok AVG počítá vlečný průměr z posledních n vzorků, $n < N$ (N závisí na implementaci), podle vztahu

$$y_k = \frac{1}{n}(u_k + u_{k-1} + \cdots + u_{k-n+1}).$$

Není-li ještě k dispozici všech n vzorků (po startu algortimu), je výstup y nastaven na hodnotu průměru ze všech dostupných vzorků.

Vstup

u	Vstupní signál filtru	double

Výstup

y	Filtrovaný výstupní signál	double

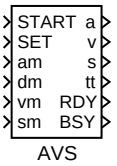
Parametr

n	Počet vzorků, ze kterých se provádí výpočet vlečného průměru ↓1 ↑10000000 ⊙10	long
nmax	Maximální hodnota parametru n (na tuto velikost se alokuje paměť) ↓1 ↑10000000 ⊙10	long

AVS – Rozběhová jednotka

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok `AVS` generuje časově optimální trajektorii pohybu z klidové polohy 0 do klidové polohy `sm` při omezení `am` na maximální zrychlení, `dm` na maximální zpomalení a `vm` na maximální rychlost. Při náběžné hraně vstupu `SET` (`off`→`on`) se provede inicializace (výpočet trajektorie) pro aktuální vstupy `am`, `dm`, `vm` a `sm`. Před první inicializací a po dobu inicializace má výstup `RDY` hodnotu `off`, potom `on`. Při náběžné hraně vstupu `START` (`off`→`on`) se spustí generování trajektorie pohybu na výstupech `a`, `v`, `s`, `tt`, přičemž tyto výstupy mají po řadě význam zrychlení, rychlosti, polohy a času. Po dobu generování trajektorie má výstup `BSY` hodnotu `on`, jinak `off`.

Vstupy

`START`	Spouštěcí signál (náběžná hrana `off`→`on`), start generování trajektorie	`bool`
`SET`	Inicializace/výpočet trajektorie podle aktuálních vstupů	`bool`
`am`	Maximální povolené zrychlení [m/s^2]	`double`
`dm`	Maximální povolené zpomalení [m/s^2]	`double`
`vm`	Maximální povolená rychlost [m/s]	`double`
`sm`	Žádaná konečná poloha [m] (počáteční poloha je 0)	`double`

Výstupy

`a`	Zrychlení [m/s^2]	`double`
`v`	Rychlost [m/s]	`double`
`s`	Poloha [m]	`double`
`tt`	Čas [s]	`double`
`RDY`	Příznak připravenosti (určuje, zda může být spuštěno generování trajektorie)	`bool`
`BSY`	Příznak probíhající operace (určuje, zda v daném okamžiku je či není generována trajektorie)	`bool`

BPF – Filtr: pásmová propusť

Symbol bloku

Licence: STANDARD

Popis funkce

Blok BPF realizuje přenos filtru druhého řádu ve tvaru

$$F_s = \frac{2\xi as}{a^2 s^2 + 2\xi as + 1},$$

kde a a ξ jsou po řadě parametry bloku fm a xi. Parametr fm určuje střed frekvenčního pásma propustnosti a xi je součinitel relativního tlumení.

Je-li ISSF = on, potom je stav filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu u.

Vstup

u	Vstupní signál filtru	double

Výstup

y	Filtrovaný výstupní signál	double

Parametry

fm	Střed pásma propustnosti [Hz]	⊙1.0	double
xi	Součinitel relativního tlumení (doporučená hodnota 0.5 až 1)	⊙0.707	double
ISSF	Ustálený stav při spuštění off ... nulový počáteční stav on .... ustálený počáteční stav		bool

CMP – Komparátor s hysterezí

Symbol bloku

Licence: STANDARD

Popis funkce

Blok CMP provádí komparaci vstupu u1 a u2 s hysterezí h podle následujících vztahů

$$\begin{aligned} Y_{-1} &= 0, \\ Y_k &= hyst(e_k),\ k = 0, 1, 2, \ldots \end{aligned}$$

kde

$$e_k = u1_k - u2_k$$

a

$$hyst(e_k) = \begin{cases} 0 & \text{pro} \quad e_k \leq -h \\ Y_{k-1} & \text{pro} \quad e_k \in (-h, h) \\ 1 & \text{pro} \quad e_k \geq h \quad (e_k > h \text{ pro } h = 0) \end{cases}$$

Indexované proměnné odpovídají hodnotám dané veličiny v cyklu, který udává index k, tzn. Y_{k-1} značí hodnotu výstupu v minulém kroku/cyklu. Hodnota Y_{-1} je použita pouze jednou při inicializaci bloku ($k = 0$), pokud je rozdíl vstupních signálů nepřekročí mez hystereze.

Vstupy

u1	První analogový vstup bloku		double
u2	Druhý analogový vstup bloku		double

Výstup

Y	Logický výstupní signál		bool

Parametr

hys	Hystereze	⊙0.5	double

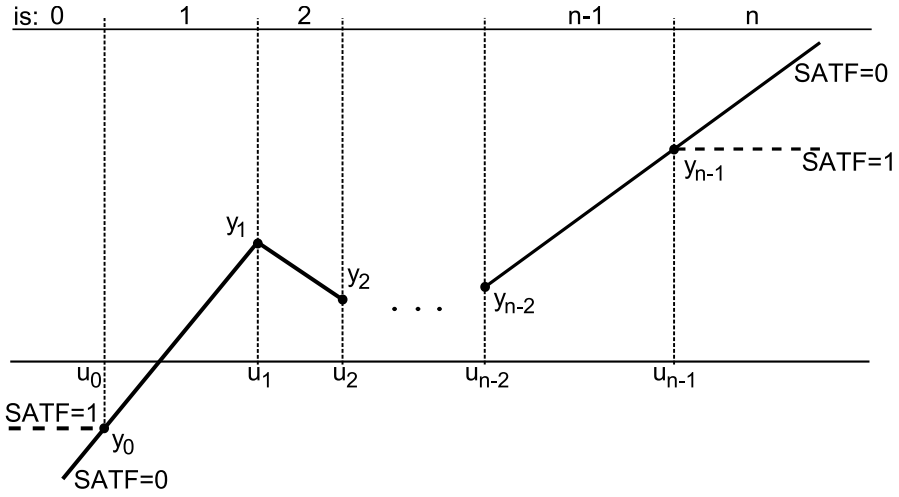
CNDR – Kompenzátor složité nelinearity

Symbol bloku

Licence: STANDARD

Popis funkce

Blok CNDR je určen pro kompenzaci složitých nelinearit pomocí po částech lineární transformace zobrazené na níže uvedeném obrázku.



V této souvislosti je důležité upozornit, že v případech $\mathtt{u} < u_0$ a $\mathtt{u} > u_{n-1}$ je výstup definován v závislosti na parametru SATF.

Vstup

u	Analogový vstupní signál	double

Výstupy

y	Analogový výstupní signál	double
is	Sektor nelinearity odpovídající vstupu u	long

Parametry

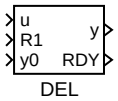
n	Počet uzlových bodů (u, y)	⊙2	long

`SATF`	Saturace v koncových uzlech `off` ... signál není omezen `on` .... saturační meze jsou aktivní	⊙`on`	`bool`
`up`	Vektor rostoucích u souřadnic uzlů	⊙`[-1 1]`	`double`
`yp`	Vektor y souřadnic uzlů	⊙`[-1 1]`	`double`

DEL – Dopravní zpoždění s inicializací

Symbol bloku

Licence: STANDARD



Popis funkce

Blok DEL realizuje zpoždění vstupního signálu u o n vzorků, tj.

$$y_k = u_{k-n}.$$

Jestliže po spuštění nebo restartu (R1: off→on→off) dosud není zapamatovaných n minulých vzorků (RDY = off), potom

$$y_k = y_0,$$

kde y_0 je inicializační vstup bloku.

Vstupy

u	Analogový vstupní signál	double
R1	Reset bloku	bool
y0	Počáteční hodnota výstupu	double

Výstupy

y	Zpožděný vstupní signál	double
RDY	Příznak připravenosti signalizující, že paměťový buffer je již naplněn vstupními vzorky	bool

Parametr

n	Zpoždení (počet vzorků) – příslušné časové zpoždění je $n \cdot T_S$, kde T_S je perioda spouštění bloku ↓0 ↑10000000 ⊙10	long
nmax	Maximální velikost parametru n (na tolik hodnot se alokuje pamět) ↓1 ↑10000000 ⊙10	long

DELM – Dopravní zpoždění

Symbol bloku

Licence: STANDARD

Popis funkce

Blok DELM realizuje časové zpoždění vstupního signálu u o čas, který vznikne zaokrouhlením parametru del na nejbližší celočíselný násobek periody T_S spouštění bloku. Po spuštění bloku do času del je výstup y = 0.

Vstup

u	Analogový vstupní signál	double

Výstup

y	Zpožděný vstupní signál	double

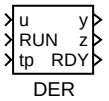
Parametr

del	Časové zpoždění [s]	⊙1.0	double
nmax	Délka vyrovnávací paměti pro dopravní zpoždění del (na tolik hodnot se alokuje pamět)	↓1 ↑10000000 ⊙10	long

DER – Derivace, filtrace a predikce z posledních n+1 vzorků

Symbol bloku

Licence: STANDARD



Popis funkce

Blok DER prokládá posledních n + 1 vzorků (n ≤ $N-1$, N závisí na implementaci) vstupního signálu u přímkou $y = at + b$ metodou nejmenších čtverců. Počátek časové osy je v každém kroku umístěn do aktuálního okamžiku vzorkování vstupu u. Ze získaných parametrů přímky a a b se počítají v případě RUN = on výstupy y a z podle vztahů:

Derivace:	y	=	a
Filtrace:	z	=	b, pro $t_p = 0$
Predikce:	z	=	$at_p + b$, pro $t_p > 0$
Postdikce:	z	=	$at_p + b$, pro $t_p < 0$

Je-li RUN = off nebo blok nemá k dispozici posledních n + 1 vzorků vstupního signálu (RDY = off), potom y = 0, z = u.

Vstupy

u	Analogový výstupní signál	double
RUN	Povolení běhu algoritmu off ... sledování (z=u) on .... filtrace (y – odhad derivace, z – odhad u v čase t_p)	bool
tp	Časový okamžik pro predikci/filtraci (tp = 0 je v aktuálním okamžiku vzorkování)	double

Výstupy

y	Odhad derivace vstupního signálu u	double
z	Predikovaný/filtrovaný výstupní signál	double
RDY	Příznak připravenosti (blok má k dispozici n + 1 vzorků)	bool

Parametr

n	Počet vzorků pro lineární interpolaci (je použito n + 1 vzorků); $1 \le n \le nmax$ ↓1 ↑10000000 ⊙10	long
nmax	Maximální hodnota parametru n (na tolik hodnot se alokuje paměť) ↓1 ↑10000000 ⊙10	long

EVAR – Vlečná střední hodnota a směrodatná odchylka

Symbol bloku

Licence: STANDARD

Popis funkce

Blok EVAR počítá střední hodnotu mu (μ) a směrodatnou odchylku si (σ) z posledních n vzorků vstupního signálu u podle vztahů

$$\mu_k = \frac{1}{\mathtt{n}} \sum_{i=0}^{n-1} \mathtt{u}_{k-i}$$

$$\sigma_k = \sqrt{\frac{1}{\mathtt{n}} \sum_{i=0}^{n-1} \mathtt{u}_{k-i}^2 - \mu_k^2}$$

kde k značí aktuální okamžik vzorkování.

Vstup

u	Analogový vstupní signál	double

Výstupy

mu	Střední hodnota vstupního signálu	double
si	Směrodatná odchylka vstupního signálu	double

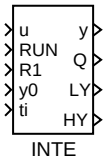
Parametr

n	Počet vzorků pro výpočet statistických ukazatelů ↓2 ↑10000000 ⊙100	long
nmax	Maximální velikost paramaetru n (na tolik hodnot se alokuje pamět) ↓1 ↑10000000 ⊙10	long

INTE – Řízený integrátor

Symbol bloku

Licence: STANDARD



Popis funkce

Blok INTE realizuje řízený integrátor s proměnnou integrační časovou konstantou ti a indikací dvou úrovní výstupu ymin a ymax. Je-li RUN = on a R1 = off, potom

$$y(t) = \frac{1}{T_i}\int_0^t u(\tau)d\tau + C,$$

kde hodnota C = y0. Je-li RUN = off a R1 = off, je výstup y zmrazen na jeho poslední hodnotu před sestupnou hranou vstupu RUN. Je-li R1 = on, potom je výstup y resetován na počáteční hodnotu y0. Integrace se provádí lichoběžníkovou metodou podle vztahu

$$y_k = y_{k-1} + \frac{T_S}{2T_i}(u_k + u_{k-1}),$$

kde T_S je perioda spouštění bloku.

Pro integraci je také možno použít blok SINT, jehož jednodušší struktura a funkčnost může být pro základní úlohy dostačující.

Vstupy

u	Analogový vstupní signál	double
RUN	Povolení běhu algoritmu off ... integrace je pozastavena ... integrace probíhá	bool
R1	Reset bloku, inicializace výstupu integrátoru na hodnotu y0	bool
y0	Počáteční hodnota výstupu	double
ti	Integrační časová konstanta	double

Výstupy

y	Výstup integrátoru	double
Q	Příznak probíhající integrace	bool
LY	Příznak dosažení spodní úrovně (y < ymin)	bool
HY	Příznak dosažení horní úrovně (y > ymax)	bool

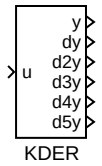
Parametry

`ymin`	Nastavení dolní úrovně	⊙`-1.0`	`double`
`ymax`	Nastavení horní úrovně	⊙`1.0`	`double`

KDER – Derivace a filtrace vstupního signálu

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok `KDER` je speciálně navržený Kalmanův filtr řádu `norder` tak, aby poskytoval odhady časových derivací řádu 0 až `norder` − 1 lokálně polynomiálních signálů, jejichž měření je zatíženo šumem. Blok je možné využít pro odhad derivací téměř libovolného vstupního signálu `u` $= u_0(t) + v(t)$ za předpokladu, že užitečný signál $u_0(t)$ a šum $v(t)$ mají odlišné frekvenční spektrum.

Blok se nastavuje pouze pomocí dvou parametrů `pbeta` a `norder`. Parametr `pbeta` je závislý na vzorkovací periodě T_S, frekvenčních vlastnostech vstupního signálu `u` a rovněž frekvenčních vlastnostech a úrovni obsaženého šumu. Platí pro něj přibližný vztah `pbeta` $\approx T_S\omega_0$. Pro správnou funkci bloku `KDER` by se frekvenční spektrum vstupního signálu `u` mělo nacházet hluboko pod zlomovou frekvencí filtru ω_0. Naopak frekvenční spektrum šumů by mělo být co možná nejdále od frekvence ω_0. Pro vyšší potlačení šumů je nutné volit nižší zlomovou frekvenci ω_0 a tím i parametr `pbeta`.

Druhý parametr `norder` je nutné volit převážně s ohledem na řád odhadovaných derivací. Ve většině případů by mělo stačit použít standardní hodnotu pro 3. řád. Vyšší hodnoty řádu derivačního filtru poskytují o něco lepší odhady derivací nepolynomiálních vstupních signálů za cenu delší doby vysledování (naladění) a vyšších výpočetních nároků.

Vstup

`u`	Vstupní signál filtru	`double`

Výstupy

`y`	Filtrovaný vstupní signál	`double`
`dy`	Odhad 1. derivace vstupního signálu	`double`
`d2y`	Odhad 2. derivace vstupního signálu	`double`
`d3y`	Odhad 3. derivace vstupního signálu	`double`
`d4y`	Odhad 4. derivace vstupního signálu	`double`
`d5y`	Odhad 5. derivace vstupního signálu	`double`

Parametry

`norder`	Řád derivačního filtru	↓2 ↑10 ⊙3	`long`
`pbeta`	Šířka pásma derivačního filtru	↓0.0 ⊙0.1	`double`

LPF – Filtr: dolní propusť

Symbol bloku

Licence: STANDARD

Popis funkce

Blok LPF realizuje přenos filtru druhého řádu ve tvaru

$$F_s = \frac{1}{a^2 s^2 + 2\xi a s + 1},$$

kde

$$a = \frac{\sqrt{\sqrt{2}\sqrt{2\xi^4 - 2\xi^2 + 1} - 2\xi^2 + 1}}{2\pi f_b}$$

a fb a ξ = xi jsou parametry bloku. Frekvence fb [Hz] určuje šířku pásma filtru a parametr xi součinitel relativního tlumení filtru. Doporučená hodnota pro Butterworthův filtr je xi = 0,71 a pro Besselův filtr xi = 0,87.

Je-li ISSF = on, potom je stav filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu u.

Vstup

u	Vstupní signál filtru	double

Výstup

y	Filtrovaný výstupní signál	double

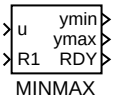
Parametry

fb	Šířka pásma [Hz] (filtr propouští frekvence v intervalu $\langle 0, \texttt{fb} \rangle$ – útlum na frekvenci fb je 3 dB, na $10 \cdot$ fb přibližně 40 dB); pro správnou funkci filtru musí platit $f_b < \frac{1}{10 T_S}$, kde T_S je perioda spouštění bloku ⊙1.0	double
xi	Součinitel relativního tlumení (doporučená hodnota 0,5 až 1) ⊙0.707	double
ISSF	Ustálený stav při spuštění off ... nulový počáteční stav on .... ustálený počáteční stav	bool

MINMAX – Vlečné minimum a maximum

Symbol bloku

Licence: STANDARD



Popis funkce

Blok MINMAX vyhodnocuje minimum a maximum z posledních n vzorků vstupního signálu u. Pokud není k dispozici n vzorků, je nastaveno RDY = off a minimum a maximum se hledá mezi dostupnými vzorky.

Vstupy

u	Analogový vstupní signál	double
R1	Reset bloku	bool

Výstupy

ymin	Nalezená minimální hodnota vstupního signálu	double
ymax	Nalezená maximální hodnota vstupního signálu	double
RDY	Příznak připravenosti (buffer je naplněn)	bool

Parametr

n	Počet prvků pro výpočet minima a maxima (délka bufferu) ↓1 ↑10000000 ⊙100	long

NSCL – Kompenzátor jednoduché nelinearity

Symbol bloku

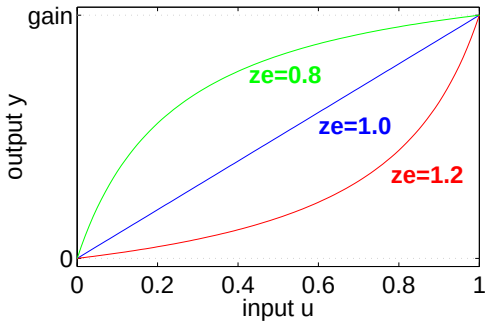
Licence: STANDARD

Popis funkce

Blok NSCL kompenzuje v praxi často se vyskytující nelinearity (např. nelinearitu servoventilu) pomocí vztahu

$$\mathtt{y} = \mathtt{gain}\frac{\mathtt{u}}{\mathtt{ze} + (1 - \mathtt{ze})\mathtt{u}},$$

kde gain a ze jsou parametry bloku. Volbou ze v intervalu $(0, 1)$ obdržíme konkávní transformaci, zatímco je-li $\mathtt{ze} > 1$, dostaneme transformaci konvexní.



Vstup

u	Analogový vstupní signál	double

Výstup

y	Analogový výstupní signál	double

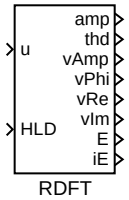
Parametry

gain	Zesílení	⊙1.0	double
ze	Tvarovací parametr	⊙1.0	double

RDFT – Vlečná diskrétní Fourierova transformace

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok RDFT počítá diskrétní Fourierovu transformaci vstupního signálu pro základní frekvenci freq (a případně několik dalších) z posledních m vzorků vstupního signálu u, kde $m = nper/freq/T_S$, tj. z časového okna o délce odpovídající nper periodám základní frekvence.

Pokud je nharm > 0, je počet vyčíslovaných vyšších harmonických frekvencí dán právě tímto parametrem. Pokud je nharm $= 0$, další vyčíslované frekvence určuje vektorový parametr freq2.

Pro každou frekvenci se vyčísluje amplituda (výstup vAmp), fáze (výstup vPhi), reálná/kosinová složka (výstup vRe) a imaginární/sinová složka (výstup vIm). Výstupy bloku jsou vektorové, takže obsahují příslušné hodnoty pro všechny analyzované frekvence. Hodnoty pro jednotlivé frekvence se získají pomocí bloků VTOR.

Vstupy

u	Analogový vstupní signál	double
HLD	Pozastavení funkce bloku	bool

Výstupy

amp	Amplituda základní frekvence (určená parametrem freq)	double
thd	Celkové harmonické zkreslení, podíl základní a vyšších harmonických (jen pokud nharm ≥ 1)	double
vAmp	Vektor amplitud pro zadané frekvence	reference
vPhi	Vektor fázových posunů pro zadané frekvence	reference
vRe	Vektor reálných částí pro zadané frekvence	reference
vIm	Vektor imaginárních částí pro zadané frekvence	reference
E	Příznak chyby	bool
iE	Kód chyby	error

i ..... obecná chyba systému REX

Parametry

`freq`	Základní frekvence	↓`0.000000001` ↑`1000000000.0` ⊙`1.0`	`double`
`nper`	Počet period signálu na kterých provádět výpočet	↓`1` ↑`10000` ⊙`10`	`long`
`nharm`	Počet monitorovaných harmonických frekvencí	↓`0` ↑`16` ⊙`3`	`long`
`ifrunit`	Jednotky pro frekvenci	↓`1` ↑`2` ⊙`1`	`long`
	`1` ..... Hz		
	`2` ..... rad/s		
`iphunit`	Jednotky pro fázový posun	↓`0` ↑`2` ⊙`1`	`long`
	`1` ..... stupně		
	`2` ..... radiány		
`freq2`	Vektor uživatelem definovaných frekvencí	⊙`[2.0 3.0 4.0]`	`double`

RLIM – Omezovač strmosti

Symbol bloku

Licence: STANDARD

Popis funkce

Blok RLIM kopíruje vstup u na výstup y, avšak maximální dovolená rychlost změny signálu je omezena. Omezení jsou definována časovými konstantami tp a tn podle následujících vztahů:

maximální nárůst za sekundu: $1/\texttt{tp}$
maximální pokles za sekundu: $-1/\texttt{tn}$

Vstup

u	Vstupní signál filtru	double

Výstup

y	Filtrovaný výstupní signál	double

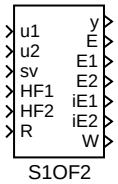
Parametry

tp	Časová konstanta určující maximální růst	⊙2.0	double
tn	Časová konstanta určující maximální pokles (pozor, tn > 0)	⊙2.0	double

S1OF2 – Výběr jednoho ze dvou analogových vstupů

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok S1OF2 určuje odděleně platnost signálů u1 a u2 stejným způsobem jako blok SAI. Je-li signál u1 (nebo u2) neplatný, potom má výstup E1 (nebo E2) hodnotu on a kód chyby je na výstupu iE1 (nebo iE2). Dále se v bloku S1OF2 vyhodnocuje odchylka vstupu u1 a u2 a nastavuje vnitřní příznak D, který má hodnotu on tehdy, jestliže posledních nd vzorků odchylek |u1 − u2| splňuje nerovnost

$$|\texttt{u1} - \texttt{u2}| > \texttt{pdev}\frac{\texttt{vmax} - \texttt{vmin}}{100},$$

kde vmin a vmax jsou po řadě dolní a horní mez vstupů u1 a u2 a pdev je dovolená procentuální odchylka signálů u1 a u2 z celkového rozsahu. Na základě zjištěné platnosti vstupů (příznaky E1 a E2) a příznaku odchýlení D se určuje zabezpečený výstup y následujícím způsobem:

(i) Je-li E1 = off **a** E2 = off **a** D = off , pak výstup y je podle parametru mode dán vztahem:

$$\texttt{y} = \begin{cases} \frac{\texttt{u1}+\texttt{u2}}{2}, & \text{pro } \texttt{mode} = 1, \\ \min(\texttt{u1}, \texttt{u2}), & \text{pro } \texttt{mode} = 2, \\ \max(\texttt{u1}, \texttt{u2}), & \text{pro } \texttt{mode} = 3, \end{cases}$$

a výstup ER má hodnotu off, nebyl-li již dříve nastaven na on.

(ii) Je-li E1 = off **a** E2 = off **a** D = on , potom y = sv a ER = on.

(iii) Je-li E1 = on **a** E2 = off (E1 = off **a** E2 = on) , potom y = u2 (y = u1) a výstup ER = off nebyl-li již dříve nastaven na on.

(iv) Je-li E1 = on **a** E2 = on , potom y = sv a ER = on.

Vstup R resetuje vnitřní příznaky chyb F1–F4 (viz. blok SAI) a příznak D. Je-li trvale R = on, potom v případě rozpoznání neplatnosti vstupu u1 (u2) je výstup E1 (E2) nahozen pouze po dobu jednoho cyklu. Naproti tomu při R = off je E1 = on (E2 = on) až do následného resetování (náběžná hrana R = off → on). Pro výstup ER platí obdobné

pravidlo. Je-li trvale `R` = `on`, pak v případě náběžné hrany vnitřního příznaku `D` (`off`→`on`) je výstup `ER` nahozen pouze po dobu jednoho cyklu. Při `R` = `off` je nastaveno `ER` = `on` až do následného resetování. Výstup `W` má hodnotu `on` pouze v případech (iii) a (iv), tzn. pokud alespoň jeden z výstupů `E1` a `E2` má hodnotu `on`, tedy pokud je alespoň jeden ze vstupních signálů označen za neplatný.

Vstupy

`u1`	První analogový vstup bloku	`double`
`u2`	Druhý analogový vstup bloku	`double`
`sv`	Náhradní hodnota pro případ neplatných vstupů `u1` a `u2`	`double`
`HF1`	Příznak hardwarové chyby vstupu `u1` `off` ... vstupní modul signálu pracuje normálně `on` .... došlo k hardwarové chybě vstupního modulu	`bool`
`HF2`	Příznak hardwarové chyby vstupu `u2` `off` ... vstupní modul signálu pracuje normálně `on` .... došlo k hardwarové chybě vstupního modulu	`bool`
`R`	Vynulování vnitřních chybových příznaků pro signály `u1` a `u2`	`bool`

Výstupy

`y`	Analogový výstupní signál	`double`
`E`	Indikátor neplatnosti výstupního signálu `y` `off` ... signál je platný `on` .... signál není platný	`bool`
`E1`	Indikátor neplatnosti vstupu `u1` `off` ... signál je platný `on` .... signál není platný	`bool`
`E2`	Indikátor neplatnosti vstupu `u2` `off` ... signál je platný `on` .... signál není platný	`bool`
`iE1`	Důvod neplatnosti vstupu `u1` `0` ..... signál je platný `1` ..... signál mimo rozsah `2` ..... signál se mění příliš málo `3` ..... signál se mění jen málo a je mimo rozsah `4` ..... signál se mění příliš mnoho `5` ..... signál se mění příliš mnoho a je mimo rozsah `6` ..... signál se mění příliš málo a příliš mnoho `7` ..... signál se mění příliš málo a příliš mnoho a je mimo rozsah `8` ..... hardwarová chyba	`long`
`iE2`	Důvod neplatnosti vstupu `u2`, viz výstup `iE1`	`long`
`W`	Varování (neplatný vstupní signál) `off` ... oba vstupní signály jsou platné `on` .... alespoň jeden vstupní signál je neplatný	`bool`

Parametry

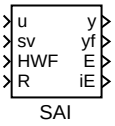
`nb`	Počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálů `u1` a `u2` ⊙10	`long`

`nc`	Počet vzorků pro testování neměnnosti (viz blok SAI, podmínka `F2`) ⊙`10`	`long`
`nbits`	Počet bitů A/D převodníku vstupního modulu (zdroje signálů `u1` a `u2`) ⊙`12`	`long`
`nr`	Počet vzorků pro testování variability (viz blok SAI, podmínka `F3`) ⊙`10`	`long`
`prate`	Maximální předpokládaná procentuální změna vstupu `u1` (`u2`) z celkového rozsahu `vmax` − `vmin` za `nr` vzorků vstupu `u1` (`u2`), viz blok SAI ⊙`10.0`	`double`
`nv`	Počet vzorků pro testování překročení rozsahu (viz blok SAI, podmínka `F4`) ⊙`1`	`long`
`vmin`	Spodní omezení na vstupní signál ⊙`-1.0`	`double`
`vmax`	Horní omezení na vstupní signál ⊙`1.0`	`double`
`nd`	Počet vzorků pro vyhodnocování odchýlení (vnitřní příznak `D`, pro `nd` = 0 je vždy `D` = `off`) ⊙`5`	`long`
`pdev`	Maximální povolená procentuální odchylka signálů `u1` a `u2` z celkového rozsahu `vmax` − `vmin` ⊙`10.0`	`double`
`mode`	Způsob výpočtu výstupu při platnosti obou vstupů (`E1` = `off`, `E2` = `off` a `D` = `off`) ⊙`1` `1` ..... průměr, $\mathtt{y} = \frac{\mathtt{u1}+\mathtt{u2}}{2}$ `2` ..... minimum, $\mathtt{y} = \min(\mathtt{u1}, \mathtt{u2})$ `3` ..... maximum, $\mathtt{y} = \max(\mathtt{u1}, \mathtt{u2})$	`long`

SAI – Zabezpečený analogový vstup

Symbol bloku

Licence: ADVANCED



Popis funkce

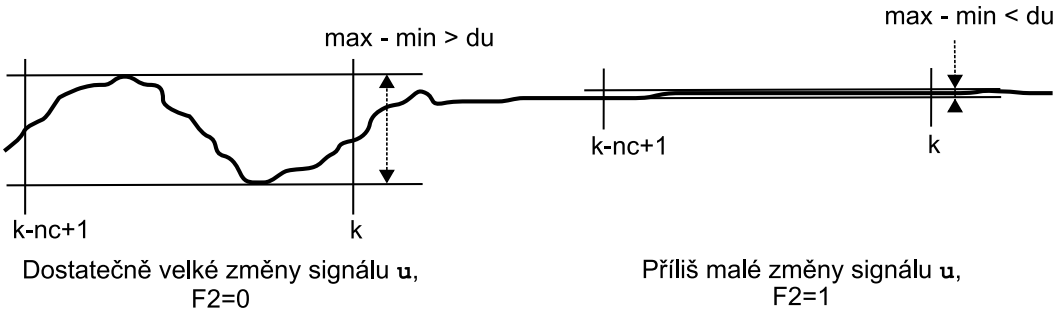
Blok SAI testuje vstupní signál u s cílem rozpoznání jeho platnosti. Vstupní signál u se považuje za neplatný (výstup E = on) v následujících případech:

F1: **Hardwarová chyba.** Vstupní signál HWF = on.

F2: **Vstupní signál u se mění příliš málo.** Posledních nc vzorků vstupu u leží v intervalu délky du,

$$\texttt{du} = \begin{cases} \frac{\texttt{vmax}-\texttt{vmin}}{2^{\texttt{nbits}}}, & \text{pro } \texttt{nbits} \in \{8,9,...,16\} \\ 0, & \text{pro } \texttt{nbits} \notin \{8,9,...,16\}, \end{cases}$$

kde vmin a vmax jsou po řadě dolní a horní mez vstupu u a nbits je počet bitů příslušného A/D převodníku. Situace, kdy je splněna podmínka příliš malé změny u, je zobrazena na následujícím obrázku:

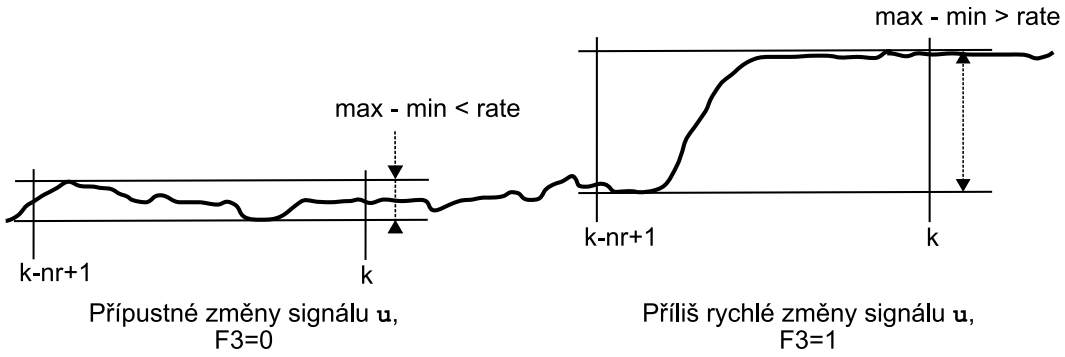


Jestliže je nastaveno nc = 0, potom podmínka F2 není splněna nikdy.

F3: **Vstupní signál u se mění příliš rychle.** Posledních nr vzorků vstupu u filtrovaného filtrem SPIKE neleží v intervalu délky rate,

$$\texttt{rate} = \texttt{prate}\frac{\texttt{vmax} - \texttt{vmin}}{100},$$

kde `prate` vyjadřuje dovolenou procentuální změnu signálu `u` z celkového rozsahu během `nr` vzorků. V bloku je zařazený SPIKE filtr s pevnými parametry `mingap` = (`vmax` − `vmin`)/100 a `q` = 2 odstraňující ze signálu úzké špičky, které by mohly způsobovat nežádoucí splnění této podmínky (blíže viz popis bloku SPIKE). Situace, kdy je splněna podmínka příliš rychlé změny, je zobrazena na následujícím obrázku:



Jestliže je nastaveno `nr` = 0, potom podmínka `F3` není splněna nikdy.

`F4`: **Vstupní signál `u` je mimo rozsah.** Posledních `nv` vzorků vstupu `u` leží mimo přípustný interval ⟨`vmin`, `vmax`⟩. Jestliže je nastaveno `nv` = 0, potom podmínka `F4` není splněna nikdy.

Je-li signál `u` platný, potom je beze změny kopírován na výstup `y`. V opačném případě je do výstupu `y` dosazena náhradní hodnota ze vstupu `sv`. V tomto případě má výstup `E` hodnotu `on` a výstup `iE` udává kód rozpoznané chyby vstupu `u` (viz tabulka níže). Vstup `R` resetuje vnitřní příznaky chyb `F1`–`F4`. Je-li trvale `R` = `on`, potom v případě rozpoznání neplatnosti vstupu `u` je výstup `E` nahozen pouze po dobu jednoho cyklu. Naproti tomu při `R` = `off` je `E` = `on` až do následného resetování (náběžná hrana `R`: `off`→`on`).

Tabulka kódů chyb `iE` podle vnitřních příznaků `F1`–`F4`:

F1	F2	F3	F4	iE
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	*	*	*	8

Parametr `nb` určuje počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálu `u`. Doporučuje se volit `nb` ≥ 5 z důvodu odeznění počátečních podmínek SPIKE filtru.

Vstupy

`u`	Analogový vstupní signál	`double`
`sv`	Náhradní hodnota při neplatném signálu `u`	`double`
`HWF`	Příznak hardwarové chyby `off` ... vstupní modul signálu pracuje normálně `on` .... došlo k hardwarové chybě vstupního modulu	`bool`
`R`	Vynulování vnitřních příznaků chyb `F1`–`F4`	`bool`

Výstupy

`y`	Analogový výstupní signál	`double`
`yf`	Výstupní signál `y` filtrovaný SPIKE algoritmem	`double`
`E`	Indikátor neplatnosti výstupního signálu `off` ... výstup je platný `yf` = `sv` `on` .... výstup není platný, `y` =	`bool`
`iE`	Důvod neplatnosti signálu `0` ..... signál je platný `1` ..... signál mimo rozsah `2` ..... signál se mění příliš málo `3` ..... signál se mění jen málo a je mimo rozsah `4` ..... signál se mění příliš mnoho `5` ..... signál se mění příliš mnoho a je mimo rozsah `6` ..... signál se mění příliš málo a příliš mnoho `7` ..... signál se mění příliš málo a příliš mnoho a je mimo rozsah `8` ..... hardwarová chyba	`long`

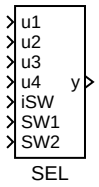
Parametry

`nb`	Počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálu `u`	⊙`10`	`long`
`nc`	Počet vzorků pro testování neměnnosti (podmínka `F2`)	⊙`10`	`long`
`nbits`	Počet bitů A/D převodníku vstupního modulu	⊙`12`	`long`
`nr`	Počet vzorků pro testování variability (podmínka `F3`)	⊙`10`	`long`
`prate`	Maximální předpokládaná procentuální změna vstupu `u` z celkového rozsahu (`vmax` − `vmin`) za `nr` vzorků vstupu `u`	⊙`10.0`	`double`
`nv`	Počet vzorků pro testování překročení rozsahu (podmínka `F4`)	⊙`1`	`long`
`vmin`	Spodní omezení na vstupní signál `u`	⊙`-1.0`	`double`
`vmax`	Horní omezení na vstupní signál `u`	⊙`1.0`	`double`

SEL – Selektor analogového signálu

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SEL nebude dále podporován, nahraďte jej blokem SELQUAD, SELOCT nebo SELHEXD. Pozor na změnu významu vstupních signálů SWn.

Blok SEL realizuje výběr zvoleného signálu ze čtyř vstupních signálů u1, u2, u3 a u4 a kopíruje ho na výstup y. Výběr se provádí podle vstupu iSW, jestliže je BINF = off nebo podle binárních vstupů SW1 a SW2 (BINF = on) dle následující tabulky:

iSW	SW1	SW2	y
0	off	off	u1
1	off	on	u2
2	on	off	u3
3	on	on	u4

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double
u3	Třetí analogový vstup bloku	double
u4	Čtvrtý analogový vstup bloku	double
iSW	Selektor aktivního signálu, který je aktivní, je-li BINF = off	long
SW1	Binární vstup pro výběr, který je aktivní, je-li BINF = on	bool
SW2	Binární vstup pro výběr, který je aktivní, je-li BINF = on	bool

Výstup

y	Zvolený signál	double

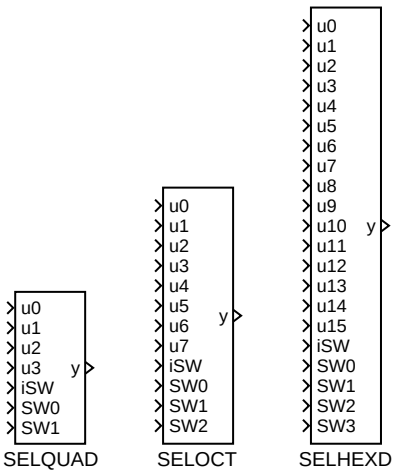
Parametr

`BINF`	Výběr pomocí binárních vstupů `off` ... zakázáno (výběr přes `iSW`) `on` .... povoleno (výběr přes `SW1` a `SW2`)	`bool`

SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky `SELQUAD`, `SELOCT` a `SELHEX` realizují výběr zvoleného signálu ze vstupních signálů a kopírují ho na výstup `y`. Jediný rozdíl mezi bloky je v počtu vstupních signálů. Výběr aktivního vstupu `u0`...`u15` se provádí podle vstupu `iSW`, jestliže je `BINF` = `off` nebo podle binárních vstupů `SW0`...`SW3` (`BINF` = `on`) dle následující tabulky:

iSW	SW0	SW1	SW2	SW3	y
0	off	off	off	off	u0
1	on	off	off	off	u1
2	off	on	off	off	u2
3	on	on	off	off	u3
4	off	off	on	off	u4
5	on	off	on	off	u5
6	off	on	on	off	u6
7	on	on	on	off	u7
8	off	off	off	on	u8
9	on	off	off	on	u9
10	off	on	off	on	u10
11	on	on	off	on	u11
12	off	off	on	on	u12
13	on	off	on	on	u13
14	off	on	on	on	u14
15	on	on	on	on	u15

Vstupy

`u0..15`	Analogové vstupní signály	`double`
`iSW`	Selektor aktivního signálu	`long`
`SW0..3`	Binární vstupy pro výběr	`bool`

Výstup

`y`	Zvolený vstupní signál	`double`

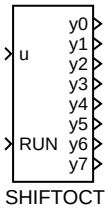
Parametr

`BINF`	Výběr pomocí binárních vstupů `off` ... zakázáno (výběr přes `iSW`) `on` .... povoleno (výběr přes `SW`n)	`bool`

SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot

Symbol bloku

Licence: STANDARD



Popis funkce

Blok realizuje funkci posuvného registru s osmi výstupy pro libovolný typ signálů. Je-li aktivní vstup `RUN`, je v každém tiku algoritmu provedeno následující přiřazení:

$$\begin{aligned} \mathtt{y}_i &= \mathtt{y}_{i-1},\ i = 1..7, \\ \mathtt{y0} &= \mathtt{u}, \end{aligned}$$

tedy hodnota na každém z výstupů `y0` až `y6` je posunuta na výstup v pořadí následující, a hodnota vstupu `u` je přenesena na výstup `y0`.

Blok pracuje s libovolným datovým typem signálu přivedeného na vstup `u`. Požadovaný datový typ je třeba nastavit parametrem `vtype`. Výstupy `y0` až `y7` jsou pak shodného datového typu.

Pokud potřebujete posun dat v registru provádět na základě triggeru, vložte před vstup `RUN` blok EDGE_.

Vstupy

`u`	Vstupní hodnota registru	`unknown`
`RUN`	Povoluje posun výstupů	`bool`

Výstupy

`y0`	První výstup bloku	`unknown`
`y1`	Druhý výstup bloku	`unknown`
`y2`	Třetí výstup bloku	`unknown`
`y3`	Čtvrtý výstup bloku	`unknown`
`y4`	Pátý výstup bloku	`unknown`
`y5`	Šestý výstup bloku	`unknown`
`y6`	Sedmý výstup bloku	`unknown`

y7	Osmý výstup bloku		unknown

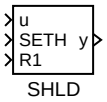
Parametry

vtype	Typ výstupů 1 ..... Bool 2 ..... Byte 3 ..... Short 4 ..... Long 5 ..... Word 6 ..... DWord 7 ..... Float 8 ..... Double -- .... 10 .... Large	⊙8	long

SHLD – Vzorkovač (sample and hold)

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SHLD je určen pro podržení hodnoty vstupního signálu u, přičemž jeho funkce je dána parametrem mode.

V případě *vynuceného vzorkování* se nastaví výstup y na hodnotu vstupu u v okamžiku náběžné hrany (off→on) řídicího vstupu SETH a zůstává konstantní až do příchodu nové náběžné hrany.

Pokud je zvoleno *držení předchozí hodnoty*, na výstup y se nastaví poslední hodnota vstupu u před příchodem vzestupné hrany na vstupu SETH. Tato hodnota je držena po celou dobu, kdy platí SETH = on. Pokud je na vstupu SETH = off, je vstup u jednoduše kopírován na výstup y.

Při režimu *držení aktuální hodnoty* se na výstup y nastaví hodnota vstupu u v okamžiku náběžné hrany. Tato hodnota je držena po celou dobu, kdy platí SETH = on. Pokud je na vstupu SETH = off, je vstup u jednoduše kopírován na výstup y.

Vstup R1 slouží k resetování bloku, inicializuje výstup y na hodnotu y0 a má prioritu před vstupem SETH.

Vstupy

u	Analogový vstupní signál		double
SETH	Vstup pro nastavení a podržení výstupní hodnoty		bool
R1	Reset bloku, R1 = on → y = y0		bool

Výstup

y	Analogový výstupní signál		double

Parametr

y0	Počáteční hodnota výstupu y		double
mode	Režim vzorkování 1 ..... Vynucené vzorkování 2 ..... Držení předchozí hodnoty 3 ..... Držení aktuální hodnoty	⊙3	long

SINT – Jednoduchý integrátor

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SINT realizuje diskrétní integrátor popsaný diferenční rovnicí

$$y_k = y_{k-1} + \frac{T_S}{2T_i}(u_k + u_{k-1}),$$

kde T_S je perioda spouštění bloku a ti je integrační konstanta. Je-li y_k mimo saturační meze ymin, ymax, potom je výstup i stav integrátoru příslušně omezen.

Pro složitější případy je možné použít blok INTE, který disponuje rozšířenou funkcionalitou.

Vstup

u	Analogový vstupní signál		double

Výstup

y	Analogový výstupní signál		double

Parametry

ti	Integrační časová konstanta	⊙1.0	double
y0	Počáteční hodnota výstupu		double
ymax	Horní saturační mez	⊙1.0	double
ymin	Dolní saturační mez	⊙-1.0	double

SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů

Symbol bloku

Licence: ADVANCED

Popis funkce

Blok SPIKE realizuje nelineární filtr odstraňující ze vstupního signálu u izolované úzké špičky (pulzy). Jeden krok SPIKE filtru provádí následující transformaci (u, y) → y:

```
delta := y - u;
if abs(delta) < gap
  then
     begin
        y := u;
        gap := gap/q;
        ifgap < mingap then gap:= mingap;
     end
  else
     begin
       if delta < 0
          then y := y + gap
          else y := y - gap;
       gap := gap * q;
     end
```

kde mingap a q jsou parametry bloku. Zvolíme-li parametr mingap dostatečně velký, potom signál prochází filtrem beze změny. Zmenšováním tohoto parametru je možné docílit stav, kdy dojde k odfiltrování nežádoucích špiček, ale jinak zůstává signál nezkreslen. Doporučená volba je 1 % z celkového rozsahu vstupního signálu u. Parametr q určuje rychlost adaptace tolerančního okénka filtru.

Vstup

u	Vstupní signál filtru	double

Výstup

y	Filtrovaný výstupní signál	double

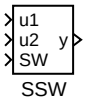
Parametry

`mingap`	Minimální velikost tolerančního okénka	⊙`0.01`	`double`
`q`	Rychlost adaptace tolerančního okénka filtru	↓`1.0` ⊙`2.0`	`double`

SSW – Jednoduchý přepínač

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SSW provádí výběr jednoho ze dvou vstupních signálů u1, u2 podle logického vstupu SW a získanou hodnotu kopíruje na výstup y. Je-li SW = off (SW = on), potom je vybraný vstup u1 (u2).

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double
SW	Přepínací signál off ... je zvolen první vstupní signál, y = u1 on .... je zvolen druhý vstupní signál, y = u2	bool

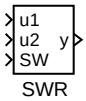
Výstup

y	Analogový výstupní signál	double

SWR – Přepínač s rampovou funkcí

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SWR provádí výběr jednoho ze dvou vstupních signálů u1, u2 podle logického vstupu SW a podle něho nastavuje výstup y. Je-li SW = off (SW = on), potom je vybraný vstup u1 (u2). Při přepnutí vstupu se výstup nezmění okamžitě, ale vysleduje vybraný vstup po rampě s definovanou strmostí. Tato strmost může být různá pro oba vstupy u1, u2 a je určena po řadě časovými konstantami t1 a t2. Po vysledování vstupu se funkce omezení strmosti vypne až do následujícího přepnutí.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double
SW	Přepínací signál off ... je zvolen vstupní signál u1 on .... je zvolen vstupní signál u2	bool

Výstup

y	Analogový výstupní signál	double

Parametry

t1	Časová konstanta omezovače strmosti, nabíhání na hodnotu u1 ⊙1.0	double
t2	Časová konstanta omezovače strmosti, nabíhání na hodnotu u2 ⊙1.0	double
y0	Počáteční hodnota výstupu, ze které se vysledovává před prvním přepnutím	double

VDEL – Dopravní zpoždění s proměnnou délkou

Symbol bloku

Licence: STANDARD

Popis funkce

Blok VDEL realizuje proměnné časové zpoždění vstupního signálu u o čas přivedený na vstup d. Přesněji, výstupní signál y je zpožděn o čas, který vznikne zaokrouhlením vstupu d na nejbližší celočíselný násobek n periody T_S spouštění bloku. Jestliže po spuštění bloku není dosud k dispozici n posledních vzorků, potom je výstup y nastaven na inicializační hodnotu y0.

Vstupy

u	Analogový vstupní signál	double
d	Časové zpoždění [s]	double

Výstup

y	Zpožděný vstupní signál	double

Parametr

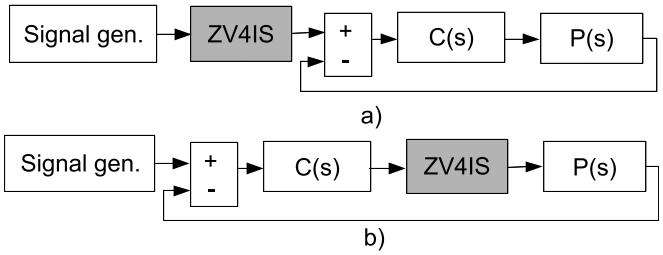
y0	Počáteční/náhradní hodnota výstupu	double
nmax	Délka vyrovnávací paměti pro dopravní zpoždění d (na tolik hodnot se alokuje pamět) ↓1 ↑10000000 ⊙10	long

ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací

Symbol bloku Licence: ADVANCED

Popis funkce

Blok ZV4IS realizuje funkci frekvenčního filtru typu pásmová zádrž. Hlavní oblastí použití je řízení pohybu mechanických systémů s pružnými částmi, kde vlivem nedostatečné tuhosti konstrukce hrozí nebezpečí vzniku reziduálních vibrací. Ty se projevují jako mechanické chvění vybuzené v důsledku momentu nebo síly, kterou aktuátory působí na pracovní mechanismus stroje. Tyto vibrace mohou mít negativní vliv na přesnost regulace, vedou ke zvýšenému opotřebení mechanických částí stroje a v krajním případě mohou způsobit nestabilitu regulačních smyček. Obecně lze tvarovací filtr využít v libovolné aplikaci pro řízení kmitavých systémů nebo pro potlačení konkrétní frekvence ve spojitém signálu.



