

# Funkční bloky systému REX

## Referenční příručka

REX Controls s.r.o.

Verze 2.50.3  
28.2.2017  
Plzeň



# Obsah

<b>1</b>	<b>Úvod</b>	<b>11</b>
1.1	Jak číst tuto příručku . . . . .	11
1.2	Formát popisu funkčních bloků . . . . .	13
1.3	Konvence pojmenování proměnných, bloků a subsystémů . . . . .	14
1.4	Kvalita signálu používaná v OPC . . . . .	15
<b>2</b>	<b>EXEC – Konfigurace exekutivy reálného času</b>	<b>17</b>
	ARC – Archiv systému REX . . . . .	18
	EXEC – Exekutiva reálného času . . . . .	20
	HMI – * Konfigurace vizualizace . . . . .	22
	INFO – * Dodatečné informace o projektu . . . . .	23
	IODRV – Vstupně-výstupní ovladač systému REX . . . . .	24
	IOTASK – Úloha řídicího systému REX spouštěná ovladačem . . . . .	26
	LPBRK – Rozpojení zpětné vazby . . . . .	27
	MODULE – Rozšiřující modul systému REX . . . . .	28
	PROJECT – * Další nastavení projektu . . . . .	29
	QTASK – Rychlá úloha řídicího systému REX . . . . .	30
	SLEEP – Časovací blok pro Simulink . . . . .	31
	SRTF – Blok pro nastavování příznaků běhu . . . . .	32
	OSCALL – Volání funkcí operačního systému . . . . .	34
	TASK – Standardní úloha řídicího systému REX . . . . .	35
	TIODRV – Vstupně-výstupní ovladač systému REX s úlohami . . . . .	37
	WWW – * Obsah pro interní webserver . . . . .	39
<b>3</b>	<b>INOUT – Bloky vstupů a výstupů systému REX</b>	<b>41</b>
	Display – * Zobrazení vstupní hodnoty . . . . .	42
	From, INSTD – Připojení signálu nebo vstupní signál . . . . .	43
	Goto, OUTSTD – Zdroj signálu nebo výstupní signál . . . . .	45
	GotoTagVisibility – Viditelnost zdroje signálu . . . . .	47
	Inport, Outport – Vstupní a výstupní port . . . . .	48
	SubSystem – Subsystém . . . . .	50
	INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů . . . . .	51
	OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů . . . . .	53

OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací . . . . .	55
OUTRSTD – Výstupní signál s verifikací hodnoty . . . . .	57
QFC – Kódování příznaků kvality signálu . . . . .	58
QFD – Dekódování příznaků kvality signálu . . . . .	59
VIN – Ověření kvality vstupního signálu . . . . .	60
VOUT – Nastavení kvality výstupního signálu . . . . .	62
<b>4 MATH – Matematické bloky</b>	<b>63</b>
ABS_ – Absolutní hodnota . . . . .	65
ADD – Součet dvou signálů . . . . .	66
ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů . . . . .	67
CNB – Booleovská (logická) konstanta . . . . .	68
CNE – Předdefinovaná konstanta . . . . .	69
CNI – Celočíselná konstanta . . . . .	70
CNR – Reálná konstanta . . . . .	71
DIF_ – Blok difference . . . . .	72
DIV – Dělení dvou signálů . . . . .	73
EAS – Rozšířené sčítání a odečítání . . . . .	74
EMD – Rozšířené násobení a dělení . . . . .	75
FNX – Výpočet hodnoty funkce jedné proměnné . . . . .	76
FNXY – Výpočet hodnoty funkce dvou proměnných . . . . .	78
GAIN – Násobení konstantou . . . . .	80
GRADS – Gradientní optimalizace . . . . .	81
IADD – Celočíselné sčítání . . . . .	83
ISUB – Celočíselné odčítání . . . . .	85
IMUL – Celočíselné násobení . . . . .	87
IDIV – Celočíselné dělení . . . . .	89
IMOD – Zbytek po celočíselném dělení . . . . .	90
LIN – Lineární interpolace . . . . .	91
MUL – Násobení dvou signálů . . . . .	92
POL – Vyhodnocení polynomu . . . . .	93
REC – Převrácená hodnota . . . . .	94
REL – Relační operace dvou signálů . . . . .	95
RTOI – Konverze reálného čísla na celé číslo . . . . .	96
SQR – Druhá mocnina . . . . .	97
SQRT_ – Druhá odmocnina . . . . .	98
SUB – Odčítání dvou signálů . . . . .	99
<b>5 ANALOG – Zpracování analogových signálů</b>	<b>101</b>
ABSR0T – Zpracování dat z absolutního snímače polohy . . . . .	103
ASW – Přepínač s automatickou volbou vstupu . . . . .	105
AVG – Filtr: vlečný průměr . . . . .	107
AVS – Rozběhová jednotka . . . . .	108
BPF – Filtr: pásmová propust' . . . . .	109

CMP – Komparátor s hysterezí . . . . .	110
CNDR – Kompenzátor složité nelinearity . . . . .	111
DEL – Dopravní zpoždění s inicializací . . . . .	113
DELM – Dopravní zpoždění . . . . .	114
DER – Derivace, filtrace a predikce z posledních $n+1$ vzorků . . . . .	115
EVAR – Vlečná střední hodnota a směrodatná odchylka . . . . .	116
INTE – Řízený integrátor . . . . .	117
KDER – Derivace a filtrace vstupního signálu . . . . .	119
LPF – Filtr: dolní propust . . . . .	121
MINMAX – Vlečné minimum a maximum . . . . .	122
NSCL – Kompenzátor jednoduché nelinearity . . . . .	123
RDFT – Vlečná diskretní Fourierova transformace . . . . .	124
RLIM – Omezovač strmosti . . . . .	126
S1OF2 – Výběr jednoho ze dvou analogových vstupů . . . . .	127
SAI – Zabezpečený analogový vstup . . . . .	130
SEL – Selektor analogového signálu . . . . .	133
SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu . . . . .	135
SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot . . . . .	137
SHLD – Vzorkovač (sample and hold) . . . . .	139
SINT – Jednoduchý integrátor . . . . .	140
SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů . . . . .	141
SSW – Jednoduchý přepínač . . . . .	143
SWR – Přepínač s rampovou funkcí . . . . .	144
VDEL – Dopravní zpoždění s proměnnou délkou . . . . .	145
ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací . . . . .	146
<b>6 GEN – Generátory signálů</b>	<b>151</b>
ANLS – Řízený generátor po částech lineární funkce . . . . .	152
BINS – Řízený generátor binární posloupnosti . . . . .	154
BIS – Generátor binární posloupnosti . . . . .	156
MP – Ručně generovaný pulz . . . . .	157
PRBS – Pseudonáhodná binární posloupnost . . . . .	158
SG, SGI – Řízený generátor signálu . . . . .	160
<b>7 REG – Bloky pro regulaci</b>	<b>163</b>
ARLY – Relé s předstihem . . . . .	165
FLCU – Fuzzy regulátor . . . . .	166
FRID – * Identifikace frekvenční charakteristiky . . . . .	168
I3PM – Identifikace modelu se třemi parametry . . . . .	170
LC – Derivační kompenzátor . . . . .	172
LLC – Integračně-derivační kompenzátor . . . . .	173
MCU – Jednotka pro ruční zadávání . . . . .	174
PIDAT – PID regulátor s reléovým autotunerem . . . . .	176
PIDE – PID regulátor se statikou . . . . .	179

PIDGS – PID regulátor s přepínáním sad parametrů . . . . .	181
PIDMA – PID regulátor s momentovým autotunerem . . . . .	183
PIDU – PID regulátor . . . . .	189
PIDUI – PID regulátor s parametry na vstupech . . . . .	192
POUT – Pulzní výstup . . . . .	194
PRGM – Programátor . . . . .	195
PSMPC – Prediktivní „pulse-step“ regulátor . . . . .	197
PWM – Blok šířkové modulace . . . . .	201
RLY – Relé s hysterezí . . . . .	203
SAT – Saturace výstupu s proměnnými mezemi . . . . .	204
SC2FA – Stavový regulátor systému 2. řádu s autotunerem . . . . .	206
SCU – Krokový regulátor s polohovou zpětnou vazbou . . . . .	212
SCUV – Krokový regulátor s rychlostním výstupem . . . . .	215
SELU – Selektor aktivního regulátoru . . . . .	219
SMHCC – Regulátor pro procesy s topením a chlazením . . . . .	221
SMHCCA – * Regulátor pro procesy s topením a chlazením s autotunerem . . . . .	225
SWU – Přepínač vstupu pro vysledování . . . . .	228
TSE – Třístavový prvek . . . . .	229
<b>8 LOGIC – Logické řízení</b>	<b>231</b>
AND_ – Logický součin dvou signálů . . . . .	232
ANDQUAD, ANDOCT, ANDHEXD – Logický součin osmi signálů . . . . .	233
ATMT – Automat pro sekvenční řízení . . . . .	234
BDOCT, BDHEXD – Bitové demultiplexery . . . . .	237
BITOP – Bitová operace dvou celočíselných signálů . . . . .	238
BMOCT, BMHEXD – Bitový multiplexer . . . . .	239
COUNT – Řízený čítač . . . . .	240
EATMT – Extended finite-state automaton . . . . .	241
EDGE_ – Detekce hrany logického signálu . . . . .	244
INTSM – Bitový posun a maska nad celým číslem . . . . .	245
ISSW – Jednoduchý přepínač celočíselných signálů . . . . .	246
ITOI – Transformace celých a binárních čísel . . . . .	247
NOT_ – Logická negace . . . . .	249
OR_ – Logický součet dvou signálů . . . . .	250
ORQUAD, OROCT, ORHEXD – Logický součet více signálů . . . . .	251
RS – Klopný obvod . . . . .	252
SR – Klopný obvod . . . . .	253
TIMER_ – Vícefunkční časovač . . . . .	254
<b>9 TIME – Bloky pro práci s časem</b>	<b>257</b>
DATE_ – Aktuální datum . . . . .	258
DATETIME – Čtení, nastavování a konverze času . . . . .	259
TIME – Aktuální čas . . . . .	262
WSCH – Týdenní časovač . . . . .	263

<b>10 ARC – Archivace dat</b>	<b>265</b>
10.1 Funkce archivačního subsystému	266
10.2 Generování alarmů u a událostí	267
ALB, ALBI – Alarmy pro logickou hodnotu	267
ALN, ALNI – Alarmy pro číselnou hodnotu	269
10.3 Záznam trendů	271
ACD – Archivní komprese s použitím delta kritéria	271
TRND – Záznam trendů v reálném čase	273
TRNDV – Záznam trendů v reálném čase (vektorová forma)	276
TRNDLF – * Záznam trendů v reálném čase (lock-free)	278
TRNDVLF – * Záznam trendů v reálném čase (pro vektory, lock-free)	280
10.4 Správa archivů	281
AFLUSH – Vynucené zapsání archivu	281
<b>11 STRING – Bloky pro práci s řetězci</b>	<b>283</b>
CNS – * Textová konstanta	284
CONCAT – * Spojení stringů (podle vzoru)	285
FIND – * Nalezení textu	286
LEN – * Délka textu	287
MID – * Výřez textu	288
PJROCT – * Získání číselných hodnot z textu ve formátu JSON	289
PJSOCT – * Získání textových hodnot z textu ve formátu JSON	291
REGEXP – Regular expresion parser	293
REPLACE – * Náhrada textu	294
RTOS – * Konverze čísla na text	295
SELSOCT – * Výběr textu z několika vstupů	296
STOR – * Koverze textu na číslo	297
<b>12 PARAM – Bloky pro manipulaci s parametry</b>	<b>299</b>
GETPA – Blok pro vzdálené získání vektorového parametru	300
GETPR, GETPI, GETPB – Bloky pro vzdálené získání parametru	302
GETPS – * Blok pro vzdálené získání parametru typu string	304
PARA – Blok s vektorovým parametrem nastavitelným ze vstupu	305
PARR, PARI, PARB – Bloky s nastavitelným parametrem ze vstupu	306
PARS – * Blok s parametrem typu string nastavitelným ze vstupu	308
SETPA – Blok pro vzdálené nastavování vektorového parametru	309
SETPR, SETPI, SETPB – Bloky pro vzdálené nastavování parametru	311
SETPS – * Blok pro vzdálené nastavování parametru typu string	313
SGSLP – Nastavování, čtení, ukládání a načítání parametrů	314
SILO – Uložení vstupního signálu, načtení výstupního signálu	318
SILOS – Uložení vstupního řetězce, načtení výstupního řetězce	319

<b>13 MODEL – Simulace dynamických systémů</b>	<b>321</b>
CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním . . . . .	322
CSSM – Stavový model spojitého lineárního systému . . . . .	325
DDELSSM – Stavový model diskrétního lineárního systému s dopravním zpožděním . . . . .	328
DSSM – Stavový model diskrétního lineárního systému . . . . .	330
FMUCS – Import modelu FMU CS (pro Co-Simulation) . . . . .	332
FMUINFO – Informace o importovaném modelu FMU . . . . .	334
FOPDT – Model systému 1. řádu s dopravním zpožděním . . . . .	336
MDL – Model procesu . . . . .	337
MDLI – Model procesu s proměnnými parametry . . . . .	338
MVD – Motorizovaný pohon ventilu . . . . .	339
SOPDT – Model systému 2. řádu s dopravním zpožděním . . . . .	340
<b>14 MATRIX – Bloky pro maticové a vektorové operace</b>	<b>343</b>
CNA – * Konstantní pole (vektor/matice) . . . . .	344
RTOV – Vektorový multiplexer . . . . .	345
SWVMR – Přepínač vektorového/maticového/odkazovacího signálu . . . . .	347
VTOR – Vektorový demultiplexer . . . . .	348
<b>15 SPEC – Speciální bloky</b>	<b>349</b>
EPC – Blok pro spouštění externích programů . . . . .	350
HTTP – * Blok pro generování požadavků HTTP GET a POST . . . . .	353
SMTP – * Blok pro odesílání e-mailových oznámení přes SMTP . . . . .	355
RDC – Komunikační blok . . . . .	357
REXLANG – Volně programovatelný blok . . . . .	362
<b>16 MC_SINGLE – Řízení pohybu v jedné ose</b>	<b>379</b>
RM_Axis – Osa pro řízení pohybu . . . . .	382
MC_AccelerationProfile, MCP_AccelerationProfile – Generování trajektorie (zrychlení) . . . . .	388
MC_Halt, MCP_Halt – Zastavení pohybu (přerušitelné) . . . . .	392
MC_HaltSuperimposed, MCP_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné) . . . . .	394
MC_Home, MCP_Home – Nalezení výchozí polohy . . . . .	395
MC_MoveAbsolute, MCP_MoveAbsolute – Pohyb do pozice (absolutní souřadnice) . . . . .	397
MC_MoveAdditive, MCP_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	401
MC_MoveRelative, MCP_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění) . . . . .	404
MC_MoveSuperimposed, MCP_MoveSuperimposed – Pohyb do pozice (přídavný pohyb) . . . . .	407



MC_MoveContinuousAbsolute, MCP_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice) . . . . .	410
MC_MoveContinuousRelative, MCP_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	414
MC_MoveVelocity, MCP_MoveVelocity – Pohyb konstantní rychlostí . . . . .	418
MC_PositionProfile, MCP_PositionProfile – Generování trajektorie (poloha) . . . . .	421
MC_Power – Aktivace osy . . . . .	425
MC_ReadActualPosition – Skutečná poloha osy . . . . .	426
MC_ReadAxisError – Chyba osy . . . . .	427
MC_ReadBoolParameter – Čtení parametru (logická hodnota) . . . . .	428
MC_ReadParameter – Čtení parametru (číselná hodnota) . . . . .	429
MC_ReadStatus – Stav osy . . . . .	431
MC_Reset – Nulování chyb osy . . . . .	433
MC_SetOverride, MCP_SetOverride – Nastavení násobivých faktorů na ose . . . . .	434
MC_Stop, MCP_Stop – Zastavení pohybu . . . . .	436
MC_TorqueControl, MCP_TorqueControl – Řízení síly/momentu . . . . .	438
MC_VelocityProfile, MCP_VelocityProfile – Generování trajektorie (rychlost) . . . . .	441
MC_WriteBoolParameter – Nastavení parametru (logická hodnota) . . . . .	445
MC_WriteParameter – Nastavení parametru (číselná hodnota) . . . . .	446
RM_AxisOut – Výstupní blok osy . . . . .	448
RM_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení) . . . . .	450
RM_Track – Sledování a krokování . . . . .	452
<b>17 MC _MULTI – Řízení pohybu více os</b> . . . . .	<b>455</b>
MC_CamIn, MCP_CamIn – Zapnutí vačky . . . . .	456
MC_CamOut – Vypnutí vačky . . . . .	460
MCP_CamTableSelect – Definice vačky . . . . .	462
MC_CombineAxes, MCP_CombineAxes – Kombinace pohybu dvou os do třetí . . . . .	464
MC_GearIn, MCP_GearIn – Zapnutí konstantního převodového poměru . . . . .	467
MC_GearInPos, MCP_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici . . . . .	470
MC_GearOut – Vypnutí konstantního převodového poměru . . . . .	475
MC_PhasingAbsolute, MCP_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice) . . . . .	477
MC_PhasingRelative, MCP_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění) . . . . .	480
<b>18 MC _COORD – Koordinované řízení pohybu</b> . . . . .	<b>483</b>
RM_AxesGroup – Skupina os pro koordinované řízení pohybu . . . . .	489
RM_Feed – * MC krmič ??? . . . . .	493
RM_Gcode – * CNC řízení pohybu . . . . .	494
MC_AddAxisToGroup – Přidání osy do skupiny os . . . . .	496

MC_UngroupAllAxes – Odebrání všech os ze skupiny . . . . .	497
MC_GroupEnable – Převedení skupiny do stavu GroupStandby . . . . .	498
MC_GroupDisable – Převedení skupiny do stavu GroupDisabled . . . . .	499
MC_SetCartesianTransform, MCP_SetCartesianTransform – Kartézská transformace . . . . .	500
MC_ReadCartesianTransform – Přčtení použité kartézské transformace . . . . .	503
MC_GroupSetPosition, MCP_GroupSetPosition – Nastavení polohového offsetu skupiny os . . . . .	504
MC_GroupReadActualPosition – Aktuální poloha skupiny os . . . . .	506
MC_GroupReadActualVelocity – Aktuální rychlost skupiny os . . . . .	507
MC_GroupReadActualAcceleration – Aktuální zrychlení skupiny os . . . . .	508
MC_GroupStop, MCP_GroupStop – Zastavení koordinovaného pohybu . . . . .	509
MC_GroupHalt, MCP_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné) . . . . .	512
MC_GroupInterrupt, MCP_GroupInterrupt – Přerušení pohybu skupiny os . . . . .	517
MC_GroupContinue – Pokračování v přerušném pohybu . . . . .	518
MC_GroupReadStatus – Stav skupin os . . . . .	519
MC_GroupReadError – Chyby ve skupině os . . . . .	521
MC_GroupReset – Nulování chyb os ve skupině . . . . .	522
MC_MoveLinearAbsolute, MCP_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice) . . . . .	523
MC_MoveLinearRelative, MCP_MoveLinearRelative – Pohyb do pozice po přímkách (relativní souřadnice) . . . . .	527
MC_MoveCircularAbsolute, MCP_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice) . . . . .	531
MC_MoveCircularRelative, MCP_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice) . . . . .	536
MC_MoveDirectAbsolute, MCP_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice) . . . . .	541
MC_MoveDirectRelative, MCP_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice) . . . . .	545
MC_MovePath, MCP_MovePath – Generování obecné trajektorie v prostoru . . . . .	549
MC_GroupSetOverride, MCP_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině . . . . .	552
<b>A Seznam funkčních bloků a jejich licencování</b>	<b>555</b>
<b>B Chybové kódy systému REX</b>	<b>565</b>
<b>Literatura</b>	<b>571</b>
<b>Rejstřík</b>	<b>573</b>

*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.