Tvarovací filtr je možné použít dvěma způsoby. Zapojením v *otevřené smyčce* (viz obrázek výše nahoře) je modifikován referenční signál přicházející od obsluhy nebo od generátoru trajektorie z vyšší úrovně řízení. Výhodou tohoto uspořádání je, že dynamika vlastního filtru neovlivňuje chování podřízené zpětnovazební smyčky. Podmínkou správné funkce je korektní nastavení regulátoru $C(s)$ ve zpětné vazbě, který musí pracovat v lineárním režimu. V opačném případě může dojít ke zkreslení frekvenčního spektra akční veličiny a tím k vybuzení nežádoucích kmitů na rezonančních frekvencích stroje (ve schématu $P(s)$). Záporem přímovazebního zapojení je absence tlumení při působení vnějších poruch. Tuto nevýhodu odstraňuje *zapojení v uzavřené smyčce* (na obrázku dole), kdy filtr je umístěn za zpětnovazební regulátor a tvaruje přímo akční veličinu. V této variantě jsou kompenzovány vibrace vybuzené jak v důsledku změny referenčního signálu tak vlivem působících vnějších poruch. Nevýhodou tohoto zapojení je zanesení dynamického zpoždění do zpětné vazby a tím nutnost přeladit vnitřní regulátor.

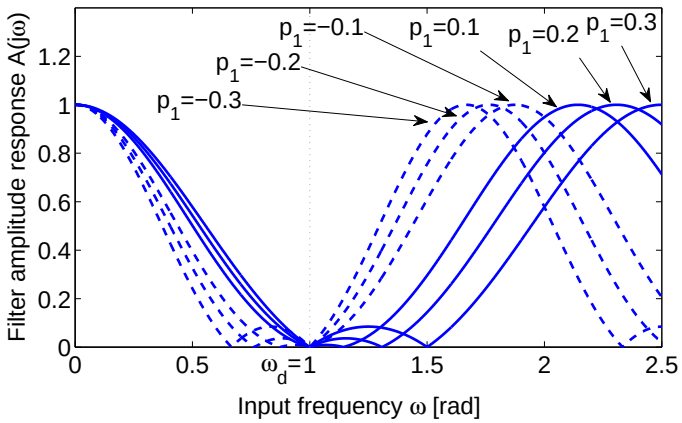
Vlastní algoritmus filtrace lze popsat v časové oblasti vztahem

$$\mathtt{y}(t) = A_1\mathtt{u}(t-t_1) + A_2\mathtt{u}(t-t_2) + A_3\mathtt{u}(t-t_3) + A_4\mathtt{u}(t-t_4)$$

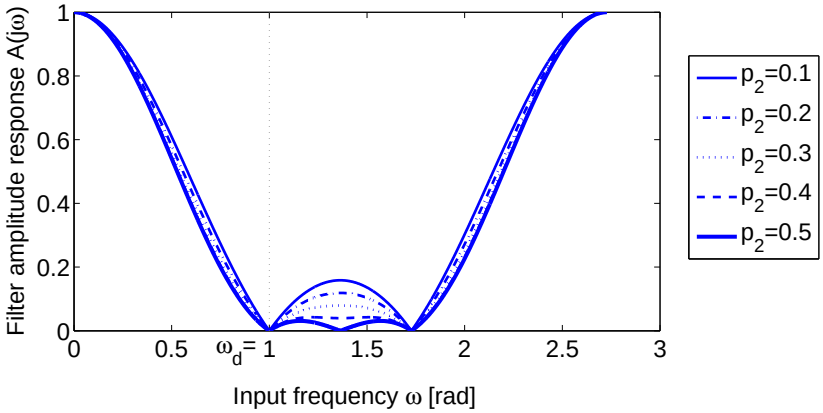
Filtr má tedy strukturu sumy vážených dopravních zpoždění kde zesílení $A_1..A_4$ a hodnoty zpoždění $t_1..t_4$ závisí na volbě typu filtru, frekvenci a tlumení kmitavého módu systému. Výhodou uvedené struktury oproti klasickým dynamickým notch filtrům užívaným v regulační technice je konečná impulzní odezva, která je důležitá zejména v aplikacích řízení pohybu, zaručená stabilita a monotónní přechodová charakteristika filtru a obecně menší zpoždění zaváděné do cesty signálu.

Pro správnou funkci filtru je třeba zadat vlastní frekvenci `omega` a tlumení `xi` kmitavého módu, který má být potlačen. Parametr `ipar` pak udává typ tvarovacího filtru, pro `ipar` = `1` je použit jeden z deseti základních filtrů, které se volí parametrem `istype`. Jednotlivé typy se liší tvarem frekvenční charakteristiky a šířkou nepropustného pásma. Při přesné znalosti `omega` a `xi` je vhodný filtr typu ZV nebo ZVD (Zero Vibration), které dosahují nejrychlejší odezvy na vstupní signál. Při velké neurčitosti v modelu systému/signálu lze použít robustní filtry UEI (Extra Insensitive) nebo UTHEI, které dosahují velké šířky nepropustného pásma za cenu delší odezvy filtru. Číslo na konci názvu filtru odpovídá maximální přípustné procentuální úrovni vybuzených vibrací pro dané `omega` a `xi` (jedno, dvě nebo pět procent).

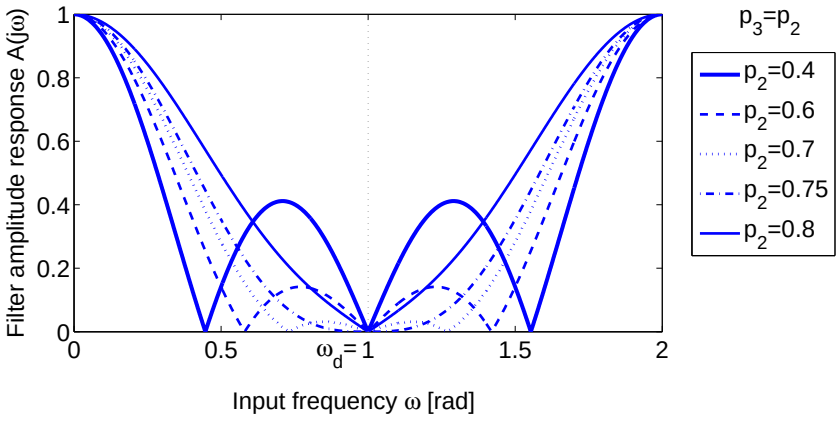
Pro jemné ladění filtru lze použít kompletní parametrizaci volbou `ipar` = `2`, pro kterou se zpřístupní volné parametry `p_alpha`,`p_a2` a `p_a3`. Ty určují tvar frekvenční charakteristiky filtru a lze je použít pro nalezení optimálního kompromisu mezi robustností a zavedeným zpožděním.



Parametr asymetrie `p_alpha` určuje relativní polohu sedla nepropustné oblasti frekvenční charakteristiky filtru vzhledem k nastavené frekvenci `omega`. Kladná hodnota znamená posun vpravo do oblasti vyšších frekvencí, záporná do opačného směru, nulová hodnota vede na symetrickou charakteristiku (viz obrázek výše). S parametrem `p_alpha` souvisí také délka filtru, tedy celkové zpoždění zavedené do cesty vstupního signálu, obecně menší hodnota znamená pomalejší filtr s větším zpožděním. Asymetrické filtry jsou vhodné v případech, kdy frekvence, která má být tlumena je proměnná a pohybuje se v určitém intervalu nad nebo pod nominální známou hodnotou, přičemž vyšší pravděpodobnost se předpokládá na jednom z okrajů intervalu (asymetrická hustota pravděpodobnosti).



Parametr necitlivosti `p_a2` určuje šířku a úroveň útlumu nepropustného pásma filtru. Větší hodnota znamená širší nepropustnou oblast s větším tlumením. Pro praktické aplikace je doporučeno nastavit hodnotu `p_a2` $= 0.5$ pro maximální robustnost tvarovacího filtru vůči chybě v modelu systému/signálu.



Doplňkový parametr `p_a3` je nutné nastavit pro symetrické filtry (volba `p_alpha` $=$ 0). V tomto případě je pro praktické použití vhodné volit *shodné hodnoty* `p_a2` $=$ `p_a3` na intervalu $< 0, 0.75 >$. Menší hodnoty vedou na rychlejší filtry s úzkým nepropustným pásmem, větší pak na robustní tvarovače s širokým pásmem útlumu a delší odezvou (viz obrázek).

Vstup

`u`	Vstupní signál filtru	`double`

Výstupy

`y`	Filtrovaný výstupní signál	`double`
`E`	Příznak chyby	`bool`
	`off` ... bez chyby `on` .... nastala chyba	

Parametry

`omega`	Vlastní frekvence	⊙`1.0`	`double`
`xi`	Součinitel relativního tlumení		`double`
`ipar`	Specifikace	⊙`1`	`long`
	`1` ..... základní typy tvarovačů signálu		
	`2` ..... kompletní parametrizace		
`istype`	Typ	⊙`2`	`long`
	`1` ..... ZV		
	`2` ..... ZVD		
	`3` ..... ZVDD		
	`4` ..... MISZV		
	`5` UEI1		
	`6` UEI2		
	`7` UEI5		
	`8` UTHEI1		
	`9` UTHEI2		
	`10` UTHEI5		
`p_alpha`	Asymetrie tvarovače	⊙`0.2`	`double`
`p_a2`	Necitlivost	⊙`0.5`	`double`
`p_a3`	Přídavný parametr (pouze pro $p\_alpha = 0$)	⊙`0.5`	`double`
`nmax`	Délka vyrovnávací paměti pro dopravní zpoždění (na tolik hodnot se alokuje pamět)	↓`1` ↑`10000000` ⊙`10`	`long`

Kapitola 6

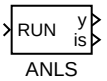
GEN – Generátory signálů

Obsah

ANLS – Řízený generátor po částech lineární funkce

Symbol bloku

Licence: STANDARD



Popis funkce

Blok `ANLS` generuje na výstupu `y` po částech lineární funkci zadanou uzlovými body `t1,y1`; `t2,y2`; `t3,y3`; `t4,y4`. Počáteční hodnota `y` je definována parametrem `y0`. Start generování funkce (časový okamžik 0) je určen náběžnou hranou vstupu `RUN`. V intervalu $\langle t_i, t_{i+1}\rangle, i = 0, \ldots, 3, t_0 = 0$ je výstup `y` definován vztahem

$$y = y_i + \frac{y_{i+1} - y_i}{t_{i+1} - t_i}(t - t_i).$$

Je-li $t_i = t_{i+1}$, potom se výstup `y` mění v čase t_i skokem z hodnoty y_i na hodnotu y_{i+1}. Generování funkce je předčasně ukončeno v případě, že `RUN` = `off` (výstup je resetován na `y0` a `is` na `0`), nebo jestliže $t > t*$, kde $t*$ je rovno času t_i, kde index $i \leq 4$ je největší možné celé číslo takové, že $t_1 < \ldots < t_i$. Po tomto tzv. normálním ukončení si výstup podrží svoji předcházející hodnotu. Má-li parametr `RPT` hodnotu `on`, potom se po normálním ukončení spustí opětovné generování funkce podle stejného algoritmu atd. Takto lze například generovat obdélníkový, pilovitý nebo lichoběžníkový signál.

Vstup

`RUN`	Povolení generování posloupnosti		`bool`

Výstupy

`y`	Analogový výstupní signál		`double`
`is`	Index aktivního časového úseku		`long`

Parametry

`y0`	Počáteční hodnota výstupu		`double`
`t1`	Čas uzlového bodu 1	⊙`1.0`	`double`
`y1`	Hodnota uzlového bodu 1		`double`
`t2`	Čas uzlového bodu 2	⊙`1.0`	`double`
`y2`	Hodnota uzlového bodu 2	⊙`1.0`	`double`
`t3`	Čas uzlového bodu 3	⊙`2.0`	`double`
`y3`	Hodnota uzlového bodu 3	⊙`1.0`	`double`

`t4`	Čas uzlového bodu 4	⊙`2.0`	`double`
`y4`	Hodnota uzlového bodu 4		`double`
`RPT`	Opakování sekvence		`bool`
	`off` ... zakázáno	`on` .... povoleno	

BINS – Řízený generátor binární posloupnosti

Symbol bloku

Licence: STANDARD

Popis funkce

Blok BINS generuje na výstupu Y zadanou binární posloupnost. Počáteční hodnota Y je definována parametrem Y0. Start generování posloupnosti (časový okamžik 0) je určen náběžnou hranou vstupu START. V časech t1, t2,...,t8 se mění hodnota výstupu Y na hodnotu opačnou (off→on, on→off). V případě, že je parametr RPT nastaven na off, nastane poslední přepnutí výstupu v čase t_i, jestliže $t_{i+1} < t_i$. Výstup si poté podrží svoji poslední hodnotu. Má-li však parametr RPT hodnotu on, potom se místo posledního přepnutí vrátí blok do svého původního stavu Y0, interní čas bloku je nastaven na 0 a generování posloupnosti se periodicky opakuje. Dojde-li ke změně parametrů bloku při běhu, potom se nové parametry uplatní až při následném spuštění generování posloupnosti. Poznamenejme, že k opětovnému spuštění generování posloupnosti může dojít i za běhu generátoru.

Časové okamžiky přepnutí jsou interně zaokrouhlovány na nejbližší celý násobek periody spouštění bloku, což může vést např. k vymizení pulzů, které jsou užší než $T_S/2$ nebo spojení více po sobě jdoucích úzkých pulzů do jednoho širokého pulzu. Je proto důrazně doporučováno zadávat okamžiky přepnutí jako celé násobky periody spouštění bloku.

Vstup

START	Spouštěcí signál (náběžná hrana)		bool

Výstupy

Y	Logický výstupní signál		bool
is	Index aktivního časového úseku		long

Parametry

Y0	Počáteční hodnota výstupu off ... vypnuto/nepravda on .... zapnuto/pravda		bool
t1	Okamžik přepnutí 1 [s]	⊙1.0	double
t2	Okamžik přepnutí 2 [s]	⊙2.0	double

`t3`	Okamžik přepnutí 3 [s]	⊙`3.0`	`double`
`t4`	Okamžik přepnutí 4 [s]	⊙`4.0`	`double`
`t5`	Okamžik přepnutí 5 [s]	⊙`5.0`	`double`
`t6`	Okamžik přepnutí 6 [s]	⊙`6.0`	`double`
`t7`	Okamžik přepnutí 7 [s]	⊙`7.0`	`double`
`t8`	Okamžik přepnutí 8 [s]	⊙`8.0`	`double`
`RPT`	Opakování sekvence `off` ... zakázáno `on` .... povoleno		`bool`

BIS – Generátor binární posloupnosti

Symbol bloku

Licence: STANDARD

Popis funkce

Blok BIS generuje na výstupu Y zadanou binární posloupnost. Okamžiku spuštění exekuce bloku je přiřazen čas 0. Počáteční hodnota výstupu Y je definována parametrem Y0. V časech t1, t2,...,t8 se mění hodnota výstupu Y na hodnotu opačnou (off→on, on→off). V případě, že je parametr RPT nastaven na off, nastane poslední přepnutí výstupu v čase t_i, jestliže $t_{i+1} < t_i$. Výstup si poté podrží svoji poslední hodnotu. Má-li však parametr RPT hodnotu on, potom se místo posledního přepnutí vrátí blok do svého původního stavu Y0, interní čas bloku je nastaven na 0 a generování posloupnosti se periodicky opakuje.

Časové okamžiky přepnutí jsou interně zaokrouhlovány na nejbližší celý násobek periody spouštění bloku, což může vést např. k vymizení pulzů, které jsou užší než $T_S/2$ nebo spojení více po sobě jdoucích úzkých pulzů do jednoho širokého pulzu. Je proto důrazně doporučováno zadávat okamžiky přepnutí jako celé násobky periody spouštění bloku.

Výstupy

Y	Logický výstupní signál	bool
is	Index aktivního časového úseku	long

Parametry

Y0	Počáteční hodnota výstupu off ... vypnuto/nepravda on .... zapnuto/pravda		bool
t1	Okamžik přepnutí 1 [s]	⊙1.0	double
t2	Okamžik přepnutí 2 [s]	⊙2.0	double
t3	Okamžik přepnutí 3 [s]	⊙3.0	double
t4	Okamžik přepnutí 4 [s]	⊙4.0	double
t5	Okamžik přepnutí 5 [s]	⊙5.0	double
t6	Okamžik přepnutí 6 [s]	⊙6.0	double
t7	Okamžik přepnutí 7 [s]	⊙7.0	double
t8	Okamžik přepnutí 8 [s]	⊙8.0	double
RPT	Opakování sekvence off ... zakázáno on .... povoleno		bool

MP – Ručně generovaný pulz

Symbol bloku

Licence: STANDARD

Popis funkce

Blok MP generuje na výstupu Y pulz délky pwidth při náběžné hraně parametru BSTATE (off→on). Hodnota parametru BSTATE je algoritmem bloku okamžitě shozena na hodnotu off (BSTATE představuje signál z krátce stisknutého tlačítka). Je-li RPTF = on, potom je aktivní opětovné nahození BSTATE během generování pulsu a výsledkem je prodloužení generovaného pulsu. Je-li RPFT = off, je nahození BSTATE během generování výstupního pulzu neúčinné.

Blok MP reaguje pouze na náběžnou hranu parametru BSTATE, nelze ho tedy použít k vygenerování pulsu ihned při startu exekutivy. K tomuto účelu použijte blok BIS.

Výstup

Y	Logický výstupní signál	bool

Parametry

pwidth	Šířka pulzu [s]	⊙1.0	double
BSTATE	Aktivace výstupního pulzu		bool
	off ... žádná činnost		
	on .... vygenerování výstupního pulzu		
RPTF	Povolení prodloužení pulzu		bool
	off ... zakázáno on .... povoleno		

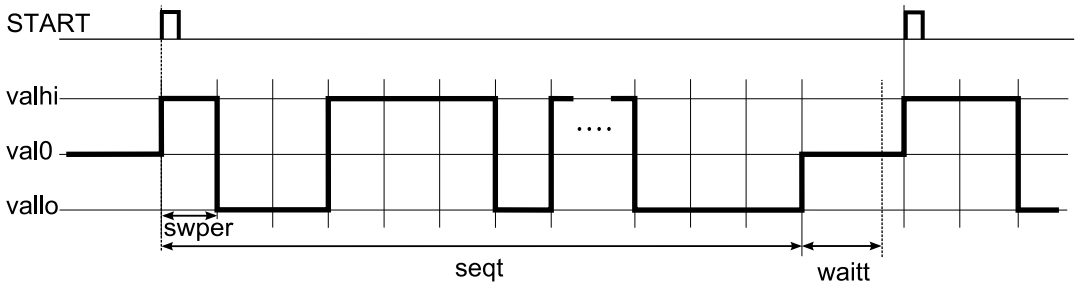
PRBS – Pseudonáhodná binární posloupnost

Symbol bloku

Licence: STANDARD

Popis funkce

Blok PRBS generuje pseudonáhodnou binární posloupnost. Způsob generování je zřejmý z níže uvedeného obrázku.



Počáteční a konečná hodnota posloupnosti je val0. Z této hodnoty je generování startováno náběžnou hranou na vstupu START (off→on). V tom okamžiku se výstup y přepne z hodnoty val0 na hodnotu valhi a dále se přepíná na druhou možnou hladinu s časovou periodou swper a pravděpodobností swprob. Tak se postupuje až do uběhnutí času seqt. Posloupnost je ukončena opět hodnotou val0. Následuje prodleva waitt sloužící pro ustálení odezvy řízené soustavy. Teprve poté je možné odstartovat generování nové posloupnosti. V případě potřeby je možné generování posloupnosti přerušit vstupem BRK = on.

Vstupy

START	Spouštěcí signál (náběžná hrana)		bool
BRK	Signál pro přerušení		bool

Výstupy

y	Vygenerovaná pseudonáhodná binární posloupnost		double
BSY	Příznak probíhající operace		bool

Parametry

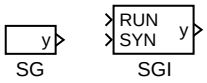
val0	Počáteční a koncová hodnota		double
valhi	Horní hladina výstupu y	⊙1.0	double

`vallo`	Dolní hladina výstupu `y`	⊙`-1.0`	`double`
`swper`	Perioda náhodného přepínání výstupu `y` mezi hladinami [s]	⊙`1.0`	`double`
`swprob`	Pravděpodobnost přepnutí	↓`0.0` ↑`1.0` ⊙`0.2`	`double`
`seqt`	Délka posloupnosti [s]	⊙`10.0`	`double`
`waitt`	Doba ustáleni [s]	⊙`2.0`	`double`

SG, SGI – Řízený generátor signálu

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky SG a SGI mohou generovat podle zvoleného typu signálu isig periodické funkce: sinus, obdélník (se střídou 1), pilovitý signál a bílý šum s rovnoměrným rozdělením. Amplitudu a frekvenci výstupního signálu y určují po řadě parametr amp a freq. Výstup y v případech isig $\in \{1, 2, 3\}$ může být navíc fázově posunut podle parametru phase $\in (0, 2\pi)$.

V případě bloku SGI je možné navíc synchronizovat počátek generování výstupů u více generátorů SGI pomocí vstupů RUN a SYN. Vstup SYN je možné používat za běhu po změně parametrů bloku.

Vstupy

RUN	Povolení běhu algoritmu, spuštění generátoru	bool
SYN	Synchronizační signál	bool

Výstup

y	Analogový výstupní signál	double

Parametry

isig	Typ generovaného signálu 1 ..... sinusový signál 2 ..... obdélníkový signál se střídou 1 3 ..... pilovitý signál 4 ..... bílý šum s rovnoměrným rozdělením	⊙1	long
amp	Amplituda generovaného signálu	⊙1.0	double
freq	Frekvence generovaného signálu	⊙1.0	double
phase	Fázový posun generovaného signálu		double
offset	Hodnota přičítaná k výstupu	⊙1.0	double
ifrunit	Jednotky pro frekvenci 1 ..... Hz 2 ..... rad/s	⊙1	long

`iphunit`	Jednotky pro fázový posun `1` ..... stupně `2` ..... radiány	⊙`1`	`long`

Kapitola 7

REG – Bloky pro regulaci

Obsah

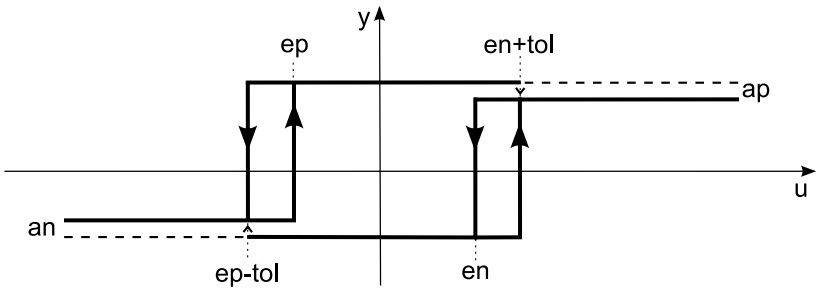
ARLY – Relé s předstihem

Symbol bloku

Licence: STANDARD

Popis funkce

Blok ARLY transformuje vstupní analogový signál u na výstupní analogový signál y podle níže uvedeného obrázku.



Vstup

u	Analogový vstupní signál	double

Výstup

y	Analogový výstupní signál	double

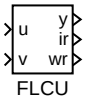
Parametry

ep	Mez pro přepnutí do stavu „Zapnuto“	⊙-1.0	double
en	Mez pro přepnutí do stavu „Vypnuto“	⊙1.0	double
tol	Toleranční mez pro amplitudu superponovaného šumu vstupu u	↓0.0 ⊙0.5	double
ap	Hodnota výstupu y ve stavu „Zapnuto“	⊙1.0	double
an	Hodnota výstupu y ve stavu „Vypnuto“	⊙-1.0	double
y0	Počáteční hodnota výstupu y po spuštění		double

FLCU – Fuzzy regulátor

Symbol bloku

Licence: ADVANCED



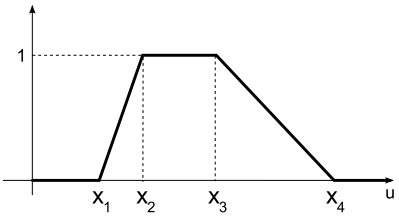
Popis funkce

Blok FLCU realizuje jednoduchý fuzzy regulátor se dvěma vstupy a jedním výstupem. Dostatečný úvod do problematiky fuzzy řízení je uveden v textu [4].

Funkce bloku je jednoznačně určena lichoběžníkovými funkcemi příslušnosti jazykových výrazů vstupů u a v, dále impulsními funkcemi příslušnosti jazykových výrazů výstupu y a konečně expertními pravidly. Pravidla mají následující tvar:

$$\text{Jestliže (u je } U_i\text{) AND (v je } V_j\text{), potom (y je } Y_k\text{)},$$

kde $U_i, i = 1, \ldots, \texttt{nu}$ jsou jazykové výrazy příslušné ke vstupu u; $V_j, j = 1, \ldots, \texttt{nv}$ jsou jazykové výrazy příslušné ke vstupu v a $Y_k, k = 1, \ldots, \texttt{ny}$ jsou jazykové výrazy příslušné k výstupu y. Lichoběžníkové (trojúhelníkové) funkce příslušnosti odpovídající vstupům u a v jsou definovány čtyřmi čísly podle následujícího obrázku



U trojúhelníkových funkcí nejsou všechna čísla $x_1, \ldots, x_4$ vesměs různá. Matice funkcí příslušnosti vstupů u a v se potom skládají z řádků $[x_1, x_2, x_3, x_4]$. Matice mfu a mfv jsou tedy po řadě typu $(\texttt{nu} \times 4)$ a $(\texttt{nv} \times 4)$.

Impulsní funkce příslušnosti prvního řádu odpovídající výstupu y se zapisují jako trojice

$$\texttt{y}_k,\ a_k,\ b_k,$$

kde $\texttt{y}_k$ je hodnota výstupu přiřazená jazykovému výrazu $Y_k, k = 1, \ldots, \texttt{ny}$ v případě $a_k = b_k = 0$. Je-li $a_k \neq 0$ a $b_k \neq 0$, potom je výrazu Y_k přiřazena hodnota $\texttt{y}_k + a_k\texttt{u} + b_k\texttt{v}$. Matice funkcí příslušnosti výstupu sty je typu $(\texttt{ny} \times 3)$ a skládá se po řadě z řádků $[\texttt{y}_k, a_k, b_k], k = 1, \ldots, \texttt{ny}$.

Soubor pravidel se skládá též jako matice a její řádky jsou $[i_l, j_l, k_l, w_l], l = 1, \ldots, \texttt{nr}$, kde i_l, j_l a k_l označuje jistý jazykový výraz příslušný po řadě vstupu u, v a výstupu y.

Číslo w_l udává váhu pravidla v procentech $w_l \in \{0, 1, \ldots, 100\}$. Tímto způsobem lze jednoduše některé pravidlo zdůraznit, popřípadě vypustit.

Vstupy

`u`	První analogový vstup bloku	`double`
`v`	Druhý analogový vstup bloku	`double`

Parametry

`umax`	Horní omezení vstupu `u`	⊙`1.0`	`double`
`umin`	Dolní omezení vstupu `u`	⊙`-1.0`	`double`
`nu`	Počet funkcí příslušnosti – vstup `u`	↓`1` ↑`25` ⊙`3`	`long`
`vmax`	Horní omezení vstupu `v`	⊙`1.0`	`double`
`vmin`	Dolní omezení vstupu `v`	⊙`-1.0`	`double`
`nv`	Počet funkcí příslušnosti – vstup `v`	↓`1` ↑`25` ⊙`3`	`long`
`ny`	Počet funkcí příslušnosti – výstup `y`	↓`1` ↑`100` ⊙`3`	`long`
`nr`	Počet pravidel	↓`1` ↑`25` ⊙`3`	`long`
`mfu`	Matice funkcí příslušnosti – vstup `u`	⊙`[-1 -1 -1 0; -1 0 0 1; 0 1 1 1]`	`double`
`mfv`	Matice funkcí příslušnosti – vstup `v`	⊙`[-1 -1 -1 0; -1 0 0 1; 0 1 1 1]`	`double`
`sty`	Matice funkcí příslušnosti – výstup `y`	⊙`[-1 0 0; 0 0 0; 1 0 0]`	`double`
`rls`	Matice pravidel	⊙`[1 2 3 100; 1 1 1 100; 1 0 3 100]`	`byte`

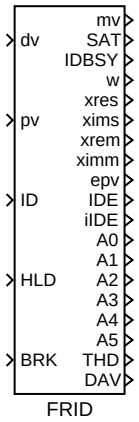
Výstupy

`y`	Analogový výstupní signál	`double`
`ir`	Dominantní pravidlo	`long`
`wr`	Stupeň pravdivosti dominantního pravidla	`double`

FRID – * Identifikace frekvenční charakteristiky

Symbol bloku

Licence: ADVANCED



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

dv	Proměnná dopředné vazby	double
pv	Řízená veličina	double
ID	Zahájení ladicího experimentu	bool
HLD	Pozastavení	bool
BRK	Ukončení ladicího experimentu	bool

Parametry

ubias	Stejnosměrná složka budicího signálu	
uamp	Amplituda budicího signálu	⊙1.
wb	Počáteční frekvence [rad/s]	⊙1.
wf	Koncová frekvence [rad/s]	⊙10.
isweep	Režim rozmítání 1 ..... logaritmické 2 ..... lineární	⊙
cp	Rychlost rozmítání	⊙0.99
iavg	Počet hodnot pro průměrování	⊙1

obw	Šířka pásma 1 ..... Malá 2 ..... Střední 3 ..... Velká	⊙2	long
stime	Doba ustálení [s]	⊙10.0	double
umax	Maximální amplituda generátoru	⊙1.0	double
thdmin	Minimální požadované zkreslení	⊙0.1	double
adapt_rc	Rychlost změny amplitudy rozmítání	⊙0.001	double
pv_max	Maximální požadovaná amplituda PV	⊙1.0	double
pv_sat	Maximální dovolená amplituda PV	⊙2.0	double
ADAPT_EN	Zapnutí adaptace amplitudy generátoru	⊙on	bool
immode	Režim měření 1 ..... Ruční volba frekvencí 2 ..... Lineárně nmw frekvencí v intervalu <wb,wf> 3 ..... Logaritmicky nmw frekvencí v intervalu <wb,wf> 4 ..... Automatická detekce důležitých frekvencí (N/A)	⊙1	long
nwm	Počet frekvencí v automatickém režimu		long
wm	Pole frekvencí pro ruční režim [array of rad/s]	⊙[2.0 4.0 6.0 8.0]	double

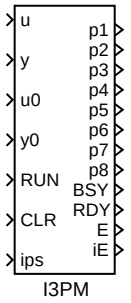
Výstupy

mv	Akční zásah regulátoru (manipulated variable)	double
SAT	Saturace	bool
IDBSY	Příznak probíhajícího ladicího experimentu	bool
w	Actuální frekvence [rad/s]	double
xres	Reálná složka přenosu (rozmítání)	double
xims	Imaginární složka přenosu (rozmítání)	double
xrem	Reálná složka přenosu (změřená)	double
ximm	Imaginární složka přenosu (změřená)	double
epv	Odhad PV	double
IDE	Příznak chyby	bool
iIDE	Kód chyby	long
A0	Odhad stejnosměrné složky přenosu	double
A1	Odhad amplitudy 1. harmonické	double
A2	Odhad amplitudy 2. harmonické	double
A3	Odhad amplitudy 3. harmonické	double
A4	Odhad amplitudy 4. harmonické	double
A5	Odhad amplitudy 5. harmonické	double
THD	Celkové harmonické zkreslení	double
DAV	Platná data	bool

I3PM – Identifikace modelu se třemi parametry

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok I3PM identifikuje tříparametrový model soustavy metodou zobecněných momentů.

Vstupy

u	Vstup identifikované soustavy	double
y	Výstup identifikované soustavy	double
u0	Ustálená hodnota vstupu	double
y0	Ustálená hodnota výstupu	double
RUN	Spuštění identifikace	bool
CLR	Nulování bloku	bool
ips	Význam výstupních signálů	long

- 0 ..... model prvního řádu s dopravním zpožděním
  - p1 ... zesílení
  - p2 ... dopravní zpoždění
  - p3 ... časová konstanta
- 1 ..... momenty vstupu a výstupu
  - p1 ... parametr $mu0$
  - p2 ... parametr $mu1$
  - p3 ... parametr $mu2$
  - p4 ... parametr $my0$
  - p5 ... parametr $my1$
  - p6 ... parametr $my2$
- 2 ..... momenty procesu
  - p1 ... parametr $mp0$
  - p2 ... parametr $mp1$
  - p3 ... parametr $mp2$

`3` ..... charakteristická čísla procesu
`p1` ... parametr κ
`p2` ... parametr μ
`p3` ... parametr σ^2
`p4` ... parametr σ

Výstupy

`pi`	Identifikované parametry v závislosti na `ips`, $i = 1, ..., 8$	`double`
`BSY`	Příznak probíhající identifikace	`bool`
`RDY`	Příznak připravenosti	`bool`
`E`	Příznak chyby	`bool`
`iE`	Kód chyby	`long`

`1` ..... předčasné ukončení (`RUN` = `off`)
`2` ..... `mu0` = 0
`3` ..... `mp0` = 0
`4` ..... $\sigma^2 < 0$

Parametry

`tident`	Délka identifikace [s]	⊙`100.0`	`double`
`irtype`	Typ regulátoru	⊙`6`	`long`

`1` ..... D `3` ..... ID `5` ..... PD `7` ..... PID
`2` ..... I `4` ..... P `6` ..... PI

`ispeed`	Požadovaná rychlost uzavřené smyčky	⊙`2`	`long`

`1` ..... pomalá uzavřená smyčka
`2` ..... středně rychlá uzavřená smyčka
`3` ..... rychlá uzavřená smyčka

LC – Derivační kompenzátor

Symbol bloku

Licence: STANDARD

Popis funkce

Blok LC realizuje diskrétní simulátor přenosu derivačního článku

$$C(s) = \frac{\mathtt{td} * s}{\frac{\mathtt{td}}{\mathtt{nd}} * s + 1},$$

kde td je derivační konstanta a nd je parametr určující vliv parazitního filtru prvního řádu. Doporučená hodnota nd je $2 \leq \mathtt{nd} \leq 10$. Je-li ISSF = on, potom je stav parazitního filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu u.

Pro diskretizaci přenosu $C(s)$ je použita přesná diskretizace v okamžicích vzorkování.

Vstup

u	Analogový vstupní signál	double

Výstup

y	Analogový výstupní signál	double

Parametry

td	Derivační časová konstanta	⊙1.0	double
nd	Parametr filtru derivační složky	⊙10.0	double
ISSF	Ustálený stav při spuštění off ... nulový počáteční stav on .... ustálený počáteční stav		bool

LLC – Integračně-derivační kompenzátor

Symbol bloku

Licence: STANDARD

Popis funkce

Blok LLC realizuje diskrétní simulátor přenosu integračně-derivačního článku

$$C(s) = \frac{\mathtt{a} * \mathtt{tau} * s + 1}{\mathtt{tau} * s + 1},$$

kde tau je časová konstanta jmenovatele a její a-násobek (a * tau) je časová konstanta čitatele. Je-li ISSF = on, potom je stav integračního článku nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu u.

Tento blok je ideální pro simulaci lineárních systémů prvního řádu s dopravním zpožděním (FOPDT) Stačí použít nulový parametr a.

Pro diskretizaci přenosu $C(s)$ je použita přesná diskretizace v okamžicích vzorkování.

Vstup

u	Analogový vstupní signál		double

Parametry

tau	Časová konstanta	⊙1.0	double
a	Koeficient pro výpočet časové konstanty čitatele		double
ISSF	Ustálený stav při spuštění off ... nulový počáteční stav on .... ustálený počáteční stav		bool

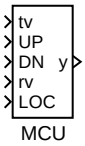
Výstup

y	Analogový výstupní signál		double

MCU – Jednotka pro ruční zadávání

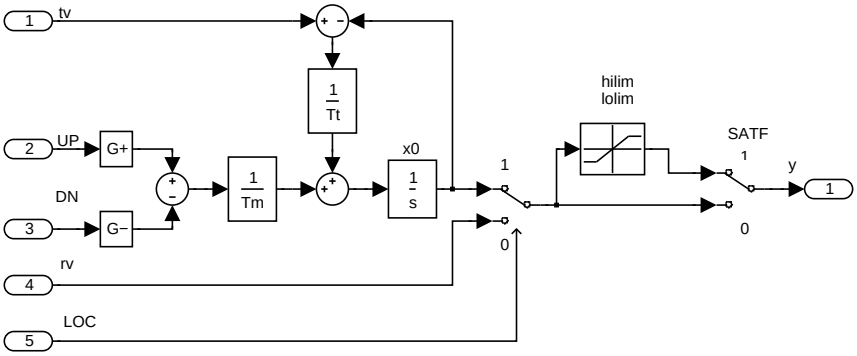
Symbol bloku

Licence: STANDARD



Popis funkce

V lokálním režimu (`LOC` = `on`) je blok `MCU` určen k ručnímu zadávání výstupu `y` pomocí tlačítek „více“ (vstup `UP`) a „méně“ (vstup `DN`). Strmost najíždění z počáteční hodnoty `y0` na žádanou hodnotu je určena integrační konstantou `tm` a dobou stlačení ovládacích tlačítek. Po uplynutí každých `ta` sekund je strmost vždy násobena faktorem `q`, až do vypršení doby `tf`. Rozsah výstupu `y` může být omezen (`SATF` = `on`) saturačními mezemi `lolim` a `hilim`. V případě, že žádné z tlačítek není stlačeno (`UP` = `off` a `DN` = `off`), vysleduje výstup `y` vstupní hodnotu `tv`. Rychlost vysledování je dána integrační časovou konstantou `tt`. V případě `LOC` = `off` je vstup `rv` s případnými omezeními (`SATF` = `on`) kopírován na výstup `y`. Podrobná funkce bloku je přímo patrná z obrázku s vnitřním schématem bloku.



Vstupy

`tv`	Veličina pro vysledování	`double`
`UP`	Signál UP (nahoru, více)	`bool`
`DN`	Signál DN (dolů, méně)	`bool`
`rv`	Hodnota pro externí zadávání výstupu v režimu `LOC` = `off`	`double`
`LOC`	Lokální nebo vzdálený režim	`bool`

Výstup

`y`	Analogový výstupní signál		`double`

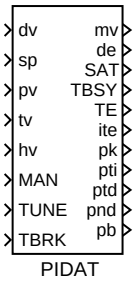
Parametry

`tt`	Časová konstanta vysledování vstupní hodnoty `tv`	⊙`1.0`	`double`
`tm`	Počáteční časová konstanta strmosti najíždění	⊙`100.0`	`double`
`y0`	Počáteční hodnota výstupu		`double`
`q`	Faktor určující velikost změny strmosti najíždění	⊙`5.0`	`double`
`ta`	Interval, po kterém dochází ke zvýšení strmosti [s]	⊙`4.0`	`double`
`tf`	Interval, po kterém se strmost již dále nemění [s]	⊙`8.0`	`double`
`SATF`	Saturace `off` ... signál není omezen `on` .... saturační meze jsou aktivní		`bool`
`hilim`	Horní saturační mez	⊙`1.0`	`double`
`lolim`	Dolní saturační mez	⊙`-1.0`	`double`

PIDAT – PID regulátor s reléovým autotunerem

Symbol bloku

Licence: AUTOTUNING



Popis funkce

Blok PIDAT má zcela stejné regulační funkce jako blok PIDU. Navíc je vybaven funkcí automatického nastavování parametrů regulátoru. Pro využití této funkce je nutné převést řízený systém do přibližně ustáleného stavu (ve vhodném pracovním bodě), zvolit požadovaný typ regulátoru (PI nebo PID) a aktivovat vstup TUNE hodnotou on (start identifikačního experimentu). V následném identifikačním experimentu je řízený proces regulován pomocí speciálního adaptivního reléového regulátoru a ze získaného záznamu vstupu a výstupu procesu je odhadnut vhodný bod jeho frekvenční charakteristiky. Na základě toho jsou poté určeny parametry regulátoru. Amplitudu reléového regulátoru (úroveň vybuzení systému) je možné nastavit parametrem amp a jeho hysterezi parametrem hys. Zvolíme-li hys = 0, potom se hystereze relé určí automaticky na základě odhadu úrovně šumu měření regulované veličiny. Během identifikačního experimentu je TBSY = on. Po řádném skončení experimentu je TE = off a vypočítané parametry se objeví na výstupech pk, pti, ptd, pnd, pb. Váhový koeficient c je uvažován c = 0. Skončil-li experiment s chybou, je TE = on a ite blíže specifikuje důvod chyby. Při výskytu chyby se doporučuje zvětšit parametr amp. Jeho volbu usnadňuje zabudovaná funkce, která parametr amp automaticky zmenšuje při hrozbě překročení maximální dovolené odchylky maxdev regulované veličiny od jejího počátečního ustáleného stavu. Identifikační experiment je možné předčasně ukončit aktivací vstupu TBRK.

Vstupy

dv	Proměnná dopředné vazby	double
sp	Požadovaná hodnota (setpoint)	double
pv	Řízená veličina	double
tv	Veličina pro vysledování	double
hv	Hodnota výstupu v manuálním režimu	double

`MAN`	Manuální nebo automatický režim `off` ... automatický režim `on` .... manuální režim	`bool`
`TUNE`	Zahájení ladicího experimentu	`bool`
`TBRK`	Ukončení ladicího experimentu	`bool`

Výstupy

`mv`	Akční zásah regulátoru (manipulated variable)	`double`
`de`	Regulační odchylka	`double`
`SAT`	Saturace `off` ... lineární zákon řízení `on` .... výstup regulátoru je saturován	`bool`
`TBSY`	Příznak probíhajícího ladicího experimentu	`bool`
`TE`	Příznak chyby během ladění `off` ... Ladění proběhlo bez chyby `on` .... Během ladění se vyskytla chyba	`bool`
`ite`	Kód chyby (během probíhajícího ladícího experimentu očekávaný čas v sekundách do jeho konce) `1000` .. příliš nízký poměr užitečného signálu k šumu měření `1001` .. příliš velká hystereze reléového regulátoru `1002` .. příliš přísné pravidlo pro ukončení `1003` .. příliš velká chyba při určování fáze identifikovaného bodu	`long`
`pk`	Navržené zesílení regulátoru	`double`
`pti`	Navržená integrační časová konstanta regulátoru	`double`
`ptd`	Navržená derivační časová konstanta regulátoru	`double`
`pnd`	Navržený parametr filtru derivační složky	`double`
`pb`	Navržený váhový faktor pro proporcionální složku	`double`

Parametry

`irtype`	Typ regulátoru `1` ..... D `4` ..... P `7` ..... PID `2` ..... I `5` ..... PD `3` ..... ID `6` ..... PI	⊙`6`	`long`

`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
			`bool`

`k`	Zesílení regulátoru K	⊙`1.0`	`double`
`ti`	Integrační časová konstanta T_i	⊙`4.0`	`double`
`td`	Derivační časová konstanta T_d	⊙`1.0`	`double`
`nd`	Parametr N filtru derivační složky	⊙`10.0`	`double`
`b`	Váhový faktor pro proporcionální složku	⊙`1.0`	`double`
`c`	Váhový faktor pro derivační složku		`double`

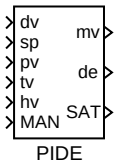
`tt`	Časová konstanta vysledování. Pro regulátory bez integrační složk nemá žádný význam. ⊙1.		
`hilim`	Horní mez akčního zásahu regulátoru ⊙1.		
`lolim`	Dolní mez akčního zásahu regulátoru ⊙-1.		
`iainf`	Druh apriorní informace ⊙ `1` ..... žádná apriorní informace `2` ..... astatický proces `3` ..... proces nízkého řádu `4` ..... statický proces + požadavek na aperiodickou odezv uzavřené smyčky `5` ..... statický proces + požadavek na středně rychlou odezv uzavřené smyčky `6` ..... statický proces + požadavek na rychlou odezvu uzavřen smyčky		
`k0`	Statické zesílení procesu (musí být zadáno v případě `iainf` = 3, 4, 5 ⊙1.		
`n1`	Maximální počet půlperiod pro nalezení bodu frekvenčn charakteristiky ⊙2		
`mm`	Maximální počet půlperiod pro průměrování ⊙		
`amp`	Amplituda reléového regulátoru ⊙0.		
`uhys`	Hystereze reléového regulátoru		
`ntime`	Čas vymezený pro odhad amplitudy šumu na počátku experimentu [s] ⊙5.0		`double`
`rerrap`	Ukončovací hodnota relativní chyby amplitudy kmitů ⊙0.1		`double`
`aerrph`	Ukončovací hodnota absolutní chyby fáze odhadovaného bodu ⊙10.0		`double`
`maxdev`	Maximální přípustná odchylka regulované veličiny od ustáleného stavu ⊙1.0		`double`

Parametry `n1`, `mm`, `ntime`, `rerrap` a `aerrph` se nedoporučuje měnit.

PIDE – PID regulátor se statikou

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok PIDE je základní blok pro vytvoření úplného modifikovaného regulátoru PI(D), který se liší od standardního PI(D) regulátoru (blok PIDU) tím, že má zadané konečné statické zesílení (ve skutečnosti se zadává velikost odchylky ε, která způsobí saturaci výstupu). V nejjednodušším případě může pracovat zcela samostatně a plnit standardní funkci modifikovaného PID regulátoru s dvěma stupni volnosti v automatickém (MAN = off) nebo manuálním režimu (MAN = on).

V automatickém režimu v lineární oblasti realizuje zákon řízení daný vztahem

$$U(s) = \pm K \left[bW(s) - Y(s) + \frac{1}{T_i s + \beta} E(s) + \frac{T_d s}{\frac{T_d s}{N} + 1} (cW(s) - Y(s)) \right] + Z(s),$$

kde

$$\beta = \frac{K\varepsilon}{1 - K\varepsilon}$$

a $U(s)$ je obraz akční veličiny mv, $W(s)$ je obraz požadované hodnoty sp, $Y(s)$ je obraz regulované veličiny pv, $E(s)$ je Laplaceova transformace regulační odchylky, $Z(s)$ je obraz dopředné vazby dv a $K, T_i, T_d, N, \varepsilon\ (= b_p/100), b, c$ jsou parametry regulátoru. Znaménko pravé strany závisí na parametru RACT. Rozsah řídicí veličiny mv je omezen saturačními mezemi lolim a hilim. Propojením výstupu mv se vstupem tv a vhodnou volbou parametru tt dosáhneme žádaného chování regulátoru při dosažení saturačních hodnot mv. Odstraníme tak nežádoucí unášení integrační složky (wind up effect) a současně s tím zajistíme bezrázové přepínání (bumpless transfer) automatického a manuálního režimu.

V manuálním režimu je vstup hv (po případném omezení) kopírován na výstup mv. Signál připojený na vstup tv zajišťuje v tomto režimu příslušné vysledování vnitřního stavu regulátoru pro následné bezrázové přepnutí do automatického režimu (pro $\varepsilon > 0$ však vysledování není zcela přesné).

Vstupy

dv	Proměnná dopředné vazby	double
sp	Požadovaná hodnota (setpoint)	double

`pv`	Řízená veličina	`double`
`tv`	Veličina pro vysledování	`double`
`hv`	Hodnota výstupu v manuálním režimu	`double`
`MAN`	Manuální nebo automatický režim `off` ... automatický režim `on` .... manuální režim	`bool`

Výstupy

`mv`	Akční zásah regulátoru (manipulated variable)	`double`
`de`	Regulační odchylka	`double`
`SAT`	Saturace `off` ... lineární zákon řízení `on` .... výstup regulátoru je saturován	`bool`

Parametry

`irtype`	Typ regulátoru `1` ..... D `4` ..... P `7` ..... PID `2` ..... I `5` ..... PD `3` ..... ID `6` ..... PI	⊙`6`	`long`

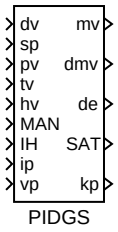
`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
			`bool`

`k`	Zesílení regulátoru K	⊙`1.0`	`double`
`ti`	Integrační časová konstanta T_i	⊙`4.0`	`double`
`td`	Derivační časová konstanta T_d	⊙`1.0`	`double`
`nd`	Parametr N filtru derivační složky	⊙`10.0`	`double`
`b`	Váhový faktor pro proporcionální složku	⊙`1.0`	`double`
`c`	Váhový faktor pro derivační složku		`double`
`tt`	Časová konstanta vysledování. Pro regulátory bez integrační složky nemá žádný význam.	⊙`1.0`	`double`
`bp`	Hodnota regulační odchylky, která způsobí, že výstup regulárotu `mv` v ustáleném stavu je roven hodnotě 100		`double`
`hilim`	Horní mez akčního zásahu regulátoru	⊙`1.0`	`double`
`lolim`	Dolní mez akčního zásahu regulátoru	⊙`-1.0`	`double`

PIDGS – PID regulátor s přepínáním sad parametrů

Symbol bloku

Licence: ADVANCED



Popis funkce

Regulační funkce bloku PIDGS je přesně shodná s blokem PIDU. Blok PIDGS má však až šest sad základních parametrů, které je možné bezrázově přepínat pomocí vstupu ip (index sady parametrů) nebo vstupu vp (přepínací analogová veličina). V případě použití přepínací analogové veličiny je třeba zadat GSCF = on a vektor příslušných přepínacích mezí thsha. Sady parametrů jsou poté přepínány takto: sada 0 je pro vp < thrsha(0), sada 1 pro thrsha(1) < vp < thrsha(2) atd. až sada 5 pro thrsha(5) < vp. Index aktuální sady je k dispozici na výstupu kp.