**16 MC\_SINGLE – Řízení pohybu v jedné ose**

Bloky této podknihovny byly vyvinuty dle normy PLCopen Motion Control pro řízení pohybu v jedné ose.

**17 MC\_MULTI – Řízení pohybu ve více osách**

Bloky této podknihovny byly vyvinuty dle normy PLCopen Motion Control pro řízení pohybu ve více osách.

**18 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. [ $\odot$ <def>] [ $\downarrow$ <min>] [ $\uparrow$ <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  $\odot$  a podobně i minimální příp. maximální přípustou hodnotu, uvozenou symbolem  $\downarrow$ , příp.  $\uparrow$ . Všechny tyto tři hodnoty mohou být uvedeny ve druhém sloupci, ale nejsou povinné (jsou umístěny v [ ]). Pokud není uvedena hodnota  $\odot$ <def>, je vždy tato hodnota nulová. Není-li uvedena hodnota  $\downarrow$ <min> příp.  $\uparrow$ <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
<b>Bitová pole</b>	<b>Kvalita</b>		<b>Substatus</b>				<b>Omezení</b>	
	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

---

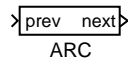
ARC – Archiv systému REX . . . . .	18
EXEC – Exekutiva reálného času . . . . .	20
HMI – * Konfigurace vizualizace . . . . .	22
INFO – * Dodatečné informace o projektu . . . . .	23
IODRV – Vstupně-výstupní ovladač systému REX . . . . .	24
IOTASK – Úloha řídicího systému REX spouštěná ovladačem . . . . .	26
LPBRK – Rozpojení zpětné vazby . . . . .	27
MODULE – Rozšiřující modul systému REX . . . . .	28
PROJECT – * Další nastavení projektu . . . . .	29
QTASK – Rychlá úloha řídicího systému REX . . . . .	30
SLEEP – Časovací blok pro Simulink . . . . .	31
SRTF – Blok pro nastavování příznaků běhu . . . . .	32
OSCALL – Volání funkcí operačního systému . . . . .	34
TASK – Standardní úloha řídicího systému REX . . . . .	35
TIODRV – Vstupně-výstupní ovladač systému REX s úlohami . . . . .	37
WWW – * Obsah pro interní webserver . . . . .	39

---

## 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

<b>prev</b>	Vstup sloužící pro připojení prvního archivu na výstup <b>Archives</b> bloku <b>EXEC</b> nebo k připojení na výstup <b>next</b> předchozího archivu	<b>long</b>
-------------	---	-------------

### Výstup

<b>next</b>	Výstup sloužící pro zřetězování archivů připojením na vstup <b>prev</b> následujícího archivu	<b>long</b>
-------------	---	-------------

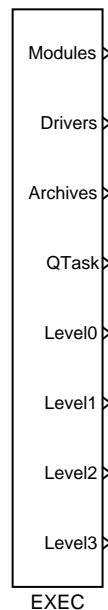
## Parametry

<b>atype</b>	Typ archivu	⊙1	long
	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)		
<b>asize</b>	Velikost archivu (v bytech)	↓256 ⊙102400	long
<b>nmarks</b>	Počet časových značek pro urychlení sekvenčního vyhledávání v archivu	↓2 ⊙720	long
<b>ldaymax</b>	Maximální velikost archivu za den [byte]	↓1000 ↑2147480000 ⊙1048576	large
<b>period</b>	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 konfiguruje 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, nejnižší 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

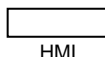
Modules	Výstup pro připojování rozšiřujících modulů systému REX, viz blok <a href="#">MODULE</a>	long
Drivers	Výstup pro připojování vstupně výstupních ovladačů systému REX, viz bloky <a href="#">IODRV</a> a <a href="#">TIODRV</a>	long
Archives	Výstup pro konfiguraci archivů, viz blok <a href="#">ARC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

## Parametry

target	Cílové zařízení 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 vizualizace**

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.

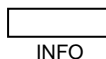
## Parametry

<code>IncludeHMI</code>	Zahrnout soubory HMI do projektu	<input type="radio"/> on	bool
<code>HmiDir</code>	Výstupní adresář pro soubory vizualizace (HMI)	<input type="radio"/> hmi	string
<code>SourceDir</code>	Zdrojový adresář	<input type="radio"/> hmisrc	string
<code>GenerateWebWatch</code>	Vygenerovat WebWatch vizualizaci z MDL souborů	<input type="radio"/> on	bool
<code>GenerateRexHMI</code>	Při překladu projektu vygenerovat HMI ze SVG a JS souborů	<input type="radio"/> on	bool
<code>RedirectToHMI</code>	Webserver bude automaticky přesměrovávat na stránku s HMI	<input type="radio"/> on	bool
<code>Compression</code>	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.

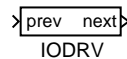
### Parametry

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".

Zbýlé 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

<code>prev</code>	Vstup sloužící pro k připojení prvního ovladače na výstup <code>Drivers</code> bloku <code>EXEC</code> nebo k připojení na výstup <code>next</code> předchozího ovladače	<code>long</code>
-------------------	--	-------------------

## Výstup

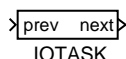
<code>next</code>	Výstup sloužící pro zřetězování ovladačů připojením na vstup <code>prev</code> následujícího ovladače	<code>long</code>
-------------------	---	-------------------

## Parametry

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

## 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 nevyhovuje, 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

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

## 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 umísťuje 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

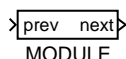
u	Vstupní signál	double
---	----------------	--------

### 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.

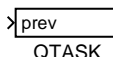
### Parametry

<code>CompileParams</code>	Parametry příkazového řádku RexComp	<code>string</code>
<code>TargetURL</code>	Cílová URL adresa	<code>string</code>

## 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

<b>prev</b>	Vstup, sloužící pro k připojení k výstupu <b>QTask</b> bloku <a href="#">EXEC</a>	<b>long</b>
-------------	---	-------------

### Parametry

<b>factor</b>	Násobek času <b>tick</b> bloku <a href="#">EXEC</a> určující periodu úlohy ( $\text{factor} * \text{tick}$ )	<b>long</b>
		⊙1
<b>stack</b>	Velikost zásobníku (v bytech)	⊙10240 <b>long</b>
<b>filename</b>	Jméno souboru s příponou <b>.mdl</b> obsahující algoritmus úlohy; není-li jméno zadáno, je jméno souboru určeno jménem tohoto bloku (v hlavním souboru projektu) doplněném příponou <b>.mdl</b>	<b>string</b>

## 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).

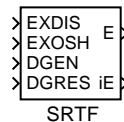
### Parametr

<b>ts</b>	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

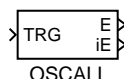
E	Příznak chyby off ... bez chyby on .... nastala chyba	bool
iE	Kód chyby (při E = on) 0 ..... bez chyby 1 ..... objekt nebyl nalezen, neplatný parametr <b>bname</b> 2 ..... interní chyba systému REX (nesprávné ukazatele) 3 ..... příznak se nepodařilo nastavit (timeout)	long

## Parametr

bname	Úplná cesta k bloku (objektu), rozlišuje malá a velká písmena. Jednotlivé vrstvy jsou oddělovány tečkami, názvy objektů kromě úloh ( <b>TASK</b> , <b>QTASK</b> ) 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 ( <b>IOTASK</b> ) 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

<code>TRG</code>	Spuštění zvolené akce	<code>bool</code>
------------------	-----------------------	-------------------

### Výstupy

<code>E</code>	Příznak chyby	<code>bool</code>
<code>iE</code>	Kód chyby	<code>long</code>
	<code>i</code> ..... obecná chyba systému <code>REX</code>	

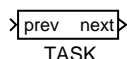
### Parametr

<code>action</code>	Systémová funkce	⊙1 <code>long</code>
	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 tiků č. **start** do tiků č. **stop**, přičemž parametry **start** a **stop** musí splňovat podmínku  $0 \leq \text{start} < \text{stop} \leq \text{ntick< i>}$ . Navíc musí být splněna podmínka postupného spouštění úloh kontrolovaná překladačem **RexComp** říkající, že parametr **stop** předchází úlohy nesmí být větší než parametr **start** úlohy následující (interval 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

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

### Výstup

<b>next</b>	Výstup sloužící pro zřetězování úloh dané úrovně připojením na vstup <b>prev</b> následující úlohy téže úrovně	<b>long</b>
-------------	--	-------------

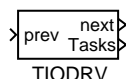
## Parametry

<b>factor</b>	Faktor spouštění, násobek periody $\text{tick} * \text{ntick} \langle i \rangle$ bloku $i$ -té výpočetní úrovně bloku <b>EXEC</b> určující periodu úlohy ( $\text{factor} * \text{tick} * \text{ntick} \langle i \rangle$ ) $\odot 1$	<b>long</b>
<b>start</b>	Číslo tiky periody dané výpočetní úrovně, na kterém má být úloha spuštěna $\downarrow 0 \uparrow \text{ntick} \langle i \rangle$	<b>long</b>
<b>stop</b>	Číslo tiky periody dané výpočetní úrovně, do kterého má být úloha dokončena $\downarrow \text{start} + 1 \uparrow \text{ntick} \langle i \rangle$	<b>long</b>
<b>stack</b>	Velikost zásobníku (v bytech) $\odot 10240$	<b>long</b>
<b>filename</b>	Jméno souboru s příponou <b>.mdl</b> obsahující algoritmus úlohy. Není-li jméno zadáno, je jméno souboru určeno jménem tohoto bloku (v hlavním souboru projektu) doplněným příponou <b>.mdl</b> .	<b>string</b>

## 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

<b>prev</b>	Vstup sloužící pro k připojení prvního ovladače na výstup <b>Drivers</b> bloku <b>EXEC</b> nebo k připojení na výstup <b>next</b> předchozího ovladače	long
-------------	--	------

### Výstupy

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

### Parametry

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

<code>timer</code>	Ovladač je zdrojem pro časování	<code>bool</code>
--------------------	---------------------------------	-------------------

## 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

<b>Source</b>	Zdrojový adresář	<b>string</b>
<b>Target</b>	Cílový adresář	<b>string</b>
<b>Compression</b>	Aktivovat kompresi dat	<b>bool</b>





## Kapitola 3

# INOUT – Bloky vstupů a výstupů systému REX

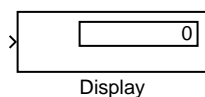
### Obsah

Display – * Zobrazení vstupní hodnoty . . . . .	42
From, INSTD – Připojení signálu nebo vstupní signál . . . . .	43
Goto, OUTSTD – Zdroj signálu nebo výstupní signál . . . . .	45
GotoTagVisibility – Viditelnost zdroje signálu . . . . .	47
Inport, Outport – Vstupní a výstupní port . . . . .	48
SubSystem – Subsystém . . . . .	50
INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů . . . . .	51
OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů . . . . .	53
OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací . . . . .	55
OUTRSTD – Výstupní signál s verifikací hodnoty . . . . .	57
QFC – Kódování příznaků kvality signálu . . . . .	58
QFD – Dekódování příznaků kvality signálu . . . . .	59
VIN – Ověření kvality vstupního signálu . . . . .	60
VOUT – Nastavení kvality výstupního signálu . . . . .	62

## 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
---	----------------	---------

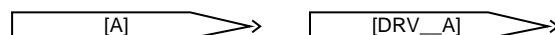
### Parametry

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

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

## Parametr

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

## Goto, OUTSTD – Zdroj signálu nebo výstupní signál

Symbole 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**. Těto 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

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

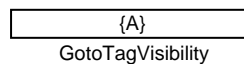
## Parametry

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

## 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 `Rex-Comp` a ve výsledné konfiguraci není obsažen, protože v reálném čase nevykonává žádnou činnost.

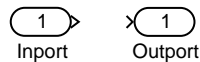
### Parametr

<code>GotoTag</code>	Odkaz na parametr <code>GotoTag</code> bloku <code>Goto</code> , jehož viditelnost je dána <code>string</code> umístěním tohoto bloku <code>GotoTagVisibility</code>
----------------------	--

## 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 přechíslovány. Pozor, pokud jsou přechíslovány 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

<b>value</b>	Hodnota odcházející na výstupní připojení nebo do bloku <b>Inport</b>	<b>unknown</b>
--------------	---	----------------

### Výstup

<b>value</b>	Hodnota přicházející ze vstupního připojení nebo bloku <b>Outport</b>	<b>unknown</b>
--------------	---	----------------



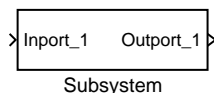
## Parametr

Port	Číslo portu bloku <code>Inport</code> nebo <code>Outport</code>	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.

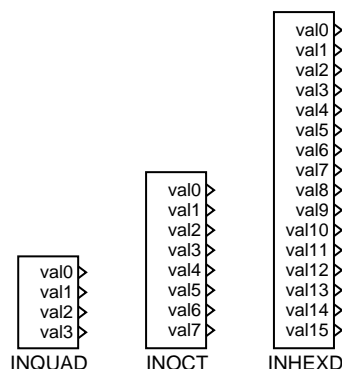
### Výstupy

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 popsany 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.

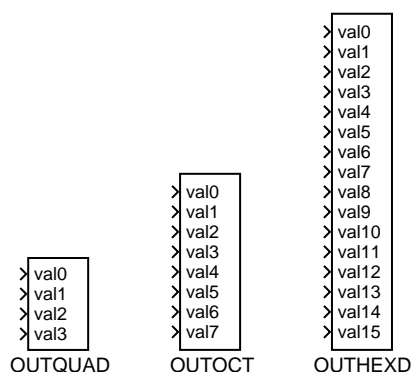
## Výstupy

<code>vali</code>	Vstupní signály z procesu přivedené prostřednictvím I/O ovladačů do řídicího algoritmu. Typ a umístění jednotlivých signálů je popsáno v uživatelské příručce příslušného ovladače.	<code>unknown</code>
-------------------	---	----------------------

## 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 popsání způsobu 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.