Vstupy

dv	Proměnná dopředné vazby		double
sp	Požadovaná hodnota (setpoint)		double
pv	Řízená veličina		double
tv	Veličina pro vysledování		double
hv	Hodnota výstupu v manuálním režimu		double
MAN	Manuální nebo automatický režim off ... automatický režim on .... manuální režim		bool
IH	Zastavení integrace off ... integrování povoleno on .... integrování pozastaveno		bool
ip	Index sady parametrů	↓0 ↑5	long
vp	Přepínací veličina		double

Výstupy

mv	Akční zásah regulátoru (manipulated variable)	double
dmv	Rychlostní výstup regulátoru (diference)	double
de	Regulační odchylka	double

SAT	Saturace off ... lineární zákon řízení on .... výstup regulátoru je saturován	bool
kp	Index aktuální sady parametrů	long

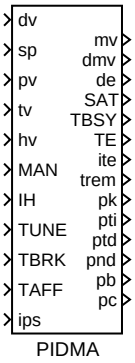
Parametry

hilim	Horní mez akčního zásahu regulátoru	⊙1.
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.
dz	Pásmo necitlivosti	
icotype	Typ výstupu regulátoru 1 ..... analogový výstup 2 ..... šířkově modulovaný výstup (PWM) 3 ..... krokový regulátor s polohovou zpětnou vazbou (SCU) 4 ..... krokový regulátor bez polohové zpětné vazby (SCUV)	⊙
npars	Počet sad parametrů	⊙
GSCF	Přepínání parametrů podle vstupu vp off ... přepínání indexem sady parametrů on .... přepínání analogovým signálem	
hys	Hystereze pro přepínání podle vstupu vp	
irtypea	Vektor typů regulátoru 1 ..... D 4 ..... P 7 ..... PID 2 ..... I 5 ..... PD 3 ..... ID 6 ..... PI	⊙[6 6 6 6 6 6
RACTA	Vektor příznaků obráceného působení výstupu regulátoru 0 ..... vyšší mv → vyšší pv 1 ..... vyšší mv → nižší pv	⊙[0 0 0 0 0 0
ka	Vektor zesílení regulátoru K	⊙[1.0 1.0 1.0 1.0 1.0 1.0
tia	Vektor integračních časových konstant T_i	⊙[4.0 4.0 4.0 4.0 4.0 4.0
tda	Vektor derivačních časových konstant T_d	⊙[1.0 1.0 1.0 1.0 1.0 1.0
nda	Vektor parametrů filtru derivační složky N	⊙[10.0 10.0 10.0 10.0 10.0 10.0
ba	Váhové faktory pro proporcionální složku	⊙[1.0 1.0 1.0 1.0 1.0 1.0
ca	Váhové faktory pro derivační složku	⊙[0.0 0.0 0.0 0.0 0.0 0.0
tta	Vektor časových konstant vysledování. Pro regulátory bez integrač složky nemá žádný význam.	⊙[1.0 1.0 1.0 1.0 1.0 1.0
thrsha	Vektor mezí přepínací veličiny	⊙[0.1 0.2 0.3 0.4 0.5 0] double

PIDMA – PID regulátor s momentovým autotunerem

Symbol bloku

Licence: AUTOTUNING



Popis funkce

V automatickém režimu (MAN = off) realizuje blok PIDMA řídicí zákon PID regulátoru se dvěma stupni volnosti ve tvaru

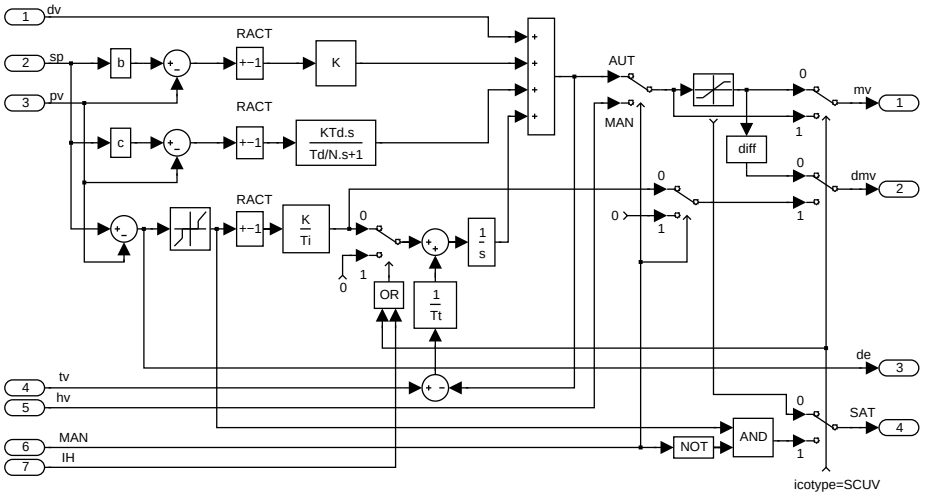
$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s}\left[W(s) - Y(s)\right] + \frac{T_d s}{\frac{T_d}{N}s + 1}\left[cW(s) - Y(s)\right] \right\} + Z(s)$$

kde $U(s)$ je Laplaceova transformace řídicí veličiny mv, $W(s)$ je Laplaceova transformace požadované hodnoty sp, $Y(s)$ je Laplaceova transformace regulované veličiny pv, $Z(s)$ je Laplaceova transformace dopředné vazby dv a K, T_i, T_d, N, b, c jsou parametry regulátoru. Znaménko pravé strany závisí na parametru RACT. Rozsah řídicí veličiny mv (polohového výstupu regulátoru) je omezen parametry hilim, lolim. Parametr dz udává pásmo necitlivosti v integrační složce regulátoru. Navíc integrační složka může být vypnuta a zafixována na své aktuální hodnotě vstupem IH = on. Pro správnou funkci regulátoru je nutné propojit výstup regulátoru mv se vstupem tv a správně nastavit časovou konstantu vysledovávání tt (doporučená hodnota je tt $\approx \sqrt{T_i T_d}$, pro PI regulátor tt $\approx 2 \cdot \sqrt{T_i}$). Tím bude zaručen bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správná funkce regulátoru při saturaci výstupu mv (tzv. antiwindup). Přídavné výstupy dmv, de a SAT poskytují po řadě rychlostní výstup regulátoru (diference mv), regulační odchylku a příznak saturace výstupu regulátoru mv.

Jestliže je blok PIDMA propojen s blokem SCUV (za účelem realizace krokového regulátoru bez polohové zpětné vazby), potom parametr icotype musí být nastaven na hodnotu 4 a význam výstupů mv, dmv a SAT je v tomto případě pozměněn: výstup mv je roven součtu P a D složky regulátoru, zatímco výstup dmv poskytuje diferenci jeho I složky a výstup SAT nese informaci pro blok SCUV, zda je regulační odchylka de v automatickém režimu menší než pásmo necitlivosti dz. Pro případ propojení bloků PIDMA

a SCUV se navíc doporučuje volit váhový koeficient požadované hodnoty pro derivační složku `c` rovný nule.

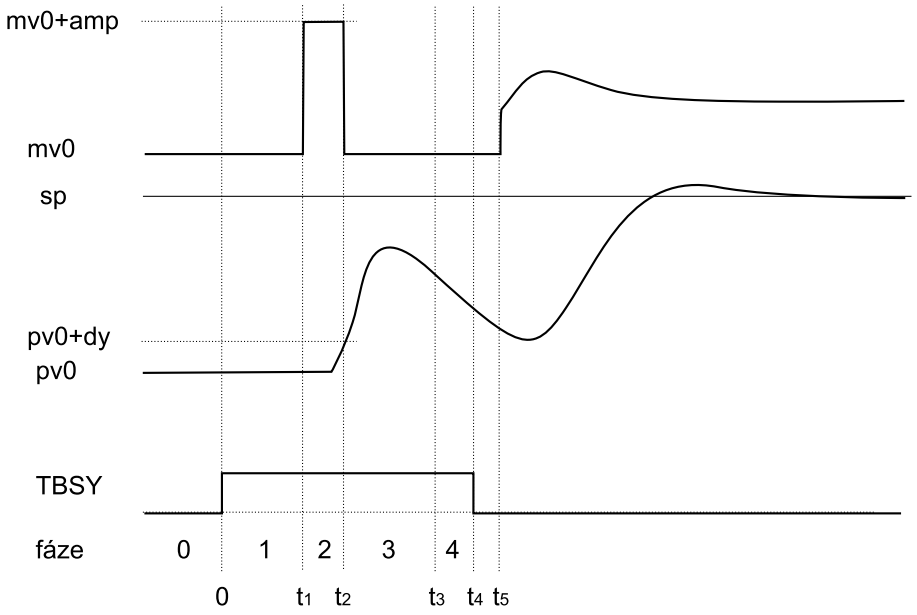
V manuálním režimu (`MAN` = `on`) je vstup `hv` kopírován na výstup `mv`. Celková regulační funkce bloku `PIDMA` je zřejmá z následujícího obrázku.



Blok `PIDMA` rozšiřuje řídicí funkci standardního PID regulátoru o vestavěné automatické nastavování parametrů (PID autotuner). Před spuštěním autotuneru musí operátor ve vhodném pracovním bodě dosáhnout ustáleného stavu a zvolit požadovaný typ regulátoru `ittype` (`PI` nebo `PID`) a nastavit další parametry autotuneru (`iainf`, `DGC`, `tdg`, `tn`, `amp`, `dy` a `ispeed`). Identifikační experiment se startuje vstupem `TUNE` (vstupem `TBRK` jej lze předčasně ukončit). V tomto módu (`TBSY` = `on`) je nejprve odhadnut drift a šum regulované veličiny (ve specifikovaném čase `tdg`+`tn`) a poté je na vstup procesu aplikován pravoúhlý puls. Z odezvy procesu jsou odhadnuty první tři momenty jeho impulsní odezvy. Amplituda pulsu se nastavuje parametrem `amp`. Puls je ukončen poté, co se hodnota regulované veličiny `pv` změní o více, než určuje tolerance (práh) `dy` (zadává se vždy jako kladné číslo). Pokud je nastaven příznak `DGC`, používá se při zpracování signálu speciální kompenzace trendu signálu. Odhad času zbývajícího do konce procesu ladění je přiveden na výstup `trem`.

Pokud experiment skončí úspěšně (`TE` = `off`) a vstup `ips` = `0`, objeví se optimální parametry na výstupech `pk`, `pti`, `ptd`, `pnd`, `pb`, `pc`. V opačném případě (`TE` = `on`) určuje výstup `ite` kód chyby experimentu. Další hodnoty vstupu `ips` jsou rezervovány pro speciální účely.

Funkce autotuneru je demonstrována na následujícím obrázku.



Během identifikačního experimentu výstup `ite` indikuje jednotlivé fáze činnosti autotuneru. Ve fázi odhadu strmosti odeznívání odezvy (`ite` = `-4`) může být proces ladění předčasně manuálně ukončen. V tomto případě jsou parametry regulátoru řádně navrženy, avšak jejich možná nepřesnost je indikována varovným kódem `ite` = `100`.

Po ukončení experimentu (`TBSY` `on`→`off`) je funkce regulátoru závislá na nastaveném režimu (manuální, automatický). Jestliže `TAFF` = `on`, potom jsou navržené parametry okamžitě použity.

Vstupy

`dv`	Proměnná dopředné vazby	`double`
`sp`	Požadovaná hodnota (setpoint)	`double`
`pv`	Řízená veličina	`double`
`tv`	Veličina pro vysledování	`double`
`hv`	Hodnota výstupu v manuálním režimu	`double`
`MAN`	Manuální nebo automatický režim `off` ... automatický režim `on` .... manuální režim	`bool`
`IH`	Zastavení integrace `off` ... integrování povoleno `on` .... integrování pozastaveno	`bool`
`TUNE`	Zahájení ladicího experimentu nebo vynucení přechodu do další fáze experimentu	`bool`
`TBRK`	Ukončení ladicího experimentu	`bool`
`TAFF`	Přijetí výsledků ladicího experimentu `off` ... parametry jsou pouze vypočítány `on` .... parametry jsou dosazeny do řídicího algoritmu	`bool`

`ips` Význam výstupních signálů `pk`, `pti`, `ptd`, `pnd`, `pb` a `pc` `long`

- `0` ..... navržené parametry `k`, `ti`, `td`, `nd`, `b` a `c` PID regulátoru
- `1` ..... momenty procesu: zesílení (`pk`), míra zpoždění soustavy (`pti`), míra délky odezvy soustavy (`ptd`)
- `2` ..... tříparametrový model procesu prvního řádu s dopravním zpožděním: zesílení (`pk`), dopravní zpoždění (`pti`), časová konstanta (`ptd`)
- `3` ..... tříparametrový model procesu druhého řádu s násobnou časovou konstantou a dopravním zpožděním: zesílení (`pk`), dopravní zpoždění (`pti`), časová konstanta (`ptd`)
- `4` ..... odhad mezí intervalu pro manuální doladění zesílení `k` PID regulátoru (`irtype` = 7): horní mez $\mathtt{k}_{hi}$ (`pk`), dolní mez $\mathtt{k}_{lo}$ (`pti`)
- `>99` ... slouží pro diagnostické účely

Výstupy

`mv` Akční zásah regulátoru (manipulated variable)

`dmv` Rychlostní výstup regulátoru (diference)

`de` Regulační odchylka

`SAT` Saturace
- `off` ... lineární zákon řízení
- `on` .... výstup regulátoru je saturován

`TBSY` Příznak probíhajícího ladicího experimentu

`TE` Příznak chyby během ladění
- `off` ... Ladění proběhlo bez chyby
- `on` .... Během ladění se vyskytla chyba

`ite` Kód chyby

Kódy chyb ladění (po experimentu):
- `0` ..... bez chyby
- `1` ..... příliš malá hodnota prahu pro ukončení pulzu
- `2` ..... příliš velká amplituda pulzu
- `3` ..... nebylo dosaženo ustáleného stavu
- `4` ..... příliš malá amplituda pulzu
- `5` ..... nebylo dosaženo ustáleného stavu
- `6` ..... při experimentu došlo k saturaci výstupu regulátoru
- `7` ..... pro vybraný typ regulátoru není podporován automatické nastavování
- `8` ..... nedodržena podmínka monotónnosti procesu
- `9` ..... selhání extrapolace
- `10` .... neočekávané hodnoty momentů (fatální)
- `11` .... ruční přerušení experimentu uživatelem
- `12` .... nesprávný směr řídicí veličiny (změňte parametr `RACT`)
- `100` ... ruční ukončení ladění (varování)

Kódy fází ladění (během experimentu):

- `0` ..... čekání na ustálený stav před začátkem experimentu
- `-1` .... odhad driftu a šumu (parametry `tdg` a `tn`)
- `-2` .... generování obdélníkového pulzu (pulz končí při změně `pv` o hodnotu větší než `dy`)
- `-3` .... hledání vrcholu odezvy
- `-4` .... odhad rychlosti ustalování odezvy

Poznámka k ukončování fází ladění:

- `TUNE` .. Náběžná hrana vstupu `TUNE` během fází `-2`, `-3` and `-4` způsobuje předčasné ukončení dané fáze a přechod do fáze následující (nebo ukončení experimentu ve fázi `-4`).

`trem`	Odhad času do ukončení experimentu [s]		`double`
`pk`	Navržené zesílení regulátoru K (`ips` = `0`)		`double`
`pti`	Navržená integrační časová konstanta regulátoru T_i (`ips` = `0`)		`double`
`ptd`	Navržená derivační časová konstanta regulátoru T_d (`ips` = `0`)		`double`
`pnd`	Navržený parametr filtru derivační složky N (`ips` = `0`)		`double`
`pb`	Navržený váhový faktor pro proporcionální složku (`ips` = `0`)	`double`	
`pc`	Navržený váhový faktor pro derivační složku (`ips` = `0`)	`double`	

Parametry

`irtype`	Typ regulátoru `1` ..... D `3` ..... ID `5` ..... PD `7` ..... PID `2` ..... I `4` ..... P `6` ..... PI	⊙`6`	`long`

`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
			`bool`

`k`	Zesílení regulátoru K	⊙`1.0`	`double`
`ti`	Integrační časová konstanta T_i	⊙`4.0`	`double`
`td`	Derivační časová konstanta T_d	⊙`1.0`	`double`
`nd`	Parametr filtru derivační složky N	⊙`10.0`	`double`
`b`	Váhový faktor pro proporcionální složku	⊙`1.0`	`double`
`c`	Váhový faktor pro derivační složku		`double`
`tt`	Časová konstanta vysledování. Pro regulátory bez integrační složky nemá žádný význam.	⊙`1.0`	`double`
`hilim`	Horní mez akčního zásahu regulátoru	⊙`1.0`	`double`
`lolim`	Dolní mez akčního zásahu regulátoru	⊙`-1.0`	`double`
`dz`	Pásmo necitlivosti		`double`
`icotype`	Typ výstupu regulátoru `1` ..... analogový výstup `2` ..... šířkově modulovaný výstup (PWM) `3` ..... krokový regulátor s polohovou zpětnou vazbou (SCU) `4` ..... krokový regulátor bez polohové zpětné vazby (SCUV)	⊙`1`	`long`
`ittype`	Požadovaný typ regulátoru pro návrh `6` ..... PI regulátor `7` ..... PID regulátor	⊙`6`	`long`

`iainf` Druh apriorní informace ⊙

`1` ..... proces bez integrátoru
`2` ..... proces s integračním chováním

`DGC` Kompenzace gradientu trendu ⊙`on` `bool`

`off` ... zakázáno `on` .... povoleno

`tdg` Doba odhadu gradientu trendu [s] ⊙`60.0` `double`

`tn` Doba odhadování šumu [s] ⊙`5.0` `double`

`amp` Amplituda pulzu ⊙`0.5` `double`

`dy` Práh pro ukončení pulsu (absolutní změna od ustálené hodnoty `pv`) `double`
↓`0.0` ⊙`0.1`

`ispeed` Požadovaná rychlost uzavřené smyčky ⊙`2` `long`

`1` ..... požadována pomalá uzavřená smyčka
`2` ..... požadována středně rychlá uzavřená smyčka
`3` ..... požadována rychlá uzavřená smyčka

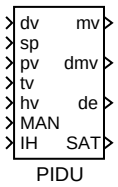
`ipid` Forma PID regulátoru ⊙`1` `long`

`1` ..... paralelní realizace
`2` ..... sériová realizace

PIDU – PID regulátor

Symbol bloku

Licence: STANDARD



Popis funkce

Blok PIDU je základní blok pro vytvoření úplného regulátoru PID (P, I, PI, PD, PID, PI+S). V nejjednodušším případě může pracovat zcela samostatně a plnit standardní funkci PID regulátoru se dvěma stupni volnosti v automatickém (MAN = off) nebo manuálním režimu (MAN = on).

V automatickém režimu (MAN = off) realizuje blok PIDU řídicí zákon PID regulátoru se dvěma stupni volnosti ve tvaru

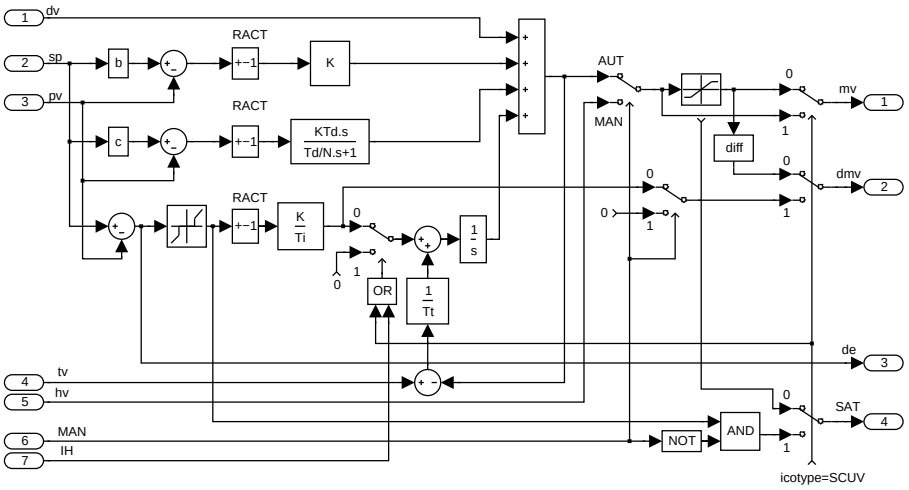
$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s}\left[W(s) - Y(s)\right] + \frac{T_d s}{\frac{T_d}{N}s + 1}\left[cW(s) - Y(s)\right] \right\} + Z(s)$$

kde $U(s)$ je Laplaceova transformace řídicí veličiny mv, $W(s)$ je Laplaceova transformace požadované hodnoty sp, $Y(s)$ je Laplaceova transformace regulované veličiny pv, $Z(s)$ je Laplaceova transformace dopředné vazby dv a K, T_i, T_d, N, b, c jsou parametry regulátoru. Znaménko pravé strany závisí na parametru RACT. Rozsah řídicí veličiny mv (polohového výstupu regulátoru) je omezen parametry hilim, lolim. Parametr dz udává pásmo necitlivosti v integrační složce regulátoru. Navíc integrační složka může být vypnuta a zafixována na své aktuální hodnotě vstupem IH (IH = on). Pro správnou funkci regulátoru je nutné propojit výstup regulátoru mv se vstupem tv a správně nastavit časovou konstantu vysledovávání tt (doporučená hodnota je $\texttt{tt} \approx \sqrt{T_i T_d}$, pro PI regulátor $\texttt{tt} \approx 2 \cdot \sqrt{T_i}$). Tím bude zaručen bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správná funkce regulátoru při saturaci výstupu mv (tzv. antiwindup). Přídavné výstupy dmv, de a SAT poskytují po řadě rychlostní výstup regulátoru (difference mv), regulační odchylku a příznak saturace výstupu regulátoru mv.

Jestliže je blok PIDU propojen s blokem SCUV (za účelem realizace krokového regulátoru bez polohové zpětné vazby), potom parametr icotype musí být nastaven na hodnotu 4 a význam výstupů mv, dmv a SAT je v tomto případě pozměněn: výstup mv je roven součtu P a D složky regulátoru, zatímco výstup dmv poskytuje diferenci jeho I složky a výstup SAT nese informaci pro blok SCUV, zda je regulační odchylka de v automatickém režimu menší než pásmo necitlivosti dz. Pro případ propojení bloků PIDU

a SCUV se navíc doporučuje volit váhový koeficient požadované hodnoty pro derivační složku `c` rovný nule.

V manuálním režimu (`MAN` = `on`) je vstup `hv` kopírován na výstup `mv`, pokud nenaráží na horní či dolní omezení výstupu regulátoru. Celková regulační funkce bloku `PIDU` je zřejmá z následujícího obrázku.



Vstupy

`dv`	Proměnná dopředné vazby	`double`
`sp`	Požadovaná hodnota (setpoint)	`double`
`pv`	Řízená veličina	`double`
`tv`	Veličina pro vysledování	`double`
`hv`	Hodnota výstupu v manuálním režimu	`double`
`MAN`	Manuální nebo automatický režim	`bool`
	`off` ... automatický režim	
	`on` .... manuální režim	
`IH`	Zastavení integrace	`bool`
	`off` ... integrování povoleno	
	`on` .... integrování pozastaveno	

Výstupy

`mv`	Akční zásah regulátoru (manipulated variable)	`double`
`dmv`	Rychlostní výstup regulátoru (diference)	`double`
`de`	Regulační odchylka	`double`
`SAT`	Saturace	`bool`
	`off` ... lineární zákon řízení	
	`on` .... výstup regulátoru je saturován	

Parametry

`irtype`	Typ regulátoru `1` ..... D `4` ..... P `7` ..... PID `2` ..... I `5` ..... PD `3` ..... ID `6` ..... PI	⊙`6`	`long`

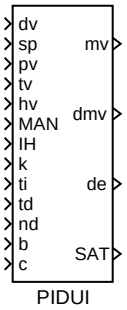
`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
			`bool`

`k`	Zesílení regulátoru K	⊙`1.0`	`double`
`ti`	Integrační časová konstanta T_i	⊙`4.0`	`double`
`td`	Derivační časová konstanta T_d	⊙`1.0`	`double`
`nd`	Parametr N filtru derivační složky	⊙`10.0`	`double`
`b`	Váhový faktor pro proporcionální složku	⊙`1.0`	`double`
`c`	Váhový faktor pro derivační složku		`double`
`tt`	Časová konstanta vysledování. Pro regulátory bez integrační složky nemá žádný význam.	⊙`1.0`	`double`
`hilim`	Horní mez akčního zásahu regulátoru	⊙`1.0`	`double`
`lolim`	Dolní mez akčního zásahu regulátoru	⊙`-1.0`	`double`
`dz`	Pásmo necitlivosti		`double`
`icotype`	Typ výstupu regulátoru `1` ..... analogový výstup `2` ..... šířkově modulovaný výstup (`PWM`) `3` ..... krokový regulátor s polohovou zpětnou vazbou (`SCU`) `4` ..... krokový regulátor bez polohové zpětné vazby (`SCUV`)	⊙`1`	`long`

PIDUI – PID regulátor s parametry na vstupech

Symbol bloku

Licence: ADVANCED



Popis funkce

Regulační funkce bloku PIDUI je přesně shodná s blokem PIDU. Jediný rozdíl spočívá v tom, že základní parametry PID algoritmu jsou vyvedeny na vstupy. V důsledku toho je lze pohodlně měnit v závislosti na výstupech jiných bloků. Tímto způsobem lze realizovat speciální adaptivní PID regulátory.

Vstupy

dv	Proměnná dopředné vazby	double
sp	Požadovaná hodnota (setpoint)	double
pv	Řízená veličina	double
tv	Veličina pro vysledování	double
hv	Hodnota výstupu v manuálním režimu	double
MAN	Manuální nebo automatický režim off ... automatický režim on .... manuální režim	bool
IH	Zastavení integrace off ... integrování povoleno on .... integrování pozastaveno	bool
k	Zesílení regulátoru K	double
ti	Integrační časová konstanta T_i	double
td	Derivační časová konstanta T_d	double
nd	Parametr N filtru derivační složky	double
b	Váhový faktor pro proporcionální složku	double
c	Váhový faktor pro derivační složku	double

Výstupy

`mv`	Akční zásah regulátoru (manipulated variable)	`double`
`dmv`	Rychlostní výstup regulátoru (diference)	`double`
`de`	Regulační odchylka	`double`
`SAT`	Saturace	`bool`
	`off` ... lineární zákon řízení	
	`on` .... výstup regulátoru je saturován	

Parametry

`irtype`	Typ regulátoru ⊙`6`	`long`
	`1` ..... D `4` ..... P `7` ..... PID	
	`2` ..... I `5` ..... PD	
	`3` ..... ID `6` ..... PI	
`RACT`	Převrácené působení výstupu regulátoru	`bool`
	`off` ... vyšší `mv` → vyšší `pv`	
	`on` .... vyšší `mv` → nižší `pv`	
`tt`	Časová konstanta vysledování. Pro regulátory bez integrační složky nemá žádný význam. ⊙`1.0`	`double`
`hilim`	Horní mez akčního zásahu regulátoru ⊙`1.0`	`double`
`lolim`	Dolní mez akčního zásahu regulátoru ⊙`-1.0`	`double`
`dz`	Pásmo necitlivosti	`double`
`icotype`	Typ výstupu regulátoru ⊙`1`	`long`
	`1` ..... analogový výstup	
	`2` ..... šířkově modulovaný výstup (`PWM`)	
	`3` ..... krokový regulátor s polohovou zpětnou vazbou (`SCU`)	
	`4` ..... krokový regulátor bez polohové zpětné vazby (`SCUV`)	

POUT – Pulzní výstup

Symbol bloku

Licence: STANDARD

Popis funkce

Blok POUT tvaruje vstupní pulzy U takovým způsobem, že délka výstupního pulzu Y je alespoň dtime sekund a prodleva mezi dvěma sousedními výstupními pulzy je minimálně btime sekund. Vstupní pulz, který přijde po sestupné hraně výstupního signálu dříve, než uplyne čas btime, nezpůsobí žádnou odezvu na výstupu Y.

Vstup

U	Logický vstupní signál		bool

Výstup

Y	Logický výstupní signál		bool

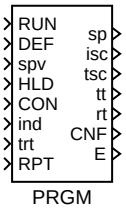
Parametry

dtime	Minimální trvání výstupního pulzu [s]	⊙1.0	double
btime	Minimální prodleva mezi sousedními výstupními pulzy [s]	⊙1.0	double

PRGM – Programátor

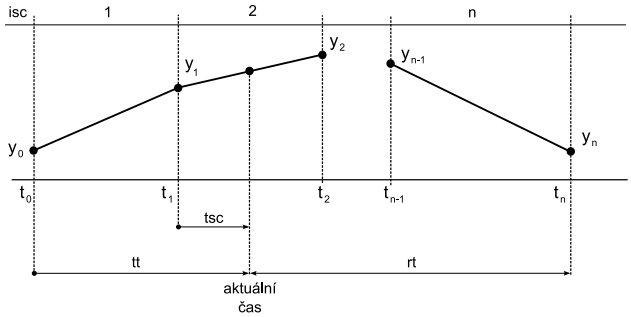
Symbol bloku

Licence: STANDARD



Popis funkce

Blok PRGM je určen pro generování časových funkcí (programů) složených z n lineárních částí definovaných $(n+1)$ rozměrnými vektory $tm = [t_0, ..., t_n]$ času a požadovaných hodnot $y = [y_0, ..., y_n]$ (generovaná křivka je spojitá po částech lineární, viz. obrázek). Nejčastěji je používán pro generování požadované hodnoty regulátoru. Generování programu je spuštěno vstupem RUN = on; přechod zpět na RUN = off vrací stav programátoru do základního stavu. Vstup DEF nastaví sp na hodnotu spv a po vymizení hodnoty DEF = on se pokračuje přejetím po rampě na nejbližší následující uzel, čas přitom není narušen. Vstup HLD = on zmrazí výstupní hodnotu sp a všechny výstupní časy (tsc, tt, rt), po vymizení hodnoty HLD = on se pokračuje z okamžiku zmrznutí dále podle programu. Je-li při přechodu HLD on→off nastaven vstup CON = on, nepokračuje se od okamžiku zmrazení, ale najede se do uzlového bodu s indexem ind po rampě za čas trt. Index uzlového bodu ind musí být rovný nebo větší než aktuálně prováděný sektor (v okamžiku HLD on→off). Je-li RPT = on, potom se program generuje opakovaně.



Vstupy

RUN	Povolení generování časové funkce programu	bool
DEF	Inicializace sp na hodnotu spv	bool
spv	Inicializační hodnota	double

HLD	Zmrazení výstupu a výstupních časů	bool
CON	Pokračování od uzlového bodu ind	bool
ind	Index uzlového bodu pro pokračování	long
trt	Čas pro dosažení požadovaného uzlu ind	double
RPT	Příznak opakování generování časové funkce	bool

Výstupy

sp	Požadovaná hodnota (hodnota časové funkce v daném čase)	double
isc	Aktuální sektor funkce	long
tsc	Čas od začátku sektoru	double
tt	Čas od startu generování časové funkce	double
rt	Čas do konce programu	double
CNF	Příznak sledování nakonfigurované křivky	bool
E	Chyba, časy uzlů nejsou seřazeny vzestupně	bool

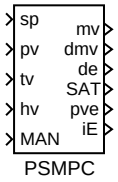
Parametry

n	Počet sektorů	↓1 ↑10000000 ⊙2	long
tmunits	Jednotky pro zadávání časů 1 ..... sekundy 2 ..... minuty 3 ..... hodiny	⊙1	long
tm	(n + 1)-rozměrný vektor vzestupně uspořádaných časů	⊙[0 1 2]	double
y	(n + 1)-rozměrný vektor hodnot časové funkce	⊙[0 1 0]	double

PSMPC – Prediktivní „pulse-step“ regulátor

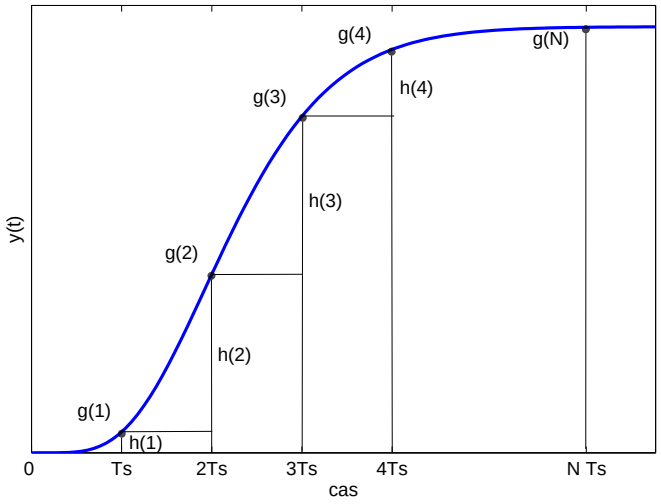
Symbol bloku

Licence: ADVANCED



Popis funkce

Funkční blok PSMPC (Pulse Step Model Predictive Control) je určen pro realizaci vysoce kvalitních regulátorů pro obtížně regulovatelné lineární časově invariantní soustavy s omezením akční veličiny (např. soustavy s dopravním zpožděním nebo s neminimální fází). Zvlášť výhodný je pro případy, kdy je požadován velmi rychlý přechod z jedné hodnoty regulované veličiny na druhou bez překmitu. Regulátor PSMPC však může být obecně použit všude tam, kde je běžně nasazován standardní PID regulátor a kde žádáme vysokou kvalitu regulace.



PSMPC je prediktivní regulátor s explicitně zadaným intervalovým omezením akční veličiny. Pro účely predikce je použit model ve tvaru diskrétní přechodové charakteristiky $g(j), j = 1, ..., N$. Na obrázku výše je naznačen způsob, jakým lze tuto posloupnost získat ze spojité přechodové charakteristiky. Poznamenejme, že N musí být zvoleno dostatečně velké, aby přechodová charakteristika byla popsána až do ustáleného stavu ($NT_S > t_{95}$, kde T_S je perioda vzorkování regulátoru a t_{95} je doba ustálení na 95 % konečné hodnoty).

Pro systémy s monotónní přechodovou charakteristikou je alternativně možné použít momentový množinový model [5] a popsat systém pouze třemi charakteristickými čísly κ, μ a σ^2, které je možno určit z jednoduchého pulzního experimentu. Řízený systém pak aproximujeme buď přenosem prvého řádu s dopravním zpožděním

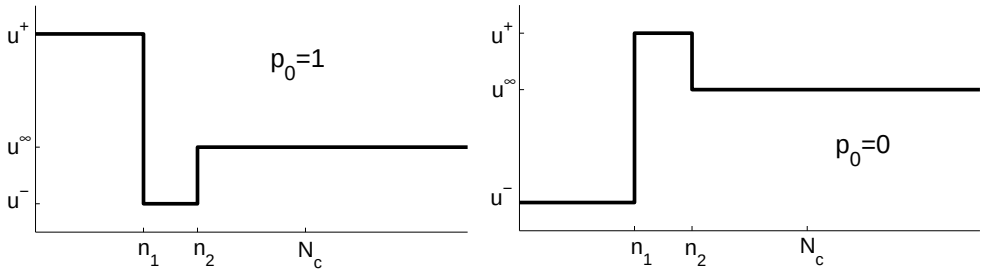
$$F_{FOPDT}(s) = \frac{K}{\tau s + 1} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = \tau + D, \quad \sigma^2 = \tau^2 \tag{7.1}$$

nebo přenosem druhého řádu s dopravním zpožděním

$$F_{SOPDT}(s) = \frac{K}{(\tau s + 1)^2} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = 2\tau + D, \quad \sigma^2 = 2\tau^2 \tag{7.2}$$

se stejnými charakteristickými čísly. Typ aproximace se zadává parametrem `imtype`.

Pro zjednodušení on-line optimalizace v otevřené smyčce je množina přípustných posloupností řízení omezena pouze na posloupnosti ve tvaru "pulz-skok"zobrazené na obrázku níže.



Poznamenejme, že každá taková posloupnost je jednoznačně určena jen třemi čísly $n_1, n_2 \in \{0, \ldots, N_C\}$ a $u^\infty \in \langle u^-, u^+ \rangle$, kde $N_C \in \{0, 1, \ldots\}$ je horizont řízení a u^-, u^+ označují po řadě zadanou dolní a horní mez akční veličiny regulátoru. On-line optimalizace (vzhledem k n_1, n_2 a u^∞) spočívá v minimalizaci kritéria

$$I = \sum_{i=N_1}^{N_2} \hat{e}(k+i|k)^2 + \lambda \sum_{i=0}^{N_C} \Delta\hat{u}(k+i\,|k)^2 \to \min, \tag{7.3}$$

kde $\hat{e}(k+i|k)$ je v kroku k predikovaná regulační odchylka na intervalu predikce $i \in \{N_1, N_2\}$, $\Delta\hat{u}(k+i|k)$ jsou diference řídicího signálu na intervalu $i \in \{0, N_C\}$ a λ je koeficient penalizace změn akční veličiny. Pro nalezení optima úlohy (7.3) je použita kombinace metody nejmenších čtverců a hrubé síly. Hodnota u^∞ je určena pro všechny přípustné kombinace p_0, n_1 a n_2 a následně je z nich vybrána optimální řídicí sekvence pro řízení v otevřené smyčce. Ve skutečnosti je však vždy aplikován pouze prvý krok této řídicí sekvence a v další vzorkovací periodě je celý optimalizační proces zopakován. Tím se řídicí strategie mění na zpětnovazební řízení.

Parametry prediktivního regulátoru, kromě modelu řízené soustavy a omezení jejího vstupu, jsou horizont řízení N_C, horizont predikce N_1, N_2 a koeficient λ. Pouze poslední uvedený parametr je určen pro ruční doladění kvality regulace při rutinním uvádění do provozu. V případě použití modelu soustavy ve tvaru přenosu (7.1) nebo (7.2) jsou

parametry N_1, N_2 zvoleny automaticky na základě charakteristických čísel μ, σ^2. Regulátor potom může být efektivně laděn „ručně“ pouze seřizováním charakteristických čísel procesu κ, μ, σ^2.

Varování

Při použití bloku `PSMPC` pro simulaci v systému Matlab/Simulink je třeba zajistit, aby parametr `nsr` byl dostatečně velký, tak aby jím definovaný buffer pojmul interně vygenerovanou přechodovou charakteristiku určenou z FOPDT nebo SOPDT modelu. V opačném případě dojde k havárii systému Matlab/Simulink.

Vstupy

`sp`	Požadovaná hodnota (setpoint)	`double`
`pv`	Řízená veličina	`double`
`tv`	Veličina pro vysledování (použitý řídicí signál)	`double`
`hv`	Hodnota výstupu v manuálním režimu	`double`
`MAN`	Manuální nebo automatický režim `off` ... automatický režim `on` .... manuální režim	`bool`

Výstupy

`mv`	Akční zásah regulátoru (manipulated variable)	`double`
`dmv`	Rychlostní výstup regulátoru (diference)	`double`
`de`	Regulační odchylka	`double`
`SAT`	Saturace `off` ... lineární zákon řízení `on` .... výstup regulátoru je saturován	`bool`
`pve`	Predikovaná hodnota regulované veličiny na základě zadaného modelu	`double`
`iE`	Kód chyby `0` ..... bez chyby `1` ..... nesprávný FOPDT model `2` ..... nesprávný SOPDT model `3` ..... chyba v zadání přechodové charakteristiky	`long`

Parametry

`nc`	Délka horizontu řízení (N_C)	⊙`5`	`long`
`np1`	Začátek koincidenčního intervalu (N_1)	⊙`1`	`long`
`np2`	Konec koincidenčního intervalu (N_2)	⊙`10`	`long`
`lambda`	Koeficient penalizace změn řízení (λ)	⊙`0.05`	`double`
`umax`	Horní mez akčního zásahu regulátoru (u^+)	⊙`1.0`	`double`
`umin`	Dolní mez akčního zásahu regulátoru (u^-)	⊙`-1.0`	`double`

`imtype`	Typ modelu řízené soustavy `1` ..... model prvního řádu `2` ..... model druhého řádu `3` ..... přechodová charakteristika	⊙`3`	`long`
`kappa`	Statické zesílení (κ)	⊙`1.0`	`double`
`mu`	Míra zpoždění soustavy (μ)	⊙`20.0`	`double`
`sigma`	Míra délky odezvy soustavy ($\sqrt{\sigma^2}$)	⊙`10.0`	`double`
`nsr`	Délka diskrétní přechodové charakteristiky (N), pozor na varování uvedené výše	↓`10` ↑`10000000` ⊙`11`	`long`
`sr`	Diskrétní přechodová charakteristika ($[g(1), \ldots, g(N)]$)	⊙`[0 0.2642 0.5940 0.8009 0.9084 0.9596 0.9826 0.9927 0.9970 0.9988 0.9995]`	`double`

PWM – Blok šířkové modulace

Symbol bloku

Licence: STANDARD

Popis funkce

Blok PWM provádí pulzně šířkovou modulaci vstupního signálu z intervalu od -1 do +1. Užitím tohoto bloku je možné realizovat proporcionální akční veličinu i u akčních členů s jedním (např. topení zapnuto/vypnuto) nebo dvěma (např. topení zapnuto/vypnuto a chlazení zap./vyp.) binárními vstupy. Šířka L výstupního pulzu je určena vztahem:

$$L = \texttt{pertm} * |\texttt{u}| \,,$$

kde pertm je perioda modulace. Je-li u > 0 (u < 0), pulz je generován na výstupu UP (DN). Z praktických důvodů je však délka generovaného pulzu dále upravována podle zadaných parametrů bloku. Faktor asymetrie asyfac definuje poměr mezi délkou negativního pulzu DN a délkou pozitivního pulzu UP. Modifikované délky se počítají podle vztahů:

$$\text{jestliže} \quad \texttt{u} > 0 \quad \text{potom} \quad L(\texttt{UP}) := \begin{cases} L & \text{pro} \quad \texttt{asyfac} \leq 1.0 \\ L/\texttt{asyfac} & \text{pro} \quad \texttt{asyfac} > 1.0 \end{cases}$$

$$\text{jestliže} \quad \texttt{u} < 0 \quad \text{potom} \quad L(\texttt{DN}) := \begin{cases} L * \texttt{asyfac} & \text{pro} \quad \texttt{asyfac} \leq 1.0 \\ L & \text{pro} \quad \texttt{asyfac} > 1.0 \end{cases}$$

které pro libovolnou hodnotu asyfac>0 zajišťují, že maximální délka generovaných pulzů je rovna pertm. Dále, jestliže vypočtená délka pulzu je menší než dtime, potom je výsledná délka nastavena na nulu. Jestliže se vypočtená délka pulzu liší od pertm méně než btime, potom je výsledná délka nastavena na pertm. Jestliže kladný pulz UP je následovaný záporným pulzem DN nebo obráceně, potom pozdější pulz je v případě potřeby posunut tak, že vzdálenost mezi těmito dvěma pulzy je alespoň offtime. Jestliže SYNCH = on, potom změna vstupu u způsobí okamžitý přepočet délky výstupního pulzu za předpokladu, že není splněna synchronizační podmínka mezi začátkem periody modulace a okamžikem změny vstupu u.

Vstup

u	Analogový vstupní signál	double

Výstupy

`UP`	Signál UP (nahoru, více)		`bool`
`DN`	Signál DN (dolů, méně)		`bool`

Parametry

`pertm`	Perioda šířkové modulace [s]	⊙`10.0`	`double`
`dtime`	Minimální trvání výstupního pulzu [s]	⊙`0.1`	`double`
`btime`	Minimální prodleva mezi pulzy [s]	⊙`0.1`	`double`
`offtime`	Minimální prodleva mezi pulzy opačné polarity [s]	⊙`1.0`	`double`
`asyfac`	Faktor asymetrie	⊙`1.0`	`double`
`SYNCH`	Synchronizační příznak pro začátek periody `off` ... synchronizace vypnuta `on` .... synchronizace zapnuta		`bool`

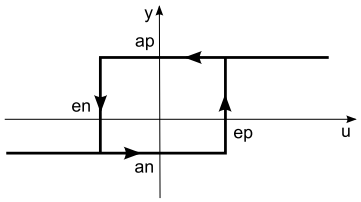
RLY – Relé s hysterezí

Symbol bloku

Licence: STANDARD

Popis funkce

Blok RLY transformuje vstupní analogový signál u na výstupní analogový signál y podle níže uvedeného obrázku.



Vstup

u	Analogový vstupní signál	double

Výstup

y	Analogový výstupní signál	double

Parametry

ep	Hodnota u > ep způsobí y = ap („Zapnuto“)	⊙1.0	double
en	Hodnota u < en způsobí y = an („Vypnuto“)	⊙-1.0	double
ap	Výstup ve stavu „Zapnuto“	⊙1.0	double
an	Výstup ve stavu „Vypnuto“	⊙-1.0	double
y0	Počáteční hodnota výstupu y po spuštění		double

SAT – Saturace výstupu s proměnnými mezemi

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SAT kopíruje vstup u na výstupu y, pokud pro vstupní veličinu platí lolim ≤ u a u ≤ hilim, kde lolim a hilim jsou stavové proměnné bloku. Je-li u < lolim (resp. u > hilim), potom y = lolim (y = hilim). Horní a dolní limit jsou buď pevné hodnoty dané po řadě parametry bloku hilim0 a lolim0 (případ HLD = on) nebo jsou řízeny vstupy hi a lo (HLD = off). Maximální rychlost změny aktivních mezí hilim a lolim je dána časovými konstantami tp a tn. Parametr tp určuje maximální kladnou strmost a tn maximální zápornou strmost změn hilim a lolim. Omezení strmosti změn mezí je aktivní i v případě, že hodnoty mezí měníme ručně (HLD = on) pomocí parametrů hilim0 a lolim0. Pro možnost okamžitých změn saturačních mezí je potřeba nastavit tp = 0 a tn = 0. Výstupy HL a LL signalizují po řadě horní a dolní saturaci.

Pokud je to potřeba, parametry hilim0 a lolim0 jsou použity jako počáteční hodnoty pro saturační meze řízené vstupními signály.

Vstupy

u	Analogový vstupní signál	double
hi	Horní saturační mez pro případ HLD = off	double
lo	Dolní saturační mez pro případ HLD = off	double

Výstupy

y	Analogový výstupní signál	double
HL	Příznak saturace na horní mezi	bool
LL	Příznak saturace na dolní mezi	bool

Parametry

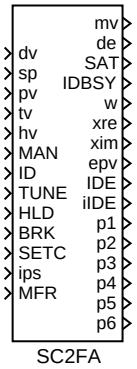
tp	Časová konstanta rychlosti změn aktivních hodnot mezí v kladném směru	⊙1.0	double
tn	Časová konstanta rychlosti změn aktivních hodnot mezí v záporném směru	⊙1.0	double
hilim0	Horní omezení výstupu (platné pro HLD = on)	⊙1.0	double
lolim0	Dolní omezení výstupu (platné pro HLD = on)	⊙-1.0	double

`HLD` Pevné saturační meze ⊙`on` `bool`
`off` ... proměnné meze `on` .... pevné meze

SC2FA – Stavový regulátor systému 2. řádu s autotunerem

Symbol bloku

Licence: AUTOTUNING



Popis funkce

Funkční blok SC2FA realizuje stavový regulátor pro systém druhého řádu (7.4) s frekvenčním autotunerem. Je vhodný především pro aktivní řízení (zatlumení) kmitavých systémů s velmi slabým tlumením ($\xi < 0{,}1$). Může však být použit též jako samonastavující se regulátor pro libovolný systém, který lze s dostatečnou přesností popsat přenosem ve tvaru

$$F(s) = \frac{b_1 s + b_0}{s^2 + 2\xi\Omega s + \Omega^2}, \tag{7.4}$$

kde $\Omega > 0$ je přirozená (netlumená) frekvence, ξ, $0 < \xi < 1$, je koeficient tlumení a b_1, b_0 jsou libovolná reálná čísla. Blok pracuje ve dvou režimech, v režimu Identifikace a návrhu a v režimu Regulace.

Režim „Identifikace a návrhu“ ze zapíná nastavením binárního vstupu `ID` = `on`. Vlastní proces identifikace a návrhu se spouští náběžnou hranou vstupu `RUN`. Na výstupu bloku `mv` se poté objeví budící harmonický signál se stejnoměrnou složkou `ubias`, amplitudou `uamp` a frekvencí ω postupně probíhající interval $\langle$`wb`, `wf`$\rangle$. Aktuální frekvence ω je přitom kopírována na výstup `w`. Rychlost změny (rozmítání) frekvence je dána parametrem `cp`, který udává relativní zmenšení počáteční periody $T_b = \frac{2\pi}{\texttt{wb}}$ budící sinusovky za čas T_b, tedy

$$c_p = \frac{\texttt{wb}}{\omega(T_b)} = \frac{\texttt{wb}}{\texttt{wb}e^{\gamma T_b}} = e^{-\gamma T_b}. \tag{7.5}$$

Hodnota parametru `cp` se obvykle pohybuje v intervalu `cp` $\in \langle 0{,}95; 1)$. Čím menší je koeficient tlumení řízeného systému, tím více se musí `cp` blížit k jedné.