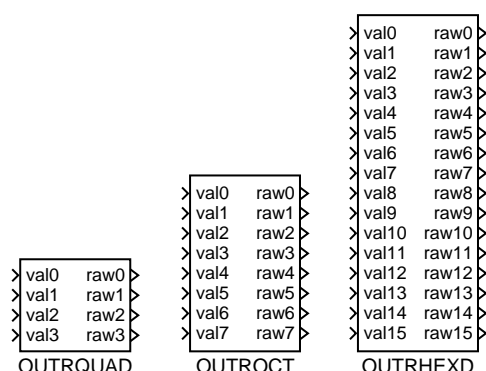
## Vstupy

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

## OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací

Symbody 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 dvojitým 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 prostřednictvím I/O ovladačů. Typ a umístění jednotlivých signálů je popsáno v uživatelské příručce příslušného ovladače. **unknown**

## Výstupy

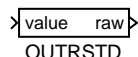
<code>rawi</code>	Zpětná informace od ovladače o výsledku nastavení odpovídajícího výstupu. Typ a význam signálů je popsán v uživatelské příručce příslušného ovladače.	<code>unknown</code>
-------------------	---	----------------------



## 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 řídicího algoritmu o výsledku zápisu dvojitým 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

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

### Výstup

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

## 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

<b>iq</b>	Základní příznaky kvality, viz tab. <a href="#">1.2</a> , str. <a href="#">16</a>	long
<b>is</b>	Doplňující příznaky kvality, viz <a href="#">[1]</a>	long
<b>il</b>	Příznaky dosažení mezních úrovní, viz <a href="#">[1]</a>	long

### Výstup

<b>iqf</b>	Bitová kombinace vstupních signálů <b>iq</b> , <b>is</b> a <b>il</b>	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 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 **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

<b>iqf</b>	Příznaky kvality, které mají být dekomponovány na složky <b>iq</b> , <b>is</b> a <b>il</b>	<b>long</b>
------------	--	-------------

### Výstupy

<b>iq</b>	Příznaky základního typu kvality, viz tabulku <a href="#">1.2</a> , str. <a href="#">16</a>	<b>long</b>
<b>is</b>	Doplňující příznaky kvality, viz <a href="#">[1]</a>	<b>long</b>
<b>il</b>	Příznaky dosažení mezních úrovní, viz <a href="#">[1]</a>	<b>long</b>

## 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

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

### Výstupy

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

## Parametr

GU	Připustnost kvality UNCERTAIN	bool
	off ... kvalita UNCERTAIN je nepřipustná	
	on .... kvalita UNCERTAIN je připustná	

## 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

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

### Výstup

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

## Kapitola 4

# MATH – Matematické bloky

### Obsah

---

ABS_ – Absolutní hodnota . . . . .	65
ADD – Součet dvou signálů . . . . .	66
ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů . . . . .	67
CNB – Booleovská (logická) konstanta . . . . .	68
CNE – Předdefinovaná konstanta . . . . .	69
CNI – Celočíselná konstanta . . . . .	70
CNR – Reálná konstanta . . . . .	71
DIF_ – Blok difference . . . . .	72
DIV – Dělení dvou signálů . . . . .	73
EAS – Rozšířené sčítání a odečítání . . . . .	74
EMD – Rozšířené násobení a dělení . . . . .	75
FNX – Výpočet hodnoty funkce jedné proměnné . . . . .	76
FNXY – Výpočet hodnoty funkce dvou proměnných . . . . .	78
GAIN – Násobení konstantou . . . . .	80
GRADS – Gradientní optimalizace . . . . .	81
IADD – Celočíselné sčítání . . . . .	83
ISUB – Celočíselné odčítání . . . . .	85
IMUL – Celočíselné násobení . . . . .	87
IDIV – Celočíselné dělení . . . . .	89
IMOD – Zbytek po celočíselném dělení . . . . .	90
LIN – Lineární interpolace . . . . .	91
MUL – Násobení dvou signálů . . . . .	92
POL – Vyhodnocení polynomu . . . . .	93
REC – Převrácená hodnota . . . . .	94
REL – Relační operace dvou signálů . . . . .	95
RTOI – Konverze reálného čísla na celé číslo . . . . .	96

SQR – Druhá mocnina . . . . .	97
SQRT_ – Druhá odmocnina . . . . .	98
SUB – Odčítání dvou signálů . . . . .	99

---



## 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,

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

### Vstup

<b>u</b>	Analogový vstupní signál	double
----------	--------------------------	--------

### Výstupy

<b>y</b>	Absolutní hodnota vstupního signálu	double
<b>sgn</b>	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

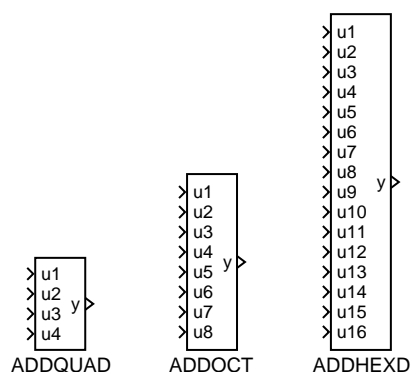
### Výstup

y	Součet vstupních signálů	double
---	--------------------------	--------

## ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky ADDQUAD, ADDOCT a ADDHEXD sčítají více (až 16) vstupních signálů. Parametr *n1* udává seznam vstupů, které se místo přičtení odečítají. Pokud je tento parametr prázdný, tak blok provádí funkci  $y = u1 + u2 + u3 + u4 + u5 + u6 + u7 + \dots + u16$ . Pokud bude například  $n1=2,5,7$ , tak bude realizována funkce  $y = u1 - u2 + u3 + u4 - u5 + u6 - u7 + \dots + u16$ .

Pro jednoduché operace sčítání a odečítání můžete použít bloky [ADD](#) a [SUB](#).

### Vstupy

u1..u16	Analogové vstupní signály	double
---------	---------------------------	--------

### Výstup

y	Výsledná hodnota	double
---	------------------	--------

### Parametr

n1	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 decimálně 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
---	-------------------------	------

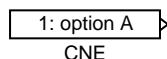
### Parametr

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

## 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

<b>yenum</b>	Konstanta ze seznamu	⊙1: option A	string
<b>pupstr</b>	Definice seznamu konstant	⊙1: option A 2: option B 3: option C	string

### Výstup

<b>iy</b>	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 difference

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	bool
	<code>off</code> ... v prvním cyklu bude na výstupu $y = u$ <code>on</code> ... v prvním cyklu bude výstup $y = 0$	



## 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  $E = \text{on}$  a na výstup  $y$  je dána náhradní hodnota  $y = \text{yerr}$ .

### Vstupy

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

### Výstupy

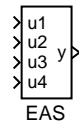
y	Podíl vstupních signálů	double
E	Indikátor chyby – dělení nulou	bool

### 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
-----	---------------------------	--------

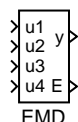
## Parametry

$a$	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

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

V případě, že jmenovatel vztahu (4.1) je roven 0, nastaví se výstup  $E = \text{on}$  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 určena parametrem **ifn**.

Tabulka funkcí bloku FNX:

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

Tabulka funkcí bloku **FNXY**:

ifn: zkratka	funkce	omezení u1, u2
1: <b>atan2</b>	arcus tangens $u1/u2$	–
2: <b>fmod</b>	zbytek po dělení $u1/u2$	$u2 \neq 0.0$
3: <b>pow</b>	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  $y = \mathbf{yerr}$ .

### Vstupy

<b>u1</b>	První analogový vstup bloku	<b>double</b>
<b>u2</b>	Druhý analogový vstup bloku	<b>double</b>

### Výstupy

<b>y</b>	Výsledek vybrané funkce	<b>double</b>
----------	-------------------------	---------------

<b>E</b>	Příznak chyby	<b>bool</b>
	<b>off</b> ... bez chyby	
	<b>on</b> .... nastala chyba	

## Parametry

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

## 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	<code>double</code>
-----	--------------------------	---------------------

Výstup

$y$	Analogový výstupní signál	<code>double</code>
-----	---------------------------	---------------------

Parametr

$k$	Zesílení	<code>⊙1.0</code> <code>double</code>
-----	----------	---------------------------------------



## GRADS – Gradientní optimalizace

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

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

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

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

### Vstupy

<b>f</b>	Hodnota minimalizované funkce $f(\cdot)$ v bodě $\mathbf{x}$	double
<b>x0</b>	Startovní bod optimalizace	double
<b>START</b>	Spouštěcí signál (reaguje na náběžnou hranu)	bool
<b>BRK</b>	Signál pro předčasné přerušení	bool

### Výstupy

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

## Parametry

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

## 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  $-32768 \dots +32767$ , 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	$\downarrow -9,22E+18 \uparrow 9,22E+18$	long
i2	Druhý celočíselný vstup bloku	$\downarrow -9,22E+18 \uparrow 9,22E+18$	long

### Výstupy

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

### Parametry

vtype	Typ hodnoty, může nabývat hodnot:	⊙4 long
	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)	

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

## 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  $-32768 \dots +32767$  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	$\downarrow -9,22E+18$ $\uparrow 9,22E+18$	long
i2	Druhý celočíselný vstup bloku	$\downarrow -9,22E+18$ $\uparrow 9,22E+18$	long

### Výstupy

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

### Parametry

vtype	Číselný typ	⊙4 long
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)	

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

## 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  $-32768 \dots +32767$  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	$\downarrow -9,22E+18 \uparrow 9,22E+18$	long
i2	Druhý celočíselný vstup bloku	$\downarrow -9,22E+18 \uparrow 9,22E+18$	long

### Výstupy

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

### Parametry

vtype	Číselný typ	⊙4 long
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)	

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



## IDIV – Celočíselné dělení

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok IDIV dělí dva vstupní celočíselné signály  $n = i1 \div i2$ , kde  $\div$  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	$\downarrow -9,22E+18 \uparrow 9,22E+18$	long
i2	Druhý celočíselný vstup bloku	$\downarrow -9,22E+18 \uparrow 9,22E+18$	long

### Výstupy

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

### Parametr

vtype	Typ hodnoty, může nabývat hodnot:	$\odot 1$ long
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)	
nerr	Náhradní hodnota pro případ chyby	$\odot 1$ 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:	⊙1	long
	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)		
nerr	Náhradní hodnota pro případ chyby	⊙1	long

## LIN – Lineární interpolace

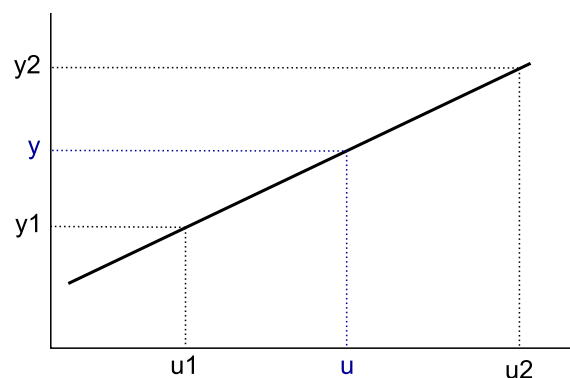
Symbol bloku

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:

$$y = a_0 + a_1 u + a_2 u^2 + a_3 u^3 + a_4 u^4 + a_5 u^5 + a_6 u^6 + a_7 u^7 + a_8 u^8.$$

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

### Vstup

$u$	Analogový vstupní signál	double
-----	--------------------------	--------

### Parametry

$a_i$	Koeficient $i$ -té mocniny vstupu, $i = 0, 1, \dots, 8$	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  $E = \text{on}$  a na výstupu **y** je náhradní hodnota **yerr**.

### Vstup

<b>u</b>	Analogový vstupní signál	<b>double</b>
----------	--------------------------	---------------

### Výstupy

<b>y</b>	Analogový výstupní signál	<b>double</b>
<b>E</b>	Indikátor chyby – dělení nulou	<b>bool</b>

### Parametr

<b>yerr</b>	Náhradní hodnota pro případ chyby	⊙1.0 <b>double</b>
-------------	-----------------------------------	--------------------

## REL – Relační operace dvou signálů

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **REL** vyhodnocuje binární relaci  $u1 \circ u2$  z hodnot vstupů a podle výsledku relace „ $\circ$ “ 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

<b>u1</b>	První analogový vstup bloku	double
<b>u2</b>	Druhý analogový vstup bloku	double

### Výstup

<b>Y</b>	Logický výstupní signál indikující platnost relace	bool
----------	--	------

### Parametr

<b>irel</b>	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:

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

kde  $\text{round}(r)$  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	<code>double</code>
-----	--------------------------	---------------------

### Výstup

$i$	Zaokrouhlený a zkonvertovaný vstupní signál	<code>long</code>
-----	---	-------------------



## 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

<b>u</b>	Analogový vstupní signál	double
----------	--------------------------	--------

### Výstupy

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

### Parametr

<b>yerr</b>	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

---

ABSR0T – Zpracování dat z absolutního snímače polohy . . . . .	103
ASW – Přepínač s automatickou volbou vstupu . . . . .	105
AVG – Filtr: vlečný průměr . . . . .	107
AVS – Rozběhová jednotka . . . . .	108
BPF – Filtr: pásmová propust' . . . . .	109
CMP – Komparátor s hysterezí . . . . .	110
CNDR – Kompenzátor složité nelinearity . . . . .	111
DEL – Dopravní zpoždění s inicializací . . . . .	113
DELM – Dopravní zpoždění . . . . .	114
DER – Derivace, filtrace a predikce z posledních $n+1$ vzorků . . .	115
EVAR – Vlečná střední hodnota a směrodatná odchylka . . . . .	116
INTE – Řízený integrátor . . . . .	117
KDER – Derivace a filtrace vstupního signálu . . . . .	119
LPF – Filtr: dolní propust' . . . . .	121
MINMAX – Vlečné minimum a maximum . . . . .	122
NSCL – Kompenzátor jednoduché nelinearity . . . . .	123
RDFT – Vlečná diskretní Fourierova transformace . . . . .	124
RLIM – Omezovač strmosti . . . . .	126
S10F2 – Výběr jednoho ze dvou analogových vstupů . . . . .	127
SAI – Zabezpečený analogový vstup . . . . .	130
SEL – Selektor analogového signálu . . . . .	133
SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu . . . .	135
SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot . . . .	137
SHLD – Vzorkovač (sample and hold) . . . . .	139

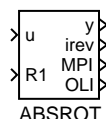
SINT – Jednoduchý integrátor . . . . .	140
SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů . . . . .	141
SSW – Jednoduchý přepínač . . . . .	143
SWR – Přepínač s rampovou funkcí . . . . .	144
VDEL – Dopravní zpoždění s proměnnou délkou . . . . .	145
ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací . . . . .	146

---

## 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.  $5^\circ$  až  $355^\circ$  (popř.  $-175^\circ$  až  $+175^\circ$ ), ale potřebujeme řídit pohyb o několik otáček. Blok předpokládá spojitý signál, takže při přechodu z například  $355^\circ$  na  $5^\circ$  předpokládá, že nastala další otáčka a úhel je ve skutečnosti  $365^\circ$ .

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

<b>u</b>	Signál z absolutního snímače polohy	<b>double</b>
<b>R1</b>	Reset bloku (nastavení výstupu do základního intervalu)	<b>bool</b>

### Výstupy

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

### Parametry

<b>lolim</b>	Dolní mez pro údaj z čidla	$\odot -3.141592654$	<b>double</b>
<b>hilim</b>	Horní mez pro údaj z čidla	$\odot 3.141592654$	<b>double</b>
<b>tol</b>	Rozsah pro indikaci středové pozice	$\odot 0.5$	<b>double</b>
<b>hys</b>	Hystereze pro indikaci středové pozice		<b>double</b>

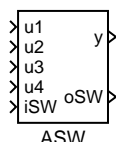
RESR	Příznak pro resetování počítadla otáček	bool
	off ... resetovat pouze vypočtenou polohu y	
	on .... vynulovat i počítadlo otáček irev	



## ASW – Přepínač s automatickou volbou vstupu

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok ASW ukládá na výstup  $y$  hodnotu jednoho ze vstupů vstup  $u1, \dots, u4$  nebo jeden z parametrů  $p1, \dots, p4$ . 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  $u1, \dots, u4$  nebo parametrů  $p1, \dots, p4$ , který se naposledy změnil. Pokud se změnil 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  $p1, \dots, p4$  (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 nadřazených 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  $u1, \dots, u4$  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

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

Není-li ještě k dispozici všech  $n$  vzorků (po startu algoritmu), 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	long
	↓1 ↑10000000 ⊙10	
$n_{\max}$	Maximální hodnota parametru $n$ (na tuto velikost se alokuje paměť)	long
	↓1 ↑10000000 ⊙10	

## 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

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

### Výstupy

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

## BPF – Filtr: pásmová propust

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok BPF realizuje přenos filtru druhého řádu ve tvaru

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

kde  $a$  a  $\xi$  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

<b>u</b>	Vstupní signál filtru	<b>double</b>
----------	-----------------------	---------------

### Výstup

<b>y</b>	Filtrovaný výstupní signál	<b>double</b>
----------	----------------------------	---------------

### Parametry

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

## CMP – Komparátor s hysterezí

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

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

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

kde

$$e_k = u1_k - u2_k$$

a

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

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

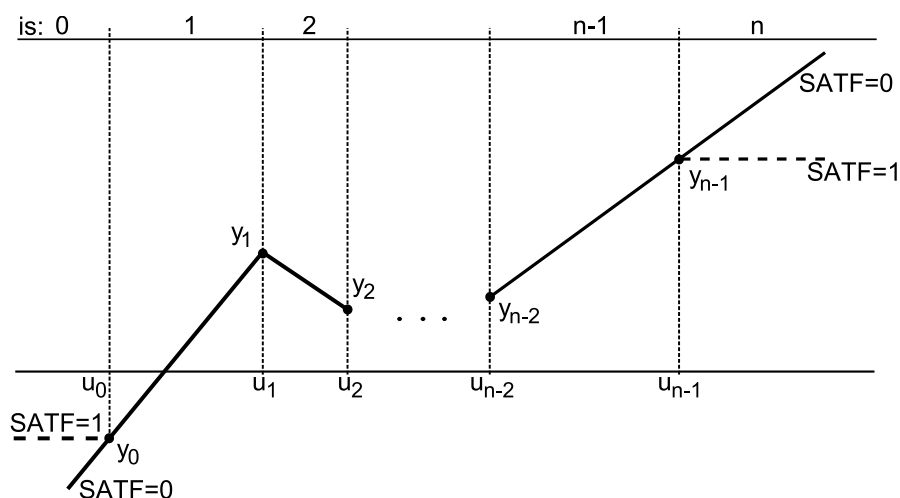
Symbol bloku

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  $u < u_0$  a  $u > u_{n-1}$  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
-----	------------------------------	---------

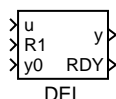
SATF	Saturace v koncových uzlech	⊙on	bool
	off ... signál není omezen on ... saturační meze jsou aktivní		
up	Vektor rostoucích $u$ souřadnic uzlů	⊙[-1 1]	double
yp	Vektor $y$ souřadnic uzlů	⊙[-1 1]	double



## 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 ( $R1$ :  $\text{off} \rightarrow \text{on} \rightarrow \text{off}$ ) dosud není zapamatovaných  $n$  minulých vzorků ( $RDY = \text{off}$ ), potom

$$y_k = y_0,$$

kde  $y_0$  je inicializační vstup bloku.

### Vstupy

$u$	Analogový vstupní signál	double
$R1$	Reset bloku	bool
$y_0$	Počáteční hodnota výstupu	double

### Výstupy

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

### Parametr

$n$	Zpoždění (počet vzorků) – příslušné časové zpoždění je $n \cdot T_S$ , kde $T_S$ je perioda spouštění bloku	long ↓0 ↑10000000 ⊙10
$n_{\max}$	Maximální velikost parametru $n$ (na tolik hodnot se alokuje paměť)	long ↓1 ↑10000000 ⊙10

## 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

<b>u</b>	Analogový vstupní signál	<b>double</b>
----------	--------------------------	---------------

### Výstup

<b>y</b>	Zpožděný vstupní signál	<b>double</b>
----------	-------------------------	---------------

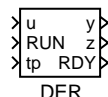
### Parametr

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

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

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok DER prokládá posledních  $n + 1$  vzorků ( $n \leq N - 1$ ,  $N$  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  $RUN = \text{on}$  výstupy  $y$  a  $z$  podle vztahů:

$$\begin{aligned} \text{Derivace: } y &= a \\ \text{Filtrace: } z &= b, \text{ pro } t_p = 0 \\ \text{Predikce: } z &= at_p + b, \text{ pro } t_p > 0 \\ \text{Postdikce: } z &= at_p + b, \text{ pro } t_p < 0 \end{aligned}$$

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

### Vstupy

$u$	Analogový výstupní signál	double
$RUN$	Povolení běhu algoritmu	bool
	off ... sledování ( $z=u$ )	
	on .... filtrace ( $y$ – odhad derivace, $z$ – odhad $u$ v čase $t_p$ )	
$t_p$	Časový okamžik pro predikci/filtraci ( $t_p = 0$ 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); $1 \leq n \leq nmax$	long ↓1 ↑10000000 ⊙10
$nmax$	Maximální hodnota parametru $n$ (na tolik hodnot se alokuje paměť)	long ↓1 ↑10000000 ⊙10

## EVAR – Vlečná střední hodnota a směrodatná odchylka

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **EVAR** počítá střední hodnotu **mu** ( $\mu$ ) a směrodatnou odchylku **si** ( $\sigma$ ) z posledních **n** vzorků vstupního signálu **u** podle vztahů

$$\mu_k = \frac{1}{n} \sum_{i=0}^{n-1} u_{k-i}$$

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

kde  $k$  značí aktuální okamžik vzorkování.

### Vstup

<b>u</b>	Analogový vstupní signál	<b>double</b>
----------	--------------------------	---------------

### Výstupy

<b>mu</b>	Střední hodnota vstupního signálu	<b>double</b>
<b>si</b>	Směrodatná odchylka vstupního signálu	<b>double</b>

### Parametr

<b>n</b>	Počet vzorků pro výpočet statistických ukazatelů	<b>long</b>
		↓2 ↑10000000 ⊖100
<b>nmax</b>	Maximální velikost parametru <b>n</b> (na tolik hodnot se alokuje paměť)	<b>long</b>
		↓1 ↑10000000 ⊖10

## 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

<b>u</b>	Analogový vstupní signál	<b>double</b>
<b>RUN</b>	Povolení běhu algoritmu	<b>bool</b>
<b>R1</b>	Reset bloku, inicializace výstupu integrátoru na hodnotu <b>y0</b>	<b>bool</b>
<b>y0</b>	Počáteční hodnota výstupu	<b>double</b>
<b>ti</b>	Integrační časová konstanta	<b>double</b>

### Výstupy

<b>y</b>	Výstup integrátoru	<b>double</b>
<b>Q</b>	Příznak probíhající integrace	<b>bool</b>
<b>LY</b>	Příznak dosažení spodní úrovně ( $y < y_{\min}$ )	<b>bool</b>
<b>HY</b>	Příznak dosažení horní úrovně ( $y > y_{\max}$ )	<b>bool</b>

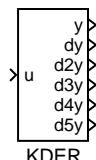
## Parametry

<code>ymin</code>	Nastavení dolní úrovně	⊖1.0	<code>double</code>
<code>ymax</code>	Nastavení horní úrovně	⊕1.0	<code>double</code>

## 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  $\text{pbeta} \approx T_S \omega_0$ . Pro správnou funkci bloku **KDER** by se frekvenční spektrum vstupního signálu  $u$  mělo nacházet hluboko pod zlomovou frekvencí filtru  $\omega_0$ . Naopak frekvenční spektrum šumů by mělo být co možná nejdále od frekvence  $\omega_0$ . Pro vyšší potlačení šumů je nutné volit nižší zlomovou frekvenci  $\omega_0$  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 propust'

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  $f_b$  a  $\xi = \mathbf{xi}$  jsou parametry bloku. Frekvence  $f_b$  [Hz] určuje šířku pásma filtru a parametr  $\mathbf{xi}$  součinitel relativního tlumení filtru. Doporučená hodnota pro Butterworthův filtr je  $\mathbf{xi} = 0,71$  a pro Besselův filtr  $\mathbf{xi} = 0,87$ .

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

### Vstup

$\mathbf{u}$	Vstupní signál filtru	<code>double</code>
--------------	-----------------------	---------------------

### Výstup

$\mathbf{y}$	Filtrovaný výstupní signál	<code>double</code>
--------------	----------------------------	---------------------

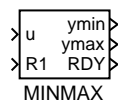
### Parametry

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

## 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  $RDY = \text{off}$  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)	long
	↓1 ↑100000000 Ⓢ100	

## NSCL – Kompenzátor jednoduché nelinearity

Symbol bloku

Licence: [STANDARD](#)

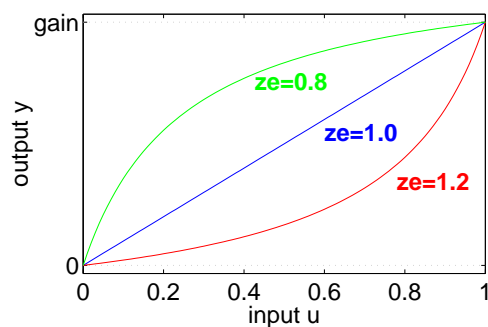


### Popis funkce

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

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

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



### Vstup

u	Analogový vstupní signál	double
---	--------------------------	--------

### Výstup

y	Analogový výstupní signál	double
---	---------------------------	--------

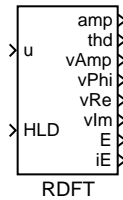
### Parametry

gain	Zesílení	⊙1.0	double
ze	Tvarovací parametr	⊙1.0	double

## RDFT – Vlečná diskretní Fourierova transformace

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok RDFT počítá diskretní Fourierovu transformaci vstupního signálu pro základní frekvenci **freq** (a případně několik dalších) z posledních **m** vzorků vstupního signálu **u**, kde  $m = n_{per}/freq/T_s$ , 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

<b>u</b>	Analogový vstupní signál	double
<b>HLD</b>	Pozastavení funkce bloku	bool

### Výstupy

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

## Parametry

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

## 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  $t_p$  a  $t_n$  podle následujících vztahů:

$$\begin{aligned} \text{maximální nárůst za sekundu:} & \quad 1/t_p \\ \text{maximální pokles za sekundu:} & \quad -1/t_n \end{aligned}$$

## Vstup

$u$	Vstupní signál filtru	double
-----	-----------------------	--------

## Výstup

$y$	Filtrovaný výstupní signál	double
-----	----------------------------	--------

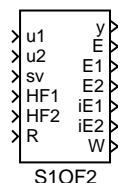
## Parametry

$t_p$	Časová konstanta určující maximální růst	⊙2.0	double
$t_n$	Časová konstanta určující maximální pokles (pozor, $t_n > 0$ )	⊙2.0	double

## S10F2 – Výběr jednoho ze dvou analogových vstupů

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok **S10F2** 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 **S10F2** 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

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

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:

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

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  $R = \text{on}$ , pak v případě náběžné hrany vnitřního příznaku  $D$  ( $\text{off} \rightarrow \text{on}$ ) je výstup  $ER$  nahozen pouze po dobu jednoho cyklu. Při  $R = \text{off}$  je nastaveno  $ER = \text{on}$  až do následného resetování. Výstup  $W$  má hodnotu  $\text{on}$  pouze v případech (iii) a (iv), tzn. pokud alespoň jeden z výstupů  $E1$  a  $E2$  má hodnotu  $\text{on}$ , 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 $u1$ $\text{off}$ ... vstupní modul signálu pracuje normálně $\text{on}$ .... došlo k hardwarové chybě vstupního modulu	bool
$HF2$	Příznak hardwarové chyby vstupu $u2$ $\text{off}$ ... vstupní modul signálu pracuje normálně $\text{on}$ .... 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 $y$ $\text{off}$ ... signál je platný $\text{on}$ .... signál není platný	bool
$E1$	Indikátor neplatnosti vstupu $u1$ $\text{off}$ ... signál je platný $\text{on}$ .... signál není platný	bool
$E2$	Indikátor neplatnosti vstupu $u2$ $\text{off}$ ... signál je platný $\text{on}$ .... 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) $\text{off}$ ... oba vstupní signály jsou platné $\text{on}$ .... alespoň jeden vstupní signál je neplatný	bool

### Parametry

$nb$	Počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálů $u1$ a $u2$	long
------	---	------

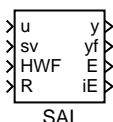


<b>nc</b>	Počet vzorků pro testování neměnnosti (viz blok <a href="#">SAI</a> , podmínka F2)	long
	⊙10	
<b>nbits</b>	Počet bitů A/D převodníku vstupního modulu (zdroje signálů <b>u1</b> a <b>u2</b> )	long
	⊙12	
<b>nr</b>	Počet vzorků pro testování variability (viz blok <a href="#">SAI</a> , podmínka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

## SAI – Zabezpečený analogový vstup

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

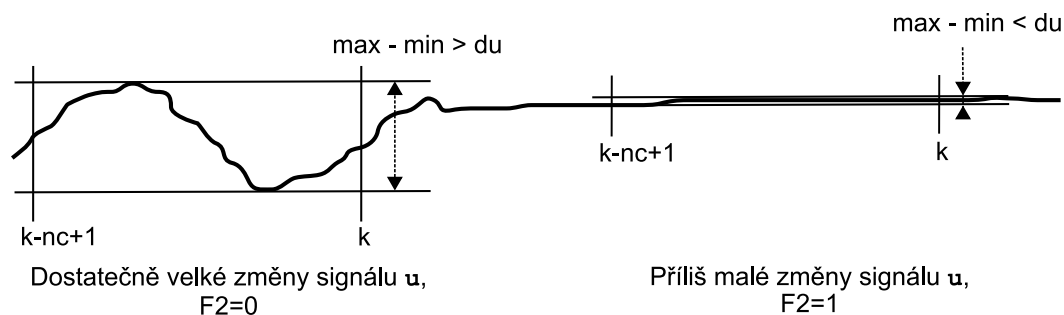
Blok **SAI** testuje vstupní signál  $u$  s cílem rozpoznání jeho platnosti. Vstupní signál  $u$  se považuje za neplatný (výstup  $E = \text{on}$ ) v následujících případech:

**F1: Hardwarová chyba.** Vstupní signál  $\text{HWF} = \text{on}$ .

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

$$du = \begin{cases} \frac{v_{\max} - v_{\min}}{2^{n_{\text{bits}}}}, & \text{pro } n_{\text{bits}} \in \{8, 9, \dots, 16\} \\ 0, & \text{pro } n_{\text{bits}} \notin \{8, 9, \dots, 16\}, \end{cases}$$

kde  $v_{\min}$  a  $v_{\max}$  jsou po řadě dolní a horní mez vstupu  $u$  a  $n_{\text{bits}}$  je počet bitů příslušného A/D převodníku. Situace, kdy je splněna podmínka příliš malé změny  $u$ , je zobrazena na následujícím obrázku:

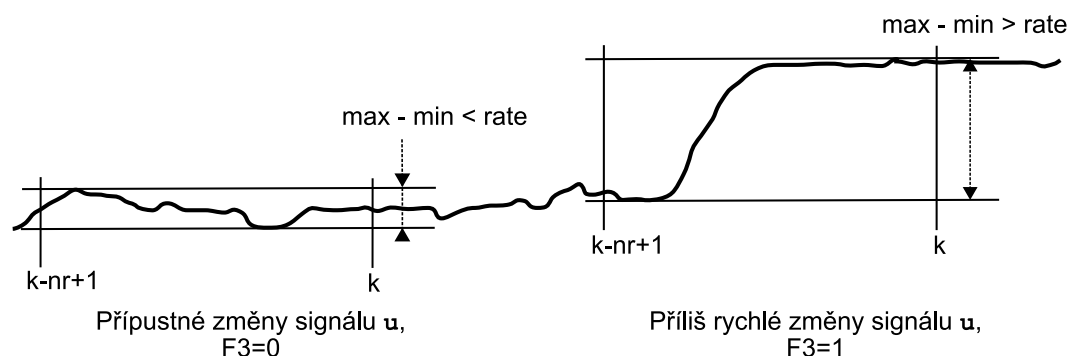


Jestliže je nastaveno  $nc = 0$ , potom podmínka F2 není splněna nikdy.

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

$$rate = prate \frac{v_{\max} - v_{\min}}{100},$$

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  $\langle vmin, vmax \rangle$ . 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`  $\geq 5$  z důvodu odeznění počátečních podmínek **SPIKE** filtru.

## Vstupy

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

## Výstupy

<b>y</b>	Analogový výstupní signál	<b>double</b>
<b>yf</b>	Výstupní signál <b>y</b> filtrovaný <b>SPIKE</b> algoritmem	<b>double</b>
<b>E</b>	Indikátor neplatnosti výstupního signálu <b>off</b> ... výstup je platný <b>yf = sv</b> <b>on</b> .... výstup není platný, <b>y =</b>	<b>bool</b>
<b>iE</b>	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	<b>long</b>

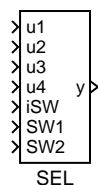
## Parametry

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

## 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

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

### Výstup

<i>y</i>	Zvolený signál	double
----------	----------------	--------

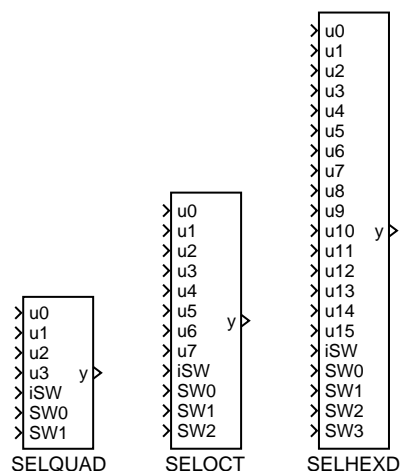
## Parametr

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

## 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

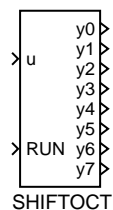
BINF	Výběr pomocí binárních vstupů	bool
	off ... zakázáno (výběr přes iSW)	
	on .... povoleno (výběr přes SWn)	



## 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:

$$\begin{aligned} y_i &= y_{i-1}, \quad i = 1..7, \\ y_0 &= u, \end{aligned}$$

tedy hodnota na každém z výstupů  $y_0$  až  $y_6$  je posunuta na výstup v pořadí následující, a hodnota vstupu  $u$  je přenesena na výstup  $y_0$ .

Blok pracuje s libovolným datovým typem signálu přivedeného na vstup  $u$ . Požadovaný datový typ je třeba nastavit parametrem **vtype**. Výstupy  $y_0$  až  $y_7$  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

$y_0$	První výstup bloku	unknown
$y_1$	Druhý výstup bloku	unknown
$y_2$	Třetí výstup bloku	unknown
$y_3$	Čtvrtý výstup bloku	unknown
$y_4$	Pátý výstup bloku	unknown
$y_5$	Šestý výstup bloku	unknown
$y_6$	Sedmý výstup bloku	unknown

y7

Osmý výstup bloku

unknown

## Parametry

vtype

Typ výstupů

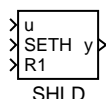
⊙8 long

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

## 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 ( $\text{off} \rightarrow \text{on}$ ) 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  $y_0$  a má prioritu před vstupem **SETH**.

### Vstupy

$u$	Analogový vstupní signál	double
<b>SETH</b>	Vstup pro nastavení a podržení výstupní hodnoty	bool
<b>R1</b>	Reset bloku, $R1 = \text{on} \rightarrow y = y_0$	bool

### Výstup

$y$	Analogový výstupní signál	double
-----	---------------------------	--------

### Parametr

$y_0$	Počáteční hodnota výstupu $y$	double
<code>mode</code>	Režim vzorkování	⊙3 long
	1 . . . . . Vynucené vzorkování	
	2 . . . . . Držení předchozí hodnoty	
	3 . . . . . Držení aktuální hodnoty	

## SINT – Jednoduchý integrátor

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok SINT realizuje diskrétní integrátor popsany diferencní rovnicí