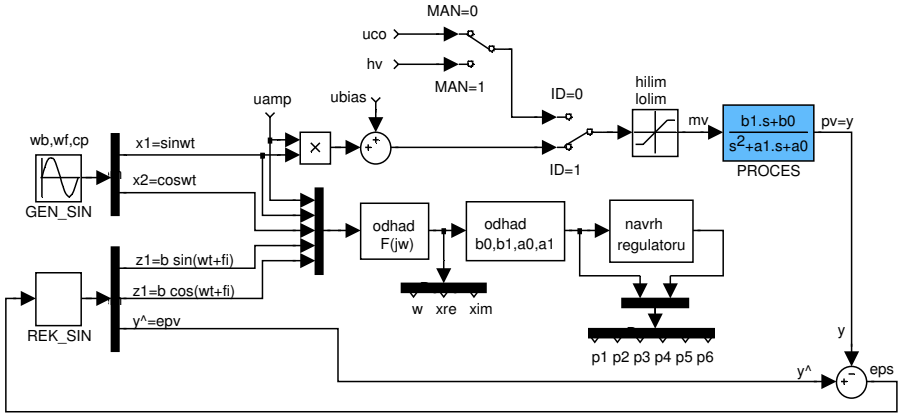
Identifikace systému se spouští náběžnou hranou vstupu RUN zároveň s generátorem budícího signálu se startovací frekvencí $\omega =$ wb. Po uplynutí stime se startuje výpočet odhadu aktuálního bodu frekvenční charakteristiky. Jeho reálná a imaginární část se průběžně kopíruje po řadě na výstupy xre a xim. Je-li parametr bloku MANF nastaven na 0, potom se v procesu identifikace dvakrát zastaví rozmítání frekvence na dobu stime a to v okamžicích, kdy jsou poprvé dosaženy body s fázovým zpožděním ph1 a ph2. Přednastavené hodnoty parametrů ph1 a ph2 jsou po řadě $-60°$ a $-120°$ a mohou být změněny na libovolné hodnoty v intervalu $(-360°, 0°)$, přičemž ph1 > ph2. Po uplynutí stime sekund při zastavení ve fázi ph1, resp. ph2 se spočítá průměr posledních iavg naměřených bodů (průměrováním tedy získáme odhad příslušného bodu frekvenční charakteristiky) pro následný výpočet parametrického modelu ve tvaru (7.4). Je-li MANF = on, potom je možné provést „navzorkování“ dvou bodů frekvenční charakteristiky ručně pomocí vstupu HLD. Vstup HLD = on zastaví rozmítání frekvence a opětovné nastavení HLD = off vede k jeho pokračování. Ostatní funkce jsou identické.

V případě potřeby je možné proces identifikace přerušit vstupem BRK = on. Jsou-li již v tomto okamžiku oba dva body pro parametrickou identifikaci určeny, pokračuje se v návrhu regulátoru normálním způsobem. V opačném případě je proces ukončen bez návrhu regulátoru a výstup IDE = on signalizuje chybu.

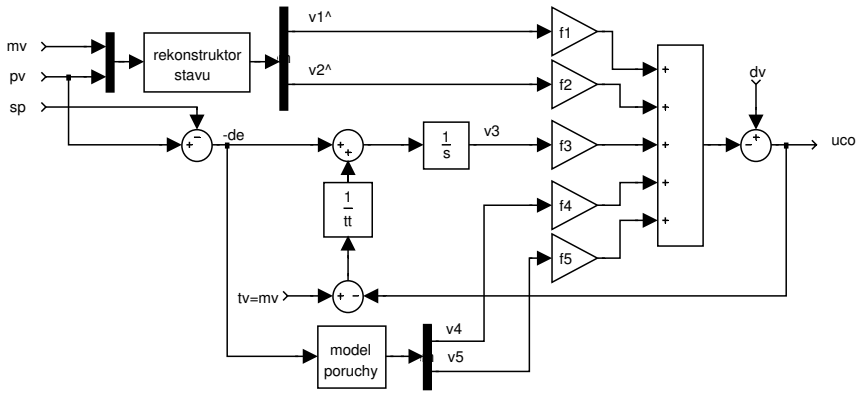
Během vlastní „identifikace a návrhu“ je výstup IDBSY nastaven na 1. Po skončení je shozen na 0. Při bezchybném návrhu regulátoru je výstup IDE = off a výstup iIDE signalizuje jednotlivé fáze identifikačního experimentu. Přibližování k prvnímu bodu je iIDE $= -1$, zastavení v prvním bodě iIDE $= 1$, přibližování k druhému bodu je iIDE $= -2$, zastavení v druhém bodě iIDE $= 2$ a poslední fáze po zastavení v druhém bodě je iIDE $= -3$. Jestliže identifikace skončí s chybou, pak je IDE = on a číslo na výstupu iIDE specifikuje příslušnou chybu.

Vypočtené parametry stavového regulátoru jsou instalovány okamžitě do algoritmu řízení, jestliže vstup SETC je trvale nastaven na on. V opačném případě se provede nastavení parametrů až po ukončení návrhu na náběžnou hranu vstupu SETC. Výsledky parametrické identifikace a návrhu stavového regulátoru je možné získat na výstupech p1, p2, ..., p6 vhodným nastavením vstupu ips. Náběžná hrana na vstupu MFR po skončení identifikace (IDBSY = off) odstartuje generování frekvenční charakteristiky získaného parametrického modelu na výstupech w, xre, xim. Takto je možno porovnat její průběh s „přímo odměřenou“ frekvenční charakteristikou systému.

V režimu „regulace“ (binární vstup ID = off) může regulátor pracovat v manuálním módu (MAN = on) nebo v automatickém módu. Jestliže je blok regulátoru spuštěn (při studeném startu) s hodnotou vstupu ID = off, potom se předpokládá, že zadané parametry bloku mb0, mb1, ma0 a ma1 odpovídají dříve určeným koeficientům b_0, b_1, a_0 a a_1 přenosu řízeného systému a automaticky proběhne návrh stavového regulátoru. Je-li regulátor navíc v automatickém módu a SETC = on, potom zákon řízení od počátku využívá nově navržené parametry. Takto lze vypustit identifikaci při opakovaném spuštění bloku.



Na výše uvedeném obrázku je zjednodušené vnitřní schéma samonastavujícího se stavového regulátoru, část frekvenční identifikace. Na spodním obrázku je stavová zpětná vazba s rekonstruktorem stavu a ošetřením unášení integrační složky. Na obrázku není znázorněna skutečnost, že blok návrhu regulátoru v části frekvenční identifikace automaticky nastaví parametry rekonstruktoru stavu a koeficienty `f1`, `f2`, ..., `f5` stavové zpětné vazby.



Model řízeného systému je brán jako systém 2. řádu s přenosem ve tvaru (7.4). Jednoduchými úpravami lze dojít k přenosům

$$F(s) = \frac{(b_1 s + b_0)}{s^2 + a_1 s + a_0} \tag{7.6}$$

a

$$F(s) = \frac{K_0 \Omega^2 (\tau s + 1)}{s^2 + 2\xi\Omega s + \Omega^2}. \tag{7.7}$$

Parametry těchto přenosů je možné po skončení identifikace přečíst z výstupů `p1`,...,`p6`. Význam těchto výstupů se mění při změně vstupu `ips`, avšak pouze pokud neběží identifikace (tedy `IDBSY` = `off`).

Vstupy

`dv`	Proměnná dopředné vazby	`double`
`sp`	Požadovaná hodnota (setpoint)	`double`
`pv`	Řízená veličina	`double`
`tv`	Veličina pro vysledování	`double`
`hv`	Hodnota výstupu v manuálním režimu	`double`
`MAN`	Manuální nebo automatický režim `off` ... automatický režim `on` .... manuální režim	`bool`
`ID`	Režim identifikace nebo regulace `off` ... režim regulátoru `on` .... režim identifikace a návrhu	`bool`
`TUNE`	Zahájení ladicího experimentu, start generátor harmonického budicího signálu (`off`→`on`)	`bool`
`HLD`	Zastavení rozmítání frekvence	`bool`
`BRK`	Signál pro přerušení identifikačního experimentu	`bool`
`SETC`	Přijmutí a nastavení parametrů regulátoru `off` ... parametry jsou pouze vypočítány `on` .... parametry jsou přijaty ihned po vypočtení `off`→`on` jednorázové přijetí vypočtených parametrů	`bool`
`ips`	Význam výstupních signálů `0` ..... Dva body frekvenční charakteristiky v komplexní rovině `p1` ... frekvence 1. změřeného bodu v rad/s `p2` ... reálná část 1. bodu `p3` ... imaginární část 1. bodu `p4` ... frekvence 2. změřeného bodu v rad/s `p5` ... reálná část 2. bodu `p6` ... imaginární část 2. bodu `1` ..... Model druhého řádu ve tvaru (7.6) `p1` ... parametr b_1 `p2` ... parametr b_0 `p3` ... parametr a_1 `p4` ... parametr a_0 `2` ..... Model druhého řádu ve tvaru (7.7) `p1` ... parametr K_0 `p2` ... parametr τ `p3` ... parametr Ω v rad/s `p4` ... parametr ξ `p5` ... parametr Ω v Hz `p6` ... rezonanční frekvence modelu v Hz `3` ..... Parametry stavové zpětné vazby `p1` ... parametr f_1 `p2` ... parametr f_2 `p3` ... parametr f_3 `p4` ... parametr f_4 `p5` ... parametr f_5	`long`
`MFR`	Generování frekvenční charakteristiky modelu na výstupy `w`, `xre` a `xim` (`off`→`on` spouští generování)	`bool`

Výstupy

mv	Akční zásah regulátoru (manipulated variable)	double
de	Regulační odchylka	double
SAT	Saturace off ... lineární zákon řízení on .... výstup regulátoru je saturován	bool
IDBSY	Příznak probíhající identifikace off ... identifikace neprobíhá on .... běží identifikační experiment	bool
w	Odhad bodu frekvenční charakteristiky - frekvence v rad/s	double
xre	Odhad bodu frekvenční charakteristiky - reálná část	double
xim	Odhad bodu frekvenční charakteristiky - imaginární část	double
epv	Rekonstruovaný signál pv (pro účely ručního ladění rekonstruktoru)	double
IDE	Příznak chyby identifikace off ... identifikace proběhla v pořádku on .... identifikace skončila s chybou	bool
iIDE	Kód chyby 101 ... vzorkovací frekvence je příliš nízká 102 ... chyba identifikace jednoho nebo dvou bodů frekvenční charakteristiky 103 ... saturace výstupu během identifikace 104 ... je zadán nebo spočten nesprávný model procesu	long
p1..p6	Výsledky identifikace a návrhu regulátoru	double

Parametry

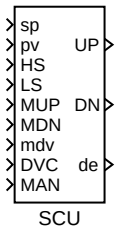
ubias	Stejnosměrná složka budicího harmonického signálu	
uamp	Amplituda budicího harmonického signálu	⊙1.
wb	Počátek frekvenčního intervalu [rad/s]	⊙1.
wf	Konec frekvenčního intervalu [rad/s]	⊙10.
isweep	Způsob rozmítání frekvence 1 ..... logaritmické 2 ..... lineární (zatím není implementováno)	⊙
cp	Rychlost rozmítání	↓0.5 ↑1.0 ⊙0.99
iavg	Počet vzorků pro průměrování	⊙1
alpha	Relativní poloha pólů rekonstruktoru (ve fázi identifikace)	⊙2.
xi	Koeficient tlumení rekonstruktoru (ve fázi identifikace)	⊙0.70
MANF	Ruční výběr bodů frekvenční charakteristiky off ... zakázáno on .... povoleno	
ph1	Fázové zpoždění prvního bodu ve stupních	⊙-60.
ph2	Fázové zpoždění druhého bodu ve stupních	⊙-120.
stime	Doba ustáleni [s]	⊙10.
ralpha	Relativní poloha pólů rekonstruktoru	⊙4.
rxi	Koeficient tlumení rekonstruktoru	⊙0.70

acl1	Relativní poloha 1. dvojice pólů uzavřené smyčky	⊙1.0	double
xicl1	Tlumení 1. dvojice pólů uzavřené smyčky	⊙0.707	double
INTGF	Příznak rozšíření o integrátor	⊙on	bool
	off ... stavový model neobsahuje integrátor on .... integrátor je zahrnut ve stavovém modelu		
apcl	Relativní poloha reálného pólu	⊙1.0	double
DISF	Příznak rozšíření o model poruchy		bool
	off ... stavový model neobsahuje model poruchy on .... model poruchy je zahrnut ve stavovém modelu		
dom	Přirozená frekvence modelu poruchy	⊙1.0	double
dxi	Koeficient tlumení modelu poruchy		double
acl2	Relativní poloha 2. dvojice	⊙2.0	double
xicl2	Tlumení 2. dvojice pólů uzavřené smyčky	⊙0.707	double
tt	Časová konstanta vysledování	⊙1.0	double
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	double
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	double
mb1p	Koeficient přenosu řízeného systému (b_1)		double
mb0p	Koeficient přenosu řízeného systému (b_2)	⊙1.0	double
ma1p	Koeficient přenosu řízeného systému (a_1)	⊙0.2	double
ma0p	Koeficient přenosu řízeného systému (a_0)	⊙1.0	double

SCU – Krokový regulátor s polohovou zpětnou vazbou

Symbol bloku

Licence: STANDARD



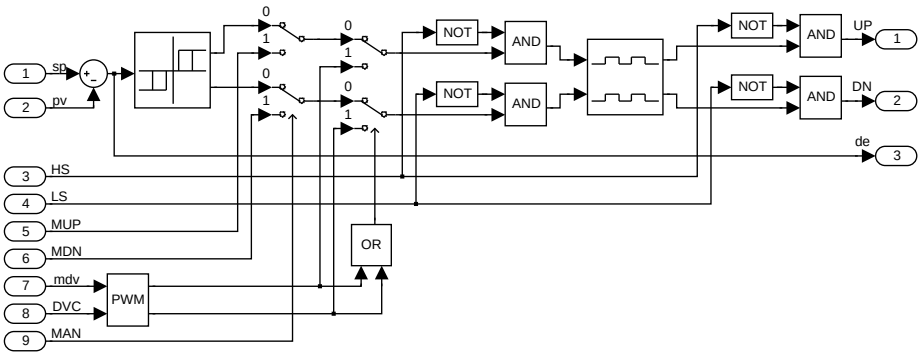
Popis funkce

Blok SCU je polohový regulátor servoventilu s třístavovým výstupem. Ve spojení s nadřazeným blokem PIDU nebo od něho odvozeným (PIDMA, atd.) je určen k realizaci třístavového krokového regulátoru s polohovou zpětnou vazbou.

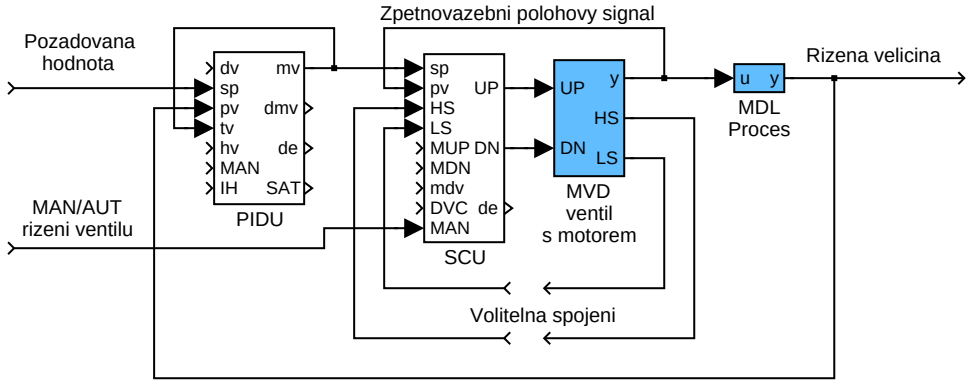
Blok SCU nejprve zpracovává regulační odchylku sp − pv na třístavový výstup symetrickým třístavovým algoritmem s parametry (práhy) thron a throff (viz blok TSE, uvažujte parametry ep = thron, epoff = throff, en = -thron a enoff = -throff). Parametr RACT určuje, pro kterou polaritu odchylky jsou generovány pulsy UP (více) nebo DN (méně). Výstupy symetrického třístavového algoritmu jsou dále zpracovávány tak, že délka libovolného generovaného pulsu (UP, DN) na výstupu bloku je alespoň dtime a prodleva mezi dvěma po sobě jdoucími pulsy opačné polarity je alespoň btime. Jsou-li dostupné signály od koncových spínačů servoventilu, potom by měly být připojeny na vstupy HS (horní spínač) a LS (dolní spínač).

K dispozici je také sada vstupů pro manuální ovládání. Přepnutí do manuálního režimu je možné pomocí vstupu MAN = on, pak lze s motorem pohybovat tam a zpět pomocí signálů MUP a MDN, eventuelně lze pomocí vstupu mdv nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou (off→on) na vstupu DVC.

Celková funkce bloku SCU je dostatečné zřejmá z následujícího diagramu.



Úplný třístavový krokový regulátor s polohovou zpětnou vazbou je zobrazen na následujícím obrázku.



Vstupy

`sp`	Požadovaná hodnota (výstup primárního regulátoru)	`double`
`pv`	Řízená veličina (poloha servopohonu ventilu)	`double`
`HS`	Horní koncový spínač (příznak, že poloha ventilu je na horní mezi)	`bool`
`LS`	Dolní koncový spínač (příznak, že poloha ventilu je na spodní mezi)	`bool`
`MUP`	Manuální povel UP (nahoru, přidej)	`bool`
`MDN`	Manuální povel DN (dolů, uber)	`bool`
`mdv`	Ruční diferenční hodnota (požadovaný přírůstek/úbytek polohy, mající vyšší prioritu než přímé signály `MUP`/`MDN`)	`double`
`DVC`	Přijetí ruční diferenční hodnoty (`off`→`on`)	`bool`
`MAN`	Manuální nebo automatický režim `off` ... automatický režim `on` .... manuální režim	`bool`

Výstupy

`UP`	Signál UP (nahoru, více)	`bool`
`DN`	Signál DN (dolů, méně)	`bool`

`de`	Regulační odchylka		`double`

Parametry

`thron`	Mez pro zapnutí	↓`0.0` ⊙`0.02`	`double`
`throff`	Mez pro vypnutí	↓`0.0` ⊙`0.01`	`double`
`dtime`	Minimální trvání výstupního pulzu [s]	↓`0.0` ⊙`0.1`	`double`
`btime`	Minimální prodleva mezi pulzy [s]	↓`0.0` ⊙`0.1`	`double`

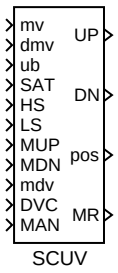
`RACT`	Převrácené působení výstupu regulátoru
	`off` ... vyšší `mv` → vyšší `pv`
	`on` .... vyšší `mv` → nižší `pv`
			`bool`

`trun`	Časová konstanta motoru	↓`0.0` ⊙`10.0`	`double`

SCUV – Krokový regulátor s rychlostním výstupem

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SCUV nahrazuje polohový regulátor SCU v úplné regulační smyčce s třístavovým krokovým regulátorem, jestliže polohový signál servoventilu není dostupný anebo dostatečně spolehlivý. Nadřazený regulátor PIDU (nebo odvozený) je propojen s blokem SCUV signály mv, dmv a SAT (výstupy bloku PIDU a vstupy bloku SCUV).

Jestliže je nadřazený regulátor typu PI nebo PID (CWOI = off), potom jsou všechny tři vstupy mv, dmv a SAT bloku SCUV zpracovávány speciálním integračním algoritmem a symetrickým třístavovým algoritmem s parametry (práhy) thron a throff (viz blok TSE, uvažujte parametry ep = thron, epoff = throff, en = -thron a enoff = -throff). Vzniklé pulsy (více, méně) jsou dále upravovány tak, že délka libovolného generovaného pulsu (UP, DN) na výstupu bloku je alespoň dtime a prodleva mezi dvěma po sobě jdoucími pulsy opačné polarity je alespoň btime. Parametr RACT určuje směr otáčení motoru. Poznamenejme, že nadřazený regulátor PIDU musí mít nastavení icotype = 4. Blok SCUV rekonstruuje rychlostní výstup nadřazeného regulátoru ze vstupů mv a dmv. Navíc, jestliže regulační odchylka nadřazeného regulátoru je menší než pásmo necitlivosti (SAT = on), potom je výstup vnitřního integrátoru bloku SCUV nulován. Takto je dosaženo klidu servoventilu při dostatečně malé regulační odchylce nadřazeného regulátoru (|de| < dz – viz popis bloku PIDU).

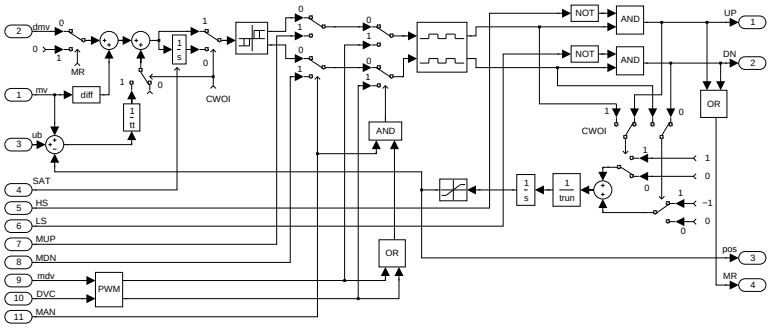
Poloha servoventilu pos je odhadována dalším vnitřním integrátorem s časovou konstantou trun. Jsou-li dostupné signály od koncových spínačů servoventilu, potom by měly být připojeny na vstupy HS (horní spínač) a LS (dolní spínač).

Jestiže je nadřazený regulátor typu P nebo PD (CWOI = on), potom může být regulační odchylka nadřazeného regulátoru manuálně odstraněna vhodným nastavením vstupu ub. V tomto případě je řídicí algoritmus bloku SCUV) lehce modifikován. Je užita rekonstruovaná hodnota polohy servoventilu pos a parametry thron, throff a tt musí být pečlivě nastaveny pro potlačení střídání pulsů více a méně v ustáleném stavu.

K dispozici je také sada vstupů pro manuální ovládání. Přepnutí do manuálního režimu je možné pomocí vstupu MAN = on, pak lze s motorem pohybovat tam a zpět

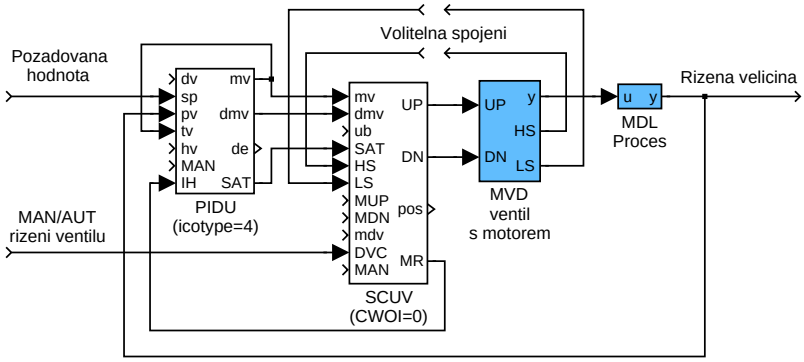
pomocí signálů `MUP` a `MDN`, eventuelně lze pomocí vstupu `mdv` nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou (`off`→`on`) na vstupu `DVC`.

Celková funkce bloku `SCUV` je zřejmá z následujícího diagramu:

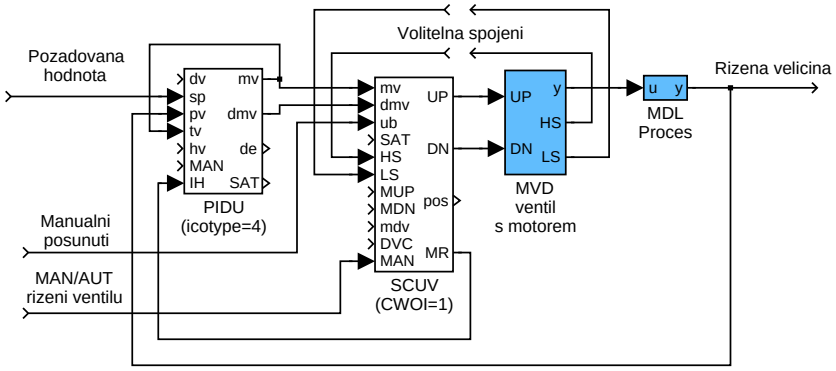


Úplné třístavové krokové regulátory bez polohové zpětné vazby jsou zobrazeny na následujících obrázcích:

Primarni regulator s integraci: I, PI, PID



Primarni regulator bez integrace: P, PD



Vstupy

`mv`	Akční zásah regulátoru (manipulated variable)	`double`
`dmv`	Rychlostní výstup regulátoru (diference)	`double`
`ub`	Posunutí (jen pokud je primární regulátor typu P nebo PD)	`double`

SAT	Nulování interního integrátoru (propojen s výstupem SAT primárního regulátoru)	bool
HS	Horní koncový spínač (příznak, že poloha ventilu je na horní mezi)	bool
LS	Dolní koncový spínač (příznak, že poloha ventilu je na spodní mezi)	bool
MUP	Manuální povel UP (nahoru, přidej)	bool
MDN	Manuální povel DN (dolů, uber)	bool
mdv	Ruční diferenční hodnota (požadovaný přírůstek/úbytek polohy, mající vyšší prioritu než přímé signály MUP/MDN)	double
DVC	Přijetí ruční diferenční hodnoty off→on	bool
MAN	Manuální nebo automatický režim off ... Automatický režim on .... Manuální režim	bool

Výstupy

UP	Signál UP (nahoru, více)	bool
DN	Signál DN (dolů, méně)	bool
pos	Simulovaná poloha motoru	double
MR	Požadavek na běh motoru off ... motor neběží (UP = off a DN = off) on .... motor se má pohybovat (UP = on nebo DN = on)	bool

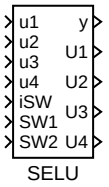
Parametry

thron	Mez pro zapnutí	↓0.0 ⊙0.02	double
throff	Mez pro vypnutí	↓0.0 ⊙0.01	double
dtime	Minimální trvání výstupního pulzu [s]	↓0.0 ⊙0.1	double
btime	Minimální prodleva mezi dvěma následujícími pulzy [s]	↓0.0 ⊙0.1	double
RACT	Převrácené působení výstupu regulátoru off ... vyšší mv → vyšší pv on .... vyšší mv → nižší pv		bool
trun	Časová konstanta motoru (určuje dobu, za kterou se motor posune o hodnotu jedna)	↓0.0 ⊙10.0	double
CWOI	Regulátor bez integrační složky off ... primární regulátor s integrátorem (I, PI, PID) on .... primární regulátor bez integrátoru (P, PD)		bool
tt	Časová konstanta vysledování	↓0.0 ⊙1.0	double

SELU – Selektor aktivního regulátoru

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SELU je určen pro přepínání aktivního regulátoru v případě selektorové regulace. Provádí výběr jednoho ze vstupních signálů u1, u2, u3, u4 a kopíruje ho na výstup y buď podle celočíselného vstupu iSW (je-li parametr bloku BINF = off) nebo podle binárních vstupů SW1 a SW2 (BINF = on) dle následující tabulky.

iSW	SW1	SW2	y	U1	U2	U3	U4
0	off	off	u1	off	on	on	on
1	off	on	u2	on	off	on	on
2	on	off	u3	on	on	off	on
3	on	on	u4	on	on	on	off

Z této tabulky je patrný též význam logických výstupů U1, U2, U3, U4, které se používají jako vstupy bloků SWU pro realizaci funkce vysledování neaktivních regulátorů v selektorové regulaci.

Vstupy

u1..u4	Signály, ze kterých bude jeden vybrán	double
iSW	Selektor aktivního signálu, použit pokud BINF = off	long
SW1	Binární vstup pro výběr, použit pokud BINF = on	bool
SW2	Binární vstup pro výběr, použit pokud BINF = on	bool

Výstupy

y	Analogový výstupní signál	double
U1..U4	Binární signály pro selektorovou regulaci	bool

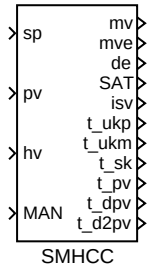
Parametr

`BINF`	Výběr pomocí binárních vstupů `off` ... zakázáno (výběr přes `iSW`) `on` .... povoleno (výběr přes `SW1` a `SW2`)	`bool`

SMHCC – Regulátor pro procesy s topením a chlazením

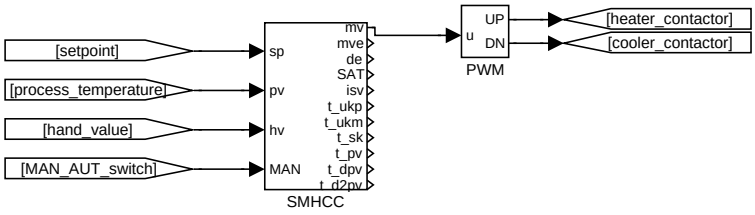
Symbol bloku

Licence: ADVANCED



Popis funkce

Regulátor SMHCC (Sliding Mode Heating/Cooling Controller) je snadno nastavitelný regulátor pro kvalitní regulaci teplotních soustav s dvoustavovým (ON-OFF) topením a dvoustavovým (ON-OFF) chlazením. Klasickým příkladem takových soustav je plastikářský lis. SMHCC může být samozřejmě nasazen i na jiné soustavy, kde se dosud běžně používají konvenční termostaty. Pro zajištění správné funkce je nutné blok SMHCC doplnit blokem PWM (Pulse Width Modulation), jak je patrné z následujícího obrázku.



Blok SMHCC pracuje se dvěma časovými periodami. První perioda T_S je vzorkovací perioda měřené teploty a je rovněž rovna periodě, se kterou se blok regulátoru SMHCC spouští. Druhá perioda $T_C = i_{pwmc} T_S$ je perioda řízení, se kterou blok SMHCC generuje akční zásahy. Tato perioda T_C je totožná s periodou cyklu bloku PWM. V každém okamžiku, když se změní akční zásah mv bloku SMHCC, algoritmus bloku PWM přepočte šířku pulsu a spustí nový PWM cyklus. Třetí perioda, kterou je třeba stanovit, je perioda spouštění T_R bloku PWM. Obecně může být $T_R \neq T_S$. Pro dosažení co nejlepší kvality řízení je doporučeno nastavit periodu T_S na minimální možnou hodnotu (i_{pwmc} na maximální možnou hodnotu), poměr T_C/T_S maximální, ale T_C by měla být dostatečně malá vzhledem k dynamice procesu. Pro aplikace v plastikářském průmyslu jsou doporučeny následující hodnoty:

$$T_S = 0.1,\ i_{pwmc} = 100,\ T_C = 10s,\ T_R = 0.01s.$$

Zákon řízení bloku regulátoru SMHCC v automatickém režimu (MAN=off) je založen na diskrétní technice dynamického řízení s klouzavým režimem a dále je použit speciální filtr třetího řádu pro odhad první a druhé derivace regulační odchylky.

Po změně požadované hodnoty sp (setpoint) se regulátor dostane do první fáze, tzv. přibližovací, kdy diskrétní proměnná klouzavého režimu

$$s_k \stackrel{\triangle}{=} \ddot{e}_k + 2\xi\Omega\dot{e}_k + \Omega^2 e_k$$

je stlačena do nuly. Ve výše uvedené definici neznámé $e_k, \dot{e}_k, \ddot{e}_k$ po řadě označují filtrovanou regulační odchylku (pv-sp), první a druhou derivaci e_k v čase k. Parametry ξ a Ω jsou popsány níže. V druhé fázi (kvazi klouzavý režim) je proměnná s_k držena v okolí nulové hodnoty pomocí patřičných zásahů řízení, režim topení se střídá s režimem chlazení. Amplitudy topení a chlazení se adaptují tak, aby se dosáhlo přibližně $s_k = 0$. V důsledku toho je hypotetická spojitá proměnná klouzavého režimu

$$s \stackrel{\triangle}{=} \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e$$

stále přibližně nulová. Jinak řečeno regulační odchylka e je popsána diferenciální rovnicí druhého řádu

$$s \stackrel{\triangle}{=} \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e = 0.$$

Z toho plyne, že vývoj e může být ovlivněn volbou parametrů ξ a Ω. Poznamenejme, že pro stabilní chování musí být splněno $\xi > 0$ a $\Omega > 0$. Typická optimální hodnota ξ leží v intervalu $[0.1, 8]$. Optimální hodnota Ω je silně závislá na řízeném procesu, pomalejší procesy mají menší hodnotu a rychlejší větší. Doporučená hodnota Ω pro začátek ladění parametrů je $\pi/(5T_C)$.

Řídicí veličena mv je obvykle v intervalu $[-1, 1]$. Kladná hodnota odpovídá topení, záporná chlazení, např. mv $= 1$ znamená plné topení. Omezení na mv může být zadáno parametry hilim_p a hilim_m. Omezení může být potřebné, když existuje velká nesymetrie mezi topením a chlazením. Jestliže je například chlazení mnohem agresivnější než topení, je vhodné nastavit hilim_p $= 1$ and hilim_m < 1. Pokud chceme omezení aplikovat pouze v intervalu po změně požadované hodnoty sp, volíme u0_p a u0_m tak, že platí u0_p $\leq$ hilim_p a u0_m $\leq$ hilim_m.

Hodnoty amplitud proměnných pro topení a chlazení t_ukp, t_ukm se automaticky adaptují speciálním algoritmem tak, aby byl dosažen kvazi klouzavý režim, ve kterém se střídají znaménka sk po každém kroku. V tomto případě se výstup isv přepíná mezi 1 a -1. Rychlost adaptace amplitud topení a chlazení je dána časovými konstantami taup a taum. Obě tyto časové konstanty musí být dostatečně velké, aby zajistily správnou funkci adaptace, ale jejich jemné doladění není nezbytné pro výslednou kvalitu regulace. Pro úplnost dodejme, že mv je určena na základě amplitud t_ukp a t_ukm podle následujícího výrazu

$$if \quad (\texttt{sk} < 0.0) \quad then \quad \texttt{mv} = \texttt{t\_ukp} \quad else \quad \texttt{mv} = -\texttt{t\_ukm}\,.$$

Dále je třeba říci, že dosažení kvazi klouzavého režimu nastává velmi zřídka, protože řízené procesy obsahují dopravní zpoždění a působí na ně poruchy. Vhodným indikátorem

kvality "klouzání"je opět výstup `isv`. Pro jemné doladění je možno v mimořádných případech použít parametr `beta` definující šířku pásma derivačního filtru. Ve většině případů však vyhovuje přednastavená hodnota `beta` = 0.1.

V manuálním režimu (`MAN` = `on`) je vstup regulátoru `hv` kopírován po případném omezení saturačními mezemi na výstup `mv`.

Vstupy

`sp`	požadovaná hodnota (setpoint)	`double`
`pv`	regulovaná veličina (process variable)	`double`
`hv`	výstup regulátoru v manuálním režimu (hand value)	`double`
`MAN`	režim činnosti regulátoru `0` ..... automatický režim `1` ..... manuální režim	`bool`

Výstupy

`mv`	řídicí veličina (manipulated variable)	`double`
`de`	regulační odchylka (deviation error) `de` = `sp` − `pv`	`double`
`SAT`	příznak saturace `0` ..... regulátor pracuje v lineární oblasti `1` ..... výstup regulátoru je saturován, `mv` ≥ `hilim_p` nebo `mv` ≤ −`hilim_m`	`bool`
`isv`	počet kladných nebo záporných kroků přepínací proměnné sk	`long`
`t_ukp`	aktuální hodnota amplitudy topení	`double`
`t_ukm`	aktuální hodnota amplitudy chlazení	`double`
`t_sk`	přepínací proměnná sk	`double`
`t_ek`	filtrovaná regulační odchylka $-de$	`double`
`t_dek`	filtrovaná první derivace regulační odchylky t_ek	`double`
`t_2dek`	filtrovaná druhá derivace regulační odchylky t_ek	`double`

Parametry

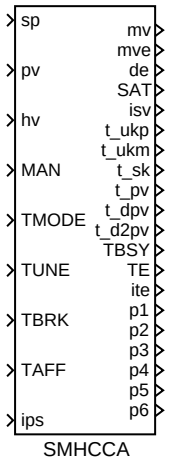
`ipwmc`	počet `PWM` cyklů během jedné periody spouštění bloku `SMHCC` (T_C/T_S)	`long`
`xi`	relativní tlumení `xi` > 0	`double`
`om`	přirozená frekvence `om` > 0	`double`
`taup`	časová konstanta adaptace amplitudy topení v sekundách	`double`
`taum`	časová konstanta adaptace amplitudy chlazení v sekundách	`double`
`beta`	šířka pásma derivačního filtru; `beta` > 0; doporučená hodnota 0.1	`double`
`hilim_p`	horní saturační mez amplitudy topení (0 < `hilim_p` ≤ 1)	`double`
`hilim_m`	horní saturační mez amplitudy chlazení (0 < `hilim_m` ≤ 1)	`double`
`u0_p`	počáteční hodnota amplitudy topení po změně požadované hodnoty nebo startu bloku	`double`
`u0_m`	počáteční hodnota amplitudy chlazení po změně požadované hodnoty nebo startu bloku	`double`
`sp_dif`	Práh pro detekci změny setpointu ⊙`10.0`	`double`

`tauf`	Časová konstanta filtru ekvivalentní akční veličiny	⊙`400.0`	`double`

SMHCCA – * Regulátor pro procesy s topením a chlazením s autotunerem

Symbol bloku

Licence: AUTOTUNING



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

`sp`	Požadovaná hodnota (setpoint)	`double`
`pv`	Řízená veličina	`double`
`hv`	Hodnota výstupu v manuálním režimu	`double`
`MAN`	Manuální nebo automatický režim `off` ... automatický režim `on` .... manuální režim	`bool`
`TMODE`	Režim ladění	`bool`
`TUNE`	Zahájení ladicího experimentu	`bool`
`TBRK`	Ukončení ladicího experimentu	`bool`
`TAFF`	Přijetí výsledků ladicího experimentu `off` ... parametry jsou pouze vypočítány `on` .... parametry jsou dosazeny do řídicího algoritmu	`bool`
`ips`	Význam výstupních signálů `0` ..... parametry regulátoru `1` ..... pomocné parametry	`long`

Parametry

`ipwmc`	Délka PWM cyklu (počet vzorkovacích period bloku)	⊙100	`long`
`xi`	Relativní tlumení	↓0.5 ↑8.0 ⊙1.0	`double`
`om`	Přirozená frekvence	↓0.0 ⊙0.01	`double`
`taup`	Časová konstanta adaptace amplitudy topení [s]	⊙700.0	`double`
`taum`	Časová konstanta adaptace amplitudy chlazení [s]	⊙400.0	`double`
`beta`	Šířka pásma derivačního filtru	⊙0.01	`double`
`hilim_p`	Horní saturační mez amplitudy topení	↓0.0 ↑1.0 ⊙1.0	`double`
`hilim_m`	Horní saturační mez amplitudy chlazení	↓0.0 ↑1.0 ⊙1.0	`double`
`u0_p`	Počáteční hodnota amplitudy topení	⊙1.0	`double`
`u0_m`	Počáteční hodnota amplitudy chlazení	⊙1.0	`double`
`sp_dif`	Práh pro detekci změny setpointu	⊙10.0	`double`
`tauf`	Časová konstanta filtru ekvivalentní akční veličiny	⊙400.0	`double`
`itm`	Metoda ladění regulátoru	⊙1	`long`
	`1` ..... omezeno na symetrické procesy `2` ..... asymetrické procesy (zatím není implementováno)		
`ut_p`	Amplituda topení pro ladicí experiment	↓0.0 ↑1.0 ⊙1.0	`double`
`ut_m`	Amplituda chlazení pro ladicí experiment	↓0.0 ↑1.0 ⊙1.0	`double`

Výstupy

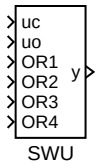
`mv`	Akční zásah regulátoru (manipulated variable)	`double`
`mve`	Ekvivalentní akční veličina	`double`
`de`	Regulační odchylka	`double`
`SAT`	Saturace	`bool`
	`off` ... lineární zákon řízení `on` .... výstup regulátoru je saturován	
`isv`	Počet kroků přepínací proměnné	`long`
`t_ukp`	Aktuální amplituda topení	`double`
`t_ukm`	Aktuální amplituda chlazení	`double`
`t_sk`	Přepínací proměnná	`double`
`t_pv`	Filtrovaná řízená veličina	`double`
`t_dpv`	Filtrovaná první derivace řízené veličiny	`double`
`t_d2pv`	Filtrovaná druhá derivace řízené veličiny	`double`
`TBSY`	Příznak probíhajícího ladicího experimentu	`bool`
`TE`	Příznak chyby během ladění	`bool`
	`off` ... ladění proběhlo bez chyby `on` .... během ladění se vyskytla chyba	

`ite`	Kód chyby `0` ..... bez chyby `1` ..... příliš zašuměné pv, zkontroluj teplotní vstup 2 `2` ..... nesprávný parametr `ut_p` `3` ..... setpoint je příliš nízký `4` ..... vzorkovací perioda je příliš nízká nebo je druhá derivace příliš zašuměná `5` ..... předčasné ukončení ladicího experimentu	`long`
`p1..p6`	Výsledky identifikace a návrhu regulátoru	`double`

SWU – Přepínač vstupu pro vysledování

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SWU je určen pro přepínání vhodného signálu na vstup pro vysledování bloků PIDU a MCU. V případě, že všechny vstupy OR1, ..., OR4 jsou logické nuly (off), potom na výstup y je kopírována hodnota vstupu uc, v opačném případě hodnota vstupu uo.

Vstupy

uc	Tento vstup je kopírován na výstup y, jestliže všechny vstupy OR1, OR2, OR3 a OR4 jsou off	double
uo	Tento vstup je kopírován na výstup, jestliže alespoň jeden vstup OR1, OR2, OR3 nebo OR4 je on	double
OR1	První logický výstup bloku	bool
OR2	Druhý logický výstup bloku	bool
OR3	Třetí logický výstup bloku	bool
OR4	Čtvrtý logický výstup bloku	bool

Výstup

y	Analogový výstupní signál	double

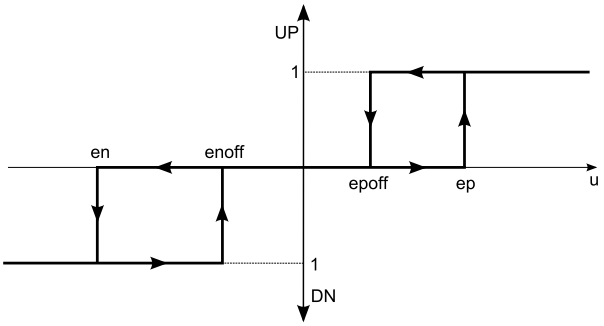
TSE – Třístavový prvek

Symbol bloku

Licence: STANDARD

Popis funkce

Blok TSE transformuje analogový vstup u na třístavový výstup (méně, nečinnost, více) podle níže uvedeného obrázku.



Vstup

u	Analogový vstupní signál	double

Výstupy

UP	Signál UP (nahoru, více)	bool
DN	Signál DN (dolů, méně)	bool

Parametry

ep	Hodnota u > ep způsobí nastavení výstupů UP = on a DN = off	⊙1.0	double
en	Hodnota u < en způsobí nastavení výstupů UP = off a DN = off	⊙-1.0	double
epoff	Je-li UP = on a u < epoff, potom UP = off	⊙0.5	double
enoff	Je-li DN = on a u > enoff, potom DN = off	⊙-0.5	double

Kapitola 8

LOGIC – Logické řízení

Obsah

AND_ – Logický součin dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok AND dělá logický součin dvou vstupních signálů U1 a U2.

Pokud potřebujete pracovat s více vstupními signály, použijte blok ANDOCT.

Vstupy

U1	První logický vstup bloku	bool
U2	Druhý logický vstup bloku	bool

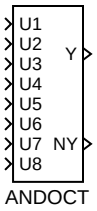
Výstupy

Y	Výstup bloku, logický součin (U1 $\wedge$ U2)	bool
NY	Negace výstupního signálu Y (NY $= \neg$Y)	bool

ANDOCT – Logický součin osmi signálů

Symbol bloku

Licence: STANDARD



Popis funkce

Blok ANDOCT dělá logický součin osmi vstupních signálů U1 až U8. Signály, jejichž seznam je uveden v parametru nl se před provedením logického součinu negují.

Pokud je tedy parametr nl prázdný, tak se provádí logický součin $Y = U1 \wedge U2 \wedge U3 \wedge U4 \wedge U5 \wedge U6 \wedge U7 \wedge U8$. Pokud bude například nl=2,6..8, pak $Y = U1 \wedge \neg U2 \wedge U3 \wedge U4 \wedge U5 \wedge \neg U6 \wedge \neg U7 \wedge \neg U8$.

Pokud pracujete s méně než 8 signály, je potřeba ošetřit nepřipojené vstupy bloku pomocí parametru nl. Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku AND_.

Vstupy

U1	První logický vstup bloku	bool
U2	Druhý logický vstup bloku	bool
U3	Třetí logický vstup bloku	bool
U4	Čtvrtý logický vstup bloku	bool
U5	Pátý logický vstup bloku	bool
U6	Šestý logický vstup bloku	bool
U7	Sedmý logický vstup bloku	bool
U8	Osmý logický vstup bloku	bool

Výstupy

Y	Výstup bloku, logický součin	bool
NY	Negace výstupního signálu Y	bool

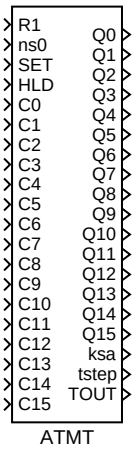
Parametr

`nl`	Seznam negovaných signálů. Zadává se ve tvaru např. `1,3..5,8`. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy `157`, binárně `10011101`.	`long`

ATMT – Automat pro sekvenční řízení

Symbol bloku

Licence: STANDARD



Popis funkce

Blok ATMT realizuje konečný automat až s 16 stavy a 16 podmínkami přechodů mezi nimi.

Aktuální stav automatu i, $i = 0, 1, \ldots, 15$, je indikován pomocí binárních výstupů Q0,Q1, ..., Q15. Pokud je automat ve stavu i, je nastaven příslušný výstup Qi = on. Aktuální stav automatu je též indikován celočíselným výstupem ksa $\in \{0, 1, \ldots, 15\}$.

Podmínky přechodů C_k, $k = 0, 1, \ldots, 15$ jsou aktivovány pomocí binárních vstupů bloku C0, C1, ..., C15. Pokud je Ck = on, je podmínka Ck splněna, naopak pro Ck = off splněna není.

Funkce automatu se zadává pomocí tabulky stavů a přechodů:

$$\begin{array}{lll} S1 & C1 & NS1 \\ S2 & C2 & NS2 \\ & & \ldots \\ Sn & Cn & NSn \end{array}$$

Každý řádek této tabulky vyjadřuje jedno pravidlo přechodu. Např. prvý řádek

$$S1 \quad C1 \quad NS1$$

má tento význam

Jestliže (aktuální stav je $S1$ AND podmínka přechodu $C1$ je splněna)
potom přejdi do následujícího stavu $NS1$

Výše uvedenou tabulku lze získat ze stavového diagramu automatu nebo z popisu automatu v jazyce SFC (Sequential Function Charts, dříve Grafcet).

Vstup `R1` = `on` resetuje stav automatu do počátečního stavu $S0$, přičemž vstup `R1` má prioritu před vstupem `SET`. Náběžná hrana na vstupu `SET` způsobí přechod z aktuálního stavu do stavu `ns0`. Vstup `HLD` = `on` zablokuje činnost automatu, tzn. automat setrvá v daném stavu i v případě, že je splněna některá podmínka přechodu, rovněž je zastaveno inkrementování času `tstep` a generování výstupu `TOUT`. Výstup `TOUT` indikuje, že automat setrval v daném stavu déle, než je povoleno. Časová omezení TOi jednotlivých stavů se definují pomocí vektoru `touts`. Pokud je $TOi = 0$, není pro daný stav nastaveno žádné časové omezení. Výstup `TOUT` je automaticky nastavován na hodnotu `off` při každém přechodu mezi stavy automatu.

Pomocí parametru `morestps` lze povolit přechod automatu o více kroků v jednom cyklu. Tuto možnost je však vždy potřeba pečlivě zvážit, zejména při použití výstupu `TOUT` v podmínkách pro přechod do dalších stavů. V takovém případě je vhodné zkonstruovat podmínku přechodu nejen pomocí výstupu `TOUT`, ale zahrnout do ní i informaci o stavu automatu `ksa`.

Součástí systému REX je také program SFCEditor, který umožňuje tvorbu SFC schémat v grafickém návrhovém prostředí. Editor se spouští z programu RexDraw kliknutím na tlačítko *Configure* na kartě parametrů bloku `ATMT`. Uživatelská příručka editoru je k dispozici jako samostatný dokument.

Vstupy

`R1`	Resetovací signál, je-li `R1` = `on`, je automat převeden do počátečního stavu `S0` (vstup `R1` má prioritu před vstupem `SET`)	`bool`
`ns0`	Do tohoto stavu přejde automat při náběžné hraně na vstupu `SET`	`long`
`SET`	Náběžná hrana na vstupu `SET` způsobí přechod z aktuálního stavu do stavu `ns0`	`bool`
`HLD`	Blokovací vstup, `HLD` = `on` zablokuje činnost automatu, stav zůstává, výstup `tstep` se neinkrementuje	`bool`
`C0...C15`	Podmínky přechodu, `C`i = `on` značí, že i-tá podmínka je splněna	`bool`

Výstupy

`Q0...Q15`	Výstupní signály určující stav automatu, aktivní je ten stav i, pro který platí `Q`i = `on`	`bool`
`ksa`	Celočíselná reprezentace stavu	`long`
`tstep`	Čas uplynulý od posledního přechodu mezi stavy	`double`
`TOUT`	Příznak překročení časového limitu pro aktuální stav	`bool`

Parametry

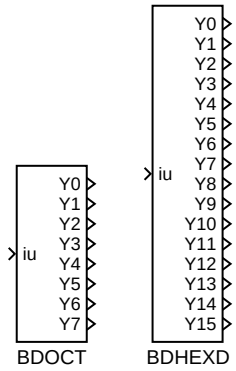
`morestps`	Povolit více přechodů mezi stavy automatu v jednom cyklu `off` ... zakázáno `on` .... povoleno		`bool`
`ntr`	Počet řádků tabulky přechodů mezi stavy	↓0 ↑64 ⊙4	`long`

`sfcname`	Jméno souboru, kam si konfigurátor bloku ukládá data (pokud se nevyplní, zvolí se automaticky podle jména bloku)	`string`
`STT`	Tabulka přechodů mezi stavy ⊙`[0 0 1; 1 1 2; 2 2 3; 3 3 0]`	`byte`
`touts`	Vektor časových limitů $TO0 \ldots TO15$ pro jednotlivé stavy $S0 \ldots S15$) ⊙`[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]`	`double`

BDOCT, BDHEXD – Bitové demultiplexery

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky `BDOCT` a `BDHEXD` pracují jako bitové demultiplexery a lze je výhodně použít pro rozebírání celočíselných signálů na jednotlivé bity. Oba bloky se od sebe liší jen počtem výstupů, blok `BDOCT` má 8 bitových výstupů, blok `BDHEXD` jich má 16. Výstupní signály `Y`*i* jsou přímo tvořeny bity signálu, který vznikne bitovým posunem vstupu `iu` o `shift` bitů vpravo, přičemž v signálu `Y0` je vždy nejnižší bit tohoto čísla.

Vstup

`iu`	Vstupní signál k dekompozici	`long`

Výstupy

`Y0...Y15`	Jednotlivé bity vstupního signálu	`bool`

Parametr

`shift`	Počet bitů, o který se posune vstupní signál `iu` před rozebráním na jednotlivé bitové výstupy ↓0 ↑31	`long`

BITOP – Bitová operace dvou celočíselných signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok BITOP provádí operaci i1 ∘ i2 na vstupních signálech po jednotlivých bitech. Výsledkem je celočíselný výstup n. Kód zvolené bitové operace je uveden v parametru iop popsaném níže. V případě bitové negace a dvojkových doplňků se operace provádí pouze se vstupem i1 (tj. operace je unární).

Vstupy

i1	První celočíselný vstup bloku	long
i2	Druhý celočíselný vstup bloku	long

Výstup

n	Výsledek bitové operace určené parametrem iop	long

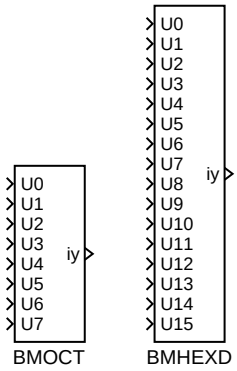
Parametr

iop	Bitová operace	⊙1	long
	1 ..... bitová negace (Bit NOT)		
	2 ..... logický součet po jednotlivých bitech (Bit OR)		
	3 ..... logický součin po jednotlivých bitech (Bit AND)		
	4 ..... logický exkluzivní součet po jednotlivých bitech (Bit XOR)		
	5 ..... posun signálu i1 doleva o i2 bitů (Shift Left)		
	6 ..... posun signálu i1 doprava o i2 bitů (Shift Right)		
	7 ..... dvojkový doplněk signálu i1 na 8 bitech (2's Complement - Byte)		
	8 ..... dvojkový doplněk signálu i1 na 16 bitech (2's Complement - Word)		
	9 ..... dvojkový doplněk signálu i1 na 32 bitech (2's Complement - Long)		

BMOCT, BMHEXD – Bitový multiplexer

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky `BMOCT` a `BMHEXD` pracují jako bitové multiplexery a lze je výhodně využít ke skládání celočíselných signálů z jednotlivých bitů. Oba bloky se od sebe liší jen počtem vstupů, blok `BMOCT` má 8 bitových vstupů, blok `BMHEXD` jich má 16. V případě, že parametr `shift` = 0, jsou jednotlivé bity výstupního signálu `iy` přímo tvořeny vstupními signály, v nejnižším bitu je vždy signál `U0`.

Vstupy

`U0...U15`	Jednotlivé bity výstupního signálu	`bool`

Výstup

`iy`	Výsledný výstupní signál	`long`

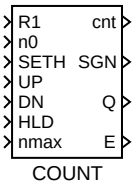
Parametr

`shift`	Počet bitů, o který se posune složená celočíselná hodnota z jednotlivých vstupů těsně před předáním na výstup `iy` ↓0 ↑31	`long`

COUNT – Řízený čítač

Symbol bloku

Licence: STANDARD



Popis funkce

Blok COUNT je určen pro obousměrné čítání pulsů – přesněji náběžných hran na vstupech UP a DN. Při výskytu náběžné hrany na vstupu UP (DN) se výstup cnt zvětší o 1 (sníží o 1). Současný výskyt náběžných hran na vstupech UP a DN indikuje výstup E jako chybu čítání. Resetování výstupu cnt na hodnotu 0 lze provést vstupem R1 (pokud je R1 = on je výstup cnt držen na hodnotě 0). Nastavení výstupu cnt na hodnotu n0 zajistí vstup SETH = on (pokud SETH = on je výstup cnt držen na hodnotě n0). Vstup R1 má vyšší prioritu než vstup SETH. Vstup HLD = on způsobí zastavení čítání. Stav čítače cnt $\geq$ nmax způsobí nastavení výstupu Q na hodnotu on.

Vstupy

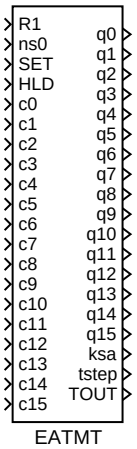
R1	Reset bloku (R1 = on)	bool
n0	Hodnota pro nastavení čítače pomocí vstupu SETH	long
SETH	Nastavení hodnoty čítače na n0 (SETH = on)	bool
UP	Vstup pro přičítání	bool
DN	Vstup pro odečítání	bool
HLD	Zmrazení čítače off ... čítač běží on .... čítač je zablokován	bool
nmax	Cílová hodnota čítače	long

Výstupy

cnt	Celkový počet načtených pulzů	long
SGN	Znaménko výstupu cnt off ... menší nebo rovno nule on .... větší než nula	bool
Q	Indikátor dosažení cílové hodnoty off ... pro cnt < nmax on .... pro cnt $\geq$ nmax	bool
E	Indikátor současného výskytu náběžných hran na vstupech UP a DN	bool

EATMT – Extended finite-state automaton

Symbol bloku Licence: ADVANCED



Popis funkce

The EATMT block implements a finite automat with at most 256 states and 256 transition rules, thus it extends the possibilities of the ATMT block.

The current state of the automat i, $i = 0, 1, \ldots, 255$ is indicated by individual bits of the integer outputs q0, q1, ..., q15. Only a single bit with index $i \,\mathrm{MOD}\, 16$ of the q($i \,\mathrm{DIV}\, 16$) output is set to 1. The remaining bits of that output and the other outputs are zero. The bits are numbered from zero, least significant bit first. Note that the DIV and MOD operators denote integer division and remainder after integer division respectively. The current state is also indicated by the ksa $\in \{0, 1, \ldots, 255\}$ output.

The transition conditions C_k, $k = 0, 1, \ldots, 255$) are activated by individual bits of the inputs c0, c1, ..., c15. The k-th transition condition is fulfilled when the ($k \,\mathrm{MOD}\, 16$)-th bit of the input c($k \,\mathrm{DIV}\, 16$) is equal to 1. The transition cannot happen otherwise.

The BMHEXD or BMOCT bitwise multiplexers can be used for composition of the input signals c0, c1, ..., c15 from individual Boolean signals. Similarly the output signals q0, q1, ..., q15 can be decomposed using the BDHEXD or BDOCT bitwise demultiplexers.

The automat function is defined by the following table of transitions:

$$\begin{array}{ccc} S1 & C1 & FS1 \\ S2 & C2 & FS2 \\ & & \ldots \\ Sn & Cn & FSn \end{array}$$

Each row of this table represents one transition rule. For example the first row

$$S1 \quad C1 \quad FS1$$

has the meaning

`If` ($S1$ is the current state `AND` transition condition $C1$ is fulfilled)
`then` proceed to the following state $FS1$.

The above described meaning of the table row holds for $C1 < 1000$. Negation of the $(C1 - 1000)$-th transition condition is assumed for $C1 \geq 1000$.

The above mentioned table can be easily constructed from the automat state diagram or SFC description (Sequential Function Charts, formerly Grafcet).

The `R1` = `on` input resets the automat to the initial state $S0$. The `SET` input allows manual transition from the current state to the `ns0` state when rising edge occurs. The `R1` input overpowers the `SET` input. The `HLD` = `on` input freezes the automat activity, the automat stays in the current state regardless of the `c`i input signals and the `tstep` timer is not incremented. The `TOUT` output indicates that the machine remains in the given state longer than expected. The time limits TOi for individual states are defined by the `touts` array. There is no time limit for the given state when TOi is set to zero. The `TOUT` output is set to `off` whenever the automat changes its state.

It is possible to allow more state transitions in one cycle by the `morestps` parameter. However, this option must be thoroughly considered and tested, namely when the `TOUT` output is used in transition conditions. In such a case it is strongly recommended to incorporate the `ksa` output in the transition conditions as well.

The development tools of the REX Control System include also the SFCEditor program. You can create SFC schemes graphically using this tool. Run this editor from RexDraw by clicking the *Configure* button in the parameter dialog of the `EATMT` block.

Vstupy

`R1`	Reset signal, `R1` = `on` brings the automat to the initial state `S0`; the `R1` input overpowers the `SET` input	`bool`
`ns0`	This state is reached when rising edge occurs at the the `SET` input	`long`
`SET`	The rising edge of this signal forces the transition to the `ns0` state	`bool`
`HLD`	The `HLD` = `on` freezes the automat, no transitions occur regardless of the input signals, `tstep` is not increasing	`bool`
`c0...c15`	Transition conditions, each input signal contains 16 transition conditions, see details above	

Výstupy

`q0...q15`	Output signals indicating the current state of the automat, see details above	`long`
`ksa`	Integer code of the active state	`long`
`tstep`	Time elapsed since the current state was reached; the timer is set to `0` whenever a state transition occurs	`double`
`TOUT`	Flag indicating that the time limit for the current state was exceeded	`bool`

Parametry

`morestps`	Allow multiple transitions in one cycle of the automat `off` ... Disabled `on` .... Enabled	`bool`
`ntr`	Number of state transition table rows ↓0 ↑1024 ⊙4	`long`
`sfcname`	Jméno souboru, kam si konfigurátor bloku ukládá data (pokud se nevyplní, zvolí se automaticky podle jména bloku)	`string`
`STT`	State transition table (matrix) ⊙`[0 0 1; 1 1 2; 2 2 3; 3 3 0]`	`short`
`touts`	Vector of timeouts `TO0`, `TO1`, ..., `TO255` for the states $S0$, $S1$, ..., $S255$ ⊙`[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]`	`double`

EDGE_ – Detekce hrany logického signálu

Symbol bloku

Licence: STANDARD

Popis funkce

Blok EDGE_ detekuje na vstupním signálu U náběžnou (off→on), sestupnou (on→off) nebo obě uvedené hrany podle hodnoty parametru iedge. V případě nalezení požadované hrany (změny vstupního signálu) je na jeden krok nastavena hodnota výstupu Y na on. Po dobu, kdy se hodnota vstupního signálu nemění je hodnota výstupu Y rovna off. Hodnota výstupu Y zůstane nulová i v případě, že v parametru iedge je zvolena náběžná (sestupná) hrana a v signálu se vyskytne hrana opačná, tj. sestupná (náběžná).

Vstup

U	Logický vstupní signál	bool

Výstup

Y	Indikace výskytu zvolené hrany	bool

Parametr

iedge	Typ detekovaných hran 1 ..... náběžná hrana 2 ..... sestupná hrana 3 ..... obě hrany	⊙1	long

INTSM – Bitový posun a maska nad celým číslem

Symbol bloku

Licence: STANDARD

Popis funkce

Blok INTSM provádí bitový posun vstupního čísla i o shift bitů doprava (pro kladný shift) nebo doleva (záporný shift). Volné bity vzniklé posunem jsou vyplněny nulami.

Výstupní hodnota n je logickým součinem (AND) bitově posunutého vstupu i a bitové masky mask.

Typické využití bloku spočívá v extrakci hodnoty jednoho nebo více sousedních bitů z určité pozice v celočíselném registru vyčteném z externího systému.

Vstup

i	Celočíselný signál pro zpracování		long

Parametry

shift	Bitový posun (záporné číslo=doleva, kladné číslo=doprava)	↓-31 ↑31	long
mask	Bitová maska (aplikovaná po bitovém posunu)	↓XXX ↑XXX ⊙XXX	dword

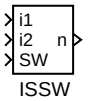
Výstup

n	Výsledná celočíselná hodnota	long

ISSW – Jednoduchý přepínač celočíselných signálů

Symbol bloku

Licence: STANDARD



Popis funkce

Blok ISSW je jednoduchý přepínač celočíselných vstupních signálů i1 a i2 na základě logického vstupu SW. Jestliže SW je off, pak výstup n je roven signálu i1. Jestliže SW je on, pak výstup n je roven signálu i2.

Vstupy

i1	První celočíselný vstup bloku	long
i2	Druhý celočíselný vstup bloku	long
SW	Přepínací signál off ... je zvolen vstupní signál i1 on .... je zvolen vstupní signál i2	bool

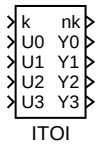
Výstup

n	Celočíselný výstupní signál	long

ITOI – Transformace celých a binárních čísel

Symbol bloku

Licence: STANDARD



Popis funkce

Blok ITOI přiřazuje vstupnímu číslu k respektive binárnímu číslu (U3 U2 U1 U0)$_2$ z množiny $\{0, 1, 2, \ldots, 15\}$ výstupní číslo nk a jeho binární reprezentaci (Y3 Y2 Y1 Y0)$_2$ z téže množiny. Příslušné zobrazení je popsáno tabulkou

k	0	1	2	...	15
nk	n0	n1	n2	...	n15

kde n0, ..., n15 jsou dány převodním vektorem fktab. Je-li BINF = off, potom se za významný považuje vstup k, zatímco pro BINF = on se za vstup bloku považuje číslo (U3 U2 U1 U0)$_2$.

Vstupy

k	Celočíselný vstupní signál	long
U0	Binární číslice vstupu s vahou 1	bool
U1	Binární číslice vstupu s vahou 2	bool
U2	Binární číslice vstupu s vahou 4	bool
U3	Binární číslice vstupu s vahou 8	bool

Výstupy

nk	Celočíselný výstupní signál	long
Y0	Binární číslice výstupu s vahou 1	bool
Y1	Binární číslice výstupu s vahou 2	bool
Y2	Binární číslice výstupu s vahou 4	bool
Y3	Binární číslice výstupu s vahou 8	bool

Parametry

BINF	Transformace hodnoty z binárních vstupů off ... zakázáno (transformace vstupu k) on .... povoleno (vstupem je hodnota (U3 U2 U1 U0)$_2$)	⊙on	bool

`fktab`	Cílové hodnoty převodní tabulky ⊙`[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]`	`byte`

NOT_ – Logická negace

Symbol bloku

Licence: STANDARD

Popis funkce

Blok NOT dělá logickou negaci vstupního signálu.

Vstup

U	Logický vstupní signál	bool

Výstup

Y	Logický výstupní signál (Y = ¬U)	bool

OR_ – Logický součet dvou signálů

Symbol bloku

Licence: STANDARD

Popis funkce

Blok OR dělá logický součet dvou vstupních signálů U1 a U2.

Pokud potřebujete pracovat s více vstupními signály, použijte blok OROCT.

Vstupy

U1	První logický vstup bloku	bool
U2	Druhý logický vstup bloku	bool

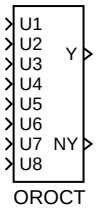
Výstupy

Y	Logický výstupní signál (U1 $\vee$ U2)	bool
NY	Negace výstupního signálu Y (NY $= \neg$Y)	bool

OROCT – Logický součet osmi signálů

Symbol bloku

Licence: STANDARD



Popis funkce

Blok OROCT provádí logický součet osmi vstupních signálů U1 až U8. Signály, jejichž seznam je uveden v parametru nl se před provedení logického součinu negují.

Pokud je tedy parametr nl prázdný, tak se provádí logický součet $Y = U1 \vee U2 \vee U3 \vee U4 \vee U5 \vee U6 \vee U7 \vee U8$. Pokud bude například nl=2,6..8, pak $Y = U1 \vee \neg U2 \vee U3 \vee U4 \vee U5 \vee \neg U6 \vee \neg U7 \vee \neg U8$.

Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku OR_.

Vstupy

U1	První logický vstup bloku	bool
U2	Druhý logický vstup bloku	bool
U3	Třetí logický vstup bloku	bool
U4	Čtvrtý logický vstup bloku	bool
U5	Pátý logický vstup bloku	bool
U6	Šestý logický vstup bloku	bool
U7	Sedmý logický vstup bloku	bool
U8	Osmý logický vstup bloku	bool

Parametr

nl	Seznam negovaných signálů. Zadává se ve tvaru např. 1,3..5,8. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	long

Výstupy

Y	Výstup bloku, logický součet	bool
NY	Negace výstupního signálu Y	bool

RS – Klopný obvod

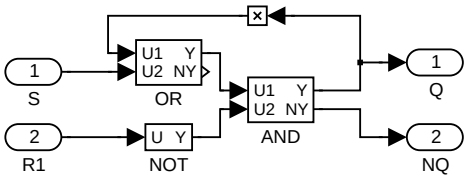
Symbol bloku

Licence: STANDARD

Popis funkce

Blok RS je klopný obvod, který v případě, že vstup S má hodnotu on, nastaví trvale výstup Q na on. Druhý vstupní signál R1 resetuje výstup Q na hodnotu off a to i tehdy, když vstup S má hodnotu on. Výstup NQ je pouhou negací výstupu Q.

Funkce bloku je dobře patrná z obrázku vnitřní struktury bloku.



Vstupy

S	Nahození klopného obvodu, nastaví výstup Q na on	bool
R1	Přednostní shození klopného obvodu, nastaví výstup Q na off	bool

Výstupy

Q	Stav klopného obvodu	bool
NQ	Negace výstupního signálu Q	bool

SR – Klopný obvod

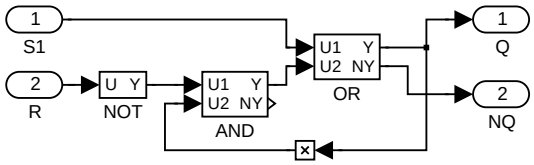
Symbol bloku

Licence: STANDARD

Popis funkce

Blok `SR` je klopný obvod, který v případě, že vstup `S1` má hodnotu `on`, nastaví trvale výstup `Q` na `on`. Druhý vstupní signál `R` resetuje výstup `Q` na hodnotu `off`, ale pouze tehdy, když vstup `S1` má hodnotu `off`. Výstup `NQ` je pouhou negací výstupu `Q`.

Funkce bloku je dobře patrná z obrázku vnitřní struktury bloku.



Vstupy

`S1`	Přednostní nahození klopného obvodu, nastaví výstup `Q` na `on`, má přednost před vstupem `R`	`bool`
`R`	Shození klopného obvodu, nastaví výstup `Q` na `off`	`bool`

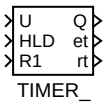
Výstupy

`Q`	Stav klopného obvodu	`bool`
`NQ`	Negace výstupního signálu `Q`	`bool`

TIMER_ – Vícefunkční časovač

Symbol bloku

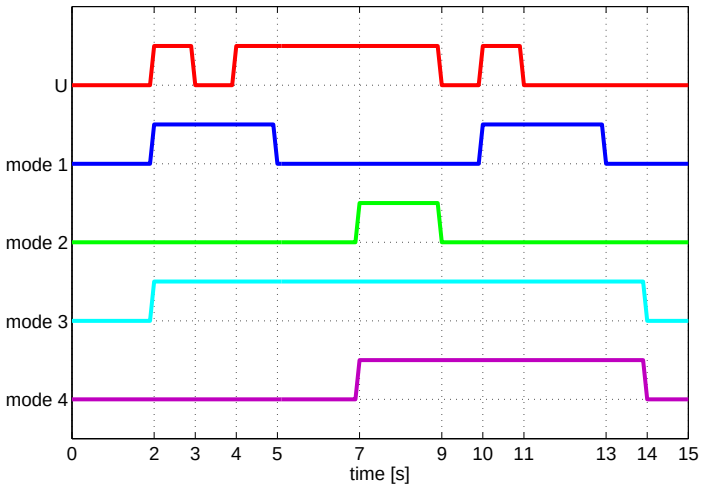
Licence: STANDARD



Popis funkce

Blok TIMER_ umožňuje buď vygenerovat impuls zadané délky pt (v sekundách) nebo filtrovat pulzy na vstupním signálu U užší než pt sekund. Režim funkce bloku se volí pomocí parametru mode. Čítání času je možno pozastavit pomocí vstupu HLD.

Následující obrázek ilustruje chování bloku v jednotlivých režimech při nastavení pt = 3:



Vstupy

U	Signál spouštějící časovač	bool
HLD	Pozastavení časovače	bool

Výstupy

Q	Výstupní signál časovače	bool
et	Doba uplynulá od startu časovače [s]	double

Parametry

`mode` Režim činnosti časovače ⊙`1` `long`

- `1` ..... generovaný pulz – na výstupu je pulz délky `pt` sekund, který začíná náběžnou hranou na vstupu `U`; další náběžné hrany na vstupu `U` během trvání pulzu jsou ignorovány
- `2` ..... zpožděné zapnutí – signál ze vstupu `U` je kopírován na výstup `Q` tak, že začátek impulzu na výstupu je opožděn o `pt` sekund proti začátku pulzu na vstupu; pulzy kratší než `pt` sekund se na výstupu neobjeví
- `3` ..... zpožděné vypnutí – signál ze vstupu `U` je kopírován na výstup `Q` tak, že konec impulzu na výstupu je opožděn o `pt` sekund proti konci pulzu na vstupu; pokud je mezera mezi vstupními pulzy kratší než `pt` sekund, výstup je trvale aktivní
- `4` ..... zpožděná změna – výstupní signál `Q` se přepne na hodnotu vstupu `U` až tehdy, když je vstup po dobu `pt` sekund neměnný

`pt` Doba časování [s] ⊙`1.0` `double`

Kapitola 9

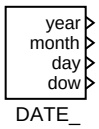
TIME – Bloky pro práci s časem

Obsah

DATE_ – Aktuální datum

Symbol bloku

Licence: STANDARD



Popis funkce

Výstupy bloku DATE_ odpovídají datu operačního systému. Pro pokročilé operace s časem a datem použijte blok DATETIME.

Výstupy

year	Rok		long
month	Měsíc		long
day	Den		long
dow	Den v týdnu, první den je neděle (1)		long

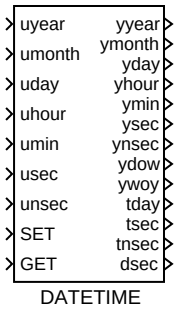
Parametr

tz	Časové pásmo 1 ..... lokální čas 2 ..... UTC	⊙1	long

DATETIME – Čtení, nastavování a konverze času

Symbol bloku

Licence: STANDARD



Popis funkce

Blok DATETIME je určen pro pokročilé operace s časem řídicího systému REX a operačního systému.

Blok umožňuje synchronizaci hodin operačního systému a řídicího systému REX. V okamžiku spuštění exekutivy systému REX jsou hodiny synchronizovány, ale během dlouhodobého provozu se mohou tyto dva údaje rozcházet (např. při přechodu na letní čas). Pokud je potřeba provést opětovnou synchronizaci, hodiny systému REX se při náběžné hraně (off→on) na vstupu SET aktualizují dle vstupů a parametrů bloku.

Je však důrazně doporučeno neaktualizovat hodiny systému REX, pokud je řízený stroj či technologie v provozu, neboť by to mohlo vést k nepředvídatelnému chování.

Pokud je potřeba číst nebo konvertovat údaje o čase, je možno příslušnou akci spustit náběžnou hranou (off→on) na vstupu GET a hodnoty přečíst na výstupech bloku. Výstupy začínající na 't' označují celkový počet daných jednotek od 1.1.2000 UTC.

Pokud jsou nastaveny parametry getper a setper na nenulové hodnoty, je čtení a nastavování hodin prováděno periodicky.

Při menší odchylce hodin systému REX a operačního systému, než udává parametr settol, nejsou hodiny systému REX nastaveny jednorázově, synchronizace probíhá postupně. Toho je dosaženo zanedbatelnými změnami v časování exekutivy systému REX, čímž po nějaké době dojde k dosažení synchronizace. Následně je použito standardní časování systému REX.

Pro jednoduché čtení data a/nebo času použijte bloky DATE_ a TIME.

Vstupy

uyear	Vstup pro nastavení roku	long
umonth	Vstup pro nastavení měsíce	long
uday	Vstup pro nastavení dne	long

`uhour`	Vstup pro nastavení hodin		`long`
`umin`	Vstup pro nastavení minut		`long`
`usec`	Vstup pro nastavení sekund		`long`
`unsec`	Vstup pro nastavení nanosekund	↓`-9,22E+18` ↑`9,22E+18`	`large`
`SET`	Nastavení času pomocí náběžné hrany		`bool`
`GET`	Přečtení času pomocí náběžné hrany		`bool`

Výstupy

`yyear`	Rok	`long`
`ymonth`	Měsíc	`long`
`yday`	Den	`long`
`yhour`	Hodiny	`long`
`ymin`	Minuty	`long`
`ysec`	Sekundy	`long`
`ynsec`	Nanosekundy	`long`
`ydow`	Den v týdnu	`long`
`ywoy`	Týden v roce	`long`
`tday`	Počet dní od začátku epochy	`long`
`tsec`	Počet sekund od začátku epochy	`long`
`tnsec`	Počet nanosekund od začátku epochy	`large`
`dsec`	Počet sekund od půlnoci	`long`

Parametry

`isetmode`	Zdroj podle kterého nastavit čas
	`1` ..... čas OS
	`2` ..... vstupy bloku
	`3` ..... vstup `unsec`
	`4` ..... vstup `usec`
	`5` ..... vstup `unsec` relativně
		⊙`1`	`long`

`igetmode`	Zdroj ze kterého přečíst čas pro nastavení či konverzi
	`1` ..... čas OS
	`2` ..... vstupy bloku
	`3` ..... vstup `unsec`
	`4` ..... vstup `usec`
	`5` ..... vstup `uday`
	`6` ..... čas systému REX
		⊙`6`	`long`

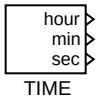
`settol`	Tolerance pro nastavení času systému REX [s]	⊙`1.0`	`double`
`setper`	Perioda nastavování času [s] (0=bez opakování)		`double`
`getper`	Perioda čtení času [s] (0=bez opakování)	⊙`0.001`	`double`
`FDOW`	První den v týdnu je neděle `off` ... týden začíná pondělím `on` .... týden začíná nedělí		`bool`

`tz`	Časové pásmo `1` ..... lokální čas `2` ..... UTC	⊙`1` `long`

TIME – Aktuální čas

Symbol bloku

Licence: STANDARD



Popis funkce

Výstupy bloku TIME odpovídají času operačního systému. Pro pokročilé operace s časem a datem použijte blok DATETIME.

Výstupy

hour	Hodiny	long
min	Minuty	long
sec	Sekundy	long

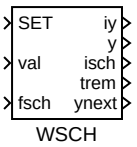
Parametr

tz	Časové pásmo 1 ..... lokální čas 2 ..... UTC	⊙1	long

WSCH – Týdenní časovač

Symbol bloku

Licence: STANDARD



Popis funkce

Blok `WSCH` je určen pro generování týdenních programů, například pro vytápění (den, noc, útlum), větrání (high, low, off), osvětlení, zavlažování apod. Jeho výstupy mohou být využity pro spínání jednotlivých zařízení nebo pro regulaci jejich výkonu.

V běžném provozu jsou v průběhu týdne na výstupech `iy` a `y` generovány hodnoty dle tabulky `wst`, která obsahuje trojice hodnot *den-hodina-hodnota*. Například zápis `[2 6.5 21.5]` znamená, že se v úterý v 6:30 hodin ráno nastaví na výstup `y` hodnota 21.5 a na výstupu `iy` bude hodnota 22 (zaokrouhlení na celé číslo). Jednotlivé trojice hodnot se oddělují středníkem.

Dny jsou číslovány od `1` (pondělí) do `7` (neděle). Vyšší čísla je možno využít pro speciální denní programy, které je možno vynutit pomocí vstupu `fsch` nebo tabulky speciálních dnů `specdays`. Aktuálně platný denní program je indikován výstupem `isch`.

Rovněž je možné dočasně nastavit výstupní hodnotu pomocí vstupů `SET` a `val`. Při náběžné hraně na vstupu `SET` (`off`→`on`) je hodnota `val` zkopírována na výstup `y` a výstup `isch` je přenastaven na hodnotu 0. Ruční hodnota zůstává nastavena, dokud:

- nenastane další přepnutí výstupní hodnoty dle tabulky `wst` nebo
- není přenastavena pomocí další náběžné hrany na vstupu `SET` nebo
- není vynucen jiný denní program pomocí vstupu `fsch`.

Seznam speciálních dní `specdays` lze využít pro vynucení konkrétního denního programu v daný den. Například ve dnech státních svátků můžeme vynutit nedělní režim. Datum se zadává ve formátu `YYYYMMDD`. Zápis `[20160328 7]` tak znamená, že 28. března 2016 se má generovat nedělní program. Jednotlivé dvojice hodnot se oddělují středníkem.

Výstupy `trem` a `ynext` mohou být využity, pokud je potřeba provést nějaké úkony v předstihu ještě před přepnutím výstupních hodnot `iy` a `y`.

Výstup `iy` je určen pro přímé napojení na funkční bloky se vstupy typu Boolean (konverze typu `long` na `bool` se provádí automaticky).

Parametr `nmax` určuje, kolik paměti je alokováno pro pole `wst` a codespecdays. Při `nmax` = 100 může parametr `wst` obsahovat až 100 trojic *den-hodina-hodnota*. Pro běžné použití není potřeba velikost `nmax` měnit.

Vstupy

`SET`	Nastavení výstupů `y` a `iy` pomocí náběžné hrany	`bool`
`val`	Hodnota pro dočasné nastavení výstupů `y` a `iy`	`double`
`fsch`	Vynucený denní program `0` ..... provoz dle týdenního programu `1` ..... pondělí `2` ..... úterý .... `7` ..... neděle `8 a více` další denní programy dle tabulky `wst`	`long`

Výstupy

`iy`	Celočíselná výstupní hodnota	`long`
`y`	Výstupní hodnota	`double`
`isch`	Identifikace denního programu	`long`
`trem`	Čas zbývající v aktuálním intervalu [s]	`double`
`ynext`	Výstupní hodnota v dalším intervalu	`double`

Parametry

`tz`	Časové pásmo ⊙`1` `1` ..... lokální čas `2` ..... UTC	`long`
`nmax`	Velikost alokovaných polí ↓`10` ↑`1000000` ⊙`100`	`long`
`wst`	Tabulka týdenního programu (seznam trojic *den-hodina-hodnota*) ⊙`[1 0.01 18.0; 2 6.0 22.0; 2 18.0 18.0; 3 6.0 22.0; 3 18.0 18.0; 4 6.0 22.0; 4`	`double`
`specdays`	Seznam speciálních dní (seznam dvojic *datum-denní program*) ⊙`[20150406 1; 20151224 1; 20151225 1; 20151226 1; 20160101 1; 20160328 1; 20170`	`long`

Kapitola 10

ARC – Archivace dat

Obsah

Exekutiva reálného času RexCore se skládá z několika vzájemně spolupracujících subsystémů (subsystém reálného času, diagnostický subsystém, subsystém ovladačů, atd.) Jedním z těchto subsystémů je i *archivační subsystém*.

Archivační subsystém slouží k zaznamenávání a uchovávání historie řídícího systému. Funkce archivačního subsystému je předmětem první podkapitoly.

Ve zbývajících dvou podkapitolách jsou popsány bloky spolupracující s archivačním subsystémem řídicího systému REX. Podle funkce lze archivační bloky rozdělit do dvou skupin:

- Bloky pro generování alarmů a událostí
- Bloky pro zaznamenávání trendů
- Bloky pro správu archivů

10.1 Funkce archivačního subsystému

Archiv slouží v řídicím systému REX pro ukládání historie událostí, alarmů a trendů vybraných veličin. Řídicí systém může současně obsluhovat až 15 archivů v každé řídicí stanici. Systém rozlišuje tři druhy archivů:

Archiv v paměti RAM. Vhodný pro krátkodobé ukládání dat. Výhodou je rychlý přístup k uloženým datům, nevýhodou ztráta dat po restartu systému.

Archiv v zálohované paměti. Podobný archivu v paměti RAM. Největší výhodou je zachovávání uložených dat i při opakovaných restartech systému, navíc přístup k datům zůstává velmi rychlý. Nevýhodou může být někdy jeho nepříliš velká kapacita (závisí na konkrétní hardwarové platformě).

Archiv v souboru na disku. Archivy na disku jsou soubory speciálního formátu. Výhodami jsou snadná přenositelnost (kopírování) a zejména velký rozsah dat omezený jen kapacitou disku. Nevýhodou je pomalejší přístup k datům.

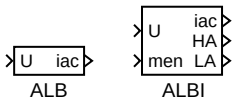
Daná hardwarová platforma nemusí podporovat všechny druhy archivů. Příznaky podporovaných druhů archivů jsou součástí verze řídicího systému cílového zařízení a lze je zjistit v programu RexView po kliknutí na jméno (IP adresu) cílového zařízení ve stromu exekutivy. Nachází se na kartě `Target` v levé spodní části.

10.2 Generování alarmů u a událostí

ALB, ALBI – Alarmy pro logickou hodnotu

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky ALB a ALBI jsou určeny pro generování alarmů nebo událostí při změně logické hodnoty přivedené na vstup U. Výstup iac indikuje stav alarmu (události). Parametr men určuje, při jaké změně vstupu U bude alarm generován. Dále bude popsán blok ALBI. Blok ALB se liší pouze tím, že nemá výstupy HA, LA a men, iACK není jeho vstupem, ale parametrem.

Události a alarmy jsou v systému REX rozlišeny pomocí parametru lvl. Pokud je $1 \leq$ lvl ≤ 127, jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah $128 \leq$ lvl ≤ 255 je určen pro události, u nichž se zapisuje pouze okamžik, kdy daná událost nastala.

Vstupy

U	Logický vstupní signál	bool
men	Povolení alarmů	long
	0 ..... žádný alarm není povolen	
	1 ..... povoleno generování dolního alarmu (LA, sestupná hrana vstupu U)	
	2 ..... povoleno generování horního alarmu (HA, náběžná hrana vstupu U)	
	3 ..... povoleno generování obou alarmů	
iACK	Potvrzení alarmů (při náběžné hraně)	byte
	1 ..... potvrzení dolního alarmu (LA)	
	2 ..... potvrzení horního alarmu (HA)	
	3 ..... potvrzení obou alarmů	

Výstupy

iac	Kód aktuálního stavu alarmového bloku	long
	0 ..... žádný alarm není aktivní	
	1 ..... dolní alarm (LA) je aktivní	
	2 ..... horní alarm (HA) je aktivní	
	256 ... dolní alarm není potvrzený	
	512 ... dolní alarm není potvrzený	

	Kladné hodnoty kódů mohou být sčítány, např. hodnota `514` značí, že nepotvrzený horní alarm. Ne všechny kombinace však mají smysl.	
`HA`	Indikátor horního alarmu	`bool`
`LA`	Indikátor dolního alarmu	`bool`
`NACK`	Indikátor nepotvrzení alarmu	`bool`

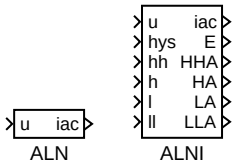
Parametry

`arc`	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. `1,3..5,8`. Událost bude uložena do všech uvedených archivů (detaily o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy `157`, binárně `10011101`.	`word`
`id`	Identifikační kód alarmu v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REX (tzn. ve všech archivačních blocích). ⊙1	`word`
`lvl`	Úroveň (závažnost) alarmu, určující, zda jde o skutečný alarm či jen o událost. ↓1 ⊙1	`byte`
`Desc`	Řetězec blíže specifikující daný alarm či událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REX. ⊙`Alarm Description`	`string`

ALN, ALNI – Alarmy pro číselnou hodnotu

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky ALN a ALNI jsou určeny pro generování dvouúrovňových alarmů nebo událostí při překročení (podkročení) číselné hodnoty vstupu u některé z horních mezí h, hh (dolních mezí l, ll). Výstup iac indikuje stav alarmu (události). Vhodnými alarmovými mezemi lze zvolit, při jaké změně vstupu u bude alarm generován. Dále bude popsán blok ALNI. Blok ALN se liší pouze tím, že nemá výstupy HHA, HA, LA, LLA a místo vstupů hys, hh, h, l, ll, iACK má stejně pojmenované parametry.

Události a alarmy jsou v systému REX rozlišeny pomocí parametru lvl. Pokud je $1 \leq$ lvl ≤ 127, jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah $128 \leq$ lvl ≤ 255 je určen pro události, u nichž se zapisuje pouze okamžik, kdy daná událost nastala.

Vstupy

u	Analogový vstupní signál, podle jehož hodnoty se generují alarmy	double
hys	Velikost hystereze, určující ukončení alarmu. Význam hystereze i ostatních vstupů je dobře patrný z grafu v příkladu k bloku ALNI. ↓0.0 ↑10000000000.0	double
hh	Mez pro druhý horní alarm. Musí být větší než mez h.	double
h	Mez pro horní alarm. Musí být větší než mez l.	double
l	Mez pro dolní alarm. Musí být větší než mez ll.	double
ll	Mez pro druhý dolní alarm	double
iACK	Potvrzení alarmů 1 ..... potvrzení dolního alarmu (LA) 2 ..... potvrzení horního alarmu (HA) 4 ..... potvrzení druhého dolního alarmu (LLA) 8 ..... potvrzení druhého horního alarmu (HHA) Alarm se potvrdí při náběžné hraně. Hodnoty kódů mohou být sčítány, např. hodnota 15 značí potvrzení všech alarmů.	byte

V případě, že stačí daným blokem generovat jen jednoúrovňové alarmy, stačí nastavit lvl2=0. Alternativně je možné druhou horní mez hh nastavit na větší a druhou dolní

mez `ll` na menší hodnotu, než může vstup `u` dosáhnout.

Výstupy

`iac`	Kód aktuálního stavu alarmového bloku `0` ..... žádný alarm není aktivní ani nepotvrzený `1` ..... dolní alarm (`LA`) je aktivní `2` ..... horní alarm (`HA`) je aktivní `4` ..... druhý dolní alarm (`LLA`) je aktivní `8` ..... druhý horní alarm (`HHA`) je aktivní `256` ... dolní alarm (`LA`) není potvrzen `512` ... horní alarm (`HA`) není potvrzen `1024` .. druhý dolní alarm (`LLA`) není potvrzen `2048` .. druhý horní alarm (`HHA`) není potvrzen `-1` .... nesprávné uspořádání alarmových mezí Kladné hodnoty kódů mohou být sčítány, např. hodnota `12` značí, že současně probíhají oba horní alarmy. Ne všechny kombinace však mají smysl.	`long`
`E`	Příznak chyby uspořádání alarmových mezí `off` ... bez chyby `on` .... nastala chyba	`bool`
`HHA`	Indikátor druhého horního alarmu	`bool`
`HA`	Indikátor (prvního) horního alarmu	`bool`
`LA`	Indikátor (prvního) dolního alarmu	`bool`
`LLA`	Indikátor druhého dolního alarmu	`bool`
`NACK`	Indikátor nepotvrzení alarmu	`bool`

Parametry

`acls`	Třída alarmu (typ proměnné, která bude do archivu ukládána) ⊙`8` `1` ..... Bool `4` ..... Long `7` ..... Float `2` ..... Byte `5` ..... Word `8` ..... Double `3` ..... Short `6` ..... DWord `9` ..... Time	`byte`
`arc`	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. `1,3..5,8`. Událost bude uložena do všech uvedených archivů (detaily o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy `157`, binárně `10011101`.	`word`
`id`	Identifikační kód alarmu v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REX (tzn. ve všech archivačních blocích). ⊙`1`	`word`
`lvl1`	Úroveň (závažnost) prvních horních a dolních alarmů (`HA` a `LA`), určující, zda jde o skutečný alarm či jen o událost ↓`1` ⊙`1`	`byte`
`lvl2`	Úroveň (závažnost) druhých horních a dolních alarmů (`HHA` a `LLA`) ↓`1` ⊙`10`	`byte`
`Desc`	Řetězec blíže specifikující daný alarm či událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REX. ⊙`Alarm Description`	`string`

10.3 Záznam trendů

ACD – Archivní komprese s použitím delta kritéria

Symbol bloku

Licence: STANDARD

Popis funkce

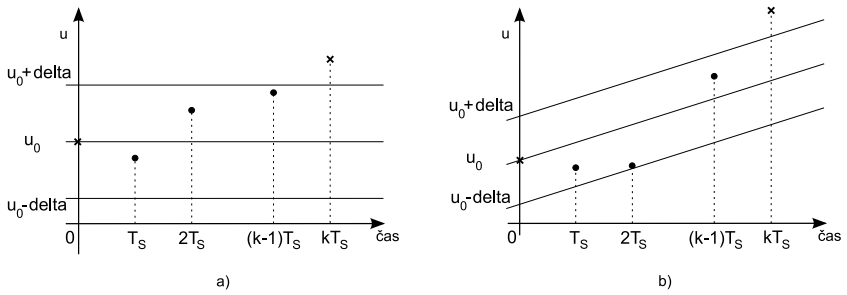
Blok `ACD` (Archive Compression using Delta criterion) je určen pro ukládání komprimovaných analogových signálů do archivu pomocí archivních událostí.

Základní myšlenkou bloku je archivovat vstupní signál `u` jen tehdy, pokud se mění. Doba mezi uložením dvou po sobě následujících hodnot signálu je v intervalu ⟨`tmin`,`tmax`⟩ sekund (doby jsou zaokrouhleny na nejbližší násobek periody vzorkování). Pokud se hodnota signálu „hodně“ mění, ukládá se signál jednou za čas `tmin`, pokud se hodnota signálu mění „málo“ nebo je konstantní, ukládá se signál jednou za čas `tmax`. Po spuštění bloku se vždy uloží první hodnota vstupu `u`, označme ji `u0`. Přesná pravidla ukládání dalších vzorků jsou určena vstupem `delta` a parametrem `TR`.

Je-li `TR`=`off`, testuje se podmínka $|\texttt{u} - \texttt{u0}| > \texttt{delta}$. Pokud je splněna a od minulého uložení uplynul alespoň čas `tmin` uloží se tato hodnota `u` do archivu a nastaví se `u0`=`u`. Je-li podmínka splněna dříve než za čas `tmin` od posledního uložení nastaví se chybový výstup `E` na `1` a počká se s uložením na první vzorek po uplynutí času `tmin`, v tomto okamžiku se nastavuje `E`=`0`. Pak se celý postup opakuje od začátku.

Je-li `TR`=`on`, pracuje blok tak, že ukládá první vzorek, který se odchyluje o více než toleranci delta od signálu s kompenzovaným trendem. Podmínka na minimální čas ukládání platí obdobně jako v předcházejícím případě.

Chování bloku v obou případech ukazuje následující obrázek: a) pro `TR`=`off`, b) pro `TR`=`on`. Ukládané vzorky jsou označeny symbolem ×.



Vstupy

`u`	Komprimovaně ukládaný signál		`double`
`delta`	Práh pro ukládání signálu do archivu	↓`0.0` ↑`10000000000.0`	`double`

Výstupy

`y`	Poslední hodnota uložená do archivu	`double`
`E`	Příznak chyby – nastaven, pokud by měl být vstup `u` uložen dřív než za čas `tmin` `off` ... bez chyby `on` .... nastala chyba	`bool`

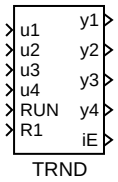
Parametry

`acls`	Třída alarmu, určující typ proměnné, která bude do archivu ukládána ⊙`8` `1` ..... Bool `4` ..... Long `7` ..... Float `2` ..... Byte `5` ..... Word `8` ..... Double `3` ..... Short `6` ..... DWord `9` ..... Time	`byte`
`arc`	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. `1,3..5,8`. Událost bude uložena do všech uvedených archivů (detaily o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy `157`, binárně `10011101`.	`word`
`id`	Identifikační kód události v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REX (tzn. ve všech archivačních blocích). ⊙`1`	`word`
`tmin`	Nejkratší čas (v sekundách) mezi dvěma uloženími hodnoty vstupu `u` do archivu ↓`0.001` ↑`1000000.0` ⊙`1.0`	`double`
`tmax`	Nejdelší čas (v sekundách) mezi dvěma uloženími hodnoty vstupu `u` do archivu ↓`1.0` ↑`1000000.0` ⊙`1000.0`	`double`
`TR`	Příznak vyhodnocování trendu signálu. Pro `TR` = `off` se vyhodnocuje odchylka od poslední uložené hodnoty, v případě `TR` = `on` odchylka od trendu posledně uložené hodnoty. ⊙`on` `off` ... vyhodnocuje se odchylka od poslední uložené hodnoty `on` .... vyhodnocuje se odchylka od trendu posledně uložené hodnoty	`bool`
`Desc`	Řetězec blíže specifikující danou událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REX. ⊙`Value Description`	`string`

TRND – Záznam trendů v reálném čase

Symbol bloku

Licence: STANDARD



Popis funkce

Blok `TRND` slouží pro ukládání průběhů až čtyř vstupních signálů `u1` až `u4` do cyklických trendových bufferů v paměti cílového zařízení (target). Výhodou bloku `TRND` je synchronní ukládání dat s během exekutivy reálného času, které umožňuje ukládat do trendu i velmi rychlé signály. Na rozdíl od asynchronního ukládání dat na nadřazeném operátorském počítači (host) nedochází ke ztrátě některých vzorků nebo jejich vícenásobnému uložení. Data lze blokem `TRND` ukládat i pro velmi krátké periody spouštění úloh.

Skutečný počet ukládaných průběhů určuje parametr `n`. V případě, že se trendové buffery s délkou `l` vzorků zaplní, začnou se přepisovat nejstarší vzorky. Do trendových bufferů se mohou ukládat data jednou za `pfac` spuštění bloku (decimace) a ukládaná data mohou být zpracována podle hodnoty parametrů `ptype1` až `ptype4`. Další decimace s faktorem `afac` může být použita pro ukládání do archivů.

Pro úsporu paměti na cílovém zařízení může být parametrem `btype` specifikován typ použitých trendových bufferů. Velikost paměti obsazená trendovými buffery je dána vztahem $s \cdot \mathtt{n} \cdot \mathtt{l}$, kde s je velikost proměnné daného typu v bytech. Přednastavený typ `Double` zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů $\mathtt{n} = 4$, délka každého trendu $\mathtt{l} = 1000$, pak pro typ `Double` je zapotřebí $8 \cdot 4 \cdot 1000 = 32000$ bytů. V případě, že by byly vstupní signály měřeny z A/D převodníku s rozlišením do 16 bitů, mohly by být ukládány v typu `Word` s velikostí 2 byty na vzorek a velikost potřebné paměti by se zmenšila na jednu čtvrtinu. Velikosti jednotlivých datových typů a jejich rozsahy jsou uvedeny v tabulce 1.1 na straně 12.

Při použití jiného typu pro trendové buffery než je typ `Double` může nastat případ, že se zpracovaná hodnota některého vstupu „nevejde“ do zvoleného typu bufferu a má hodnotu menší (větší) než je nejmenší (největší) zobrazitelné číslo v daném typu. V takovém případě se do bufferu uloží nejmenší (největší) zobrazitelné číslo v daném typu a chyba se binárně zakóduje do chybového výstupu `iE` podle následující tabulky (nepoužité bity jsou vypuštěny):

Chyba	Podkročení rozsahu				Překročení rozsahu			
Vstup	u4	u3	u2	u1	u4	u3	u2	u1
Číslo bitu	11	10	9	8	3	2	1	0
Váha bitu	2048	1024	512	256	8	4	2	1

V případě, že nastane najednou několik chyb, je výsledný chybový kód dán součtem vah jednotlivých chyb. Poznamenejme, že současné překročení a podkročení rozsahu na daném vstupu nemohou nastat zároveň.

Číst, zobrazovat a exportovat průběžně ukládaná data umožňuje diagnostický program RexView.