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

kde  $T_S$  je perioda spouštění bloku a  $t_i$  je integrační konstanta. Je-li  $y_k$  mimo saturační meze  $y_{\min}$ ,  $y_{\max}$ , 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
y <sub>max</sub>	Horní saturační mez	⊙1.0	double
y <sub>min</sub>	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  $(u, y) \rightarrow y$ :

```
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
  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	<b>double</b>
-----	-----------------------	---------------

### Výstup

$y$	Filtrovaný výstupní signál	<b>double</b>
-----	----------------------------	---------------

## 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

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

### Výstup

<b>y</b>	Analogový výstupní signál	<b>double</b>
----------	---------------------------	---------------

## 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 vybrán 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

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

### Výstup

<b>y</b>	Analogový výstupní signál	<b>double</b>
----------	---------------------------	---------------

### Parametry

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



## 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  $T_S$  spuštění bloku. Jestliže po spuštění bloku není dosud k dispozici  $n$  posledních vzorků, potom je výstup  $y$  nastaven na inicializační hodnotu  $y_0$ .

### Vstupy

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

### Výstup

$y$	Zpožděný vstupní signál	double
-----	-------------------------	--------

### Parametr

$y_0$	Počáteční/náhradní hodnota výstupu	double
$n_{\max}$	Délka vyrovnávací paměti pro dopravní zpoždění $d$ (na tolik hodnot se alokuje paměť)	long
		↓1 ↑10000000 ⊕10

## ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací

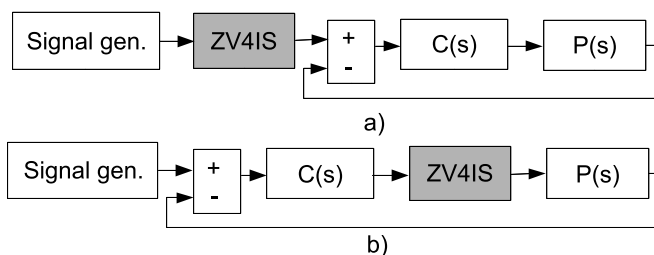
Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Blok ZV4IS realizuje funkci frekvenčního filtru typu pásmová zadrž. 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  $P(s)$ ). Záporům 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.

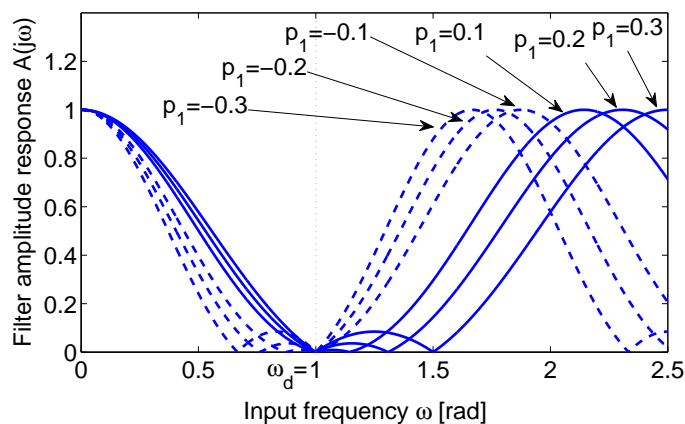
Vlastní algoritmus filtrace lze popsat v časové oblasti vztahem

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

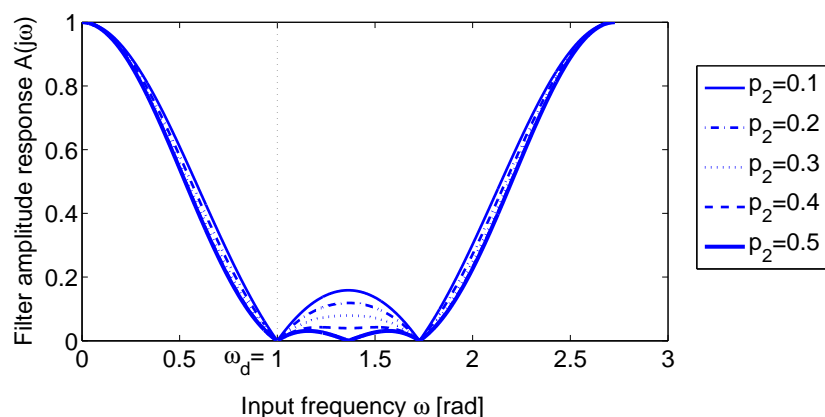
Filtr má tedy strukturu sumy vážených dopravních zpoždění kde zesílení  $A_1..A_4$  a hodnoty zpoždění  $t_1..t_4$  závisí na volbě typu filtru, frekvenci a tlumení kmitavého módu systému. Výhodou uvedené struktury oproti klasickým dynamickým notch filtrům užívaným v regulační technice je konečná impulzní 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  $\text{ipar}$  pak udává typ tvarovacího filtru, pro  $\text{ipar} = 1$  je použit jeden z deseti základních filtrů, které se volí parametrem  $\text{istype}$ . 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).

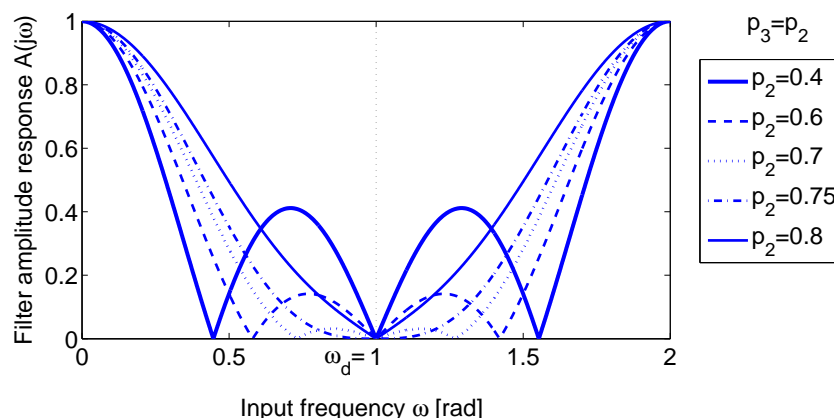
Pro jemné ladění filtru lze použít kompletní parametrizaci volbou  $\text{ipar} = 2$ , pro kterou se zpřístupní volné parametry  $\text{p\_alpha}$ ,  $\text{p\_a2}$  a  $\text{p\_a3}$ . Ty určují tvar frekvenční charakteristiky filtru a lze je použít pro nalezení optimálního kompromisu mezi robustností a zavedeným zpožděním.



Parametr asymetrie  $\text{p\_alpha}$  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  $\text{p\_alpha}$  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

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



## Kapitola 6

# GEN – Generátory signálů

### Obsah

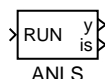
---

ANLS – Řízený generátor po částech lineární funkce . . . . .	152
BINS – Řízený generátor binární posloupnosti . . . . .	154
BIS – Generátor binární posloupnosti . . . . .	156
MP – Ručně generovaný pulz . . . . .	157
PRBS – Pseudonáhodná binární posloupnost . . . . .	158
SG, SGI – Řízený generátor signálu . . . . .	160

---

## 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  $t_1, y_1; t_2, y_2; t_3, y_3; t_4, y_4$ . Počáteční hodnota  $y$  je definována parametrem  $y_0$ . Start generování funkce (časový okamžik 0) je určen náběžnou hranou vstupu **RUN**. V intervalu  $\langle t_i, t_{i+1} \rangle, i = 0, \dots, 3, t_0 = 0$  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  $y_0$  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  $t_1 < \dots < t_i$ . 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

<b>RUN</b>	Povolení generování posloupnosti	<b>bool</b>
------------	----------------------------------	-------------

## Výstupy

<b>y</b>	Analogový výstupní signál	<b>double</b>
<b>is</b>	Index aktivního časového úseku	<b>long</b>

## Parametry

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



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

<b>START</b>	Spouštěcí signál (náběžná hrana)	<b>bool</b>
--------------	----------------------------------	-------------

### Výstupy

<b>Y</b>	Logický výstupní signál	<b>bool</b>
<b>is</b>	Index aktivního časového úseku	<b>long</b>

### Parametry

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

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		bool
	off ... zakázáno		
	on .... povoleno		

## 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

<b>Y</b>	Logický výstupní signál	bool
<b>is</b>	Index aktivního časového úseku	long

### Parametry

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

<code>pwidth</code>	Šířka pulzu [s]	⊙1.0	double
<code>BSTATE</code>	Aktivace výstupního pulzu		bool
	<code>off</code> ... žádná činnost		
	<code>on</code> .... vygenerování výstupního pulzu		
<code>RPFT</code>	Povolení prodloužení pulzu		bool
	<code>off</code> ... zakázáno	<code>on</code> ....	povoleno

## PRBS – Pseudonáhodná binární posloupnost

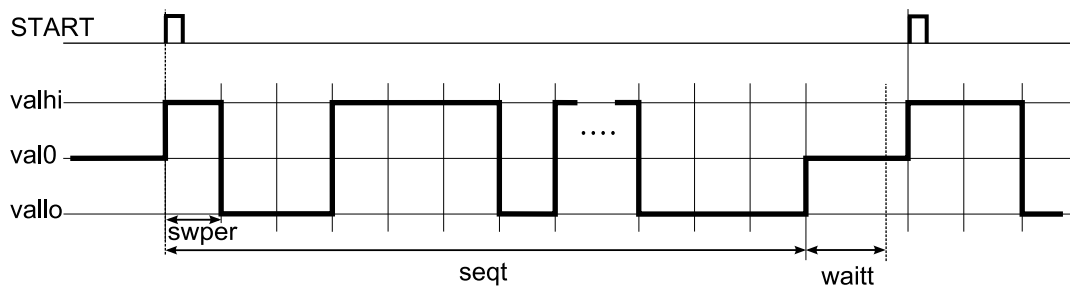
Symbol bloku

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

<code>START</code>	Spouštěcí signál (náběžná hrana)	<code>bool</code>
<code>BRK</code>	Signál pro přerušení	<code>bool</code>

### Výstupy

<code>y</code>	Vygenerovaná pseudonáhodná binární posloupnost	<code>double</code>
<code>BSY</code>	Příznak probíhající operace	<code>bool</code>

### Parametry

<code>val0</code>	Počáteční a koncová hodnota	<code>double</code>
<code>valhi</code>	Horní hladina výstupu <code>y</code>	©1.0 <code>double</code>

<code>vallo</code>	Dolní hladina výstupu $y$	$\odot -1.0$	<code>double</code>
<code>swper</code>	Perioda náhodného přepínání výstupu $y$ mezi hladinami [s]	$\odot 1.0$	<code>double</code>
<code>swprob</code>	Pravděpodobnost přepnutí	$\downarrow 0.0 \uparrow 1.0 \odot 0.2$	<code>double</code>
<code>seqt</code>	Délka posloupnosti [s]	$\odot 10.0$	<code>double</code>
<code>waitt</code>	Doba ustálení [s]	$\odot 2.0$	<code>double</code>

## 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 parametry **amp** a **freq**. Výstup **y** v případech  $\text{isig} \in \{1, 2, 3\}$  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

<b>RUN</b>	Povolení běhu algoritmu, spuštění generátoru	<b>bool</b>
<b>SYN</b>	Synchronizační signál	<b>bool</b>

### Výstup

<b>y</b>	Analogový výstupní signál	<b>double</b>
----------	---------------------------	---------------

### Parametry

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



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

`⊙1`    `long`



## Kapitola 7

# REG – Bloky pro regulaci

### Obsah

---

ARLY – Relé s předstihem . . . . .	165
FLCU – Fuzzy regulátor . . . . .	166
FRID – * Identifikace frekvenční charakteristiky . . . . .	168
I3PM – Identifikace modelu se třemi parametry . . . . .	170
LC – Derivační kompenzátor . . . . .	172
LLC – Integračně-derivační kompenzátor . . . . .	173
MCU – Jednotka pro ruční zadávání . . . . .	174
PIDAT – PID regulátor s reléovým autotunerem . . . . .	176
PIDE – PID regulátor se statikou . . . . .	179
PIDGS – PID regulátor s přepínáním sad parametrů . . . . .	181
PIDMA – PID regulátor s momentovým autotunerem . . . . .	183
PIDU – PID regulátor . . . . .	189
PIDUI – PID regulátor s parametry na vstupech . . . . .	192
POUT – Pulzní výstup . . . . .	194
PRGM – Programátor . . . . .	195
PSMPC – Prediktivní „pulse-step“ regulátor . . . . .	197
PWM – Blok šířkové modulace . . . . .	201
RLY – Relé s hysterezí . . . . .	203
SAT – Saturace výstupu s proměnnými mezemi . . . . .	204
SC2FA – Stavový regulátor systému 2. řádu s autotunerem . . . . .	206
SCU – Krokový regulátor s polohovou zpětnou vazbou . . . . .	212
SCUV – Krokový regulátor s rychlostním výstupem . . . . .	215
SELU – Selektor aktivního regulátoru . . . . .	219
SMHCC – Regulátor pro procesy s topením a chlazením . . . . .	221
SMHCCA – * Regulátor pro procesy s topením a chlazením s autotunerem . . . . .	225

SWU – Přepínač vstupu pro vysledování . . . . .	228
TSE – Třístavový prvek . . . . .	229

---

## ARLY – Relé s předstihem

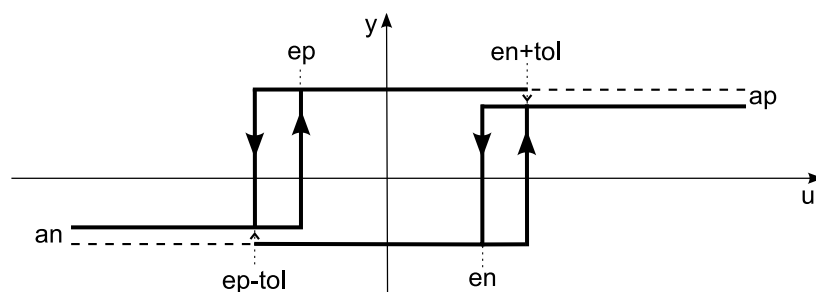
Symbol bloku

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

## FLCU – Fuzzy regulátor

Symbol bloku

Licence: **ADVANCED**



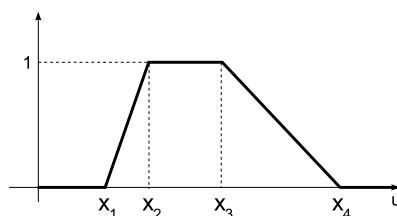
### Popis funkce

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:

Jestliže ( $u$  je  $U_i$ ) AND ( $v$  je  $V_j$ ), potom ( $y$  je  $Y_k$ ),

kde  $U_i, i = 1, \dots, nu$  jsou jazykové výrazy příslušné ke vstupu  $u$ ;  $V_j, j = 1, \dots, nv$  jsou jazykové výrazy příslušné ke vstupu  $v$  a  $Y_k, k = 1, \dots, ny$  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  $x_1, \dots, x_4$  vesměs různá. Matice funkcí příslušnosti vstupů  $u$  a  $v$  se potom skládají z řádků  $[x_1, x_2, x_3, x_4]$ . Matice  $\mathbf{mf}_u$  a  $\mathbf{mf}_v$  jsou tedy po řadě typu  $(nu \times 4)$  a  $(nv \times 4)$ .

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

$$y_k, a_k, b_k,$$

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

Soubor pravidel se skládá též jako matice a její řádky jsou  $[i_l, j_l, k_l, w_l], l = 1, \dots, nr$ , 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  $w_l \in \{0, 1, \dots, 100\}$ . Tímto způsobem lze jednoduše některé pravidlo zdůraznit, popřípadě vypustit.

## Vstupy

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

## Parametry

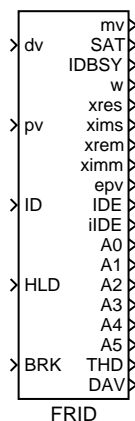
umax	Horní omezení vstupu u	$\odot 1.0$	double
umin	Dolní omezení vstupu u	$\odot -1.0$	double
nu	Počet funkcí příslušnosti – vstup u	$\downarrow 1 \uparrow 25 \odot 3$	long
vmax	Horní omezení vstupu v	$\odot 1.0$	double
vmin	Dolní omezení vstupu v	$\odot -1.0$	double
nv	Počet funkcí příslušnosti – vstup v	$\downarrow 1 \uparrow 25 \odot 3$	long
ny	Počet funkcí příslušnosti – výstup y	$\downarrow 1 \uparrow 100 \odot 3$	long
nr	Počet pravidel	$\downarrow 1 \uparrow 25 \odot 3$	long
mfu	Matice funkcí příslušnosti – vstup u	$\odot [-1 \ -1 \ -1 \ 0; \ -1 \ 0 \ 0 \ 1; \ 0 \ 1 \ 1 \ 1]$	double
mfv	Matice funkcí příslušnosti – vstup v	$\odot [-1 \ -1 \ -1 \ 0; \ -1 \ 0 \ 0 \ 1; \ 0 \ 1 \ 1 \ 1]$	double
sty	Matice funkcí příslušnosti – výstup y	$\odot [-1 \ 0 \ 0; \ 0 \ 0 \ 0; \ 1 \ 0 \ 0]$	double
rls	Matice pravidel	$\odot [1 \ 2 \ 3 \ 100; \ 1 \ 1 \ 1 \ 100; \ 1 \ 0 \ 3 \ 100]$	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	Stejnoseměrná 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	⊙2	long
	1 ..... Malá		
	2 ..... Střední		
	3 ..... Velká		
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	long
	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)		
nwm	Počet frekvencí v automatickém režimu		long
wm	Pole frekvencí pro ruční režim [array of rad/s]	double	
	⊙[2.0 4.0 6.0 8.0]		

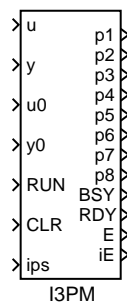
## Výstupy

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 $\mu u_0$	
	p2 ... parametr $\mu u_1$	
	p3 ... parametr $\mu u_2$	
	p4 ... parametr $\mu y_0$	
	p5 ... parametr $\mu y_1$	
	p6 ... parametr $\mu y_2$	
2	..... momenty procesu	
	p1 ... parametr $m p_0$	
	p2 ... parametr $m p_1$	
	p3 ... parametr $m p_2$	

- 3 ..... charakteristická čísla procesu  
     p1 ... parametr  $\kappa$   
     p2 ... parametr  $\mu$   
     p3 ... parametr  $\sigma^2$   
     p4 ... parametr  $\sigma$

## Výstupy

p <i>i</i>	Identifikované parametry v závislosti na ips, $i = 1, \dots, 8$	double
BSY	Příznak probíhající identifikace	bool