Vstupy

u1..u4	Analogové vstupy bloku určené pro zpracování a ukládání do trendu	double
RUN	Povolení běhu algoritmu. Data se zpracovávají a ukládají jen pokud je RUN = on.	bool
R1	Signál pro vymazání obsahu trendového bloku. Data jsou mazána při každém spuštění bloku, je-li R1 = on. Vstup má přednost před vstupem RUN.	bool

Výstupy

y1..y4	Analogové výstupy bloku nastavované jednou za pfac spuštění bloku na poslední hodnoty uložené do trendových bufferů	double
iE	Kód chyby ukládání do trendových bufferů, viz popis v textu výše	long

Parametry

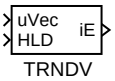
n	Počet signálů (bufferů) v trendu ↓1 ↑4 ⊙4	long
l	Počet vzorků vyhrazený v paměti pro každý buffer trendu ↓0 ↑268435000 ⊙1000	long
btype	Typ všech použitých bufferů trendu ⊙8 1 ..... Bool 4 ..... Long 7 ..... Float 2 ..... Byte 5 ..... Word 8 ..... Double 3 ..... Short 6 ..... DWord 10 .... Large	long
ptype*i*	Způsob zpracování signálu u*i*, $i = 1 \ldots 4$. Zvolený způsob se aplikuje na posledních pfac vzorků a výsledek se uloží do i-tého trendového bufferu ⊙1 1 ..... ukládání bez zpracování 2 ..... minimum z pfac vzorků 3 ..... maximum z pfac vzorků 4 ..... součet pfac vzorků 5 ..... aritmetický průměr z pfac vzorků 6 ..... směrodatná odchylka pfac vzorků 7 ..... rozptyl pfac vzorků	long

`pfac`	Násobek periody spouštění bloku pro uložení zpracovaných hodnot do trendových bufferů. Pokud je vstup `RUN` = `on`, ukládají se zpracovaná data do trendu s periodou `pfac`·T_S, kde T_S je perioda spouštění bloku ve vteřinách. ↓1 ↑1000000 ⊙1	`long`
`afac`	Archivační faktor je číslem udávajícím po kolika uložených vzorcích do trendu se mají ukládané hodnoty navíc uložit do archivů zadaných příznaky `arc`. Je-li `afac` = 0, neukládají se trendy do žádného archivu, jinak se ukládají s periodou `afac`·`pfac`·T_S, kde T_S je perioda spouštění bloku ve vteřinách. ↓0 ↑1000000	`long`
`arc`	Seznam archivů, kam budou ukládána data z trendu. Zadává se ve tvaru např. `1,3..5,8`. Data budou uložena do všech uvedených archivů (detaily o číslování archivů viz blok `ARC`. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy `157`, binárně `10011101`.	`word`
`id`	Identifikační kód trendu v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REX (tzn. ve všech archivačních blocích). ⊙1	`word`
`Title`	Text hlavičky trendu pro zobrazení v diagnostických nástrojích systému REX, např. v programu RexView ⊙`Trend Title`	`string`
`timesrc`	Zdroj časových značek. Součástí každého vzorku v trendovém bufferu je časová značka. Pro rychlé nebo krátkodobé trendy, kde nás zajímá přesný čas mezi vzorky odpovídající periodě spouštění úlohy spíše než absolutní čas, vybereme `CORETIMER` – interní technologický čas systému REX, který je inkrementován o nominální periodu s každým základním tikem. Pro dlouhodobé trendy, kde nás zajímá spíše absolutní čas sdílený s operačním systémem (a případně synchronizovaný přes NTP), vybereme `RTC`. Ostatní volby jsou určeny pouze pro ladicí nebo speciální účely. ⊙1 `1` ..... `CORETIMER` – technologický čas – aktuální tick `2` ..... `CORETIMER-PRECISE` – technologický čas – při spuštění bloku `3` ..... `RTC` – reálný čas z operačního systému – aktuální tick `4` ..... `RTC-PRECISE` – reálný čas z operačního systému – pří spuštění bloku `4` ..... `PFC` – hrubý čas s vysokým rozlišením (PerFormanceCounter)	`long`

TRNDV – Záznam trendů v reálném čase (vektorová forma)

Symbol bloku

Licence: STANDARD



Popis funkce

Blok TRNDV slouží pro ukládání průběhů vstupních signálů, které jsou bloku předávány ve vektorové podobě. Narozdíl od bloku TRND tedy umožňuje současné ukládání více než 4 signálů, konkrétně je jejich počet určen pomocí parametru n. Signály jsou ukládány do cyklických trendových bufferů v paměti cílového zařízení (target). Výhodou bloku TRNDV je synchronní ukládání dat s během exekutivy reálného času, které umožňuje ukládat do trendu i velmi rychlé signály. Na rozdíl od asynchronního ukládání dat na nadřazeném operátorském počítači (host) nedochází ke ztrátě některých vzorků nebo jejich vícenásobnému uložení. Data lze blokem TRNDV ukládat i pro velmi krátké periody spouštění úloh.

V případě, že se trendové buffery s délkou l vzorků zaplní, začnou se přepisovat nejstarší vzorky. Do trendových bufferů se mohou ukládat data jednou za pfac spuštění bloku (decimace). Další decimace s faktorem afac může být použita pro ukládání do archivů.

Pro úsporu paměti na cílovém zařízení může být parametrem btype specifikován typ použitých trendových bufferů. Velikost paměti obsazená trendovými buffery je dána vztahem $s \cdot \mathtt{n} \cdot \mathtt{l}$, kde s je velikost proměnné daného typu v bytech. Přednastavený typ Double zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů $\mathtt{n} = 4$, délka každého trendu $\mathtt{l} = 1000$, pak pro typ Double je zapotřebí $8 \cdot 4 \cdot 1000 = 32000$ bytů. V případě, že by byly vstupní signály měřeny z A/D převodníku s rozlišením do 16 bitů, mohly by být ukládány v typu Word s velikostí 2 byty na vzorek a velikost potřebné paměti by se zmenšila na jednu čtvrtinu. Velikosti jednotlivých datových typů a jejich rozsahy jsou uvedeny v tabulce 1.1 na straně 12.

Číst, zobrazovat a exportovat průběžně ukládaná data umožňuje diagnostický program RexView.

Vstupy

uVec	Vektorový signál určený k uložení	reference
HLD	Pozastavení ukládání dat do cyklických bufferů, při HLD = on se neukládají žádná data	bool

Výstup

`iE`	Kód chyby `i` ..... obecná chyba systému REX	`error`

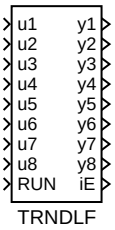
Parametry

`n`	Počet signálů (bufferů) v trendu ↓1 ↑64 ⊙8	`long`
`l`	Počet vzorků pro každý buffer trendu ↓2 ↑268435000 ⊙1000	`long`
`btype`	Typ všech použitých bufferů trendu ⊙8 `1` ..... Bool `4` ..... Long `7` ..... Float `2` ..... Byte `5` ..... Word `8` ..... Double `3` ..... Short `6` ..... DWord `10` .... Large	`long`
`pfac`	Násobek periody spouštění bloku pro uložení zpracovaných hodnot do trendových bufferů. Pokud je vstup `RUN` = `on`, ukládají se zpracovaná data do trendu s periodou `pfac`·T_S, kde T_S je perioda spouštění bloku ve vteřinách. ↓1 ↑1000000 ⊙1	`long`
`afac`	Archivační faktor je číslem udávajícím po kolika uložených vzorcích do trendu se mají ukládané hodnoty navíc uložit do archivů zadaných příznaky `arc`. Je-li `afac` = 0, neukládají se trendy do žádného archivu, jinak se ukládají s periodou `afac`·`pfac`·T_S, kde T_S je perioda spouštění bloku ve vteřinách. ↓0 ↑1000000	`long`
`arc`	Seznam archivů, kam budou ukládána data z trendu. Zadává se ve tvaru např. `1,3..5,8`. Data budou uložena do všech uvedených archivů (detaily o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy `157`, binárně `10011101`.	`word`
`id`	Identifikační kód trendu v archivu. Tento kód musí být volen jednoznačně v celé stanici s řídicím systémem REX (tzn. ve všech archivačních blocích). ⊙1	`word`
`Title`	Text hlavičky trendu pro zobrazení v diagnostických nástrojích systému REX, např. v programu RexView ⊙`Trend Title`	`string`

TRNDLF – * Záznam trendů v reálném čase (lock-free)

Symbol bloku

Licence: ADVANCED



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

u1	První analogový vstup bloku	double
u2	Druhý analogový vstup bloku	double
u3	Třetí analogový vstup bloku	double
u4	Čtvrtý analogový vstup bloku	double
u5	Pátý analogový vstup bloku	double
u6	Šestý analogový vstup bloku	double
u7	Sedmý analogový vstup bloku	double
u8	Osmý analogový vstup bloku	double
RUN	Povolení běhu algoritmu	bool

Parametry

n	Počet signálů (bufferů) v trendu	↓1 ↑8 ⊙8	long
l	Počet vzorků pro každý buffer trendu	↓0 ↑268435000 ⊙1024	long
btype	Typ všech použitých bufferů	⊙8	long

1 ..... Bool
2 ..... Byte
3 ..... Short
4 ..... Long
5 ..... Word
6 ..... DWord
7 ..... Float
8 ..... Double
-- ....
10 .... Large

`Title`	Název trendu	⊙`Trend Title`	`string`
`timesrc`	Zdroj časových značek	⊙1	`long`

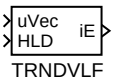
Výstupy

`y1`	První analogový výstup bloku	`double`
`y2`	Druhý analogový výstup bloku	`double`
`y3`	Třetí analogový výstup bloku	`double`
`y4`	Čtvrtý analogový výstup bloku	`double`
`y5`	Pátý analogový výstup bloku	`double`
`y6`	Šestý analogový výstup bloku	`double`
`y7`	Sedmý analogový výstup bloku	`double`
`y8`	Osmý analogový výstup bloku	`double`
`iE`	Kód chyby ukládání (po bitech)	`long`

TRNDVLF – * Záznam trendů v reálném čase (pro vektory, lock-free)

Symbol bloku

Licence: ADVANCED



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

`uVec`	Vektorový signál určený k uložení		`reference`
`HLD`	Pozastavení		`bool`

Parametry

`n`	Počet signálů (bufferů) v trendu	↓1 ↑64 ⊙8	`long`
`l`	Počet vzorků pro každý buffer trendu	↓2 ↑268435000 ⊙1024	`long`
`btype`	Typ všech použitých bufferů	⊙8	`long`
	`1` ..... Bool		
	`2` ..... Byte		
	`3` ..... Short		
	`4` ..... Long		
	`5` ..... Word		
	`6` ..... DWord		
	`7` ..... Float		
	`8` ..... Double		
	`--`		
	`10` .... Large		
`Title`	Název trendu	⊙`Trend Title`	`string`
`timesrc`	Zdroj časových značek	⊙`1`	`long`

Výstup

`iE`	Kód chyby	`error`
	`i` ..... obecná chyba systému REX	

10.4 Správa archivů

AFLUSH – Vynucené zapsání archivu

Symbol bloku Licence: STANDARD

Popis funkce

Blok AFLUSH slouží k vynucenému zapsání dat archivu na disk v situaci, kdy hrozí vypnutí napájení řídicího systému, které by vedlo ke ztrátě archivních dat, která ještě nebyla uložena na disk. V okamžiku náběžné hrany na vstupu FLUSH (off→on) se uloží data na disk bez ohledu na nastavení parametru period bloku ARC.

Vstup

FLUSH	Vynucené zapsání archivů	bool

Parametr

arc	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. 1,3..5,8. Událost bude uložena do všech uvedených archivů (detaily o číslování archivů viz blok ARC. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	word

Kapitola 11

STRING – Bloky pro práci s řetězci

Obsah

CNS – * Textová konstanta

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Parametry

scv	Textová hodnota		string
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long

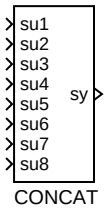
Výstup

sy	Výstupní textová hodnota	string

CONCAT – * Spojení stringů (podle vzoru)

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

su1	Vstupní textová hodnota		string
su2	Vstupní textová hodnota		string
su3	Vstupní textová hodnota		string
su4	Vstupní textová hodnota		string
su5	Vstupní textová hodnota		string
su6	Vstupní textová hodnota		string
su7	Vstupní textová hodnota		string
su8	Vstupní textová hodnota		string

Parametry

ptrn	0	⊙%1%2%3%4	string
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long

Výstup

sy	Výstupní textová hodnota	string

FIND – * Nalezení textu

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

su1	Vstupní textová hodnota		string
su2	Vstupní textová hodnota		string

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long

Výstup

pos	Poloha hledaného textu	long

LEN – * Délka textu

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstup

su	Vstupní textová hodnota		string

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long

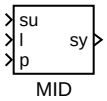
Výstup

len	Délka vstupního textu	long

MID – * Výřez textu

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

su	Vstupní textová hodnota		string
l	Délka výstupního textu		long
p	Pozice výstupního textu		long

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long

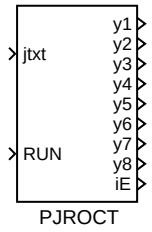
Výstup

sy	Výstupní textová hodnota	string

PJROCT – * Získání číselných hodnot z textu ve formátu JSON

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

jtxt	Text v JSON formátu		string
RUN	Povolení běhu algoritmu		bool

Parametry

name1	Jméno objektu v textu formátu JSON		string
name2	Jméno objektu v textu formátu JSON		string
name3	Jméno objektu v textu formátu JSON		string
name4	Jméno objektu v textu formátu JSON		string
name5	Jméno objektu v textu formátu JSON		string
name6	Jméno objektu v textu formátu JSON		string
name7	Jméno objektu v textu formátu JSON		string
name8	Jméno objektu v textu formátu JSON		string
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long
yerr	Náhradní hodnota pro případ chyby		double

Výstupy

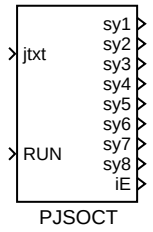
y1	Výstup bloku	double
y2	Výstup bloku	double
y3	Výstup bloku	double
y4	Výstup bloku	double
y5	Výstup bloku	double
y6	Výstup bloku	double

`y7`	Výstup bloku	`double`
`y8`	Výstup bloku	`double`
`iE`	Kód chyby	`error`

PJSOCT – * Získání textových hodnot z textu ve formátu JSON

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

jtxt	Text v JSON formátu		string
RUN	Povolení běhu algoritmu		bool

Parametry

name1	Jméno objektu v textu formátu JSON		string
name2	Jméno objektu v textu formátu JSON		string
name3	Jméno objektu v textu formátu JSON		string
name4	Jméno objektu v textu formátu JSON		string
name5	Jméno objektu v textu formátu JSON		string
name6	Jméno objektu v textu formátu JSON		string
name7	Jméno objektu v textu formátu JSON		string
name8	Jméno objektu v textu formátu JSON		string
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long

Výstupy

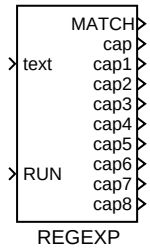
sy1	Výstupní textová hodnota		string
sy2	Výstupní textová hodnota		string
sy3	Výstupní textová hodnota		string
sy4	Výstupní textová hodnota		string
sy5	Výstupní textová hodnota		string
sy6	Výstupní textová hodnota		string
sy7	Výstupní textová hodnota		string

`sy8`	Výstupní textová hodnota	`string`
`iE`	Kód chyby	`error`

REGEXP – Regular expresion parser

Symbol bloku

Licence: ADVANCED

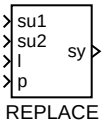


Popis funkce

REPLACE – * Náhrada textu

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

su1	Vstupní textová hodnota		string
su2	Vstupní textová hodnota		string
l	Délka původního textu		long
p	Pozice původního textu		long

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long

Výstup

sy	Výstupní textová hodnota	string

RTOS – * Konverze čísla na text

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstup

`u`	Analogový vstupní signál		`double`

Parametry

`prec`	Přesnost (počet cifer)	↓0 ↑20	`long`
`mode`	Formát výstupního textu `1` ..... nejvhodnější `2` ..... normální `3` ..... exponenciální	⊙1	`long`

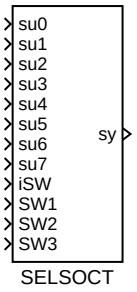
Výstup

`sy`	Výstupní textová hodnota	`string`

SELSOCT – * Výběr textu z několika vstupů

Symbol bloku

Licence: STANDARD



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

su0	Vstupní textová hodnota		string
su1	Vstupní textová hodnota		string
su2	Vstupní textová hodnota		string
su3	Vstupní textová hodnota		string
su4	Vstupní textová hodnota		string
su5	Vstupní textová hodnota		string
su6	Vstupní textová hodnota		string
su7	Vstupní textová hodnota		string
iSW	Selektor aktivního signálu		long
SW1	Binární vstup pro výběr		bool
SW2	Binární vstup pro výběr		bool
SW3	Binární vstup pro výběr		bool

Parametry

BINF	Výběr pomocí binárních vstupů		bool
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	long

Výstup

sy	Zvolený vstupní signál	string

STOR – * Koverze textu na číslo

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstup

`su`	Vstupní textová hodnota	`string`

Parametr

`yerr`	Náhradní hodnota pro případ chyby	`double`

Výstupy

`y`	Analogový výstupní signál	`double`
`E`	Příznak chyby	`bool`

Kapitola 12

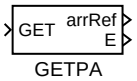
PARAM – Bloky pro manipulaci s parametry

Obsah

GETPA – Blok pro vzdálené získání vektorového parametru

Symbol bloku

Licence: STANDARD



Popis funkce

Blok GETPA slouží ke vzdálenému získávání vektorových parametrů ostatních bloků v modelu. Může pracovat ve dvou režimech, které se přepínají parametrem GETF. Pro GETF = off je na výstup arrRef vyveden vzdálený vektorový parametr při startu a dále pak při každé změně sledovaného vzdáleného parametru. Jestliže parametr GETF je on, pak bloky pracují v režimu jednorázového čtení vzdáleného parametru, který se přečte vždy, když nastane náběžná hrana (off→on) na vstupu GET.

Jméno vzdáleného parametru určuje textový parametr sc (string connection), který se zadává ve tvaru <cesta_k_bloku:jmeno_parametru>. Cesta k bloku, jehož parametr má být získán, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- Relativní – začíná v úrovni, do které je umístěn blok GETPA. V tomto případě text začíná znakem '.'. Příklady hodnot relativních cest: ".CNDR:yp", ".Lights.ATMT:touts".
- Absolutní – úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok IOTASK), je v první úrovni hierarchie uveden znak '&' následovaný názvem ovladače. Příklady hodnot absolutních cest: "uloha1.vstupy.ATMT:touts", "&EfaDrv.mereni.CNDR:yp".

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v programu RexView.

Vstup

GET	Vstup pro jednorázové přečtení parametru	bool

Výstupy

arrRef	Odkaz na pole (vektor nebo matice)	reference
E	Příznak chyby	bool

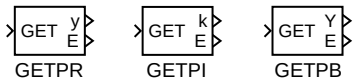
Parametry

`sc`	Jméno vzdáleného parametru		`string`
`GETF`	Načtení parametru pouze po vyžádání `off` ... režim průběžného čtení parametru `on` .... režim jednorázového přečtení parametru po náběžné hraně na vstupu `GET`		`bool`
`nmax`	Maximální velikost vektoru (pole)	⊙256	`long`

GETPR, GETPI, GETPB – Bloky pro vzdálené získání parametru

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky `GETPR`, `GETPI` a `GETPB` slouží pro vzdálené získávání parametrů ostatních bloků v modelu. Bloky mají identickou funkci, liší se pouze v typu parametru, který získávají. Blok `GETPR` je pro reálné číslo, `GETPI` pro celé číslo a `GETPB` pro Booleovskou hodnotu.

Bloky mohou pracovat ve dvou režimech, které se přepínají parametrem `GETF`. Pro `GETF` = `off` je hodnota výstupu `y` (nebo `k`, `Y`) nastavena na hodnotu vzdáleného parametru při startu a dále pak při každé změně sledovaného vzdáleného parametru. Jestliže parametr `GETF` je `on`, pak bloky pracují v režimu jednorázového čtení vzdáleného parametru, který se přečte vždy, když nastane náběžná hrana (`off`→`on`) na vstupu `GET`.

Jméno vzdáleného parametru určuje textový parametr `sc` (string connection), který se zadává ve tvaru `<cesta_k_bloku:jmeno_parametru>`. Rovněž je možné přistupovat k jednotlivým prvkům parametrů typu pole (např. parametr `tout` bloku ATMT). Toho se dosáhne pomocí hranatých závorek a čísla prvku, např. tedy `.ATMT:touts[2]`, číslování je od 0, uvedený propojovací řetězec tedy odkazuje na třetí prvek pole.

Cesta k bloku, jehož parametr má být získán, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- Relativní – začíná v úrovni, do které je umístěn daný blok `GETPR` (nebo `GETPI`, `GETPB`). V tomto případě text začíná znakem `'.'`. Příklady hodnot relativních cest: `".GAIN:k"`, `".Motor1.Poloha:ycn"`.
- Absolutní – úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok IOTASK), je v první úrovni hierarchie uveden znak `'&'` následovaný názvem ovladače. Příklady hodnot absolutních cest: `"uloha1.vstupy.lin1:u2"`, `"&EfaDrv.mereni.DER1:n"`.

Poznámka: Od verze řídicího systému REX 2.7 došlo ke změně práce s absolutními a relativními cestami. Ve starších verzích měla absolutní cesta prefix `'´` a relativní cesta neměla prefix žádný. Ke změně bylo přistoupeno z důvodu sjednocení formátu cest s blokem SGSLP. Z důvodu maximální možné kompatibility se staršími verzemi je znak `'´` na začátku řetězců ignorován, je však doporučeno cesty aktualizovat.

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v programu RexView.

Vstup

`GET`	Vstup pro jednorázové přečtení parametru	`bool`

Výstupy

`y`	Hodnota parametru, výstup bloku `GETPR`	`double`
`k`	Hodnota parametru, výstup bloku `GETPI`	`long`
`Y`	Hodnota parametru, výstup bloku `GETPB`	`bool`
`E`	Příznak chyby `off` ... bez chyby `on` .... nastala chyba	`bool`

Parametry

`sc`	Jméno vzdáleného parametru podle výše uvedených pravidel	`string`
`GETF`	Zapnutí manuálního čtení vzdáleného parametru `off` ... režim průběžného čtení parametru `on` .... režim jednorázového přečtení parametru po náběžné hraně na vstupu `GET`	`bool`

GETPS – * Blok pro vzdálené získání parametru typu string

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstup

GET	Vstup pro jednorázové přečtení parametru	bool

Parametry

sc	Jméno vzdáleného parametru	string
GETF	Načtení parametru pouze po vyžádání off ... režim průběžného čtení parametru on .... režim jednorázového přečtení parametru	bool
nmax	Rezervovaná paměť pro řetězec	long

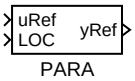
Výstupy

sy	Hodnota parametru	string
E	Příznak chyby off ... bez chyby on .... nastala chyba	bool

PARA – Blok s vektorovým parametrem nastavitelným ze vstupu

Symbol bloku

Licence: STANDARD



Popis funkce

Blok PARA je speciální blok, který kromě klasické metody zadávání svých parametrů umožnuje změnu jednoho svého parametru změnou vstupu. Parametr apar se může měnit podle vstupu uRef.

Logický vstup LOC (LOCal) určuje, zda bude hodnota vnitřního parametru apar čtena ze vstupu uRef, v tomto případě je LOC = off, nebo hodnota vnitřního parametru nebude na vstupu závislá (LOC = on). Pokud je blok v lokálním režimu LOC = on, je ve vnitřním parametru apar uložena poslední hodnota, která byla na vstupu uRef těsně před tím, než byl aktivován lokální režim (LOC = off → on).

Výstupní hodnota je shodná s hodnotou parametru yRef = apar.

Vstupy

uRef	Odkaz na pole (vektor nebo matice)	reference
LOC	Aktivace lokálního režimu off ... parametr je ovládán vstupním signálem on .... lokální režim aktivován	bool

Výstup

yRef	Odkaz na pole (vektor nebo matice)	reference

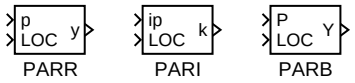
Parametry

SETS	Nastavení velikosti pole. Použijte tento příznak pro úpravu velikosti pole při nastavování vektorového parametru.		bool
apar	Počáteční hodnota parametru	⊙[0.0 1.0 2.0 3.0 4.0 5.0]	double

PARR, PARI, PARB – Bloky s nastavitelným parametrem ze vstupu

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky PARR, PARI a PARB jsou speciální bloky, které kromě klasické metody zadávání svých parametrů umožňují změnu jednoho svého parametru změnou vstupu. U bloku PARR změnu parametru par změnou vstupu p, u PARI změnu ipar vstupem ip a u PARB změnu PAR vstupem P.

Logický vstup LOC (LOCal) určuje, zda bude hodnota vnitřního parametru par (nebo ipar, PAR) čtena ze vstupu p (nebo ip, P), v tomto případě je LOC = off, nebo hodnota vnitřního parametru nebude na vstupu závislá (LOC = on). Pokud je blok v lokálním režimu LOC = on, je ve vnitřním parametru par (nebo ipar, PAR) uložena poslední hodnota, která byla na vstupu p (nebo ip, P) těsně před tím, než byl aktivován lokální režim (LOC = off → on).

Výstupní hodnota je shodná s hodnotou parametru y = par, (nebo k = ipar, Y = PAR). Bloky PARR a PARI mají navíc možnost omezení výstupního signálu y a k saturačními mezemi ⟨lolim, hilim⟩. Saturační omezení je uvažováno pouze v případě SATF = on.

Vstupy

p	Hodnota parametru (blok PARR)	double
ip	Hodnota parametru (blok PARI)	long
P	Hodnota parametru (blok PARB)	bool
LOC	Aktivace lokálního režimu off ... parametr je ovládán vstupním signálem on .... lokální režim aktivován	bool

Výstup

y	Analogový výstupní signál bloku PARR	double
k	Celočíselný výstupní signál bloku PARI	long
Y	Logický výstupní signál bloku PARB	bool

Parametry

par	Počáteční hodnota parametru bloku PARR	⊙1.0	double

`ipar`	Počáteční hodnota parametru bloku `PARI`	⊙`1`	`long`
`PAR`	Počáteční hodnota parametru bloku `PARB`	⊙`on`	`bool`
`SATF`	Omezení výstupu pro bloky `PARR` a `PARI` `off` ... signál není omezen `on` .... saturační meze jsou aktivní		`bool`
`hilim`	Horní saturační mez (bloky `PARR` a `PARI`)	⊙`1.0`	`double`
`lolim`	Dolní saturační mez (bloky `PARR` a `PARI`)	⊙`-1.0`	`double`

PARS – * Blok s parametrem typu string nastavitelným ze vstupu

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

sp	Hodnota parametru	string
LOC	Aktivace lokálního režimu	bool

Parametry

spar	Počáteční hodnota parametru	string
nmax	Rezervovaná paměť pro řetězec	long

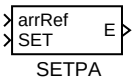
Výstup

sy	Výstupní řetězec	string

SETPA – Blok pro vzdálené nastavování vektorového parametru

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SETPA slouží ke vzdálenému nastavování vektorových parametrů ostatních bloků v modelu. Může pracovat ve dvou režimech, které se přepínají parametrem SETF. Pro SETF = off je hodnota vzdáleného parametru sc nastavena na hodnotu vstupního vektoru arrRef při startu a dále pak při každé změně vstupního signálu. Jestliže parametr SETF je on, pak blok pracuje v režimu jednorázového zápisu vzdáleného parametru, který se nastaví vždy, když nastane náběžná hrana (off→on) na vstupu SET.

Jméno vzdáleného parametru určuje textový parametr sc (string connection), který se zadává ve tvaru <cesta_k_bloku:jmeno_parametru>. Cesta k bloku, jehož parametr má být získán, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- Relativní – začíná v úrovni, do které je umístěn blok GETPA. V tomto případě text začíná znakem '.'. Příklady hodnot relativních cest: ".CNDR:yp", ".Lights.ATMT:touts".
- Absolutní – úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok IOTASK), je v první úrovni hierarchie uveden znak '&' následovaný názvem ovladače. Příklady hodnot absolutních cest: "uloha1.vstupy.ATMT:touts", "&EfaDrv.mereni.CNDR:yp".

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v programu RexView.

Vstupy

arrRef	Odkaz na pole (vektor nebo matice)	reference
SET	Vstup pro jednorázový zápis parametru	bool

Výstup

E	Příznak chyby	bool

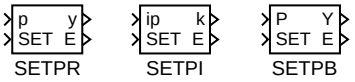
Parametry

`sc`	Jméno vzdáleného parametru	`string`
`SETF`	Nastavení parametru pouze na vyžádání `off` ... režim průběžného nastavování parametru `on` .... režim jednorázového nastavení parametru po náběžné hraně na vstupu `SET`	`bool`
`SETS`	Nastavení velikosti pole. Použijte tento příznak pro úpravu velikosti pole při nastavování vektorového parametru.	`bool`

SETPR, SETPI, SETPB – Bloky pro vzdálené nastavování parametru

Symboly bloků

Licence: STANDARD



Popis funkce

Bloky SETPR, SETPI a SETPB slouží pro vzdálené nastavování parametrů ostatních bloků v modelu. Bloky mají identickou funkci, liší se pouze v typu parametru, který nastavují. Blok SETPR je pro reálné číslo, SETPI pro celé číslo a SETPB pro Booleovskou hodnotu.

Bloky mohou pracovat ve dvou režimech, které se přepínají parametrem SETF. Pro SETF = off je hodnota vzdáleného parametru sc nastavena na hodnotu vstupního parametru p (nebo ip, P) při startu a dále pak při každé změně vstupního parametru p (nebo ip, P). V případě SETF = on bloky pracují v režimu jednorázového zápisu vzdáleného parametru, který se zapíše při každé náběžné hraně (off→on) na vstupu SET. Po úspěšném zápisu je výstup y (nebo k, Y) nastaven na zapisovanou hodnotu a chybový výstup E = off. Při neúspěšném zápisu je E = on.

Jméno vzdáleného parametru určuje textový parametr sc (string connection), který se zadává ve tvaru <cesta_k_bloku:jmeno_parametru>. Rovněž je možné přistupovat k jednotlivým prvkům parametrů typu pole (např. parametr tout bloku ATMT). Toho se dosáhne pomocí hranatých závorek a čísla prvku, např. tedy .ATMT:touts[2], číslování je od 0, uvedený propojovací řetězec tedy odkazuje na třetí prvek pole.

Cesta k bloku, jehož parametr má být nastavován, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- Relativní – začíná v úrovni, do které je umístěn daný blok SETPR (nebo SETPI, SETPB). V tomto případě text začíná znakem '.'. Příklady hodnot relativních cest: ".GAIN:k", ".Motor1.Poloha:ycn".

- Absolutní – úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být nastavován parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok IOTASK), je v první úrovni hierarchie uveden znak '&' následovaný názvem ovladače. Příklady hodnot absolutních cest: "uloha1.vstupy.lin1:u2", "&EfaDrv.mereni.DER1:n".

Poznámka: Od verze řídicího systému REX 2.7 došlo ke změně práce s absolutními a relativními cestami. Ve starších verzích měla absolutní cesta prefix '´' a relativní cesta neměla prefix žádný. Ke změně bylo přistoupeno z důvodu sjednocení formátu cest s blo-

kem SGSLP. Z důvodu maximální možné kompatibility se staršími verzemi je znak ’´ na začátku řetězců ignorován, je však doporučeno cesty aktualizovat.

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v programu RexView.

Vstupy

p	Požadovaná hodnota parametru, vstup bloku SETPR	double
ip	Požadovaná hodnota parametru, vstup bloku SETPI	double
P	Požadovaná hodnota parametru, vstup bloku SETPB	double
SET	Vstup pro jednorázový zápis parametru	bool

Výstupy

y	Hodnota parametru, výstup bloku SETPR	double
k	Hodnota parametru, výstup bloku SETPI	long
Y	Hodnota parametru, výstup bloku SETPB	bool
E	Příznak chyby off ... bez chyby on .... nastala chyba	bool

Parametry

sc	Jméno vzdáleného parametru podle výše uvedených pravidel	string
SETF	Zapnutí manuálního zápisu vzdáleného parametru off ... režim průběžného nastavování parametru on .... režim jednorázového nastavení parametru po náběžné hraně na vstupu SET	bool

SETPS – * Blok pro vzdálené nastavování parametru typu string

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

sp	Požadovaná hodnota parametru	string
SET	Vstup pro jednorázový zápis parametru	bool

Parametry

sc	Jméno vzdáleného parametru	string
SETF	Nastavení parametru pouze na vyžádání	bool
nmax	Rezervovaná paměť pro řetězec	long

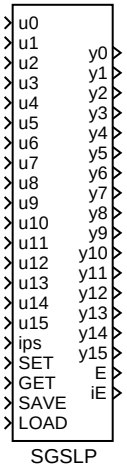
Výstupy

sy	Hodnota parametru	string
E	Příznak chyby	bool

SGSLP – Nastavování, čtení, ukládání a načítání parametrů

Symbol bloku

Licence: ADVANCED



Popis funkce

Blok `SGSLP` (z anglického *Set, Get, Save and Load Parameters*) je speciálním blokem pro správu připojených parametrů jiných bloků v konfiguraci řídicího systému REX. Blok pracuje i v systému Matlab-Simulink, jeho dosah je však omezen jen na bloky téhož souboru `.mdl`, v němž je vložen.

Blok může pracovat až se šestnácti sadami parametrů, které jsou číslovány od `0` do `15` a volí se vstupem `ips`, aktuální počet sad je určen parametrem `nps`. Je-li vstup `ips` nepřipojen, pracuje blok se sadou `ips` = 0. V každé sadě může být zkonfigurováno až 16 různých parametrů daných řetězcovými parametry `sc0` až `sc15`, takže jeden blok `SGSLP` může pracovat s maximálně 256 parametry v řídicím systému REX. Je-li řetězec `sc`*i* prázdný (nezadaný), není žádný parametr specifikován, jinak parametry `sc`*i* mohou používat dvě syntaxe:

1. `<blok>:<param>` – specifikují jeden blok se jménem `blok` s parametrem `param`. V tomto případě je použit tentýž blok a parametr pro všech `nps` sad parametrů.

2. `<blok>:<param><sep>...<blok>:<param>` – v tomto případě je pro každou sadu parametrů `ips` uvažován obecně různý parametr, první dvojice `<blok>:<param>` odpovídá `ips` = `0`. Oddělovačem `<sep>` může být buď čárka nebo středník. Specifikovaných dvojic `<blok>:<param>` by mělo být právě `nps`. V případě, že jich je méně, a má se provést některá z operací (viz níže) se sadou, pro niž specifikace bloku a parametru chybí, požadovaná operace se neprovede.

Přestože lze obecně pro každý z indexů i, $i = 0 \ldots 15$ volit různý způsob zadání sc$i$, doporučuje se pro celý blok volit buď syntaxi 1 nebo 2. První případ (několik hodnot pro stejný parametr) odpovídá např. výrobě nps druhů zboží, kde pro každé je nastavena jiná hodnota daného parametru. Druhý případ lze použít např. pro uložení co největšího počtu uživatelsky definovaných hodnot parametrů na disk (viz operaci SAVE níže), kde je vhodné blok SGSLP doplnit o logiku přepínání vstupu ips (např. pomocí bloku ATMT z knihovny LOGIC).

Pokud všechny bloky, jejichž parametry mají být nastavovány daným blokem SGSLP, leží v hierarchii bloků v nějakém subsystému nebo níže, lze výhodně použít řetězcový parametr broot, v němž se uvede jméno tohoto subsystému. Toto jméno se připojuje před každou specifikaci <blok> v parametrech sc$i$. V případě, že je broot = '.', je výsledek stejný, jako by parametr obsahoval cestu k subsystému, do nějž je daný blok SGSLP vložen (parametr se zadává bez uvozovek, ty jsou použity pouze v tomto textu pro zvýraznění jednotlivého znaku). Je-li hodnota parametru broot prázdná, musí každý výskyt <blok> v parametrech sc$i$ specifikovat úplnou cestu k bloku, v níž jsou jednotlivé hierarchické úrovně odděleny tečkami. Například tedy volba broot = . a sc0 = CNR:ycn zajišťuje propojení na blok CNR a jeho parametr ycn, který se nachází ve stejném subsystému jako blok SGSLP. Případně můžeme ponechat parametr broot prázdný a umístit znak '.' na začátek řetězce sc0. Bližší informace o cestách v systému REX jsou uvedeny u bloků GETPR a SETPR.

Blok SGSLP může při náběžné hraně (off→on) na některém ze stejnojmenných vstupů provádět následující operace:

SET – nastavit parametry dané množiny ips na hodnoty přivedené na vstupy ui. V případě, že je parametr úspěšně nastaven, je na stejnou hodnotu nastaven i výstup yi.

GET – získat parametry dané množiny ips. V případě, že je parametr úspěšně získán, je jeho hodnota nastavena na výstup yi.

SAVE – uložit parametry dané množiny ips do souboru (tzv. stavový soubor) na cílovém zařízení. Parametry a formát souboru jsou popsány níže.

LOAD – načíst parametry dané množiny ips ze souboru na cílovém zařízení. Kromě načtení parametrů při náběžné hraně vstupu LOAD se parametry sady ips0 načtou při inicializaci bloku v případě, že je hodnota parametru ips0 v rozsahu od 0 do nps − 1. Parametry a formát souboru jsou popsány níže.

Operace LOAD a SAVE pracují se souborem na cílovém zařízení, jehož jméno je uvedeno v parametru fname. Práce s parametrem fname se řídí následujícími pravidly:

- Pokud jméno souboru neobsahuje příponu, přidává se automaticky přípona .rxs (ReX Status file).
- Při ukládání bude vytvářen záložní soubor se stejným jménem, avšak s příponou modifovanou přidáním znaku '~' ihned za znak '.', např. pokud jméno souboru neobsahuje příponu, je přípona záložního souboru .~rxs.

- Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému REX na cílovém zařízení. Data se typicky ukládají na pevný disk nebo flash disk nebo jiné médium, které po vypnutí a opětovném zapnutí zachovává soubory.

Data jsou příkazem `SAVE` ukládána do textového souboru, ze kterého jsou příkazem `LOAD` načítána zpět do bloku `SGSLP`. Pro každý parametr `sc`i, $i = 0, \ldots, m$, kde $m < 16$ je maximální číslo, pro něž je parametr `sc`m neprázdný řetězec, obsahuje soubor dva řádky ve tvaru:

`"<blok>:<param>"`, ..., `"<blok>:<param>"`

`<hodnota>`, ..., `<hodnota>`

Jednotlivé položky `"<blok>:<param>"` jsou mezi sebou odděleny čárkami a jejich počet odpovídá parametru `nps`, obdobně to platí i o položkách `<hodnota>` obsahujících hodnotu parametru, jehož jméno je uvedeno ve stejné pozici v předchozím řádku. Poznamenejme, že pro `nps` $>$ `1` má první z těchto dvou řádků vždy právě uvedený tvar (dvojice `"<blok>:<param>"` se opakuje `nps`-krát) a to i v případě, že parametr `sc`i obsahuje jedinou dvojici `<blok>:<param>` (viz 1. syntaxe výše). Tato skutečnost umožňuje přecházet mezi oběma syntaxemi parametrů `sc`i, aniž by musel být soubor upravován.

Při ukládání malého počtu hodnot můžete rovněž využít blok `SILO`.

Vstupy

`u`i	i-tý analogový vstupní signál, $i = 0, \ldots, 15$	`double`
`ips`	Číslo sady parametrů (číslováno od 0)	`long`
`SET`	Přečtení vstupů `u`i a nastavení parametrů sady `ips` na jejich hodnoty	`bool`
`GET`	Přečtení parametrů sady `ips` a nastavení výstupů `y`i na jejich hodnoty	`bool`
`SAVE`	Uložení parametrů sady `ips` do souboru na disk cílového zařízení	`bool`
`LOAD`	Načtení parametrů sady `ips` ze souboru na disku cílového zařízení	`bool`

Výstupy

`y`i	i-tý analogový výstupní signál, $i = 0, \ldots, 15$	`double`
`E`	Příznak chyby `off` ... bez chyby `on` .... nastala chyba, viz výstup `iE`	`bool`

`iE`	Chybový nebo varovný výstup poslední operace `0` ..... nenastala žádná chyba ani varování `1` ..... fatální chyba volání systému Matlab (jen pro Simulink), blok dále není spouštěn `2` ..... chyba otvírání souboru pro čtení (příkaz `LOAD`) `3` ..... chyba otvírání souboru pro zápis (příkaz `SAVE`) `4` ..... nesprávný formát souboru `5` ..... dané číslo `ips` nebylo v souboru nalezeno `6` ..... jména parametru v souboru a v konfiguraci bloku si neodpovídají `7` ..... byl nalezen neočekávaný konec souboru `8` ..... chyba zápisu do souboru (plný disk?) `9` ..... chyba syntaxe parametru (chybí znak `':'`) `10` .... připojení parametru je tvořeno jen bílými znaky `11` .... nelze vytvořit záložní soubor `12` .... hodnotu parametru nelze získat operací `GET` (neexistující parametr?) `13` .... hodnotu parametru nelze nastavit operací `SET` (neexistující parametr?) `14` .... překročení času při získávání/nastavování parametru (timeout) `15` .... připojenou hodnotu (parametr) není dovoleno zapisovat `16` .... číslo sady `ips` je mimo přípustný rozsah	`long`

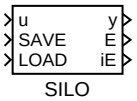
Parametry

`nps`	Počet sad parametrů ↓1 ↑16 ⊙1	`long`
`ips0`	Číslo sady parametrů, která se načte ze souboru při inicializaci bloku. Je-li `ips0` $<$ `0` nebo `ips0` $\geq$ `nps`, nečte se při inicializaci žádná sada ↓-1 ↑15	`long`
`iprec`	Počet platných číslic pro zápis hodnoty typu `double` do souboru ↓2 ↑15 ⊙12	`long`
`icolw`	Šířka sloupce v souboru. Je-li skutečná šířka menší, je doplněna zprava mezerami. Pokud je `icolw` $<$ `iprec`, nebudou žádné mezery přidávány. ↓0 ↑22	`long`
`fname`	Jméno souboru, do kterého se ukládají parametry příkazem `SAVE` a ze kterého se načítají příkazem `LOAD` ⊙`status`	`string`
`broot`	Cesta k subsystému, přidávaná na začátek specifikace bloků v parametrech `sc`i, viz popis v textu výše ⊙.	`string`
`sc`i	Řetězce specifikující připojení vstupů `u`i a výstupů `y`i, $i = 0, ..., 15$ k požadovaným parametrům, viz popis v textu výše	`string`

SILO – Uložení vstupního signálu, načtení výstupního signálu

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SILO je určen pro export nebo import jednoho signálu (hodnoty) do nebo ze souboru. Hodnota je uložena při náběžné hraně (off→on) na vstupu SAVE a po úspěšném uložení je nastavena také na výstup y. Načtení hodnoty probíhá při startu a při náběžné hraně (off→on) na vstupu LOAD. Při chybě diskové operace je na výstup y nastavena náhradní hodnota yerr.

Alternativně lze zapnout průběžné ukládání nebo čtení pomocí příslušného parametru (CSF, CLF). Diskové operace pak probíhají kontinuálně, ovšem pouze když je příslušný vstupní signál nastaven na on. Pozor však na to, že zápis/čtení pak probíhá při každém spuštění bloku, což může mít za následek nadměrné zatížení úložného zařízení, proto je potřeba použití tohoto režimu vždy důkladně zvážit.

Parametr fname určuje umístění souboru. Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému REX na cílovém zařízení.

Pro pokročilé a hromadné operace je určen blok SGSLP.

Vstupy

u	Vstupní signál	double
SAVE	Uložení vstupní hodnoty do souboru	bool
LOAD	Načtení hodnoty výstupu ze souboru	bool

Parametry

fname	Jméno souboru pro ukládání/načítání parametrů	string
CSF	Příznak pro průběžné ukládání	bool
CLF	Příznak pro průběžné načítání	bool
yerr	Náhradní hodnota pro případ chyby	double

Výstupy

y	Výstupní signál	double
E	Příznak chyby	bool
iE	Kód chyby operačního systému	long

Kapitola 13

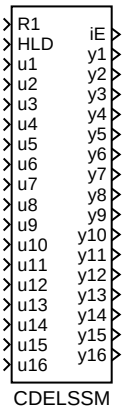
MODEL – Simulace dynamických systémů

Obsah

CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním

Symbol bloku

Licence: ADVANCED



Popis funkce

Funkční blok CDELSSM (Continuous State Space Model with time DELay) simuluje chování lineárního spojitého systému s dopravním zpožděním *del* ve stavové reprezentaci

$$\begin{aligned} \frac{dx(t)}{dt} &= A_c x(t) + B_c u(t - del),\ x(0) = x0 \\ y(t) &= C_c x(t) + D_c u(t), \end{aligned}$$

kde $x(t) \in \mathbb{R}^n$ je vektor stavu, $x0 \in \mathbb{R}^n$ je počáteční hodnota vektoru stavu, $u(t) \in \mathbb{R}^m$ je vektor vstupu, $y(t) \in \mathbb{R}^p$ je vektor výstupu. Matice $A_c \in \mathbb{R}^{n \times n}$ určuje dynamiku systému, matice $B_c \in \mathbb{R}^{n \times m}$ určuje působení vstupu na stav systému, matice $C_c \in \mathbb{R}^{p \times n}$ určuje působení stavu na výstup systému a matice $D_c \in \mathbb{R}^{p \times m}$ určuje přímé působení vstupu na výstup systému.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor $x0$ je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Simulovaný systém se nejprve převede do diskrétního (diskretizovaného) stavového modelu

$$\begin{aligned} x((k+1)T) &= A_d x(kT) + B_{d1} u((k-d)T) + B_{d2} u((k-d+1)T),\ x(0) = x0 \\ y(kT) &= C_c x(kT) + D_c u(kT), \end{aligned}$$

kde $k \in \{1, 2, \ldots\}$ je krok simulace, T je perioda spouštění bloku v [s] a d je zpoždění v krocích simulace tak, aby $(d-1)T < del \leq d.T$. Perioda T se v bloku nezadává, je určena automaticky jako perioda úlohy (TASK, QTASK nebo IOTASK), do níž je blok zařazen.

Pokud se vstup $u(t)$ mění jen v okamžicích vzorkování a mezi dvěma sousedními vzorkovacími okamžiky je konstantní (což se předpokládá), tj. $u(t) = u(kT)$ pro $t \in [kT, (k+1)T)$, pak matice A_d, B_{d1} a B_{d2} jsou určeny vztahy

$$\begin{aligned} A_d &= e^{A_c T} \\ B_{d1} &= e^{A_c(T-\Delta)} \int_0^{\Delta} e^{A_c \tau} B_c d\tau \\ B_{d2} &= \int_0^{T-\Delta} e^{A_c \tau} B_c d\tau, \end{aligned}$$

kde $\Delta = del - (d-1)T$.

Výpočet diskrétních matic A_d, B_{d1} a B_{d2} je založen na metodě popsané v [6], využívající Padéových aproximaci maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskrétního stavového modelu uvedeného výše.

Vstupy

`R1`	Resetovací signál, je-li `R1` = `on`, je stavový vektor `x` nastaven na počáteční hodnotu `x0`. Simulace se znovu spustí sestupnou hranou signálu `R1` (`on`→`off`).	`bool`
`HLD`	Zmrazení simulace po dobu, kdy je `HLD`=`on`.	`bool`
`u1..u16`	Vstupy simulovaného systému. Pro danou simulaci se používá prvních m vstupů, kde m je počet sloupců matice `Bc`.	`double`

Výstupy

`iE`	Kód chyby bloku `0` ..... vše v pořádku, blok simuluje správně `-213` .. nekompatibilita rozměrů matic stavového modelu `-510` .. úloha je špatně podmíněná (některá z pracovních matic je singulární nebo blízká singulární matici) `xxx` ... chybový kód `xxx` systému REX, více viz přílohu B	`error`
`y1..y16`	Výstupy simulovaného systému. Pro danou simulaci se používá prvních p výstupů, kde p je počet řádků matice `Cc`.	`double`

Parametry

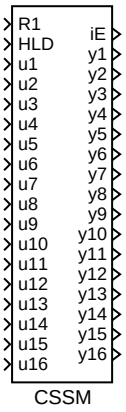
`UD`	Příznak použití matice `Dc`. Pokud je `UD`=`off`, matice `Dc` se při simulaci nepoužívá (chová se jako by byla nulová).		`bool`
`del`	Dopravní zpoždění modelu [s].	↓`0.0`	`double`

`is`	Stupeň Padéovy aproximace maticové exponenciály pro výpočet matic diskretizovaného systému. ↓`0` ↑`4` ⊙`2`	`long`
`eps`	Požadovaná přesnost Padéovy aproximace. ↓`0.0` ↑`1.0` ⊙`0.0`	`double`
`Ac`	Matice (typu [n,n]) dynamiky spojitého lineárního systému.	`double`
`Bc`	Vstupní matice (typu [n,m]) spojitého lineárního systému.	`double`
`Cc`	Výstupní matice (typu [p,n]) spojitého lineárního systému.	`double`
`Dc`	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr `UD`=`on`. Je-li `UD`=`off`, rozměry matice `Dc` se nekontrolují.	`double`
`x0`	Počáteční hodnota vektoru stavu (typu [n]) spojitého lineárního systému.	`double`

CSSM – Stavový model spojitého lineárního systému

Symbol bloku

Licence: ADVANCED



Popis funkce

Funkční blok CSSM (Continuous State Space Model) simuluje chování lineárního spojitého systému ve stavové reprezentaci

$$\begin{aligned} \frac{dx(t)}{dt} &= A_c x(t) + B_c u(t),\ x(0) = x0 \\ y(t) &= C_c x(t) + D_c u(t), \end{aligned}$$

kde $x(t) \in \mathbb{R}^n$ je vektor stavu, $x0 \in \mathbb{R}^n$ je počáteční hodnota vektoru stavu, $u(t) \in \mathbb{R}^m$ je vektor vstupu, $y(t) \in \mathbb{R}^p$ je vektor výstupu. Matice $A_c \in \mathbb{R}^{n \times n}$ určuje dynamiku systému, matice $B_c \in \mathbb{R}^{n \times m}$ určuje působení vstupu na stav systému, matice $C_c \in \mathbb{R}^{p \times n}$ určuje působení stavu na výstup systému a matice $D_c \in \mathbb{R}^{p \times m}$ určuje přímé působení vstupu na výstup systému.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor $x0$ je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Simulovaný systém se nejprve převede do diskrétního (diskretizovaného) stavového modelu

$$\begin{aligned} x((k+1)T) &= A_d x(kT) + B_d u(kT),\ x(0) = x0 \\ y(kT) &= C_c x(kT) + D_c u(kT), \end{aligned}$$

kde $k \in \{1, 2, \ldots\}$ je krok simulace, T je perioda spouštění bloku v [s]. Perioda T se v bloku nezadává, je určena automaticky jako perioda úlohy (TASK, QTASK nebo IOTASK), do níž je blok zařazen.

Pokud se vstup $u(t)$ mění jen v okamžicích vzorkování a mezi dvěma sousedními vzorkovacími okamžiky je konstantní (což se předpokládá), tj. $u(t) = u(kT)$ pro $t \in [kT, (k+1)T)$, pak matice A_d a B_d jsou určeny vztahy

$$\begin{aligned} A_d &= e^{A_c T} \\ B_d &= \int_0^T e^{A_c \tau} B_c d\tau \end{aligned}$$

Výpočet diskrétních matic A_d a B_d je založen na metodě popsané v [6], využívající Padéových aproximaci maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskrétního stavového modelu uvedeného výše.

Vstupy

R1	Resetovací signál, je-li R1 = on, je stavový vektor x nastaven na počáteční hodnotu x0. Simulace se znovu spustí sestupnou hranou signálu R1 (on→off).	bool
HLD	Zmrazení simulace po dobu, kdy je HLD=on.	bool
u1..u16	Vstupy simulovaného systému. Pro danou simulaci se používá prvních m vstupů, kde m je počet sloupců matice Bc.	double

Výstupy

iE	Kód chyby bloku 0 ..... vše v pořádku, blok simuluje správně -213 .. nekompatibilita rozměrů matic stavového modelu -510 .. úloha je špatně podmíněná (některá z pracovních matic je singulární nebo blízká singulární matici) xxx ... chybový kód xxx systému REX, více viz přílohu B	error
y1..y16	Výstupy simulovaného systému. Pro danou simulaci se používá prvních p výstupů, kde p je počet řádků matice Cc.	double

Parametry

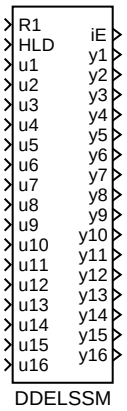
UD	Příznak použití matice Dc. Pokud je UD=off, matice Dc se při simulaci nepoužívá (chová se jako by byla nulová).	bool
is	Stupeň Padéovy aproximace maticové exponenciály pro výpočet matic diskretizovaného systému. ↓0 ↑4 ⊙2	long
eps	Požadovaná přesnost Padéovy aproximace. ↓0.0 ↑1.0 ⊙0.0	double
Ac	Matice (typu [n,n]) dynamiky spojitého lineárního systému.	double
Bc	Vstupní matice (typu [n,m]) spojitého lineárního systému.	double
Cc	Výstupní matice (typu [p,n]) spojitého lineárního systému.	double

`Dc`	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr `UD=on`. Je-li `UD=off`, rozměry matice `Dc` se nekontrolují.	`double`
`x0`	Počáteční hodnota vektoru stavu (typu [n]) spojitého lineárního systému.	`double`

DDELSSM – Stavový model diskrétního lineárního systému s dopravním zpožděním

Symbol bloku

Licence: ADVANCED



Popis funkce

Funkční blok DDELSSM (Discrete State Space Model with time DELay) simuluje chování lineárního diskrétního systému s dopravním zpožděním *del* ve stavové reprezentaci

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k-d),\ x(0) = x0 \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned}$$

kde k je krok simulace, $x(k) \in \mathbb{R}^n$ je vektor stavu, $x0 \in \mathbb{R}^n$ je počáteční hodnota vektoru stavu, $u(k) \in \mathbb{R}^m$ je vektor vstupu, $y(k) \in \mathbb{R}^p$ je vektor výstupu. Matice $A_d \in \mathbb{R}^{n \times n}$ určuje dynamiku systému, matice $B_d \in \mathbb{R}^{n \times m}$ určuje působení vstupu na stav systému, matice $C_d \in \mathbb{R}^{p \times n}$ určuje působení stavu na výstup systému a matice $D_d \in \mathbb{R}^{p \times m}$ určuje přímé působení vstupu na výstup systému. Počet kroků zpoždění d je největší celé číslo takové, že $d.T \leq del$, kde T je perioda spouštění bloku.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor $x0$ je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Při simulaci v reálném čase se v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskrétního stavového modelu uvedeného výše.

Vstupy

`R1`	Resetovací signál, je-li `R1` = `on`, je stavový vektor `x` nastaven na počáteční hodnotu `x0`. Simulace se znovu spustí sestupnou hranou signálu `R1` (`on`→`off`).	`bool`
`HLD`	Zmrazení simulace po dobu, kdy je `HLD`=`on`.	`bool`
`u1..u16`	Vstupy simulovaného systému. Pro danou simulaci se používá prvních m vstupů, kde m je počet sloupců matice `Bd`.	`double`

Výstupy

`iE`	Kód chyby bloku `0` ..... vše v pořádku, blok simuluje správně `-213` .. nekompatibilita rozměrů matic stavového modelu `xxx` ... chybový kód `xxx` systému REX, více viz přílohu B	`error`
`y1..y16`	Výstupy simulovaného systému. Pro danou simulaci se používá prvních p výstupů, kde p je počet řádků matice `Cd`.	`double`

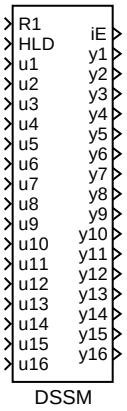
Parametry

`UD`	Příznak použití matice `Dd`. Pokud je `UD`=`off`, matice `Dd` se při simulaci nepoužívá (chová se jako by byla nulová).		`bool`
`del`	Dopravní zpoždění modelu [s].	↓`0.0`	`double`
`Ad`	Matice (typu [n,n]) dynamiky diskrétního lineárního systému.		`double`
`Bd`	Vstupní matice (typu [n,m]) diskrétního lineárního systému.		`double`
`Cd`	Výstupní matice (typu [p,n]) diskrétního lineárního systému.		`double`
`Dd`	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr `UD`=`on`. Je-li `UD`=`off`, rozměry matice `Dd` se nekontrolují.		`double`
`x0`	Počáteční hodnota vektoru stavu (typu [n]) diskrétního lineárního systému.		`double`

DSSM – Stavový model diskrétního lineárního systému

Symbol bloku

Licence: ADVANCED



Popis funkce

Funkční blok DSSM (Discrete State Space Model) simuluje chování lineárního diskrétního systému ve stavové reprezentaci

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k),\ x(0) = x0 \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned}$$

kde k je krok simulace, $x(k) \in \mathbb{R}^n$ je vektor stavu, $x0 \in \mathbb{R}^n$ je počáteční hodnota vektoru stavu, $u(k) \in \mathbb{R}^m$ je vektor vstupu, $y(k) \in \mathbb{R}^p$ je vektor výstupu. Matice $A_d \in \mathbb{R}^{n \times n}$ určuje dynamiku systému, matice $B_d \in \mathbb{R}^{n \times m}$ určuje působení vstupu na stav systému, matice $C_d \in \mathbb{R}^{p \times n}$ určuje působení stavu na výstup systému a matice $D_d \in \mathbb{R}^{p \times m}$ určuje přímé působení vstupu na výstup systému.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor $x0$ je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Při simulaci v reálném čase se v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskrétního stavového modelu uvedeného výše.

Vstupy

R1	Resetovací signál, je-li R1 = on, je stavový vektor x nastaven na počáteční hodnotu x0. Simulace se znovu spustí sestupnou hranou signálu R1 (on→off).	bool

`HLD`	Zmrazení simulace po dobu, kdy je `HLD=on`.	`bool`
`u1..u16`	Vstupy simulovaného systému. Pro danou simulaci se používá prvních m vstupů, kde m je počet sloupců matice `Bd`.	`double`

Výstupy

`iE`	Kód chyby bloku `0` ..... vše v pořádku, blok simuluje správně `-213` .. nekompatibilita rozměrů matic stavového modelu `xxx` ... chybový kód `xxx` systému REX, více viz přílohu B	`error`
`y1..y16`	Výstupy simulovaného systému. Pro danou simulaci se používá prvních p výstupů, kde p je počet řádků matice `Cd`.	`double`

Parametry

`UD`	Příznak použití matice `Dd`. Pokud je `UD=off`, matice `Dd` se při simulaci nepoužívá (chová se jako by byla nulová).	`bool`
`Ad`	Matice (typu [n,n]) dynamiky diskrétního lineárního systému.	`double`
`Bd`	Vstupní matice (typu [n,m]) diskrétního lineárního systému.	`double`
`Cd`	Výstupní matice (typu [p,n]) diskrétního lineárního systému.	`double`
`Dd`	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr `UD=on`. Je-li `UD=off`, rozměry matice `Dd` se nekontrolují.	`double`
`x0`	Počáteční hodnota vektoru stavu (typu [n]) diskrétního lineárního systému.	`double`

FOPDT – Model systému 1. řádu s dopravním zpožděním

Symbol bloku

Licence: STANDARD

Popis funkce

Blok FOPDT realizuje diskrétní simulátor lineárního systému prvního řádu s přídavným dopravním zpožděním, který je popsán následující přenosovou funkcí:

$$P(s) = \frac{\texttt{k0}}{(\texttt{tau} \cdot s + 1)} \cdot e^{-\texttt{del} \cdot s}$$

Diskrétní simulace používá přesnou diskretizaci přenosu $P(s)$ pro periodu T_S, s níž je blok FOPDT spouštěn.

Vstup

u	Analogový vstupní signál	double

Výstup

y	Analogový výstupní signál	double

Parametry

k0	Statické zesílení	⊙1.0	double
del	Dopravní zpoždění [s]		double
tau	Časová konstanta	⊙1.0	double
nmax	Délka vyrovnávací paměti pro dopravní zpoždění del (na tolik hodnot se alokuje paměť)	↓1 ↑10000000 ⊙10	long

MDL – Model procesu

Symbol bloku

Licence: STANDARD

Popis funkce

Blok MDL realizuje diskrétní simulátor spojitého systému s přenosem

$$F(s) = \frac{K_0 e^{-Ds}}{(\tau_1 s + 1)(\tau_2 s + 1)},$$

kde $K_0 > 0$ je statické zesílení k0, $D \geq 0$ je dopravní zpoždění del a $\tau_1, \tau_2 > 0$ jsou časové konstanty systému tau1 a tau2.

Vstup

u	Analogový vstupní signál	double

Výstup

y	Analogový výstupní signál	double

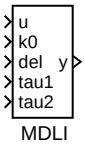
Parametry

k0	Statické zesílení	⊙1.0	double
del	Dopravní zpoždění [s]		double
tau1	První časová konstanta	⊙1.0	double
tau2	Druhá časová konstanta	⊙2.0	double
nmax	Délka vyrovnávací paměti pro dopravní zpoždění del (na tolik hodnot se alokuje paměť)	↓1 ↑10000000 ⊙10	long

MDLI – Model procesu s proměnnými parametry

Symbol bloku

Licence: STANDARD



Popis funkce

Blok MDLI realizuje diskrétní simulátor spojitého systému s přenosem

$$F(s) = \frac{K_0 e^{-Ds}}{(\tau_1 s + 1)(\tau_2 s + 1)},$$

kde $K_0 > 0$ je statické zesílení k0, $D \geq 0$ je dopravní zpoždění del a $\tau_1, \tau_2 > 0$ jsou časové konstanty systému tau1 a tau2. Na rozdíl od bloku MDL mohou být všechny parametry systému průběžně měněny ze vstupů bloku.

Vstupy

u	Analogový vstupní signál	double
k0	Statické zesílení	double
del	Dopravní zpoždění [s]	double
tau1	První časová konstanta	double
tau2	Druhá časová konstanta	double

Výstup

y	Analogový výstupní signál	double

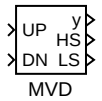
Parametry

nmax	Délka vyrovnávací paměti pro dopravní zpoždění del (na tolik hodnot se alokuje paměť) ↓1 ↑10000000 ⊙10	long

MVD – Motorizovaný pohon ventilu

Symbol bloku

Licence: STANDARD



Popis funkce

Blok MVD je určen pro simulaci servoventilu. Vstup UP (DN) představuje binární povel pro otevírání (zavírání) ventilu konstantní rychlostí 1/tv, kde tv je parametr bloku. Při UP = on (DN = on) otevírání probíhá až do úplného otevření y = hilim (úplného zavření y = hilim) ventilu. Krajní poloha otevření (zavření) je signalizována koncovým spínačem HS (LS). Počáteční poloha ventilu po spuštění je y = y0. Jestliže UP = DN = on nebo UP = DN = off, pak se poloha ventilu nemění (ani nezavírá ani neotvírá).

Vstupy

UP	Otevřít		bool
DN	Zavřít		bool

Výstupy

y	Poloha ventilu		double
HS	Horní koncový spínač		bool
LS	Dolní koncový spínač		bool

Parametry

y0	Počáteční poloha ventilu		double
tv	Čas přejezdu mezi polohami y = 0 a y = 1 [s]	⊙10.0	double
hilim	Horní mezní poloha (otevřeno)	⊙1.0	double
lolim	Dolní mezní poloha (zavřeno)		double

SOPDT – Model systému 2. řádu s dopravním zpožděním

Symbol bloku

Licence: STANDARD

Popis funkce

Blok SOPDT realizuje diskrétní simulátor lineárního systému druhého řádu s přídavným dopravním zpožděním, který je alternativně popsán, v závislosti na parametru itf, následujícími přenosovými funkcemi:

$$\begin{aligned}
\texttt{itf}=1: \quad P(s) &= \frac{\texttt{pb1}\cdot s+\texttt{pb0}}{s^2+\texttt{pa1}\cdot s+\texttt{pa0}}\cdot e^{-\texttt{del}\cdot s}\\
\texttt{itf}=2: \quad P(s) &= \frac{\texttt{k0}\,(\texttt{tau}\cdot s+1)}{(\texttt{tau1}\cdot s+1)\,(\texttt{tau2}\cdot s+1)}\cdot e^{-\texttt{del}\cdot s}\\
\texttt{itf}=3: \quad P(s) &= \frac{\texttt{k0}\cdot\texttt{om}^2\cdot(\texttt{tau}/\texttt{om}\cdot s+1)}{(s^2+2\cdot\texttt{xi}\cdot\texttt{om}\cdot s+\texttt{om}^2)}\cdot e^{-\texttt{del}\cdot s}\\
\texttt{itf}=4: \quad P(s) &= \frac{\texttt{k0}\,(\texttt{tau}\cdot s+1)}{(\texttt{tau1}\cdot s+1)\,s}\cdot e^{-\texttt{del}\cdot s}
\end{aligned}$$

Diskrétní simulace používá přesnou diskretizaci přenosu $P(s)$ pro periodu T_S, s níž je blok SOPDT spouštěn.

Vstup

u	Analogový vstupní signál		double

Výstup

y	Analogový výstupní signál		double

Parametry

itf	Tvar přenosové funkce 1 ..... obecný tvar 2 ..... reálné póly ve jmenovateli 3 ..... komplexní póly (kmitavý systém) 4 ..... systém s integrátorem	⊙1	long
k0	Statické zesílení	⊙1.0	double
tau	Časová konstanta v čitateli		double

`tau1`	První časová konstanta ve jmenovateli	⊙`1.0`	`double`
`tau2`	Druhá časová konstanta ve jmenovateli	⊙`1.0`	`double`
`om`	Vlastní frekvence	⊙`1.0`	`double`
`xi`	Relativní koeficient tlumení	⊙`1.0`	`double`
`pb0`	Koeficient čitatele: s^0	⊙`1.0`	`double`
`pb1`	Koeficient čitatele: s^1	⊙`1.0`	`double`
`pa0`	Koeficient jmenovatele: s^0	⊙`1.0`	`double`
`pa1`	Koeficient jmenovatele: s^1	⊙`1.0`	`double`
`del`	Dopravní zpoždění [s]		`double`
`nmax`	Délka vyrovnávací paměti pro dopravní zpoždění `del` (na tolik hodnot se alokuje paměť)	↓`1` ↑`10000000` ⊙`10`	`long`

Kapitola 14

MATRIX – Bloky pro maticové a vektorové operace

Obsah

CNA – * Konstantní pole (vektor/matice)

Symbol bloku

Licence: STANDARD

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Parametry

filename	Datový soubor s hodnotami oddělenými čárkou		string
TRN	Transponuj načtenou matici		bool
nmax	Rezervovaná paměť pro pole	↓2 ↑10000000 ⊙100	long
etype	Typ prvků	⊙8	long
	1 ..... Bool		
	2 ..... Byte		
	3 ..... Short		
	4 ..... Long		
	5 ..... Word		
	6 ..... DWord		
	7 ..... Float		
	8 ..... Double		
	--		
	10 .... Large		
acn	Počáteční hodnota pole	⊙[0 1 2 3]	double

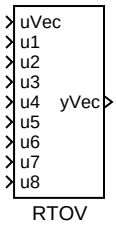
Výstup

vec	Odkaz na vektor/matici dat	reference

RTOV – Vektorový multiplexer

Symbol bloku

Licence: STANDARD



Popis funkce

Blok RTOV slouží k vytváření vektorových signálů v systému REX. Jeho vstupem jsou jednoduché analogové signály, které se sloučí do jednoho výstupního vektorového signálu.

Pokud je potřeba vytvořit signál s více než 8 položkami, je možné bloky RTOV řetězit a postupně vytvořit vektorový signál požadované dimenze.

Parametr nmax určuje maximální počet prvků ve vektoru, jinými slovy velikost místa v paměti alokovaného pro vektorový signál. Parametr offset určuje, na jakou pozici ve vektoru se má umístit hodnota prvního vstupního signálu u1 a parametr N říká, kolik vstupních signálů se má do výsledného vektoru yVec přenést.

Vstupy

uVec	Vektorový signál	reference
u1	Analogový vstupní signál	double
u2	Analogový vstupní signál	double
u3	Analogový vstupní signál	double
u4	Analogový vstupní signál	double
u5	Analogový vstupní signál	double
u6	Analogový vstupní signál	double
u7	Analogový vstupní signál	double
u8	Analogový vstupní signál	double

Výstup

yVec	Vektorový signál	reference

Parametry

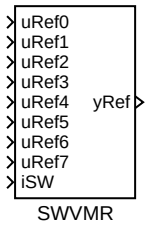
nmax	Alokovaná velikost vektoru	↓0 ⊙8	long
offset	Index prvního vstupu ve vektoru	↓0	long

`n`	Počet použitých vstupů	↓1 ↑8 ⊙8	`long`

SWVMR – Přepínač vektorového/maticového/odkazovacího signálu

Symbol bloku

Licence: STANDARD



Popis funkce

Blok SWVMR slouží k přepínání vektorových nebo maticových signálů. Rovněž umožňuje přepínat osy v algoritmech pro řízení pohybu (Motion Control, viz blok RM_Axis).

Pro přepínání jednoduchých signálů použijte blok SSW nebo jeho alternativy SWR či SELU.

Vstupy

uRef0	Vektorový signál	reference
uRef1	Vektorový signál	reference
uRef2	Vektorový signál	reference
uRef3	Vektorový signál	reference
uRef4	Vektorový signál	reference
uRef5	Vektorový signál	reference
uRef6	Vektorový signál	reference
uRef7	Vektorový signál	reference
iSW	Selektor aktivního signálu	long

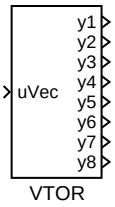
Výstup

yRef	Vektorový signál	reference

VTOR – Vektorový demultiplexer

Symbol bloku

Licence: STANDARD



Popis funkce

Blok `VTOR` rozděluje vstupní vektorový signál na jeho jednotlivé složky. Pomocí parametrů `offset` a `N` lze zvolit, od kolikátého prvku a kolik prvků se má přenést na výstupy bloku.

Vstup

`uVec`	Vektorový signál	`reference`

Výstupy

`y1`	Analogový výstupní signál	`double`
`y2`	Analogový výstupní signál	`double`
`y3`	Analogový výstupní signál	`double`
`y4`	Analogový výstupní signál	`double`
`y5`	Analogový výstupní signál	`double`
`y6`	Analogový výstupní signál	`double`
`y7`	Analogový výstupní signál	`double`
`y8`	Analogový výstupní signál	`double`

Parametry

`n`	Počet použitých výstupů	↓1 ↑8 ⊙8	`long`
`offset`	Index prvního výstupu	↓0	`long`

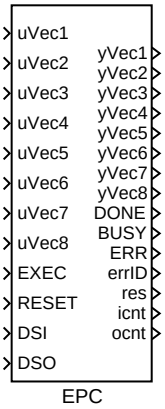
Kapitola 15

SPEC – Speciální bloky

Obsah

EPC – Blok pro spouštění externích programů

Symbol bloku Licence: ADVANCED



Popis funkce

Tento blok v okamžiku náběžné hrany (`off`→`on`) na vstupu `EXEC` spustí externí program, jehož název a parametry jsou uvedeny v parametru `cmd`. Zápis příkazu je naprosto shodný, jako by se psal na příkazovou řádku operačního systému.

Externímu programu lze předat hodnoty ze systému REX pomocí souborů. Formát těchto souborů určuje parametr `format`. V současnosti podporované formáty jsou všechny textové a velice jednoduché, takže je možné je snadno načíst do téměř libovolného programu. Například do MATLABu se soubor načte příkazem

```
hodnoty=load('-ASCII', 'epc_uVec1');
```

do SCILABu příkazem

```
hodnoty=read('/tmp/epc_uVec1',-1,32);
```

Název souboru, počet sloupců, jméno matice atd. je samozřejmě potřeba zvolit podle konkrétní aplikace. Hodnoty z externího programu zpět do systému REX se předávají analogickým způsobem (tj. opět pomocí souborů ve stejném formátu).

Blok rozlišuje dva režimy. V základním režimu je v okamžiku náběžné hrany na vstupu `EXEC` nejprve načtena aktuální hodnota na vstupech, uložena do souboru (vždy hodnoty z i-tého vstupního vektoru `uVec<i>` do i-tého souboru v parametru `ifns`). Ve vzorkovacím režimu jsou data ze vstupních vektorů ukládána do souborů v každé periodě algoritmu. V obou případech platí, že hodnoty vstupů z jednoho časového okamžiku jsou v jedné řádce souboru.

Analogicky jsou kopírována data z výstupních souborů na výstupy bloku (vždy jedna řádka z i-tého souboru v parametru `ofns` do i-tého výstupního vektoru `yVec<i>`).

Čísla vstupů, které pracují ve vzorkovacím režimu jsou uvedena v parametru `sl` (jednotlivá čísla se oddělují čárkou). Výstupy jsou vždy ve vzorkovacím režimu, přičemž

pokud v souboru nejsou další data (řádky), je ponechána předchozí hodnota. Kopírování vstupů do souboru je možné zablokovat (pozastavit) vstupem `DSI`; kopírování dat ze souborů na výstupy bloku je možné zablokovat (pozastavit) vstupem `DSO`.

Vektorové vstupy a výstupy bloku umožňují jednoduše uložit do jednoho souboru více hodnot (v každém kroku). Pro převod více jednoduchých signálů na vektor slouží blok `RTOV`. Tyto bloky lze řetězit, takže je možné vytvořit vektor téměř libovolné velikosti. Obdobně pro převod vektoru na jednoduché signály slouží blok `VTOR`, přičemž jeho vícenásobným použitím je možné získat hodnoty z libovolně velkého vektoru.

Vstupy

`uVec1..uVec8`	Vstupní vektorové signály	`reference`
`EXEC`	Náběžná hrana spouští externí program	`bool`
`RESET`	Reset bloku (smaže vstupní i výstupní soubory a zastaví externí program)	`bool`
`DSI`	Pozastavení vzorkování na vstupech	`bool`
`DSO`	Pozastavení vzorkování na výstupech	`bool`

Výstupy

`yVec1..yVec8`	Výstupní vektorové signály	`reference`
`DONE`	Příznak skončení externího programu	`bool`
`BUSY`	Příznak běhu externího programu	`bool`
`ERR`	Příznak chyby	`bool`
`errID`	Kód chyby `i` ..... obecná chyba systému REX	`error`
`res`	Návratový kód externího programu	`long`
`icnt`	Aktuální číslo vzorku na vstupech	`long`
`ocnt`	Aktuální číslo vzorku na výstupech	`long`

Parametry

`cmd`	Externí program		`string`
`ifns`	Vstupní soubory (oddělené středníkem)	⊙`epc_uVec1;epc_uVec2`	`string`
`ofns`	Výstupní soubory (oddělené středníkem)	⊙`epc_yVec1;epc_yVec2`	`string`
`sl`	Seznam čísel vzorkovacích vstupů. Zadává se ve tvaru např. `1,3..5,8`. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy `157`, binárně `10011101`.	↓0 ↑255 ⊙85	`long`
`ifm`	Maximální počet vzorků ve vstupním souboru	⊙10000	`long`
`format`	Formát vstupních a výstupních souborů `1` ..... textový (pouze hodnoty oddělené mezerou) `2` ..... CSV (desetinná tečka a čárky) `3` ..... CSV (desetinná čárka a středníky)	⊙1	`long`
`nmax`	Maximální délka výstupních vektorů	⊙10000	`long`

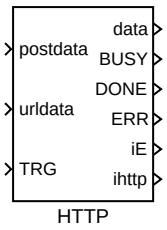
Poznámky

- Spuštěný skript má stejnou prioritu, jako task, který ji spustil. Ta je (implicitně) hodně velká (v některých případech dokonce vyšší, než tasky zpracovávající interupty v kernelu operačního systému). Pokud je toto nežádoucí (tj. zejména pokud externí skript trvá dlouho), je potřeba prioritu externího programu snížit. V linuxu se to provede tak, že příkaz napíšeme ve tvaru `chrt -o 0 extprg.sh`, kde `extprg.sh` je skript/program, který chceme spustit.
- Z implementačních důvodů je počet výstupních signálů omezen a je určen parametrem `nmax`. Parametr umožňuje zadat i hodně velká čísla, ale pro některé platformy nemusí být k dispozici dostatek paměti. Volte proto vždy co nejmenší číslo, které (s malou rezervou) dostačuje aplikaci.
- Jména souborů je potřeba psát tak, jak to vyžaduje použitý operační systém na cílové platformě. Nicméně pro vyhnutí se nečekaným potížím je doporučeno používat v názvu souboru jen písmena anglické abecedy, číslice a podtržítko. Také pozor na velikost písmen (Linux ji rozlišuje). Dále je potřeba zvážit, zda zadávat soubory s absolutní cestou nebo relativně k aktuálnímu adresáři. Zejména při vývoji aplikace se může aktuální adresář různě měnit a externí aplikace soubory nenajde.
- Z implementačních důvodů blok vytváří ještě kopie souborů uvedených v parametrech `ifns` a `ofns`. Tyto kopie mají v názvu navíc znak podtržítko.
- Parametry `ifns` a `ofns` určují umístění souborů. Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému REX na cílovém zařízení. Z důvodu výkonnosti je vhodné v tomto adresáři vytvořit symbolický link na souborový systém v RAM paměti. Na druhou stranu, pro dlouhé řady je výhodné mít soubor na disku, protože blok v případě výpadku řídicího systému po jeho opětovném spuštění naváže na předchozí data.
- Pro volání některých funkcí operačního systému lze použít i blok `OSCALL`.

HTTP – * Blok pro generování požadavků HTTP GET a POST

Symbol bloku

Licence: ADVANCED



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

`postdata`	Data vložená do HTTP požadavku	`string`
`urldata`	Data připojená k URL adrese	`string`
`TRG`	Spuštění zvolené akce	`bool`

Parametry

`url`	URL adresa pro odeslání HTTP požadavku		`string`
`method`	Typ HTTP požadavku `1` ..... GET `2` ..... POST	⊙`1`	`long`
`user`	Uživatelské jméno		`string`
`password`	Heslo		`string`
`certificate`	Certifikát pro autentifikaci		`string`
`VERIFY`	Povolení verifikace serveru (platnost certifikátu)		`bool`
`postmime`	Typ kódování pro požadavek POST	⊙`application/json`	`string`
`acceptmime`	Typ kódování pro požadavek GET	⊙`application/json`	`string`
`timeout`	Povolená doba pro dokončení operace	⊙`5.0`	`double`
`BLOCKING`	Čekání na dokončení operace		`bool`
`nmax`	Rezervovaná paměť pro řetězec	↓`0` ↑`65520`	`long`
`postmax`	postmax	↓`128` ↑`65520` ⊙`256`	`long`
`datamax`	Maximální velikost dat	↓`128` ↑`10000000` ⊙`1024`	`long`

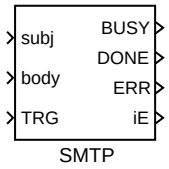
Výstupy

`data`	Data z odpovědi	`string`
`BUSY`	Odesílání HTTP požadavku	`bool`
`DONE`	HTTP požadavek byl zpracován	`bool`
`ERROR`	Příznak chyby	`bool`
`errId`	Kód chyby	`error`
`hterror`	HTTP odpověď	`long`

SMTP – * Blok pro odesílání e-mailových oznámení přes SMTP

Symbol bloku

Licence: ADVANCED



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Vstupy

`subj`	Předmět e-mailu		`string`
`body`	Tělo e-mailu		`string`
`TRG`	Spuštění zvolené akce		`bool`

Parametry

`server`	Adresa SMTP serveru		`string`
`to`	E-mail příjemce		`string`
`from`	E-mail odesílatele		`string`
`tls`	Metoda šifrování	⊙1	`long`
	1 ..... None		
	2 ..... StartTLS		
	3 ..... TLS		
`user`	Uživatelské jméno		`string`
`password`	Heslo		`string`
`domain`	domain		`string`
`auth`	Metoda autentifikace	⊙1	`long`
	1 ..... Login		
	2 ..... Plain		
`certificate`	Certifikát pro autentifikaci		`string`
`VERIFY`	Povolení verifikace serveru (platnost certifikátu)		`bool`
`timeout`	Povolená doba pro dokončení operace		`double`
`BLOCKING`	Čekání na dokončení operace		`bool`
`nmax`	Rezervovaná paměť pro řetězec	↓0 ↑65520 ⊙512	`long`
`datamax`	Maximální velikost dat	↓128 ↑65520	`long`

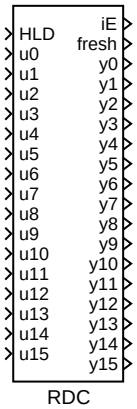
Výstupy

`BUSY`	Odesílání e-mailu	`bool`
`DONE`	E-mail byl odeslán	`bool`
`ERROR`	Příznak chyby	`bool`
`errId`	Kód chyby	`error`

RDC – Komunikační blok

Symbol bloku

Licence: ADVANCED



Popis funkce

Tento blok je speciální vstupně-výstupní blok. Hodnoty se předávají mezi dvěma bloky se stejným číslem, ale na různých počítačích (popřípadě na stejném počítači mezi dvěma Simulinky nebo Simulinkem a systémem REX). Hodnoty se předávají UDP/IP protokolem. Tento protokol je stejně rozšířený jako známější TCP/IP (tj. funguje na všech lokálních sítích LAN i na linkách sítě Internet). Algoritmus v každém kroku provádí následující operace:

- Otestuje vstup `HLD`. Pokud je `HLD` = `on`, činnost bloku končí.
- Má-li parametr `period` kladnou hodnotu, zjistí rozdíl mezi systémovým časem a časem posledního vyslání paketu. Pokud je tato doba menší než hodnota `period`, činnost bloku končí. (Pokud je hodnota parametru `period` menší nebo rovna nule, testování rozdílu času se neprovádí.)
- Vytvoří paket, který obsahuje číslo bloku, tzv. číslo `invoke` (pořadové číslo paketu), hodnoty `u0` až `u15`. Všechny hodnoty se do paketu ukládají ve standardně užívaném pořadí (tzv. network byte order), takže aplikace může běžet na libovolném počítači/procesoru.
- Odešle paket na zadanou IP adresu a port.
- Zvětší o 1 číslo `invoke`.
- Otestuje, jestli přišel nějaký paket.

- Pokud ano, otestuje, zda je paket v pořádku (souhlasí velikost, číslo bloku, číslo `invoke`).
- Pokud je paket v pořádku, nastaví výstupy `y0` až `y15` na hodnoty z přijatého paketu.
- Nastaví výstup `iE` (pokud došlo k nějaké chybě) a výstup `fresh`.

Z uvedeného popisu je zřejmé, že dvojice bloků (se stejným číslem, ale každý na jiném počítači) periodicky přenáší 16 hodnot v každém směru. Vždy se přenese `u(i)` z jednoho bloku na `y(i)` druhého bloku. Protože protokol UDP/IP (na rozdíl od TCP/IP) nemá mechanismus pro ošetření ztráty ani duplicity paketu, musí se to zajistit v algoritmu. K ošetření ztráty slouží mechanismus čísla `invoke`. To je stavová proměnná, která se zvětší o 1 při každém odeslaném paketu. Protože blok si pamatuje `invoke` číslo minulého přijatého paketu, pozná, k čemu došlo, a podle toho reaguje – pakety s číslem `invoke` menším než číslo `invoke` posledního přijatého paketu odmítá. Protože se však po ukončení a opětovném spuštění programu číslo invoke vynuluje, algoritmus přesto přijme paket s číslem menším než číslo posledního paketu, pokud je rozdíl velký (větší než 10). Z implementačních důvodů musí mít všechny bloky v jedné aplikaci stejný `local port` a v jedné aplikaci může být nejvýše 64 bloků `RDC`. Pokud by na jednom počítači běžely dva programy, které používají blok `RDC`, musí být parametr `local port` v každé aplikaci jiný.

Vstupy

`HLD`	Vstup pozastavující činnost bloku. Pokud je `HLD` = `on`, blok nevysílá ani nepřijímá žádné pakety.	`bool`
`u0..u15`	Hodnoty, které se předávají/zapisují na hodnoty `y0` až `y15` spolupracujícího bloku RDC	`double`

Výstupy

`iE`	Zobrazuje kód poslední chyby. Použitá čísla jsou v následující tabulce:	`long`

`0` ..... Bez chyby

Trvalé chyby, vznikají v inicializační části bloku, systém je nedokáže sám opravit (< 0)

- `-1` .... překročen maximální počet bloků (z interních důvodů je omezen počet bloků v jednom programu na 64)
- `-2` .... blok má jiný lokální port (z interních důvodů musí mít všechny bloky v jednom programu (jedné úloze) stejný parametr `lport`)
- `-3` .... nelze otevřít socket (protokol UDP/IP je nedostupný)
- `-4` .... nelze přiřadit lokální port (port je pravděpodobně obsazen jinou službou nebo programem)
- `-5` .... nelze nastavit tzv. neblokující mód socketu (blok `RDC` tento mód využívá a v případě chyby nemůže správně fungovat)
- `-10` ... chyba v inicializaci socketové knihovny
- `-11` ... chyba v inicializaci socketové knihovny
- `-12` ... chyba v inicializaci socketové knihovny

Přechodné chyby, mohou vzniknout ve kterémkoliv průchodu kódu, systém je dokáže sám opravit (> 0)

- `1` ..... proběhla inicializace bloku, ale ještě nebyl přijat žádný platný paket s hodnotami
- `2` ..... přijat chybný paket (chybná délka – buď došlo k chybě při přenosu a data jsou ztracena nebo může jít o konflikt s jinou službou/programem)
- `4` ..... chyba při příjmu paketu (chybu hlásí socketová knihovna)
- `8` ..... chyba při odeslání paketu (chybu hlásí socketová knihovna)

`fresh`	Udává počet sekund od přijetí posledního paketu. Má význam pro detekci chyby protilehlého bloku.	`double`
`y0..y15`	Signál přijatý ze vzdáleného bloku `RDC` – hodnoty naposledy přijatého paketu	`double`

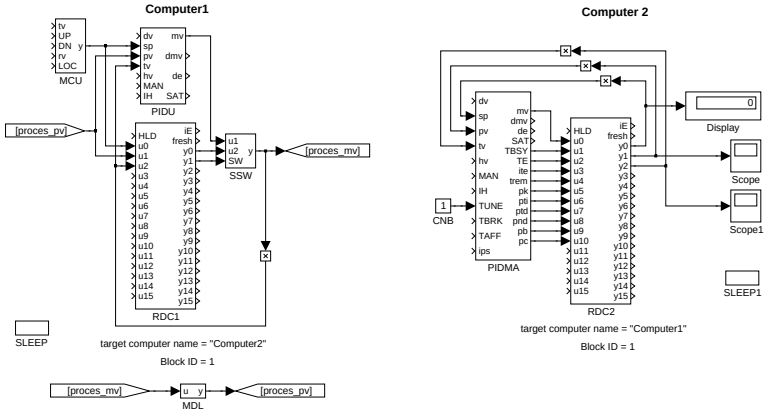
Parametry

`target`	Zde se napíše jméno nebo IP adresa počítače, kde běží spolupracující blok `RDC`. Může to být i broadcast adresa a pak spolupracující blok může být na libovolném počítači v síti, ale při malých periodách (orientačně kratších než 50ms) může docházet k zahlcení sítě. Pro typickou lokální síť (tj. IP adresa 192.168.1.x, maska 255.255.255.0) je broadcast adresa 192.168.1.255 .	`string`
`rport`	Vzdálený port (tzv. remote port). Je to vlastně upřesňující adresa nebo též adresa služby protokolu UDP/IP. Pokud nenastane kolize s jinými programy, které používají protokol UDP/IP, je vhodné tento parametr neměnit. ⊙1288	`word`
`lport`	Místní port (tzv. local port), význam podobný jako u parametru `rport`. Remote port platí pro počítač, kam je paket posílán, local port platí pro počítač, ze kterého je paket posílán. ⊙1288	`word`

`id`	Identifikátor bloku. Toto číslo se posílá v paketu a bloky ve druhém počítači podle něj poznají, pro který blok `RDC` jsou data určena. Principiálně jej přijmou všechny bloky, ale jen blok, který má stejné číslo `id`, jej akceptuje a nastaví výstupy na hodnoty z paketu. ↓1 ↑32767 ⊙1	`long`
`period`	Perioda v sekundách, určující nejkratší dobu, po které se vysílají a případně čtou došlé pakety. V případě hodnoty `period` ≤ 0, se vysílají (a případně i čtou) pakety při každém spuštění bloku. Nastavení kladné hodnoty je výhodné zejména ve spojitých modelech simulovaných systémem Simulink (při použití solveru typu `Variable step`).	`double`

Příklad

Následující obrázky představují možné použití bloku `RDC`. Příklad představuje „vzdálený autotuner“. Jeden počítač (označený `Computer1`) představuje standardní PID regulátor, který řídí technologický proces. Jeho signály `pv`, `sp`, `mv` jsou vedeny na vstupy bloku `RDC` a přenášeny na druhý počítač (označený `Computer2`). Na tomto počítači je autotuner (viz popis bloku PIDMA), který po náběžné hraně na vstupu `TUNE` provede identifikační experiment a vypočte parametry K, T_i, T_d vhodného regulátoru (výstup `pk`, `pti`, `ptd` bloku PIDMA). Aby toto mohl udělat, musí se přenášet hodnota `mv` autotuneru na akční veličinu technologického procesu. Proto je výstup `mv` (hodnota akční veličiny) a `TBSY` (slouží k přepínání mezi `mv` PID regulátoru a autotuneru). Všimněme si ještě, že hodnoty `pk`, `pti`, `ptd` jsou vyvedeny na vstupy bloku `RDC2`, takže se hodnoty přenesou na odpovídající výstupy bloku `RDC1`, kde by je bylo možné rovnou použít. Příklad je záměrně jednoduchý, aby byl dobře vidět princip bloku `RDC` a nikoliv složitost algoritmu, který lze v Simulinku vytvořit. Pro pochopení funkce si stačí uvědomit, že funkce uvedeného schématu je stejná, jako když bloky `RDC1` a `RDC2` vypustíme, zbytek obou výkresů sloučíme do jednoho a spojíme to, co původně vedlo na vstup `u0` bloku `RDC1`, s tím, co původně vedlo na výstup `y0` bloku `RDC2`, atd. pro `u1`, `y1`,....



OPC server pro blok `RDC`

Existuje OPC server, kterým se lze připojit k bloku RDC.

V popisu bloku RDC (viz výše) je uvedeno, že dva bloky RDC si vzájemně vyměňují hodnoty u a y. Jeden z této dvojice bloků může být emulován popisovaným OPC serverem. Jediný parametr, který se zadává je číslo portu. Je to lport bloku (resp. všech bloků), které OPC server emuluje. Hodnota se zadává jako parametr target name v textové podobě. Implicitní hodnota tohoto parametru je stejná, jako pro blok RDC (tj. 1288), takže ji obvykle není nutno měnit.

Pokud je přesto potřeba číslo portu změnit, tak hodnotu je možné zadat buď přímo systémovým programem Windows regedit (klíč je
SOFTWARE\REX Controls\REX_<version>\RdcOPCSvr\TargetName
- hodnotu je možné zadat do buď do sekce LocalMachine nebo CurrentUser, někdy je hodnota ve speciální podsekci VirtualStore a na 64-bitových počítačích je ještě podsekce Wow6432Node) nebo pomocí programu RexOPCcfg.exe (je součástí instalace systému REX, ale není na něj odkaz ve startmenu Windows - je potřeba v položce Key změnit text RexOPCsvr na RdcOPCsvr a požadované číslo portu zadat do pole Target name).

Server emuluje bloky všech identifikačních čísel na tomto portu. Protože takových bloků je velké množství, jsou při procházení platných signálů (tzv. browse) zobrazeny jen bloky, od kterých server dostal již nějaká data. OPC klient však může číst i zapisovat hodnoty i od ostatních bloků (čtené hodnoty jsou samozřejmě nesmyslné, ale operace čtení neselže, což je v některých případech důležité).

V adresním prostoru OPC serveru jsou položky ve tvaru

RDC<ID>.<pin>,

kde

<ID> je číslo remote/emulovaného bloku

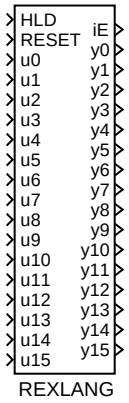
<pin> je název signálu nabývající jedné z hodnot:

- y0 až y15 – výstupy vzdáleného bloku (tj. vstupy u0 až u15 emulovaného bloku), které lze pouze zapisovat
- u0 až u15 – vstupy vzdáleného bloku (tj. výstupy y0 až y15 emulovaného bloku), které lze pouze číst
- fresh – doba uplynulá od přijetí posledního paketu v sekundách (tj. výstup fresh emulovaného bloku)

REXLANG – Volně programovatelný blok

Symbol bloku

Licence: REXLANG



Popis funkce

V některých případech se může stát, že je do řídícího algoritmu nutné implementovat funkci, kterou nelze efektivně vytvořit z dostupné množiny bloků. Pro takový účel byl vyvinut blok `REXLANG`, který implementuje algoritmus definovaný skriptovacím jazykem. Je použit skriptovací jazyk velice podobný jazyku C (nebo Java).

Skriptovací jazyk

Jak již bylo řečeno, skriptovací jazyk vychází z jazyka C a je mu velmi podobný, nicméně existují určité rozdíly a omezení:

- Jsou podpořeny jen typy `double` a `long` (lze použít i `int`, `short`, `bool`, které se interně zpracovávají jako `long`, a typ `float`, který se interně zpracovává jako `double`). Není implementován `typedef`.
- Nejsou implementovány pointery a struktury. Lze však definovat pole a používat indexy (operátor `[ ]`).
- Není zaveden operátor ',’.
- Z preprocesoru jsou podpořeny příkazy `#include`, `#define`, `#ifdef` .. [`#else` ..] `#endif`, `#ifndef` .. [`#else` ..] `#endif` (tzn. není podpořeno `#pragma` a zejména `#if` .. [`#else` ..] `#endif`).
- Nejsou implementovány standardní knihovny ANSI C, je však definována většina funkcí z `math.h` a potom některé další (viz dále).

- Jsou definována klíčová slova `input`, `output` a `parameter` pro napojení na vstupy, výstupy a parametry bloku. Dále jsou definovány systémové funkce pro řízení běhu a diagnostiku (viz dále).
- Kromě funkce `main()`, která se volá periodicky při běhu řídícího systému mohou být implementovány funkce `init()` (volá se při startu), `exit()` (volá se při ukončení řídícího algoritmu) a `parchange()` (volá se v systému REX při změně jakéhokoliv z parametrů, v systému Simulink v každém kroku).
- Ve funkcích a procedurách bez parametrů musí být v deklaraci explicitně uvedeno `void`.
- Nelze přetěžovat identifikátory, tj. nelze používat klíčová slova a názvy vestavěných funkcí jako identifikátor, nelze pojmenovat stejně globální a lokální proměnnou.
- Nelze inicializovat pole, ať už globální nebo lokální.

Syntaxe skriptovacího jazyka

Syntaxe skriptovacího jazyka vychází z jazyka C, přičemž nejsou podpořeny pointery a jiné typy než `long` a `double`. Navíc jsou definována klíčová slova `input`, `output` a `parameter`, která slouží pro odkazování na vstupy, výstupy a parametry bloku. Syntaxe je následující:

- `<typ> input(<číslo vstupu>) <jméno proměnné>;`
- `<typ> output(<číslo výstupu>) <jméno proměnné>;`
- `<typ> parameter(<číslo parametru>) <jméno proměnné>;`

Proměnné typu `input` a `parameter` lze pouze číst a do proměnných typu `output` lze pouze přiřazovat. Například:

```
double input(1) vstup; /* deklarace proměnné vstup typu double, která
                          představuje hodnotu vstupu bloku u1 */
long output(2) vystup; /* deklarace proměnné vystup typu long, která
                          představuje hodnotu výstupu bloku y2 */

vstup=3;               //nedovolený příkaz - do vstupu nelze přiřazovat
sum=vystup+1;          //nedovolený příkaz - z výstupu nelze číst hodnotu

if (vstup>1) vystup=3+vstup;  //správné použití
```

Dostupné funkce

Ve skriptovacím jazyce je možné používat následující funkce:

- **Matematické** (viz ANSI C, soubor `math.h`):
  `atan`, `sin`, `cos`, `exp`, `log`, `sqrt`, `tan`, `asin`, `acos`, `fabs`, `fmod`, `sinh`, `cosh`, `tanh`, `pow`, `atan2`, `ceil`, `floor` a `abs` Význam funkcí by měl být zřejmý, jen je potřeba zmínit, že funkce `abs` pracuje s celými čísly. Pro výpočet absolutní hodnoty desetinného čísla slouží funkce `fabs`.

- **Vektorové** (v ANSI C nejsou)

  `double max([n,]val1,...,valn)`
  : Vrací hodnotu největšího prvku. Funkce má nepovinný první parametr, který určuje počet prvků.

  `double max(n,vec)`
  : Vrací hodnotu největšího prvku z vektoru `vec`.

  `double min([n,]val1,...,valn)`
  : Vrací hodnotu nejmenšího prvku. Funkce má nepovinný první parametr, který určuje počet prvků.

  `double min(n,vec)`
  : Vrací hodnotu nejmenšího prvku z vektoru `vec`.

  `double poly([n,]x,an,...,a1,a0)`
  : Vypočte hodnotu polynomu `y` = `an` $*$ `x`$^{\texttt{n}}$ + ... + `a1` $*$ `x` + `a0`. Funkce má nepovinný první parametr, který určuje počet prvků.

  `double poly(n,x,vec)`
  : Vypočte hodnotu polynomu `y` = `vec[n]` $*$ `x`$^{\texttt{n}}$ + ... + `vec[1]` + `x` + `vec[0]`.

  `double scal(n,vec1,vec2)`
  : Vypočte skalární součin `y` = `vec1[0]` $*$ `vec2[0]` + ... + `vec1[n-1]` $*$ `vec2[n-1]`.

  `double scal(n,vec1,vec2,skip1,skip2)`
  : Vypočte skalární součin `y` = `vec1[0]`$*$`vec2[0]`+`vec1[skip1]`$*$`vec2[skip2]`+ ... + `vec1[(n-1)*skip1]` $*$ `vec2[(n-1)*skip2]`. Toto je výhodné, pokud vektory představují matice a potřebujeme vynásobit sloupce (resp. řádky, pokud je matice uložena po sloupcích).

  `double conv(n,vec1,vec2)`
  : Vypočte konvolutorní součin `y` = `vec1[0]`$*$`vec2[n-1]`+`vec1[1]`$*$`vec2[n-1]`+ ... + `vec1[n-1]` $*$ `vec2[0]`.

  `double sum(n,vec)`
  : Sečte prvky vektoru, tj. `y` = `vec[0]` + `vec[1]` + ... + `vec[n-1]`.

  `double sum([n,]val1,...,valn)`
  : Sečte prvky, tj. `y` = `val1` + `val2` + ... + `valn`. Funkce má nepovinný první parametr, který určuje počet prvků.

  `[]array([n,],an-1,...,a1,a0)`
  : Vrací pole/vektor, které obsahuje hodnoty parametrů. Funkce/operátor má nepovinný první parametr, který určuje počet prvků. Návratový typ se volí automaticky podle typu parametrů (musí být všechny stejného typu).

`[]subarray(idx,vec)`
: Vrací pole/vektor, které představuje pole `vec` od indexu `idx`. Návratový typ se volí automaticky podle typu vstupního pole.

`copyarray(count,vecSource,idxSource,vecTarget,idxTarget)`
: Kopíruje `count` hodnot z pole `vecSource` od indexu `idxSource` do pole `vecTarget` od indexu `idxTarget`. Obě pole musí být stejného typu.

`void fillarray(vector, value, count)`
: Kopíruje hodnotu `value` do `count` prvků pole `vector` (vždy od indexu0).

- **Funkce pro práci s textem** (v ANSI C jsou analogické funkce v souboru `string.h`)

`string strsub(str, index, len)`
: Vrací textový podřetězec.

`long strlen(str)`
: Vrací délku stringu (počet znaků).