RDY	Příznak připravenosti	bool
E	Příznak chyby	bool
iE	Kód chyby	long
	1 ..... předčasné ukončení (RUN = off)	
	2 ..... $\mu_0 = 0$	
	3 ..... $\mu_p = 0$	
	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 članku

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

kde **td** je derivační konstanta a **nd** je parametr určující vliv parazitního filtru prvního řádu. Doporučená hodnota **nd** je  $2 \leq nd \leq 10$ . 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

<b>u</b>	Analogový vstupní signál	<b>double</b>
----------	--------------------------	---------------

### Výstup

<b>y</b>	Analogový výstupní signál	<b>double</b>
----------	---------------------------	---------------

### Parametry

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

## LLC – Integračně-derivační kompenzátor

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok LLC realizuje diskretní simulátor přenosu integračně-derivačního članku

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

kde  $\tau$  je časová konstanta jmenovatele a její  $a$ -násobek ( $a * \tau$ ) je časová konstanta čitatele. Je-li  $ISSF = on$ , potom je stav integračního članku 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		bool
	$off \dots$ nulový počáteční stav		
	$on \dots$ ustálený počáteční stav		

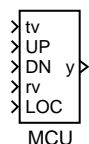
### Výstup

$y$	Analogový výstupní signál	double
-----	---------------------------	--------

## MCU – Jednotka pro ruční zadávání

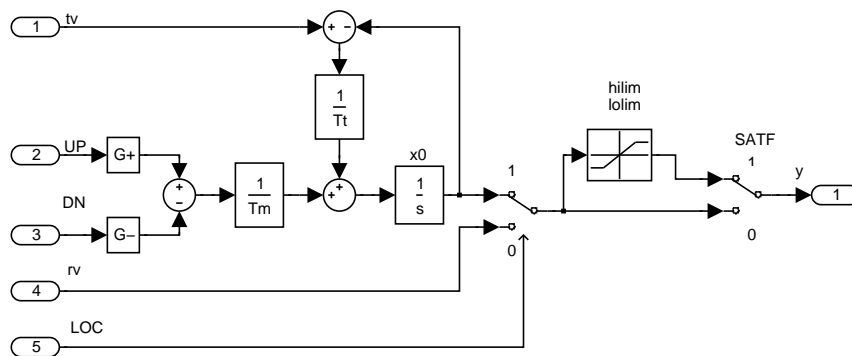
Symbol bloku

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  $y_0$  na žádanou hodnotu je určena integrační konstantou  $t_m$  a dobou stlačení ovládacích tlačítek. Po uplynutí každých  $t_a$  sekund je strmost vždy násobena faktorem  $q$ , až do vypršení doby  $t_f$ . 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
---	---------------------------	--------

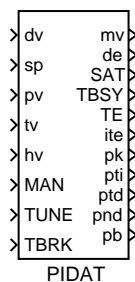
## Parametry

tt	Časová konstanta výsledování 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		bool
	off ... signál není omezen on .... saturační meze jsou aktivní		
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

<b>dv</b>	Proměnná dopředné vazby	<b>double</b>
<b>sp</b>	Požadovaná hodnota (setpoint)	<b>double</b>
<b>pv</b>	Řízená veličina	<b>double</b>
<b>tv</b>	Veličina pro výsledování	<b>double</b>
<b>hv</b>	Hodnota výstupu v manuálním režimu	<b>double</b>



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 ladicího 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

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

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  $\varepsilon$ , 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

<b>dv</b>	Proměnná dopředné vazby	<b>double</b>
<b>sp</b>	Požadovaná hodnota (setpoint)	<b>double</b>

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	

## Výstupy

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

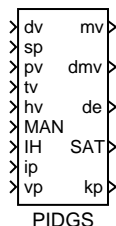
## Parametry

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

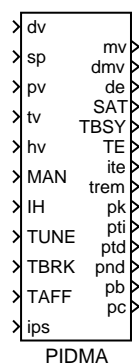
## Parametry

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

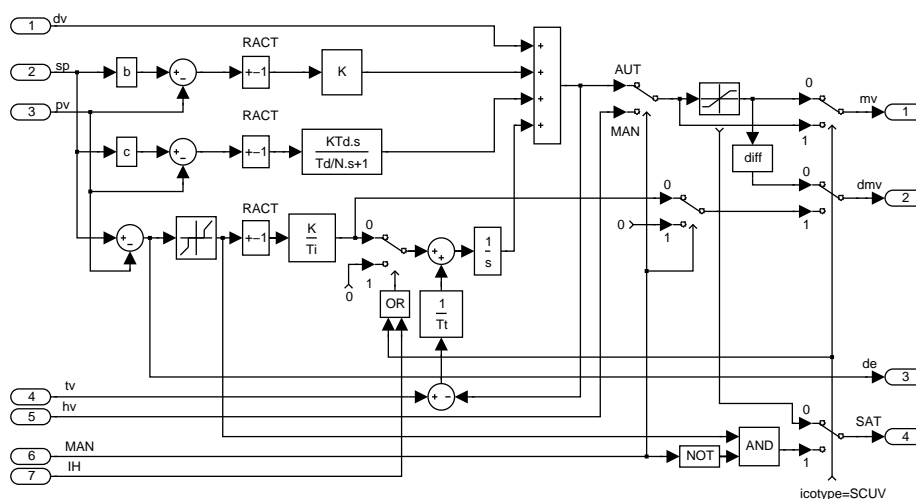
$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s} [W(s) - Y(s)] + \frac{T_d s}{\frac{T_d}{N} s + 1} [cW(s) - Y(s)] \right\} + Z(s)$$

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 vysledování **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.

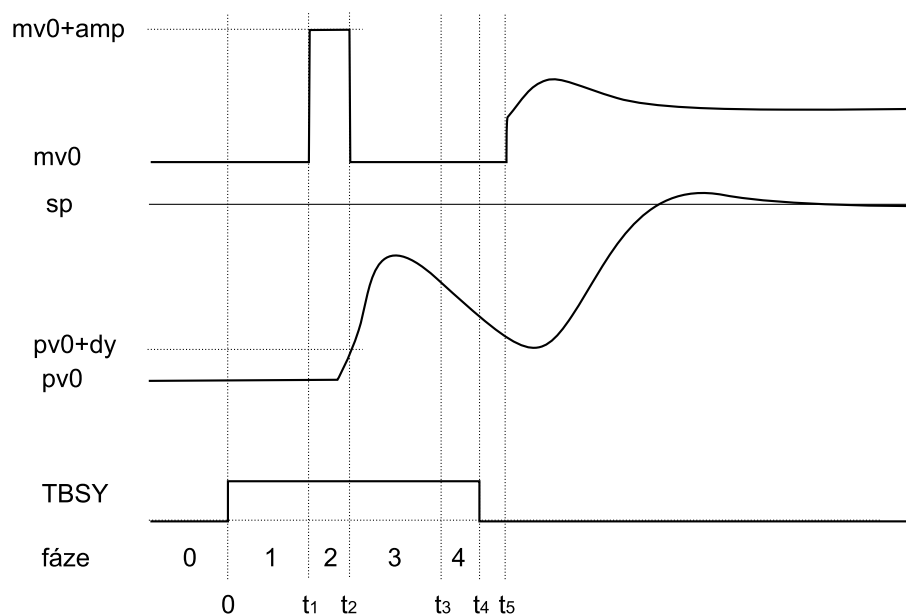


Blok **PIDMA** rozšiřuje řídicí funkci standardního PID regulátoru o vestavěné automatické nastavování parametrů (PID autotuner). Před spuštěním autotuneru musí operátor ve vhodném pracovním bodě dosáhnout ustáleného stavu a zvolit požadovaný typ regulátoru **itype** (PI nebo PID) a nastavit další parametry 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** (zadáva se vždy jako kladné číslo). Pokud je nastaven příznak **DGC**, používá se při zpracování signálu speciální kompenzace trendu signálu. Odhad času zbývajících 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

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

<b>ips</b>	Význam výstupních signálů <b>pk</b> , <b>pti</b> , <b>ptd</b> , <b>pnd</b> , <b>pb</b> a <b>pc</b>	<b>long</b>
0	..... navržené parametry <b>k</b> , <b>ti</b> , <b>td</b> , <b>nd</b> , <b>b</b> a <b>c</b> PID regulátoru	
1	..... momenty procesu: zesílení ( <b>pk</b> ), míra zpoždění soustavy ( <b>pti</b> ), míra délky odezvy soustavy ( <b>ptd</b> )	
2	..... tříparametrový model procesu prvního řádu s dopravním zpožděním: zesílení ( <b>pk</b> ), dopravní zpoždění ( <b>pti</b> ), časová konstanta ( <b>ptd</b> )	
3	..... tříparametrový model procesu druhého řádu s násobnou časovou konstantou a dopravním zpožděním: zesílení ( <b>pk</b> ), dopravní zpoždění ( <b>pti</b> ), časová konstanta ( <b>ptd</b> )	
4	..... odhad mezí intervalu pro manuální doladění zesílení <b>k</b> PID regulátoru ( <b>irtype</b> = 7): horní mez <b>k<sub>hi</sub></b> ( <b>pk</b> ), dolní mez <b>k<sub>lo</sub></b> ( <b>pti</b> )	
>99	... slouží pro diagnostické účely	

## Výstupy

<b>mv</b>	Akční zásah regulátoru (manipulated variable)
<b>dmv</b>	Rychlostní výstup regulátoru (difference)
<b>de</b>	Regulační odchylka
<b>SAT</b>	Saturace
	off ... lineární zákon řízení
	on ... výstup regulátoru je saturován
<b>TBSY</b>	Příznak probíhajícího ladicího experimentu
<b>TE</b>	Příznak chyby během ladění
	off ... Ladění proběhlo bez chyby
	on ... Během ladění se vyskytla chyba
<b>ite</b>	Kód chyby
	<i>Kódy chyb ladění (po experimentu):</i>
	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 podporováno 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 $K$ ( <code>ips = 0</code> )	double
pti	Navržená integrační časová konstanta regulátoru $T_i$ ( <code>ips = 0</code> )	double
ptd	Navržená derivační časová konstanta regulátoru $T_d$ ( <code>ips = 0</code> )	double
pnd	Navržený parametr filtru derivační složky $N$ ( <code>ips = 0</code> )	double
pb	Navržený váhový faktor pro proporcionální složku ( <code>ips = 0</code> )	double
pc	Navržený váhový faktor pro derivační složku ( <code>ips = 0</code> )	double

## Parametry

irtype	Typ regulátoru 1 ..... D    3 ..... ID    5 ..... PD    7 ..... PID 2 ..... I    4 ..... P    6 ..... PI	⊙6	long
RACT	Prevrácené 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 výsledování. 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	Pásma necitlivosti		double
icotype	Typ výstupu regulátoru 1 ..... analogový výstup 2 ..... šířkově modulovaný výstup ( <a href="#">PWM</a> ) 3 ..... krokový regulátor s polohovou zpětnou vazbou ( <a href="#">SCU</a> ) 4 ..... krokový regulátor bez polohové zpětné vazby ( <a href="#">SCUV</a> )	⊙1	long
itttype	Požadovaný typ regulátoru pro návrh 6 ..... PI regulátor 7 ..... PID regulátor	⊙6	long

<b>iainf</b>	Druh apriorní informace		
	1 ..... proces bez integrátoru		
	2 ..... proces s integračním chováním		
<b>DGC</b>	Kompenzace gradientu trendu	⊙on	bool
	off ... zakázáno      on .... povoleno		
<b>tdg</b>	Doba odhadu gradientu trendu [s]	⊙60.0	double
<b>tn</b>	Doba odhadování šumu [s]	⊙5.0	double
<b>amp</b>	Amplituda pulzu	⊙0.5	double
<b>dy</b>	Práh pro ukončení pulsu (absolutní změna od ustálené hodnoty pv)	↓0.0 ⊙0.1	double
<b>ispeed</b>	Požadovaná rychlost uzavřené smyčky	⊙2	long
	1 ..... požadována pomalá uzavřená smyčka		
	2 ..... požadována středně rychlá uzavřená smyčka		
	3 ..... požadována rychlá uzavřená smyčka		
<b>ipid</b>	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

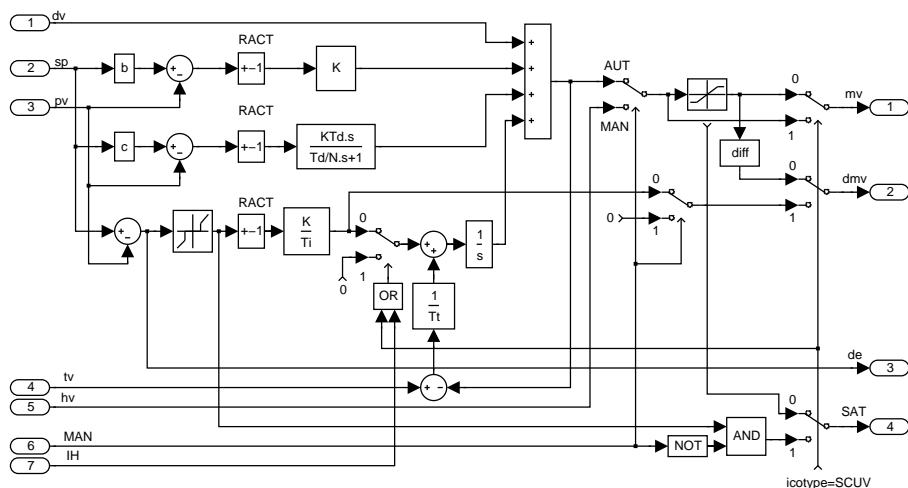
$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s} [W(s) - Y(s)] + \frac{T_d s}{\frac{T_d}{N} s + 1} [cW(s) - Y(s)] \right\} + Z(s)$$

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 vysledování **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 (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 nenarazí 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 (difference)	double
de	Regulační odchylka	double
SAT	Saturace	bool
	off ... lineární zákon řízení	
	on .... výstup regulátoru je saturován	

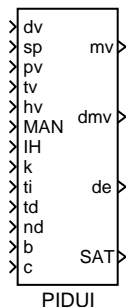
## Parametry

<code>irtype</code>	Typ regulátoru	⊙6	long
	1 ..... D            4 ..... P            7 ..... PID		
	2 ..... I            5 ..... PD		
	3 ..... ID          6 ..... PI		
<code>RACT</code>	Převrácené působení výstupu regulátoru		bool
	off ... vyšší mv → vyšší pv		
	on .... vyšší mv → nižší pv		
<code>k</code>	Zesílení regulátoru $K$	⊙1.0	double
<code>ti</code>	Integrační časová konstanta $T_i$	⊙4.0	double
<code>td</code>	Derivační časová konstanta $T_d$	⊙1.0	double
<code>nd</code>	Parametr $N$ filtru derivační složky	⊙10.0	double
<code>b</code>	Váhový faktor pro proporcionální složku	⊙1.0	double
<code>c</code>	Váhový faktor pro derivační složku		double
<code>tt</code>	Časová konstanta vysledování. Pro regulátory bez integrační složky nemá žádný význam.	⊙1.0	double
<code>hilim</code>	Horní mez akčního zásahu regulátoru	⊙1.0	double
<code>lolim</code>	Dolní mez akčního zásahu regulátoru	⊙-1.0	double
<code>dz</code>	Pásmo necitlivosti		double
<code>icotype</code>	Typ výstupu regulátoru	⊙1	long
	1 ..... analogový výstup		
	2 ..... šířkově modulovaný výstup ( <a href="#">PWM</a> )		
	3 ..... krokový regulátor s polohovou zpětnou vazbou ( <a href="#">SCU</a> )		
	4 ..... krokový regulátor bez polohové zpětné vazby ( <a href="#">SCUV</a> )		

## 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	bool
	off ... automatický režim	
	on .... manuální režim	
IH	Zastavení integrace	bool
	off ... integrování povoleno	
	on .... integrování pozastaveno	
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

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

## Parametry

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

## 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

<b>U</b>	Logický vstupní signál	<b>bool</b>
----------	------------------------	-------------

### Výstup

<b>Y</b>	Logický výstupní signál	<b>bool</b>
----------	-------------------------	-------------

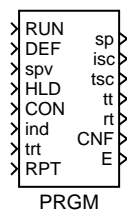
### Parametry

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

## PRGM – Programátor

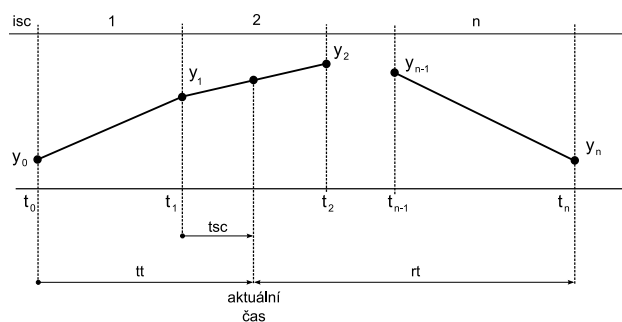
Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **PRGM** je určen pro generování časových funkcí (programů) složených z  $n$  lineárních částí definovaných  $(n + 1)$  rozměrnými vektory  $\mathbf{tm} = [t_0, \dots, t_n]$  času a požadovaných hodnot  $\mathbf{y} = [y_0, \dots, y_n]$  (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 zmrazení 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 <b>sp</b> na hodnotu <b>spv</b>	bool
spv	Inicializační hodnota	double

HLD	Zmrazení výstupu a výstupních časů	bool
CON	Pokračování od uzlového bodu <i>ind</i>	bool
ind	Index uzlového bodu pro pokračování	long
trt	Čas pro dosažení požadovaného uzlu <i>ind</i>	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

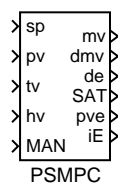
### Parametry

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

## PSMPC – Prediktivní „pulse-step“ regulátor

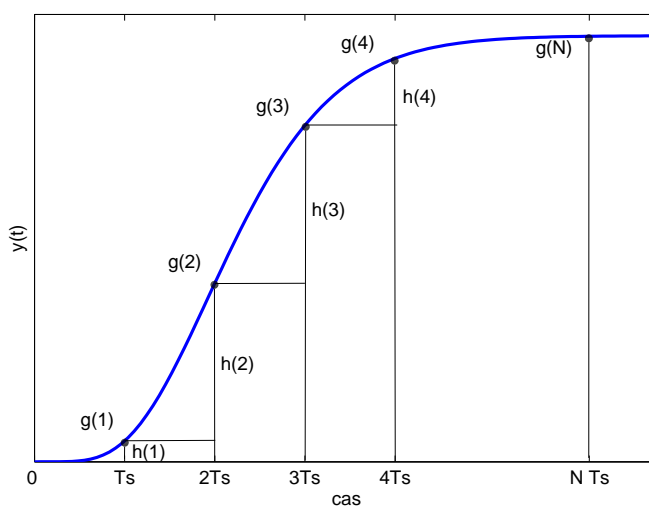
Symbol bloku

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). Zvláště 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  $g(j)$ ,  $j = 1, \dots, N$ . Na obrázku výše je naznačen způsob, jakým lze tuto posloupnost získat ze spojitě 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  $\kappa, \mu$  a  $\sigma^2$ , které je možno určit z jednoduchého pulzního experimentu. Řízený systém pak aproximujeme buď přenosem prvního řádu s dopravním zpožděním

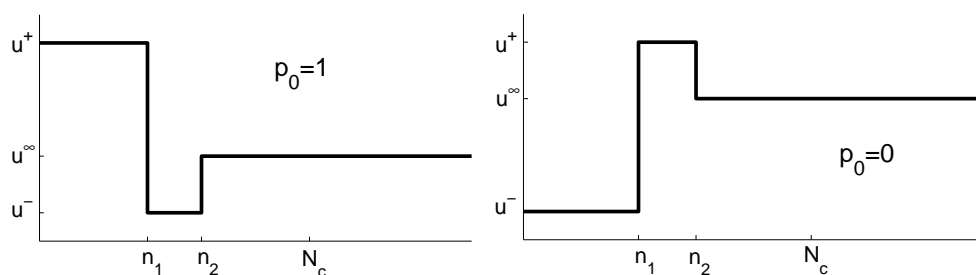
$$F_{FOPDT}(s) = \frac{K}{\tau s + 1} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = \tau + D, \quad \sigma^2 = \tau^2 \quad (7.1)$$

nebo přenosem druhého řádu s dopravním zpožděním

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

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  $n_1, n_2 \in \{0, \dots, N_C\}$  a  $u^\infty \in \langle u^-, u^+ \rangle$ , kde  $N_C \in \{0, 1, \dots\}$  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  $u^\infty$ ) spočívá v minimalizaci kritéria

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