`long strfind(str,substr) nebo long strfind(str,substr,offset)`
: Vrací polohu začátku (kolikátý znak počítáno od 0) substringu (parametr `substr`) ve stringu `str`. Prohledávání začíná od znaku s indexem `offset` (pokud není zadán, tak od začátku). Parametr `substr` může být i znak.

`long strrfind(str,substr)`
: Stejně jako předchozí funkce, ale prohledává od konce.

`string strupr(str)`
: Převede text na velká písmena.

`long str2long(str)`
: Převede text na celé číslo. Uvažuje jen tolik znaků od začátku, dokud text odpovídá číslu, zbylé znaky jsou ignorovány.

`double str2double(str)`
: Převede text na desetiné číslo. Uvažuje jen tolik znaků od začátku, dokud text odpovídá číslu, zbylé znaky jsou ignorovány.

`string long2str(num)`
: Převede celé číslo `num` na text.

`string double2str(num)`
: Převede desetinné číslo `num` na text.

`strcpy(dest,src)`
: Kopie řetězce. Funkce je zavedena z důvodu kompatibility s ANSI C. Lze použít konstrukci `dest=src` se stejným výsledkem.

`strcat(dest,src)`
: Spojení řetězců. Funkce je zavedena z důvodu komplatibility s ANSI C. Lze použít konstrukci `dest=dest+src` se stejným výsledkem.

`strcmp(str1,str2)`
: Porovnání stringů. Funkce je zavedena z důvodu komplatibility s ANSI C. Lze použít konstrukci `str1==str2` se stejným výsledkem.

`long RegExp(str,regexp,capture[])`

Porovnání stringu `str` s regulárním výrazem `regexp`. Pokud string vyhovuje, do pole `capture` se nastaví stringy odpovídající jednotlivým uzávorkovaným sekcím regulárního výrazu. `capture[0]` je vždy celý regulární výraz (první výskyt). Funkce vrací počet nastavených prvků v poli `capture` nebo záporné číslo v případě chyby. V regulárním výrazu jsou podporovány následující konstrukce:

`(?i)` ... Must be at the beginning of the regular expression. Makes the matching case-insensitive.

`^` ... Match beginning of a string

`$` ... Match end of a string

`()` ... Grouping and substring capturing

`\s` ... Match whitespace

`\S` ... Match non-whitespace

`\d` ... Match decimal digit

`\n` ... Match new line character

`\r` ... Match line feed character

`\f` ... Match vertical tab character

`\v` ... Match horizontal tab character

`\t` ... Match horizontal tab character

`\b` ... Match backspace character

`+` ... Match one or more times (greedy)

`+?` ... Match one or more times (non-greedy)

`*` ... Match zero or more times (greedy)

`*?` ... Match zero or more times (non-greedy)

`?` ... Match zero or once (non-greedy)

`x|y` ... Match x or y (alternation operator)

`\meta` ... Match one of the meta characters: `^$().[]*+?|\`

`\xHH` ... Match byte with hex value 0xHH, e.g. `\x4a`.

`[...]` ... Match any character from the set. Ranges like `[a-z]` are supported.

`[^...]` ... Match any character but the ones from the set.

`long ParseJson(json,cnt,names[],values[])`

Funkce předpokládá, že parametr `json` obsahuje text v JSON formátu. V poli `names` jsou názvy požadovaných objektů (k subpoložkám se přistupuje přes tečku, index pole se píše do `[]`), do pole `values` funkce nastaví hodnoty těchto objektů. Parametr `cnt` udává počet požadovaných objektů (délku pole `names` i `values`). Funkce vrací počet skutečně nastavených hodnot (záporná čísla znamenají chybu).

Poznámka: Textová proměnná se deklaruje stejně jako v ANSI C, tj. `char <název proměnné>[<maximální počet znaků>];`. Pro předání stringu do

funkce se používá konstrukce `char <název proměnné>[]` nebo `string <název proměnné>`.

- **Systémové** (v ANSI C nejsou)

`Trace(id,val)`
: Výpis čísla `id` a hodnoty `val`. Funkce je určena pro odladění bloku. Číslo `id` je uživatelem definovaná konstanta v rozsahu 0 až 9999 pro snadnou identifikaci výpisu. Hodnota `val` může být libovolného datového typu včetně textových řetězců (string). Zprávy se vypisují do systémového logu systému REX, při použití v Simulinku přímo do příkazového okna Matlabu.

  Pro zobrazení výpisů v programu RexView je potřeba v jeho menu `Target→Diagnostic messages` zaškrtnout položku `Information` v poli `Function block messages`. Teprve poté se zobrazí výpisy na kartě `System log`. Zároveň musí být povolen výpis zpráv z konkrétního bloku - zaškrtnutím checkboxu "Logging"v sekci "Runtime"v parametrech bloku. Ve výchozím stavu po vložení bloku z knihovny je toto povoleno.

`TraceError(id,val) TraceWarning(id,val) TraceVerbose(id,val)`
: Tyto funkce mají stejný význam jako `Trace`, avšak výpis je v jiné skupině systémového logu, kterou je potřeba nejdříve aktivovat obdobně jako u příkazu `Trace`. Výpisy úrovně "Error"se do logu zapisují vždy, nezávisle na zaškrtnutí checkboxu "Logging"na daném bloku.

`Suspend(sec)`
: Přeruší provádění kódu skriptu, pokud od jeho spuštění (v dané periodě) uplynulo více času (v sekundách), než je uvedeno. Při dalším spuštění bloku se pokračuje za tímto příkazem. Při `Suspend(0)` dojde k přerušení vždy.

`double GetPeriod()`
: Vrací vlastní periodu spouštění daného bloku ve vteřinách.

`double CurrentTime()`
: Vrací aktuální čas (v interním formátu). Používá se ve spojení s funkcí `ElapsedTime()`.

`double ElapsedTime(new_time, old_time)`
: Vrací uplynulý čas v sekundách (desetinné číslo), tj. rozdíl mezi časy určenými parametry `new_time` a `old_time`, získaným z předchozího volání funkce `CurrentTime()`.

`double Random()`
: Vrací pseudonáhodné číslo z intervalu $\langle 0, 1)$. Před voláním funkce `init()` se automaticky inicializuje generátor pseudonáhodného čísla, takže sekvence je vždy stejná.

`long QGet(var)`
: Vrací kvalitu proměnné `var` (tak jak s ní pracuje systém REX, viz bloky `QFC`, `QFD`, `VIN`, `VOUT`). Funkci je možno použít jen pro vstupy, parametry a výstupy - pro vnitřní proměnné vrací vždy 0.

`void QSet(var, value)`
: Nastaví kvalitu proměnné `var` (tak jak s ní pracuje systém REX) na hodnotu `val`. Funkci je možno použít jen pro výstupy - pro ostatní se nic nenastaví.

`long QPropag([n,]val1,...,valn)`
: Vrací kvalitu, která vznikne sloučením kvalit `val1,...,valn`. Základní pravidlo pro slučování je, že výsledná kvalita je nejhorší ze vstupních. Pokud nastavíme kvalitu výstupu bloku s použitím této funkce, tak že do parametrů dáme kvalitu všech vstupů bloku, které výstup ovlivňují, dostaneme stejné chování jako u ostatních bloků systému REX.

`double LoadValue(fileid, idx)`
: Přečte hodnotu ze souboru. Předpokládá binární soubor, kde jsou za sebou uloženy hodnoty typu `double` nebo textový soubor, kde na každé řádce je jedno číslo. Pořadí hodnoty v tomto souboru, kterou chceme přečíst udává parametr `idx` (počítá se od 0). Soubor je identifikován parametrem `fileid`. V současnosti jsou podporovány následující hodnoty:

  `0` ... soubor na disku s názvem v parametru `p0`

  `1` ... soubor na disku s názvem stejným jako název bloku rozšířený o příponu `.dat`.

  `2` ... soubor na disku s názvem v parametru `srcname`, ale s příponou .dat

  `3` ... soubor na disku s názvem `rexlang.dat` v aktuálním adresáři

  `4-7` ... stejné jako `0-3`, ale soubor je textový. Každá řádka obsahuje jedno číslo. Číslo řádku je parametr `idx` (počítáno od 0). Hodnota `idx=-1` znamená následující řádku (lze použít pro urychlení pro sekvenční čtení více hodnot).

`void SaveValue(fileid, idx, value)`
: Uloží hodnotu `value` do souboru. Význam ostatních parametrů je stejný jako u funkce `LoadValue`.

`void GetSystemTime(time)`
: Vrací hodnotu systémového času. Obvykle je to UTC, ale závisí na nastavení operačního systému. Parametr `time` musí být pole typu `long` o nejméně 8 prvcích. Funkce naplní toto pole hodnotami (po řadě) rok, měsíc, den (v měsíci), den v týdnu, hodina, minuta, sekunda, milisekunda. Na některých platformách milisekundy nejsou k dispozici (funkce vrací vždy 0ms) nebo mají jen omezenou přesnost.

`void Sleep(seconds)`
: Pozastaví vykonávání skriptu na uvedenou dobu (zadává se v parametru jako desetinné číslo v sekundách). Nejmenší čas na který lze skript pozastavit závisí na platformě, ale 0.01s a více funguje všude. Funkci je nutné používat jen ve výjimečných případech, protože se tím pozastaví vykonávání celého tasku/schématu.

`long GetExtInt(ItemID) long GetExtLong(ItemID)`
: Vrací hodnotu celočíselného vstupu/výstupu/parametru libovolného bloku v REXu určeného parametrem `ItemID`. Tento parametr je textový a má

stejný význam/strukuturu jako parametr `sc` bloku `GETPI`. Pokud hodnotu nelze získat (např. neexistující `ItemID` nebo není typu int) blok `REXLANG` skončí chybou.

`double GetExtReal(ItemID) double GetExtDouble(ItemID)`
Stejný význam jako předchozí pro desetinné číslo.

`double GetExtString(ItemID)`
Stejný význam jako předchozí pro text.

`void SetExt(ItemID, value)`
Nastaví hodnotu vstupu/výstupu/parametru libovolného bloku v REXu určeného parametrem `ItemID`, který má stejný význam jako v předchozí funkci. Nastavuje se hodnota parametru `value`, přičemž typ nastavené hodnoty (long/double/string) je určen typem parametru `value`.

`long memrd32(hMem,offset)`
Čtení fyzické paměti. Handle se získá pomocí funkce `OpenMemory`.

`long memwr32(hMem, offset, value)`
Zápis do fyzické paměti. Handle se získá funkcí `OpenMemory`.

- **Komunikační** (v ANSI C nejsou)

Tato sada funkcí slouží pro práci se sériovou linkou (RS-232 nebo RS-485), TCP/IP spojením, UDP/IP „spojením“. Zde je uveden jen stručný popis funkcí, které se pro komunikaci používají. Součástí instalace systému REX jsou příklady, které názorně ukazují způsob použití.

`long Open(long type, long lclIP, long lclPort, long rmtIP, long rmtPort)`
Otevře socket nebo COM port - podle parametru `type`. Pro TCP klient provádí rovnou connect. Vrací identifikační číslo (tzv. handle) socketu nebo portu. Pokud je záporné, otevření/spojení se nezdařilo.

`long Open(long type, string comname, long baudrate, long parity)`
Modifikace příkazu `Open()` pro otevření sériové linky.

`long Open(long type, string filename)`
Modifikace příkazu `Open()` pro otevření souboru.

`long Open(long type, string localname, long locPort, string remotename, long remPort)`
Modifikace příkazu `Open()` pro otevření TCP nebo UDP socketu.

`long OpenFile(string filename)`
Modifikace příkazu `Open()` pro otevření souboru.

`long OpenCom(string comname, long baudrate, long parity)`
Modifikace příkazu `Open()` pro otevření sériové linky.

`long OpenUDP(string localname, long lolPort, string remotename, long remPort)`
Modifikace příkazu `Open()` pro otevření UDP socketu.

`long OpenTCPsvr(string localname, long lolPort)`
Modifikace příkazu `Open()` pro otevření TCP socketu - server, naslouchání.

`long OpenTCPcli(string remotename, long remPort)`
Modifikace příkazu `Open()` pro otevření TCP socketu - klient.

`long OpenI2C(string devicename)`
: Modifikace příkazu `Open()` pro otevření I2C zařízení.

`long OpenMemory(string devicename, long baseaddr, long size)`
: Modifikace příkazu `Open()` pro mapování fyzické paměti.

`long OpenSPI(string devicename)`
: Modifikace příkazu `Open()` pro otevření SPI zařízení.

`long Close(long handle)`
: Zavře socket, sériovou linku, soubor nebo jiné zařízení otevřené pomocí funkce `Open` (nebo její varianty).

`void GetOptions(long handle, long params[])`
: Přečte parametry - nastaví aktuální hodnoty do pole `params`; pole musí být dostatečně dlouhé - aktuálně je 22 parametrů pro sériovou linku a 2 pro socket.

`void SetOptions(long handle, long params[])`
: Nastaví parametry sériové linky nebo socketu.

`long Accept(long hListen)`
: Přijme spojení navázané klientem na naslouchací socket `hListen`; vrací handle komunikačního socketu nebo chybu.

`long Read(long handle, long buffer[], long count)`
: Přijme data z linky nebo socketu nebo přečte ze souboru ; v parametru `count` je maximální počet byte, které se mají přečíst; funkce vrací počet skutečně přečtených byte nebo chybový kód; data jsou čtena tak, ze jeden byte z linky odpovídá jednomu prvku typu `long` v poli `buffer`. Ve starších verzích se funkce jmenovala `Recv`, což lze z důvodu zpětné kompatibility stále použít. Funkci lze použít také ve tvaru `long Read(long handle, string data[], long count)` (tj. místo pole na data se použije string; jeden byte ve vstupním souboru odpovídá jednomu znaku; binární soubory takto číst nelze) Chybové kódy jsou:
  - `-1` je třeba počkat na dokončení operace (funkce je totiž tzv. „neblokující“
  - `-309` čtení selhalo; chybový kód operačního systému se objevuje v logu (pokud je zapnuto logování u bloku)
  - `-307` soubor/socket není otevřen

`long Write(long handle, long buffer[], long count)`
: Odešle data na linku nebo socket; v parametru `count` je počet byte, které se mají poslat; funkce vrací počet skutečně vyslaných byte nebo chybový kód; data jsou zapisována tak, že jeden byte z linky se odpovídá jednomu prvku typu `long` v poli `buffer`. Ve starších verzích se funkce jmenovala `Send`, což lze z důvodu zpětné kompatibility stále použít. Funkci lze použít také ve tvaru `long Write(long handle, string data)` (tj. místo pole dat se použije string; jeden byte ve výstupním souboru odpovídá jednomu znaku; binární soubory takto zapisovat nelze) Chybové kody jsou:

`-1` je třeba počkat na dokončení operace (funkce je totiž tzv. „neblokující“
`-310` zápis selhal; chybový kód operačního systému se objevuje v logu (pokud je zapnuto logov
`-307` soubor/socket není otevřen

`long WriteRead(long handle, long addr, long bufW[], long cntW, long bufR[], long cntR)`
Příkaz pro komunikaci po sběrnici I2C nebo SPI. Funguje jen na zařízeních s operačním systémem Linux, která mají toto rozhraní (např. Raspberry Pi nebo ALIX). Provádí současně odeslání i příjem dat na/ze slave zařízení s adresou `addr`. Handle se získá funkcí `OpenI2C`, resp. `OpenSPI`, kde parametr funkce je jméno zařízení (dle operačního systému). Funkce vrací 0 nebo chybový kód.

`long Recv(long handle, long buffer[], long count)`
Pouze pro zajištění zpětné kompatibility. Funkce nahrazena funkcí ***`Read`****.*

`long Send(long handle, long buffer[], long count)`
Pouze pro zajištění zpětné kompatibility. Funkce nahrazena funkcí ***`Write`****.*

Ladění kódu, debugging

Pro ladění kódu je k dispozici příkaz `Trace`, viz výše. Dále lze použít výstupy bloku, které se nouužívají pro vlastní algoritmus a zapisovat do nich hodnoty různých mezivýpočtů. V závislosti na povaze algoritmu může být vhodné tyto ladící hodnoty připojit do trendu. Pokud je potřeba sledovat hodnot více, je možné do tasku přidat blok CNA (připojený na TRNDV nebo VTOR) a do hodnot v jeho poli nastavovat opět růyné mezivýsledky pomocí funkce `SetExt`.

Poznámky

- Typ vstupů `u0..u15`, výstupů `y0..y15` a parametrů `p0..p15` se určuje až při překladu zdrojového souboru bloku, podle specifikací `input`, `output` a `parameter`.
- Všechny chybové kódy `<0` nedokáže blok sám odstranit, odstraní se až novým spuštěním nebo nastavením vstupu `RESET`. Je samozřejmě nutné napřed odstranit příčinu, která chybu způsobila.
- POZOR!!! Ve funkci `init()` je sice možné číst vstupy, ale protože ostatní bloky obvykle nenastavují v init fázi výstupy, bude tam vždy 0. Nastavovat výstupy lze, ale obvykle se to nedělá.
- Parametr `srcname` je možné udávat s celou cestou. V opačném případě se soubor hledá na aktuálním adresáři a adresářích specifikovaných volbou -I v parametrech příkazové řádky programu `RexComp`.
- Všechny parametry vektorových funkcí jsou typu `double` (popřípadě vektor typu `double`) kromě parametru `n`, který je typu `long`. Také si všimněme, že funkce, které mají jen jeden vektorový parametr existují ve třech variantách:

`double funkce(val1,...,valn)`
: Vektor se zadává jako posloupnost parametrů typu `double`.

`double funkce(n,val1,...,valn)`
: Vektor se zadává jako v předchozím případě, ale navíc první parametr udává počet čísel – délku vektoru. Na rozdíl od předchozí varianty, lze v této variantě překládat zdrojový kód bez úprav překladačem jazyka C. Parametr `n` musí být přímo číslo (nikoliv tzv. `const` proměnná) a musí odpovídat počtu následujících parametrů tvořících vektor.

`double funkce(n,vec)`
: Parametr `n` je libovolný výraz typu `long` a udává počet prvků vektoru, se kterými funkce počítá.

- Nepovinný parametr `n` u vektorových funkcí se musí uvádět, pokud chceme stejný kód beze změn použít v překladači C/C++. Takové použití vyžaduje implementovat všechny nestandardní funkce, což není velký problém, ale funkce s variantním počtem parametrů musí nějak poznat jejich počet.

- Ve všech případech je třeba mít na paměti, že všechny vektory začínají prvkem s indexem 0 a dále, že program (stejně jako jazyk C) nekontroluje meze polí. Např. při definování `double vec[10], x;` (vektor s deseti prvky s indexy `0` až `9`) není zápis `x=vec[10];` ani syntaktická ani runtime chyba, ale hodnota je nedefinovaná. Dokonce lze i napsat `vec[11]=x;`, což je obzvláště nebezpečné, protože se tím přepíše nějaká jiná proměnná a program nefunguje správně, případně se může i „zaseknout“.

- Při překladu skriptu se často hlásí jen chyba `syntax error` a číslo řádky, kde nastává. Znamená to chybu v syntaxi. Pokud se zdá vše v pořádku, může to být tím, že použitý identifikátor je klíčové slovo jazyka nebo jméno vestavěné funkce.

- Všechny skoky se překládají relativně, tj. příslušný kus kódu je omezen na 32767 instrukcí (v přenositelném formátu na různé platformy).

- Do zásobníku se ukládají všechny aktuálně platné proměnné a mezivýsledky, tj.:

  – Globální proměnné a lokální `static` proměnné (trvale na začátku zásobníku)
  – Návratové adresy funkcí
  – Parametry předávané funkcím (včetně „vestavěných“)
  – Lokální proměnné funkcí
  – Návratová hodnota funkce
  – Mezivýsledky operací (například výraz `a=b+c;` se provádí tak, že se do zásobníku uloží hodnota `b`, pak (na další místo) hodnota `c` pak se provede součet, obě hodnoty se ze zásobníku zruší a vloží se tam hodnota součtu).

  Každá jednoduchá proměnná (tedy `long` i `double` se počítá za jednu položku v zásobníku. Pole se tedy počítá podle jeho délky opět bez ohledu na typ.

- Pole se do funkcí předávají odkazem. To znamená, že v zásobníku se počítá parametr jako jedna hodnota a hlavně se nepracuje s lokální kopií, ale vždy přímo s předaným polem.

- Pokud je zadána nedostatečná velikost zásobníku (méně než potřeba pro globální proměnné plus 10), volí se automaticky jako dvojnásobek potřeby pro globální proměnné (tj. předpoklad, že aktivních lokálních proměnných nebude více než globálních) plus 100 rezerva (na výpočty, parametry funkcí, lokální proměnné, pokud by globálních bylo málo).

- Při základním debug módu se kontroluje (za běhu skriptu), zda jsou všechny čtené hodnoty inicializované, zda index pole nepřesahuje deklarovanou velikost pole, přidává se několik neinicializovaných hodnot před a za každé deklarované pole (další ochrana proti nesprávnému indexu v poli), do kódu se přidávají instrukce `NOP` s argumentem číslo řádku ve zdrojovém souboru (usnadňuje dohledání v *.ill souboru). Pokud je zvolen úplný debug mód tak se navíc kontroluje, zda se program nepokouší přistupovat mimo platnou oblast dat (což je zatím nad `SP` ve stacku – neplatné hodnoty v zásobníku).

- Pod pojmem instrukce se u tohoto bloku nemyslí instrukce procesoru, ale instrukce mezikódu nezávislého na procesoru. Zdrojový kód přeložený do tohoto mezikódu je v souboru *.ill (mnemokódy pro jednotlivé instrukce, co řádka to jedna instrukce).

- Při použití sériové linky funkce `Open()` vždy nastaví binární neblokující režim bez timeoutů, 8 datových bitů, jeden stopbit, bez parity, 19200Bd. Bitovou rychlost a paritu lze nastavit přímo ve funkci `Open()` pomocí nepovinného druhého (bitrate) a třetího (parita) parametru.

- Při čtení a zápisu dat do textového souboru je potřeba počítat s tím, že se musí přečíst/zapsat celý při každém přístupu. Naproti tomu binární soubor má pevnou strukturu, takže přístup je rychlejší. Výhoda textových souborů spočívá v tom, že je lze zobrazovat i měnit bez speciálního programu.

- Na rozdíl od standardních bloků systému REX se automaticky nevolá funkce `parchange()` v inicializační fázi. Pokud je to potřeba, je nutné ji explicitně zavolat ve funkci `init()`.

- Protože operační systémy na bázi windows i linuxu přistupují k sériové lince stejně jako k souboru, je možné pomocí funkcí `Recv()` a `Send()` číst a zapisovat soubory sekvenčně po bajtech. V tomto případě se ve funkci `Open()` jako parametr `type` používají stejné hodnoty, jako ve funkci `LoadValue()` (parametr `fileid`).

- Ve funkci `WriteRead()` pro případ SPI je potřeba počítat s tím, že se data čtou i během zápisu. Pokud tedy potřebujeme zapsat do zařízení 2byte (např. číslo příkazu) a po jeho předání zařízení posílá 4byte dat, je potřeba číst 6byte, přičemž první a druhý byte (přijatý při zápisu čísla příkazu) neobsahuje platná data. Obecně

nelze ale byte přečtené při zápisu vypustit, protože u některých zařízení obsahují platná data.

- Funkce `OpenUDP()`, `OpenTCPsvr()`, `OpenTCPcli()`, podporují i IPv6 socket. Zda použít IPv4 nebo IPv6 se určí automaticky podle formátu adresy popřípadě podle toho co vrátí DNS.
- Funkce `OpenFile()` otevírá soubory v datovém adresáři systému REX(tj. v Linuxu implicitně v `\rex\data`, na Windows `C:\ProgramData\REX Controls\REX_<verze>\RexCore`). Jsou dovoleny podadresáře, ale není dovoleno `..\`. Linky se následují.

Vstupy

`HLD`	Pozastavení – kód bloku se nevykonává, je-li hodnota rovna `on`.	`bool`
`RESET`	Resetování chyby při náběžné hraně; blok se znovu inicializuje (vynulují se všechny globální proměnné a zavolá se funkce `Init()`)	`bool`
`u0..u15`	Vstupní signály, jejichž hodnoty jsou přístupné ve skriptu	`unknown`

Výstupy

`iE`	Kód chyby při běhu skriptu (kromě kódu 0 a -1 je provádění algoritmu bloku zastaveno do reinicializace vstupem `RESET` nebo novým spuštěním exekutivy) `0` ..... vše v pořádku, proběhla celá funkce `main()` popř. `init()` `-1` .... provádění programu skončilo příkazem `Suspend()`, tj. vypršel čas pro výpočet; výpočet bude při dalším spuštění bloku pokračovat tam, kde skončil `xxx` ... chybový kód `xxx` systému REX, více viz příloha B	`error`
`y0..y15`	Výstupní signály, jejichž hodnoty jsou definovány ve skriptu	`unknown`

Parametry

`srcname`	Jméno souboru se skriptem	⊙`srcfile.c`	`string`

`srctype`	Typ zdrojového souboru ⊙1 `1: C-like` Textový soubor s výše popisovanou syntaxí obdobnou jazyku C. `2: STL` Textový soubor se syntaxí dle IEC61131-3. Norma je implementována se stejnými omezeními jako C-like skript (tj. žádné struktury, z typů jen INT a REAL a STRING, vstupy bloku jsou globální VAR_INPUT, výstupy bloku jsou globální VAR_OUTPUT, parametry bloku jsou globální VAR_PARAMETER, standardní funkce dle specifikace, systémové a komunikační funkce jako v C-like) `3: RLB` Soubor v binárním formátu, který vzniká při překladu z formátu STL i C-like. Tento formát použijeme, pokud chceme předat fungující blok někomu jinému a nechceme mu dát zdrojové soubory. `4: ILL` Textový soubor, ale zapisují se mnemonické kódy instrukcí, do kterých je překládán formát STL. Dalo by se to přirovnat k assembleru. V současnosti není tento formát podporován.	`long`
`stack`	Velikost zásobníku pro všechny výpočty a proměnné. Zadává se jako počet proměnných, které se mají do zásobníku vejít. Výchozí hodnota 0 znamená, že velikost zásobníku se zvolí automaticky, což ve většině případů vyhovuje.	`long`
`debug`	Úroveň/množství ladicích kontrol a informací; větší číslo znamená více kontrol a tím pomalejší běh algoritmu; volbu `bez kontroly` se doporučuje nepoužívat (může vést až k pádu aplikace na cílové platformě při nesprávně napsaném kódu). ⊙3 `1` ..... bez kontroly `2` ..... základní kontrola `3` ..... úplná kontrola	`long`
`strs`	Velikost paměti (zásobníku) pro všechny texty. Zadává se jako počet byte/znaků, které se mají do zásobníku vejít. Výchozí hodnota 0 znamená, že velikost zásobníku se zvolí automaticky, což ve většině případů vyhovuje.	`long`
`p0..p15`	Parametry, jejichž hodnoty jsou přístupné ze skriptu	`unknown`

Příklad

Následující příklad implementuje lineární model procesu definovaný vahovou funkcí (filtr typu FIR) doplněný o saturaci na vstupu. Příklad je napsán tak, aby ukazoval různé konstrukce použitého skriptovacího jazyka. Algoritmus by bylo možné realizovat i jednodušším postupem.

```
double input(0) vstup;  //promenna 'vstup' predstavuje hodnotu u0
double output(0) vystup; //promenna 'vystup' predstavuje prirazeni do y0
double stav[20], param[20];
const long count=20;

long 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;
}

long 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;
}

long exit(void){return 0;}
```

a tentýž příklad v STL syntaxi:

```
VAR_INPUT
  vstup:REAL;  //promenna 'vstup' predstavuje hodnotu u0
END_VAR

VAR_OUTPUT
  vystup:REAL; //promenna 'vystup' predstavuje prirazeni do y0
END_VAR

VAR_PARAMETER
```

```
  tt:REAL; //promenna 'tt' predstavuje hodnotu parametru p0
END_VAR

VAR
  param, stav : ARRAY[0 .. 19] OF REAL;
END_VAR

VAR CONSTANT
  count:INT := 20;
END_VAR

FUNCTION init : INT;
VAR
  i:INT;
END_VAR

VAR CONSTANT
    a:REAL := 0.95;
END_VAR
    param[0]:=0.2; param[5]:=0.2; param[10]:=0.2; param[12]:=0.2; param[15]:=0.2;
    FOR i:=0 TO count-1 DO
        param[i] := param[i] + EXP(-i*a)/a;
        Trace(1,param[i]);
    END_FOR

    init := 0;
END_FUNCTION


FUNCTION main : INT;
VAR
  i:INT;
  soucet:REAL := 0.0;
END_VAR

    FOR i:=0 TO count-2 DO
        stav[i] := stav[i+1];
    END_FOR

    IF abs(vstup)>1 THEN
IF vstup>0.0 THEN
            stav[count-1] := 1;
ELSE
    stav[count-1] := -1;
```

```
END_IF
    ELSE
        stav[count-1] := vstup;
    END_IF

    FOR i:=0 TO count-1 DO
        soucet := soucet+stav[i]*param[count-1-i];
     IF tt>0.0 THEN
           Suspend(tt);
        ELSE
           Suspend(0.1);
END_IF
    END_FOR

    vystup := soucet;
    main := 0;
END_FUNCTION

FUNCTION exit : INT;
    exit := 0;
END_FUNCTION
```

Příloha A

Seznam funkčních bloků a jejich licencování

Aby bylo dosaženo maximální flexibility pro různé projekty, jsou funkční bloky systému REX licencovány po skupinách.

Funkční bloky ze skupiny `STANDARD` lze použít vždy, použití ostatních bloků je podmíněno aktivováním příslušné licence.

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ABS_	•	
ABSROT		ADVANCED
ACD	•	
ADD	•	
ADDOCT	•	
AFLUSH	•	
ALB	•	
ALBI	•	
ALN	•	
ALNI	•	
AND_	•	
ANDOCT	•	
ANLS	•	
ARC	•	
ARLY	•	
ASW		ADVANCED
ATMT	•	
AVG	•	
AVS		ADVANCED
BDHEXD	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
BDOCT	•	
BINS	•	
BIS	•	
BITOP	•	
BMHEXD	•	
BMOCT	•	
BPF	•	
CDELSSM		ADVANCED
CMP	•	
CNA	•	
CNB	•	
CNDR	•	
CNE	•	
CNI	•	
CNR	•	
CNS	•	
CONCAT	•	
COUNT	•	
CSSM		ADVANCED
DATE_	•	
DATETIME	•	
DDELSSM		ADVANCED
DEL	•	
DELM	•	
DER	•	
DIF_	•	
Display	•	
DIV	•	
DSSM		ADVANCED
EAS	•	
EATMT		ADVANCED
EDGE_	•	
EMD	•	
EPC		ADVANCED
EVAR	•	
EXEC	•	
FIND	•	
FLCU		ADVANCED
FNX	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
FNXY	•	
FOPDT	•	
FRID		ADVANCED
From	•	
GAIN	•	
GETPA	•	
GETPB	•	
GETPI	•	
GETPR	•	
GETPS	•	
Goto	•	
GotoTagVisibility	•	
GRADS		ADVANCED
HMI	•	
HTTP		ADVANCED
I3PM		ADVANCED
IADD	•	
IDIV	•	
IMOD	•	
IMUL	•	
INFO	•	
INHEXD	•	
INOCT	•	
Inport	•	
INQUAD	•	
INSTD	•	
INTE	•	
INTSM	•	
IODRV	•	
IOTASK	•	
ISSW	•	
ISUB	•	
ITOI	•	
KDER		ADVANCED
LC	•	
LEN	•	
LIN	•	
LLC	•	
LPBRK	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
LPF	•	
MC_AccelerationProfile		MOTION CONTROL
MC_AddAxisToGroup		COORDINATED MOTION CONTROL
MC_CamIn		MOTION CONTROL
MC_CamOut		MOTION CONTROL
MC_CombineAxes		MOTION CONTROL
MC_GearIn		MOTION CONTROL
MC_GearInPos		MOTION CONTROL
MC_GearOut		MOTION CONTROL
MC_GroupContinue		COORDINATED MOTION CONTROL
MC_GroupDisable		COORDINATED MOTION CONTROL
MC_GroupEnable		COORDINATED MOTION CONTROL
MC_GroupHalt		COORDINATED MOTION CONTROL
MC_GroupInterrupt		COORDINATED MOTION CONTROL
MC_GroupReadActualAcceleration		COORDINATED MOTION CONTROL
MC_GroupReadActualPosition		COORDINATED MOTION CONTROL
MC_GroupReadActualVelocity		COORDINATED MOTION CONTROL
MC_GroupReadError		COORDINATED MOTION CONTROL
MC_GroupReadStatus		COORDINATED MOTION CONTROL
MC_GroupReset		COORDINATED MOTION CONTROL
MC_GroupSetOverride		COORDINATED MOTION CONTROL
MC_GroupSetPosition		COORDINATED MOTION CONTROL
MC_GroupStop		COORDINATED MOTION CONTROL
MC_Halt		MOTION CONTROL
MC_HaltSuperimposed		MOTION CONTROL
MC_Home		MOTION CONTROL
MC_MoveAbsolute		MOTION CONTROL
MC_MoveAdditive		MOTION CONTROL
MC_MoveCircularAbsolute		COORDINATED MOTION CONTROL
MC_MoveCircularRelative		COORDINATED MOTION CONTROL
MC_MoveContinuousAbsolute		MOTION CONTROL
MC_MoveContinuousRelative		MOTION CONTROL
MC_MoveDirectAbsolute		COORDINATED MOTION CONTROL
MC_MoveDirectRelative		COORDINATED MOTION CONTROL
MC_MoveLinearAbsolute		COORDINATED MOTION CONTROL
MC_MoveLinearRelative		COORDINATED MOTION CONTROL
MC_MovePath		COORDINATED MOTION CONTROL
MC_MovePath_PH		COORDINATED MOTION CONTROL
MC_MoveRelative		MOTION CONTROL

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
MC_MoveSuperimposed		MOTION CONTROL
MC_MoveVelocity		MOTION CONTROL
MC_PhasingAbsolute		MOTION CONTROL
MC_PhasingRelative		MOTION CONTROL
MC_PositionProfile		MOTION CONTROL
MC_Power		MOTION CONTROL
MC_ReadActualPosition		MOTION CONTROL
MC_ReadAxisError		MOTION CONTROL
MC_ReadBoolParameter		MOTION CONTROL
MC_ReadCartesianTransform		COORDINATED MOTION CONTROL
MC_ReadParameter		MOTION CONTROL
MC_ReadStatus		MOTION CONTROL
MC_Reset		MOTION CONTROL
MC_SetCartesianTransform		COORDINATED MOTION CONTROL
MC_SetOverride		MOTION CONTROL
MC_Stop		MOTION CONTROL
MC_TorqueControl		MOTION CONTROL
MC_UngroupAllAxes		COORDINATED MOTION CONTROL
MC_VelocityProfile		MOTION CONTROL
MC_WriteBoolParameter		MOTION CONTROL
MC_WriteParameter		MOTION CONTROL
MCP_AccelerationProfile		MOTION CONTROL
MCP_CamIn		MOTION CONTROL
MCP_CamTableSelect		MOTION CONTROL
MCP_CombineAxes		MOTION CONTROL
MCP_GearIn		MOTION CONTROL
MCP_GearInPos		MOTION CONTROL
MCP_GroupHalt		COORDINATED MOTION CONTROL
MCP_GroupInterrupt		COORDINATED MOTION CONTROL
MCP_GroupSetOverride		COORDINATED MOTION CONTROL
MCP_GroupSetPosition		COORDINATED MOTION CONTROL
MCP_GroupStop		COORDINATED MOTION CONTROL
MCP_Halt		MOTION CONTROL
MCP_HaltSuperimposed		MOTION CONTROL
MCP_Home		MOTION CONTROL
MCP_MoveAbsolute		MOTION CONTROL
MCP_MoveAdditive		MOTION CONTROL
MCP_MoveCircularAbsolute		COORDINATED MOTION CONTROL
MCP_MoveCircularRelative		COORDINATED MOTION CONTROL

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
MCP_MoveContinuousAbsolute		MOTION CONTROL
MCP_MoveContinuousRelative		MOTION CONTROL
MCP_MoveDirectAbsolute		COORDINATED MOTION CONTROL
MCP_MoveDirectRelative		COORDINATED MOTION CONTROL
MCP_MoveLinearAbsolute		COORDINATED MOTION CONTROL
MCP_MoveLinearRelative		COORDINATED MOTION CONTROL
MCP_MovePath		COORDINATED MOTION CONTROL
MCP_MovePath_PH		COORDINATED MOTION CONTROL
MCP_MoveRelative		MOTION CONTROL
MCP_MoveSuperimposed		MOTION CONTROL
MCP_MoveVelocity		MOTION CONTROL
MCP_PhasingAbsolute		MOTION CONTROL
MCP_PhasingRelative		MOTION CONTROL
MCP_PositionProfile		MOTION CONTROL
MCP_SetCartesianTransform		COORDINATED MOTION CONTROL
MCP_SetKinTransform_Arm		COORDINATED MOTION CONTROL
MCP_SetOverride		MOTION CONTROL
MCP_Stop		MOTION CONTROL
MCP_TorqueControl		MOTION CONTROL
MCP_VelocityProfile		MOTION CONTROL
MCU	•	
MDL	•	
MDLI	•	
MID	•	
MINMAX	•	
MODULE	•	
MP	•	
MUL	•	
MVD	•	
NOT_	•	
NSCL	•	
OR_	•	
OROCT	•	
OSCALL	•	
OUTHEXD	•	
OUTOCT	•	
Outport	•	
OUTQUAD	•	
OUTRHEXD		ADVANCED

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
OUTROCT		ADVANCED
OUTRQUAD		ADVANCED
OUTRSTD		ADVANCED
OUTSTD	•	
PARA	•	
PARB	•	
PARI	•	
PARR	•	
PARS	•	
PIDAT		AUTOTUNING
PIDE		ADVANCED
PIDGS		ADVANCED
PIDMA		AUTOTUNING
PIDU	•	
PIDUI		ADVANCED
PJROCT	•	
PJSOCT	•	
POL	•	
POUT	•	
PRBS	•	
PRGM	•	
PROJECT	•	
PSMPC		ADVANCED
PWM	•	
QFC		ADVANCED
QFD		ADVANCED
QTASK	•	
RDC		ADVANCED
REC	•	
REGEXP		ADVANCED
REL	•	
REPLACE	•	
REXLANG		REXLANG
RLIM	•	
RLY	•	
RM_AxesGroup		COORDINATED MOTION CONTROL
RM_Axis		MOTION CONTROL
RM_AxisOut		MOTION CONTROL
RM_AxisSpline		MOTION CONTROL

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
RM_Feed		COORDINATED MOTION CONTROL
RM_Gcode		COORDINATED MOTION CONTROL
RM_GroupTrack		COORDINATED MOTION CONTROL
RM_Track		MOTION CONTROL
RS	•	
RTOI	•	
RTOS	•	
RTOV	•	
S1OF2		ADVANCED
SAI		ADVANCED
SAT	•	
SC2FA		AUTOTUNING
SCU	•	
SCUV	•	
SEL	•	
SELHEXD	•	
SELOCT	•	
SELQUAD	•	
SELSOCT	•	
SELU	•	
SETPA	•	
SETPB	•	
SETPI	•	
SETPR	•	
SETPS	•	
SG	•	
SGI	•	
SGSLP		ADVANCED
SHIFTOCT	•	
SHLD	•	
SILO	•	
SINT	•	
SLEEP	•	
SMHCC		ADVANCED
SMHCCA		AUTOTUNING
SMTP		ADVANCED
SOPDT	•	
SPIKE		ADVANCED
SQR	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
SQRT_	•	
SR	•	
SRTF		ADVANCED
SSW	•	
STOR	•	
SUB	•	
SubSystem	•	
SWR	•	
SWU	•	
SWVMR	•	
TASK	•	
TIME	•	
TIMER_	•	
TIODRV	•	
TRND	•	
TRNDLF		ADVANCED
TRNDV	•	
TRNDVLF		ADVANCED
TSE	•	
VDEL	•	
VIN		ADVANCED
VOUT		ADVANCED
VTOR	•	
WSCH	•	
WWW	•	
ZV4IS		ADVANCED

Příloha B

Chybové kódy systému REX

Kódy úspěšných operací

`0` ..... V pořádku
`-1` ..... Nepravda
`-2` ..... První hodnota je větší
`-3` ..... Druhá hodnota je větší
`-4` ..... Parametr byl změněn
`-5` ..... V pořádku, na serveru neprovedena žádná transakce
`-6` ..... Příliš velká hodnota
`-7` ..... Příliš malá hodnota
`-8` ..... Operace probíhá
`-9` ..... Upozornění ovladače systému REX
`-10` ..... V archivu nejsou další položky
`-11` ..... Položka je pole
`-12` ..... Ukončeno
`-13` ..... Konec souboru

Obecné chybové kódy

`-100` .... Nedostatek paměti
`-101` .... Předpoklad nesplněn (Assertion failure)
`-102` .... Překročení času (timeout)
`-103` .... Obecná chyba vstupní proměnné
`-104` .... Nesprávná verze konfigurace
`-105` .... Není implementováno
`-106` .... Nesprávný parametr
`-107` .... Chyba služeb COM/OLE
`-108` .... Chyba modulu systému REX - některý ovladač nebo blok není nainstalován nebo licencován
`-109` .... Chyba ovladače systému REX
`-110` .... Úlohu operačního systému se nepodařilo vytvořit

-111 .... Chyba volání funkce operačního systému
-112 .... Nesprávná verze operačního systému
-113 .... Přístup odmítnut operačním systémem
-114 .... Perioda bloku nebyla nastavena
-115 .... Selhala inicializace
-116 .... Probíhá výměna konfigurace systému REX
-117 .... Nesprávné cílové zařízení konfigurace
-118 .... Přístup odmítnut systémem REX
-119 .... Blok nebo jiný objekt není nainstalován nebo licencován
-120 .... Kontrolní součty se liší
-121 .... Objekt již existuje
-122 .... Objekt neexistuje
-123 .... Systémový uživatel nemá přiřazenou žádnou skupinu řídicího systému REX
-124 .... Špatné heslo
-125 .... Špatné uživatelské jméno nebo heslo
-126 .... Cílové zařízení není kompatibilní

Registrace tříd, chybové kódy symbolů a validačních procedur

-200 .... Neregistrovaná třída
-201 .... Třída už byla registrována
-202 .... Nedostatek místa v registru
-203 .... Index registru mimo rozsah
-204 .... Nesprávný kontext
-205 .... Nesprávný identifikátor
-206 .... Nesprávný příznak vstupu
-207 .... Nesprávná maska vstupu
-208 .... Nesprávný druh objektu
-209 .... Nesprávný typ proměnné
-210 .... Nesprávný pracovní prostor objektu
-211 .... Symbol nebyl nalezen
-212 .... Symbol je nejednoznačný
-213 .... Chyba kontroly rozsahu
-214 .... Nedostatek místa pro hledání
-215 .... Zápis do proměnné určené pouze pro čtení není dovolen
-216 .... Data nejsou připravena
-217 .... Hodnota mimo přípustný rozsah
-218 .... Chyba připojení vstupu
-219 .... Nalezena smyčka typů UNKNOWN
-220 .... Chyba při překladu jazyka REXLANG

Kódy pro streamy a souborový systém

-300 .... Přetečení streamu
-301 .... Podtečení streamu
-302 .... Vysílací chyba streamu

-303 .... Přijímací chyba streamu
-304 .... Chyba při posílání dat na cílové zařízení (download)
-305 .... Chyba při posílání dat z cílového zařízení (upload)
-306 .... Chyba vytvoření souboru
-307 .... Chyba otvírání souboru
-308 .... Chyba zavření souboru
-309 .... Chyba čtení souboru
-310 .... Chyba zápisu do souboru
-311 .... Nesprávný formát
-312 .... Chyba při komprimaci souborů
-313 .... Chyba během extrahování souborů

Chyby komunikace

-400 .... Chyba síťové komunikace
-401 .... Komunikace už byla inicializována
-402 .... Komunikace úspěšně ukončena
-403 .... Nečekané zavření komunikace
-404 .... Neznámý příkaz
-405 .... Neočekávaný příkaz
-406 .... Nečekané zavření komunikace, pravděpodobně 'příliš mnoho klientů'
-407 .... Překročení časového limitu pro komunikaci (timeout)
-408 .... Cílové zařízení nebylo nalezeno
-409 .... Spojení selhalo
-410 .... Konfigurace systému REX byla změněna
-411 .... Běh exekutivy systému REX se ukončuje
-412 .... Běh exekutivy systému REX byl ukončen
-413 .... Spojení odmítnuto
-414 .... Cílové zařízení není dostupné
-415 .... Cílové zařízení nebylo nalezeno v záznamu DNS
-416 .... Chyba při čtení ze soketu
-417 .... Chyba zápisu do soketu
-418 .... Chybná operace na soketu
-419 .... Rezervováno pro soket 1
-420 .... Rezervováno pro soket 2
-421 .... Rezervováno pro soket 3
-422 .... Rezervováno pro soket 4
-423 .... Rezervováno pro soket 5
-424 .... Nelze vytvořit kontext SSL
-425 .... Nelze načíst certifikát
-426 .... Chyba při vyjednávání spojení SSL
-427 .... Chyba verifikace certifikátu
-428 .... Rezervováno pro SSL 2
-429 .... Rezervováno pro SSL 3
-430 .... Rezervováno pro SSL 4

-431 .... Rezervováno pro SSL 5
-432 .... Relace odmítnuta
-433 .... STARTTLS odmítnuto
-434 .... Ověřovací metoda odmítnuta
-435 .... Ověření selhalo
-436 .... Chyba operace vysílání
-437 .... Chyba operace přijímání
-438 .... Komunikační příkaz selhal
-439 .... Vyrovnávací pamět pro příjem je příliš malá
-440 .... Vyrovnávací pamět pro vysílání je příliš malá
-441 .... Špatná hlavička
-442 .... Server HTTP vrátil chybu
-443 .... Server HTTP vrátil přesměrování
-444 .... Nepřípustná blokující operace
-445 .... Neplatná operace
-446 .... Komunikace ukončena
-447 .... Připojování přerušeno

Kódy numerických chyb

-500 .... Obecná numerická chyba
-501 .... Dělení nulou
-502 .... Přetečení numerického zásobníku
-503 .... Neplatná numerická instrukce
-504 .... Neplatná numerická adresa
-505 .... Nesprávný numerický typ
-506 .... Neinicializovaná numerická hodnota
-507 .... Přetečení/podtečení numerického argumentu
-508 .... Numerická chyba kontroly rozsahu
-509 .... Nesprávný rozsah indexů vektoru/matice
-510 .... Číselná hodnota příliš blízká nule

Kódy archivního systému

-600 .... Chyba prohledávání archivu
-601 .... Fatální chyba archivního semaforu
-602 .... Archiv byl smazán
-603 .... Archiv byl rekonstruován ze záložních proměnných
-604 .... Archiv byl rekonstruován z normálních proměnných
-605 .... Chyba kontrolního součtu archivu
-606 .... Chyba integrity archivu
-607 .... Byla změněna velikost archivu
-608 .... Byla překročena povolená velikost archivu

Kódy bloků pro řízení pohybu

-700 .... MC - Neplatný parametr
-701 .... MC - Mimo rozsah
-702 .... MC - Pozice není dosažitelná
-703 .... MC - Neplatný stav osy
-704 .... MC - Překročen limit momentu
-705 .... MC - Překročen časový limit
-706 .... MC - Překročena hraniční pozice
-707 .... MC - Skoková změna pozice nebo rychlosti
-708 .... MC - Base axis error or invalid state
-709 .... MC - Pohyb zastaven vstupem HALT
-710 .... MC - Pohyb zastaven polohou mimo rozsah osy
-711 .... MC - Pohyb zastaven z důvodu překročení maximální rychlosti osy
-712 .... MC - Pohyb zastaven z důvodu překročení maximálního zrzchlení osy
-713 .... MC - Pohyb zastaven koncovým spínačem
-714 .... MC - Pohyb zastaven z důvodu překročení maximální odchylky polohy (LAG)
-715 .... MC - Osa deaktivována během pohybu
-716 .... MC - Chyba generovaní přechodové křivky
-717 .... MC - nepoužito
-718 .... MC - nepoužito
-719 .... MC - nepoužito
-720 .... MC - Obecná chyba
-721 .... MC - Není implementováno
-722 .... MC - Příkaz ukončen
-723 .... MC - Rozdílná perioda osy a bloku
-724 .... MC - Blok čeká na převzetí osy

Kódy licencovacího systému

-800 .... Nepodařila se identifikace síťového rozhraní
-801 .... Nepodařila se identifikace CPU
-802 .... Nepodařila se identifikace HDD
-803 .... Neplatný kód zařízení
-804 .... Neplatný licenční klíč
-805 .... Licence nenalezena

Kódy spojené s webserverem

-900 .... Příliš rozsáhlý požadavek na webový server
-901 .... Příliš rozsáhlá odpověď webového serveru
-902 .... Neplatný formát
-903 .... Neplatný parametr

Literatura

[1] OPC Foundation. *Data Access Custom Interface Specification Version 3.00*. OPC Foundation, P.O. Box 140524, Austin, Texas, USA, 2003.

[2] REX Controls s.r.o.. *Stručný popis řídicího systému REX*, 2003.

[3] *Simulink reference, version 6*. The Mathworks, 3 Apple Hill Drive, Natick, MA, USA, 2006.

[4] Schlegel Miloš. Fuzzy regulátor: tutoriál.

[5] Miloš Schlegel, Pavel Balda, and Milan Štětina. Robustní PID autotuner: momentová metoda. *Automatizace*, 46(4):242–246, 2003.

[6] M. Schlegel and P. Balda. Diskretizace spojitého lineárního systému (in Czech). *Automatizace*, 11, 1987.

Rejstřík