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 difference řídicího signálu na intervalu  $i \in \{0, N_C\}$  a  $\lambda$  je koeficient penalizace změn akční veličiny. Pro nalezení optima úlohy (7.3) je použita kombinace metody nejmenších čtverců a hrubé síly. Hodnota  $u^\infty$  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 první 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  $\lambda$ . 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  $\mu, \sigma^2$ . Regulator potom může být efektivně laděn „ručně“ pouze seřizováním charakteristických čísel procesu  $\kappa, \mu, \sigma^2$ .

## 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

<b>sp</b>	Požadovaná hodnota (setpoint)	double
<b>p<sub>v</sub></b>	Řízená veličina	double
<b>t<sub>v</sub></b>	Velichina pro vysledování (použitý řídicí signál)	double
<b>h<sub>v</sub></b>	Hodnota výstupu v manuálním režimu	double
<b>MAN</b>	Manuální nebo automatický režim	bool
	off ... automatický režim	
	on .... manuální režim	

## Výstupy

<b>m<sub>v</sub></b>	Akční zásah regulátoru (manipulated variable)	double
<b>dm<sub>v</sub></b>	Rychlostní výstup regulátoru (difference)	double
<b>de</b>	Regulační odchylka	double
<b>SAT</b>	Saturace	bool
	off ... lineární zákon řízení	
	on .... výstup regulátoru je saturován	
<b>p<sub>ve</sub></b>	Predikovaná hodnota regulované veličiny na základě zadaného modelu	double
<b>iE</b>	Kód chyby	long
	0 ..... bez chyby	
	1 ..... nesprávný FOPDT model	
	2 ..... nesprávný SOPDT model	
	3 ..... chyba v zadání přechodové charakteristiky	

## Parametry

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

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



## 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:

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

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:

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

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 přepočítání 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		bool
	off ... synchronizace vypnuta		
	on .... synchronizace zapnuta		

## RLY – Relé s hysterezí

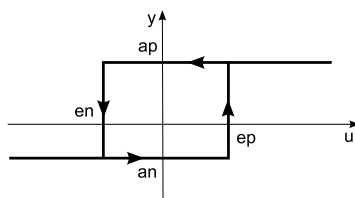
Symbol bloku

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“)	$\odot 1.0$ double
$en$	Hodnota $u < en$ způsobí $y = an$ („Vypnuto“)	$\odot -1.0$ double
$ap$	Výstup ve stavu „Zapnuto“	$\odot 1.0$ double
$an$	Výstup ve stavu „Vypnuto“	$\odot -1.0$ 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  $lolim \leq u$  a  $u \leq hilim$ , 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	double	⊙1.0
$tn$	Časová konstanta rychlosti změn aktivních hodnot mezí v záporném směru	double	⊙1.0
$hilim0$	Horní omezení výstupu (platné pro $HLD = on$ )	double	⊙1.0
$lolim0$	Dolní omezení výstupu (platné pro $HLD = on$ )	double	⊙-1.0

HLD

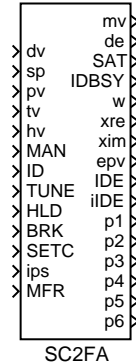
Pevné saturační meze

`off ...` proměnné meze`on ....` pevné meze⊙ `on` `bool`

## 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

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

kde  $\Omega > 0$  je přirozená (netlumená) frekvence,  $\xi$ ,  $0 < \xi < 1$ , 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 stejnou amplitudou **uamp** a frekvencí  $\omega$  postupně probíhající interval  $\langle \mathbf{wb}, \mathbf{wf} \rangle$ . Aktuální frekvence  $\omega$  je přitom kopírována na výstup **w**. Rychlost změny (rozmítání) frekvence je dána parametrem **cp**, který udává relativní zmenšení počáteční periody  $T_b = \frac{2\pi}{\mathbf{wb}}$  budící sinusovky za čas  $T_b$ , tedy

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

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

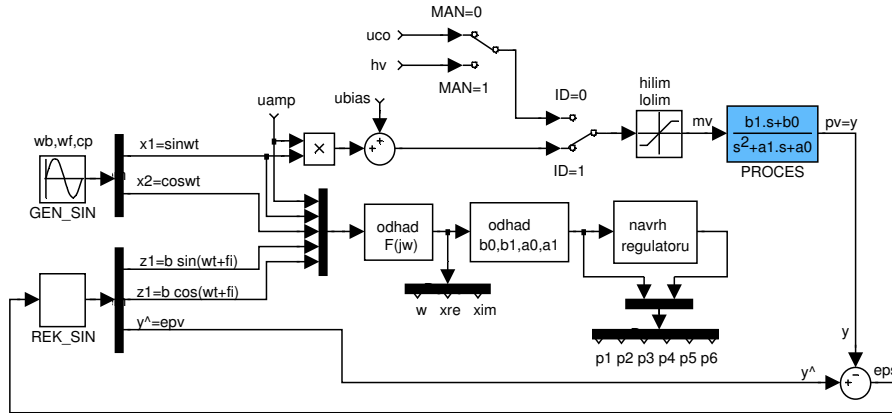
Identifikace systému se spouští náběžnou hranou vstupu **RUN** zároveň s generátorem budícího signálu se startovací frekvencí  $\omega = \mathbf{wb}$ . Po uplynutí **stime** se startuje výpočet odhadu aktuálního bodu frekvenční charakteristiky. Jeho reálná a imaginární část se průběžně kopíruje po řadě 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  $-60^\circ$  a  $-120^\circ$  a mohou být změněny na libovolné hodnoty v intervalu  $(-360^\circ, 0^\circ)$ , 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.

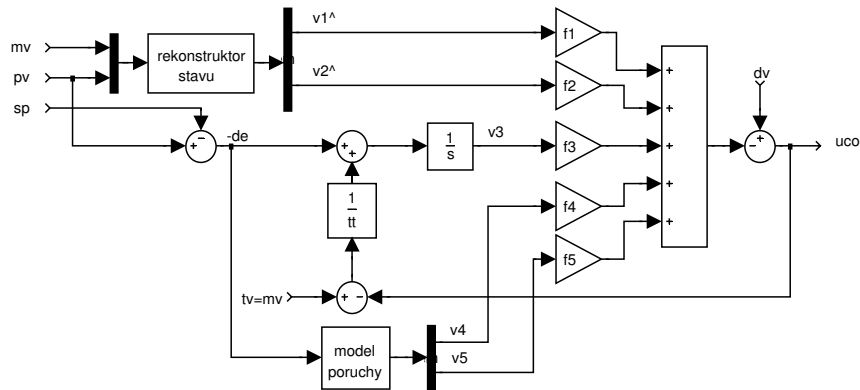
Během vlastní „identifikace a návrhu“ je výstup **IDBSY** nastaven na 1. Po skončení je shozen na 0. Při bezchybném návrhu regulátoru je výstup **IDE** = **off** a výstup **iIDE** signalizuje jednotlivé fáze identifikačního experimentu. Přibližování k prvnímu bodu je **iIDE** =  $-1$ , zastavení v prvním 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  $f_1, f_2, \dots, f_5$  stavové zpětné vazby.



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

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

a

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

Parametry těchto přenosů je možné po skončení identifikace přechíst z výstupů  $p_1, \dots, p_6$ . 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 rozmitání 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 $\tau$ p3 ... parametr $\Omega$ v rad/s p4 ... parametr $\xi$ p5 ... parametr $\Omega$ 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	bool
	off ... lineární zákon řízení	
	on .... výstup regulátoru je saturován	
IDBSY	Příznak probíhající identifikace	bool
	off ... identifikace neprobíhá	
	on .... běží identifikační experiment	
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	bool
	off ... identifikace proběhla v pořádku	
	on .... identifikace skončila s chybou	
iIDE	Kód chyby	long
	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	
p1..p6	Výsledky identifikace a návrhu regulátoru	double

## Parametry

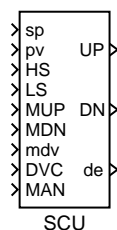
ubias	Stejnosměrná složka budicího harmonického signálu	
uamp	Amplituda budicího harmonického signálu	⊙1.
wb	Počátek frekvenčního intervalu [rad/s]	⊙1.
wf	Konec frekvenčního intervalu [rad/s]	⊙10.
isweep	Způsob rozmítání frekvence	⊙
	1 ..... logaritmické	
	2 ..... lineární (zatím není implementováno)	
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 ustálení [s]	⊙10.
ralpha	Relativní poloha pólů rekonstruktoru	⊙4.
rx1	Koeficient tlumení rekonstruktoru	⊙0.70

ac11	Relativní poloha 1. dvojice pólů uzavřené smyčky	⊙1.0	double
xic11	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
ac12	Relativní poloha 2. dvojice	⊙2.0	double
xic12	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](#)



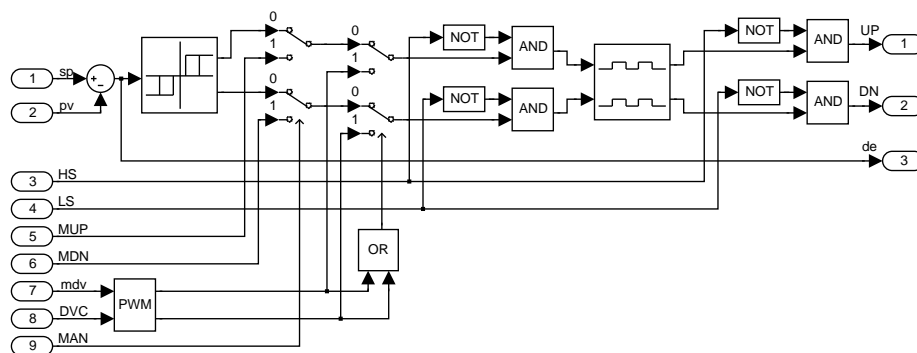
### Popis funkce

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.

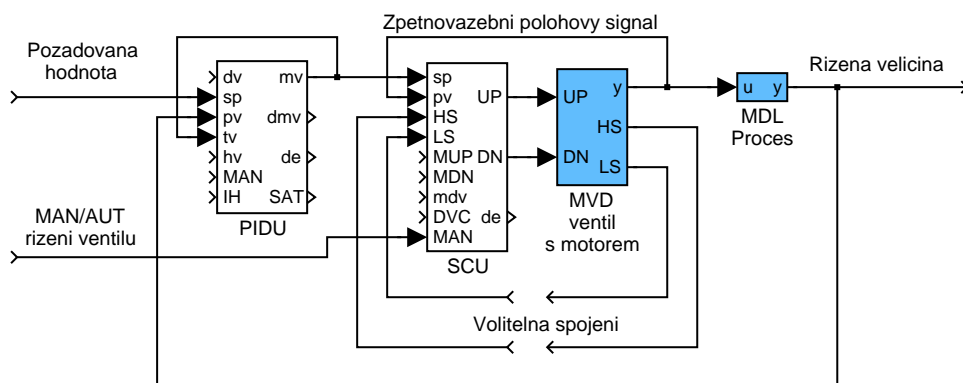
Blok **SCU** nejprve zpracovává regulační odchylku **sp** – **pv** na třístavový výstup symetrickým třístavovým algoritmem s parametry (práhy) **thron** a **throff** (viz blok **TSE**, uvažujte parametry **ep** = **thron**, **epoff** = **throff**, **en** = **-thron** a **enoff** = **-throff**). Parametr **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**, eventuálně 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 dostatečně 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

<code>de</code>	Regulační odchylka	<code>double</code>
-----------------	--------------------	---------------------

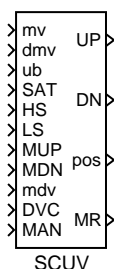
### Parametry

<code>thron</code>	Mez pro zapnutí	$\downarrow 0.0 \odot 0.02$	<code>double</code>
<code>throff</code>	Mez pro vypnutí	$\downarrow 0.0 \odot 0.01$	<code>double</code>
<code>dtime</code>	Minimální trvání výstupního pulzu [s]	$\downarrow 0.0 \odot 0.1$	<code>double</code>
<code>btime</code>	Minimální prodleva mezi pulzy [s]	$\downarrow 0.0 \odot 0.1$	<code>double</code>
<code>RACT</code>	Převrácené působení výstupu regulátoru		<code>bool</code>
	<code>off</code> ... vyšší mv $\rightarrow$ vyšší pv		
	<code>on</code> .... vyšší mv $\rightarrow$ nižší pv		
<code>trun</code>	Časová konstanta motoru	$\downarrow 0.0 \odot 10.0$	<code>double</code>

## 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).

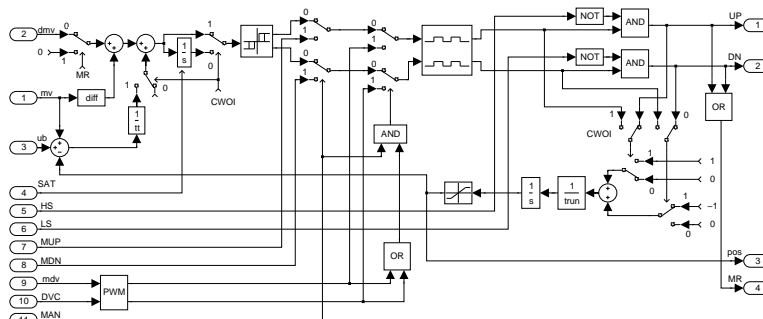
Jestliže je nadřazený regulátor typu P nebo PD (*CWOI* = *on*), potom může být regulační odchylka nadřazeného regulátoru manuálně odstraněna vhodným nastavením vstupu *ub*. V tomto případě je řídicí algoritmus bloku SCUV lehce modifikován. Je užita rekonstruovaná hodnota polohy servoventilu *pos* a parametry *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**, eventuálně lze pomocí vstupu **mdv** nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou (**off**→**on**) na vstupu **DVC**.

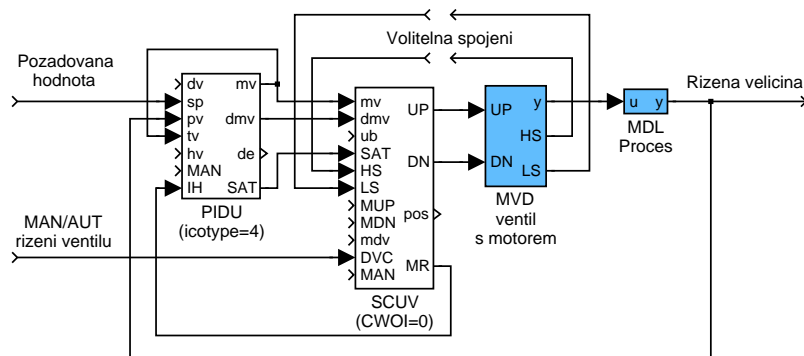


Celková funkce bloku SCUV je zřejmá z následujícího diagramu:

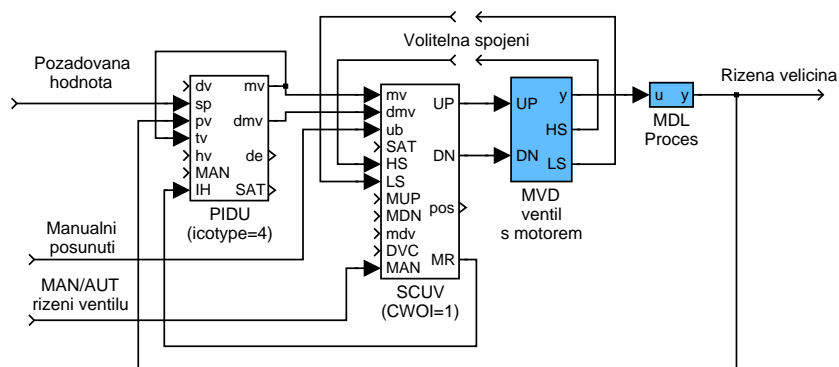


Úplné třístavové krokové regulátory bez polohové zpětné vazby jsou zobrazeny na následujících obrázcích:

**Primární regulátor s integrací: I, PI, PID**



**Primární regulátor bez integrace: P, PD**



## Vstupy

mv	Akční zásah regulátoru (manipulated variable)	double
dmv	Rychlostní výstup regulátoru (difference)	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 <code>off</code> → <code>on</code>	bool
MAN	Manuální nebo automatický režim <code>off</code> ... Automatický režim <code>on</code> .... 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 <code>off</code> ... motor neběží ( <code>UP = off</code> a <code>DN = off</code> ) <code>on</code> .... motor se má pohybovat ( <code>UP = on</code> nebo <code>DN = on</code> )	bool

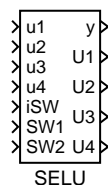
### Parametry

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 <code>off</code> ... vyšší mv → vyšší pv <code>on</code> .... vyšší mv → nižší pv		bool
trun	Časová konstanta motoru (určuje dobu, za kterou se motor posune o hodnotu jedna)	↓0.0 ⊕10.0	double
CW0I	Regulátor bez integrační složky <code>off</code> ... primární regulátor s integrátorem (I, PI, PID) <code>on</code> .... 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

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

### Výstupy

<i>y</i>	Analogový výstupní signál	double
<b>U1</b> .. <b>U4</b>	Binární signály pro selektorovou regulaci	bool

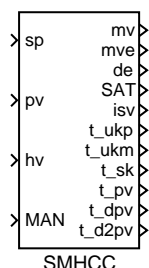
## Parametr

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

## SMHCC – Regulátor pro procesy s topením a chlazením

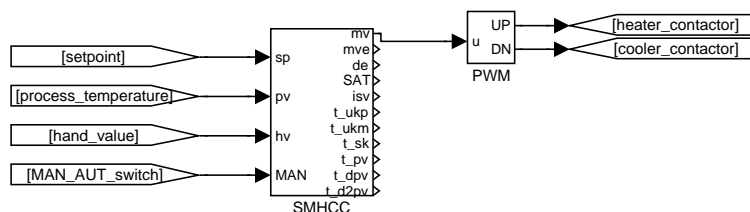
Symbol bloku

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 plastikařský 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  $T_C = i_{pwm} T_S$  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 ( $i_{pwm}$  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 plastikařském průmyslu jsou doporučeny následující hodnoty:

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

Zákon řízení bloku regulátoru **SMHCC** v automatickém režimu (**MAN=off**) je založen na diskretní 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 diskretní proměnná klouzavého režimu

$$s_k \triangleq \ddot{e}_k + 2\xi\Omega\dot{e}_k + \Omega^2 e_k$$

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  $\xi$  a  $\Omega$  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

$$s \triangleq \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e$$

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

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

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

Řídící veličina **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 ≤ hilim\_p** a **u0\_m ≤ 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

$$if \quad (sk < 0.0) \quad then \quad mv = t_{ukp} \quad else \quad mv = -t_{ukm} .$$

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

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

## Výstupy

<code>mv</code>	řídící veličina (manipulated variable)	double
<code>de</code>	regulační odchylka (deviation error) $de = sp - pv$	double
<code>SAT</code>	příznak saturace	bool
	0 ..... regulátor pracuje v lineární oblasti	
	1 ..... výstup regulátoru je satureován, $mv \geq hilim\_p$ nebo $mv \leq -hilim\_m$	
<code>isv</code>	počet kladných nebo záporných kroků přepínací proměnné <code>sk</code>	long
<code>t_ukp</code>	aktuální hodnota amplitudy topení	double
<code>t_ukm</code>	aktuální hodnota amplitudy chlazení	double
<code>t_sk</code>	přepínací proměnná <code>sk</code>	double
<code>t_ek</code>	filtrovaná regulační odchylka $-de$	double
<code>t_dek</code>	filtrovaná první derivace regulační odchylky $t_{ek}$	double
<code>t_2dek</code>	filtrovaná druhá derivace regulační odchylky $t_{ek}$	double

## Parametry

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

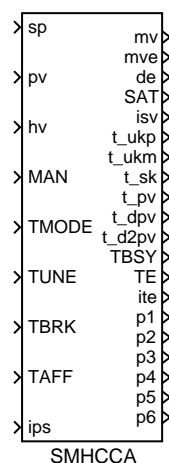
<b>tauf</b>	Časová konstanta filtru ekvivalentní akční veličiny	⊙400.0 <b>double</b>
-------------	---	----------------------



## 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	bool
	off ... automatický režim	
	on .... manuální režim	
TMODE	Režim ladění	bool
TUNE	Zahájení ladicího experimentu	bool
TBRK	Ukončení ladicího experimentu	bool
TAFF	Přijetí výsledků ladicího experimentu	bool
	off ... parametry jsou pouze vypočítány	
	on .... parametry jsou dosazeny do řídicího algoritmu	
ips	Význam výstupních signálů	long
	0 ..... parametry regulátoru	
	1 ..... pomocné parametry	

## 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

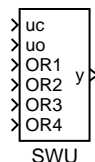
mv	Akční zásah regulátoru (manipulated variable)	double
mve	Ekvivalentní akční veličina	double
de	Regulační odchylka	double
SAT	Saturace	bool
	off ... lineární zákon řízení	
	on ... výstup regulátoru je saturován	
isv	Počet kroků přepínací proměnné	long
t_ukp	Aktuální amplituda topení	double
t_ukm	Aktuální 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	

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

## 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

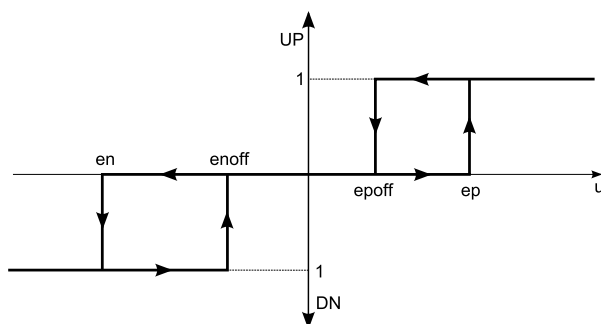
Symbol bloku

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



## Kapitola 8

# LOGIC – Logické řízení

### Obsah

---

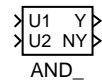
AND_ – Logický součin dvou signálů . . . . .	232
ANDQUAD, ANDOCT, ANDHEXD – Logický součin osmi signálů . . . . .	233
ATMT – Automat pro sekvenční řízení . . . . .	234
BDOCT, BDHEXD – Bitové demultiplexery . . . . .	237
BITOP – Bitová operace dvou celočíselných signálů . . . . .	238
BMOCT, BMHEXD – Bitový multiplexer . . . . .	239
COUNT – Řízený čítač . . . . .	240
EATMT – Extended finite-state automaton . . . . .	241
EDGE_ – Detekce hrany logického signálu . . . . .	244
INTSM – Bitový posun a maska nad celým číslem . . . . .	245
ISSW – Jednoduchý přepínač celočíselných signálů . . . . .	246
ITOI – Transformace celých a binárních čísel . . . . .	247
NOT_ – Logická negace . . . . .	249
OR_ – Logický součet dvou signálů . . . . .	250
ORQUAD, OROCT, ORHEXD – Logický součet více signálů . . . . .	251
RS – Klopný obvod . . . . .	252
SR – Klopný obvod . . . . .	253
TIMER_ – Vícefunkční časovač . . . . .	254

---

## 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 [ANDDOCT](#).

Vstupy

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

Výstupy

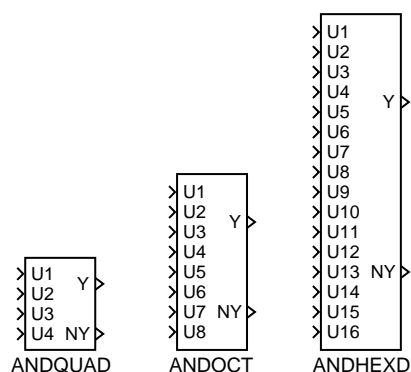
Y	Výstup bloku, logický součin ( $U1 \wedge U2$ )	bool
NY	Negace výstupního signálu Y ( $NY = \neg Y$ )	bool



## ANDQUAD, ANDOCT, ANDHEXD – Logický součin osmi signálů

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky ANDQUAD, ANDOCT a ANDHEXD vyhodnocují logický součin až 16 vstupních signálů U1 až U16. Signály, jejichž seznam je uveden v parametru n1 se před provedením logického součinu negují.

Pokud je tedy parametr n1 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  $n1=1,3..5$ , pak  $Y = \neg U1 \wedge U2 \wedge \neg U3 \wedge \neg U4 \wedge \neg U5 \wedge U6 \wedge \dots \wedge U16$ .

Pokud pracujete s méně než 8 signály, je potřeba ošetřit nepřípojené vstupy bloku pomocí parametru n1. Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku [AND\\_](#).

### Vstupy

U1..U16	Logické vstupy bloku	bool
---------	----------------------	------

### Výstupy

Y	Výsledek logické operace	bool
NY	Negace výstupního signálu Y	bool

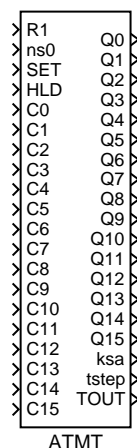
### Parametr

n1	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  $i$ ,  $i = 0, 1, \dots, 15$ , je indikován pomocí binárních výstupů  $Q_0, Q_1, \dots, Q_{15}$ . Pokud je automat ve stavu  $i$ , je nastaven příslušný výstup  $Q_i = \text{on}$ . Aktuální stav automatu je též indikován celočíselným výstupem  $\text{ksa} \in \{0, 1, \dots, 15\}$ .

Podmínky přechodů  $C_k$ ,  $k = 0, 1, \dots, 15$  jsou aktivovány pomocí binárních vstupů bloku  $C_0, C_1, \dots, C_{15}$ . Pokud je  $C_k = \text{on}$ , je podmínka  $C_k$  splněna, naopak pro  $C_k = \text{off}$  splněna není.

Funkce automatu se zadává pomocí tabulky stavů a přechodů:

$S1$	$C1$	$NS1$
$S2$	$C2$	$NS2$
		$\dots$
$S_n$	$C_n$	$NS_n$

Každý řádek této tabulky vyjadřuje jedno pravidlo přechodu. Např. první 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 setrvává 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 **moresteps** 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

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

## Výstupy

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

## Parametry

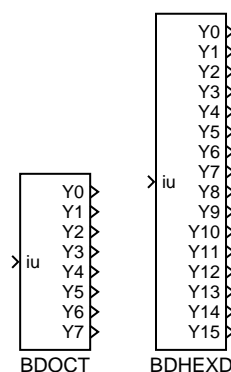
<b>moresteps</b>	Povolit více přechodů mezi stavy automatu v jednom cyklu off ... zakázáno on .... povoleno	<b>bool</b>
<b>ntr</b>	Počet řádků tabulky přechodů mezi stavy	↓0 ↑64 ⊙4 <b>long</b>

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

## 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

$Y0 \dots Y15$	Jednotlivé bity vstupního signálu	bool
----------------	-----------------------------------	------

### Parametr

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

## BITOP – Bitová operace dvou celočíselných signálů

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **BITOP** provádí operaci  $i1 \circ i2$  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

<b>i1</b>	První celočíselný vstup bloku	long
<b>i2</b>	Druhý celočíselný vstup bloku	long

### Výstup

<b>n</b>	Výsledek bitové operace určené parametrem <b>iop</b>	long
----------	--	------

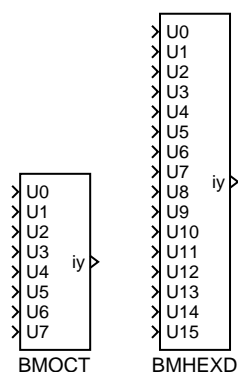
### Parametr

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

## 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

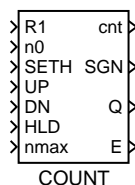
iy	Výsledný výstupní signál	long
----	--------------------------	------

### Parametr

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

## 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  $\text{cnt} \geq \text{nmax}$  způsobí nastavení výstupu **Q** na hodnotu **on**.

## Vstupy

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

## Výstupy

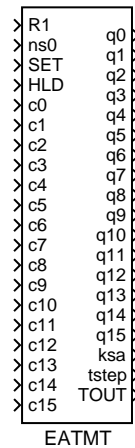
<b>cnt</b>	Celkový počet načtených pulzů	long
<b>SGN</b>	Znaménko výstupu <b>cnt</b>	bool
	off ... menší nebo rovno nule	
	on .... větší než nula	
<b>Q</b>	Indikátor dosažení cílové hodnoty	bool
	off ... pro $\text{cnt} < \text{nmax}$	
	on .... pro $\text{cnt} \geq \text{nmax}$	
<b>E</b>	Indikátor současného 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 automaton with at most 256 states and 256 transition rules, thus it extends the possibilities of the [ATMT](#) block.

The current state of the automaton  $i$ ,  $i = 0, 1, \dots, 255$  is indicated by individual bits of the integer outputs  $q0, q1, \dots, q15$ . Only a single bit with index  $i \text{ MOD } 16$  of the  $q(i \text{ DIV } 16)$  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  $ksa \in \{0, 1, \dots, 255\}$  output.

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

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

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

$S1$	$C1$	$FS1$
$S2$	$C2$	$FS2$
$\dots$		
$Sn$	$Cn$	$FSn$

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 **ci** 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 **moresteps** 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

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

### Výstupy

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

## Parametry

<b>moresteps</b>	Allow multiple transitions in one cycle of the automat <b>off</b> ... Disabled <b>on</b> .... Enabled	<b>bool</b>
<b>ntr</b>	Number of state transition table rows	$\downarrow 0 \uparrow 1024 \odot 4$ <b>long</b>
<b>sfcname</b>	Jméno souboru, kam si konfigurátor bloku ukládá data (pokud se nevyplní, zvolí se automaticky podle jména bloku)	<b>string</b>
<b>STT</b>	State transition table (matrix) $\odot [0 \ 0 \ 1; \ 1 \ 1 \ 2; \ 2 \ 2 \ 3; \ 3 \ 3 \ 0]$	<b>short</b>
<b>touts</b>	Vector of timeouts T00, T01, ..., T0255 for the states $S_0, S_1, \dots, S_{255}$ $\odot [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \ 16]$	<b>double</b>

## 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

<b>U</b>	Logický vstupní signál	<b>bool</b>
----------	------------------------	-------------

### Výstup

<b>Y</b>	Indikace výskytu zvolené hrany	<b>bool</b>
----------	--------------------------------	-------------

### Parametr

<b>iedge</b>	Typ detekovaných hran	⊙1 <b>long</b>
1	..... náběžná hrana	
2	..... sestupná hrana	
3	..... obě hrany	

## 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

<b>i</b>	Celočíselný signál pro zpracování	long
----------	-----------------------------------	------

### Parametry

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

### Výstup

<b>n</b>	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

<i>i1</i>	První celočíselný vstup bloku	long
<i>i2</i>	Druhý celočíselný vstup bloku	long
<i>SW</i>	Přepínací signál	bool
	<b>off</b> ... je zvolen vstupní signál <i>i1</i>	
	<b>on</b> .... je zvolen vstupní signál <i>i2</i>	

### Výstup

<i>n</i>	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  $(U_3 U_2 U_1 U_0)_2$  z množiny  $\{0, 1, 2, \dots, 15\}$  výstupní číslo  $nk$  a jeho binární reprezentaci  $(Y_3 Y_2 Y_1 Y_0)_2$  z téže množiny. Příslušné zobrazení je popsáno tabulkou

$k$	0	1	2	...	15
$nk$	$n_0$	$n_1$	$n_2$	...	$n_{15}$

kde  $n_0, \dots, n_{15}$  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  $(U_3 U_2 U_1 U_0)_2$ .

### Vstupy

$k$	Celočíselný vstupní signál	long
$U_0$	Binární číslice vstupu s vahou 1	bool
$U_1$	Binární číslice vstupu s vahou 2	bool
$U_2$	Binární číslice vstupu s vahou 4	bool
$U_3$	Binární číslice vstupu s vahou 8	bool

### Výstupy

$nk$	Celočíselný výstupní signál	long
$Y_0$	Binární číslice výstupu s vahou 1	bool
$Y_1$	Binární číslice výstupu s vahou 2	bool
$Y_2$	Binární číslice výstupu s vahou 4	bool
$Y_3$	Binární číslice výstupu s vahou 8	bool

### Parametry

BINF	Transformace hodnoty z binárních vstupů off ... zakázáno (transformace vstupu $k$ ) on .... povoleno (vstupem je hodnota $(U_3 U_2 U_1 U_0)_2$ )	⊙ on bool
------	--	-----------

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



## 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 ( $Y = \neg U$ )	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

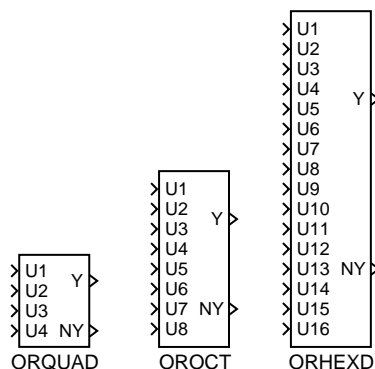
### Výstupy

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

## ORQUAD, OROCT, ORHEXD – Logický součet více signálů

Symboly bloků

Licence: [STANDARD](#)



### Popis funkce

Bloky **ORQUAD**, **OROCT**, **ORHEXD** provádí logický součet až šestnácti vstupních signálů **U1** až **U16**. Signály, jejichž seznam je uveden v parametru **n1** se před provedením logického součinu negují.

Pokud je tedy parametr **n1** prázdný, tak se provádí logický součet  $Y = U1 \vee U2 \vee U3 \vee U4 \vee U5 \vee \dots \vee U16$ . Pokud bude například **n1=1,3..5**, pak  $Y = \neg U1 \vee U2 \vee \neg U3 \vee \neg U4 \vee \neg U5 \vee U6 \vee \dots \vee U16$ .

Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku [OR\\_](#).

### Vstupy

<b>U1..U16</b>	Logické vstupy bloku	<b>bool</b>
----------------	----------------------	-------------

### Parametr

<b>n1</b>	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.	<b>long</b>
-----------	---	-------------

### Výstupy

<b>Y</b>	Výsledek logické operace	<b>bool</b>
<b>NY</b>	Negace výstupního signálu Y	<b>bool</b>

RS – Klopný obvod

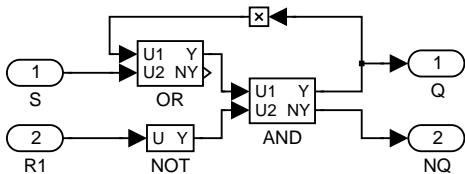
Symbol bloku

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

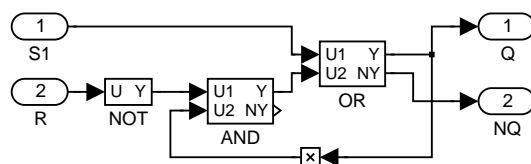
Licence: [STANDARD](#)



### 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 <b>on</b> , má přednost před vstupem R	bool
R	Shození klopného obvodu, nastaví výstup Q na <b>off</b>	bool

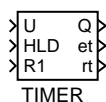
### Výstupy

Q	Stav klopného obvodu	bool
NQ	Negace výstupního signálu Q	bool

## TIMER\_ – Vícefunkční časovač

Symbol bloku

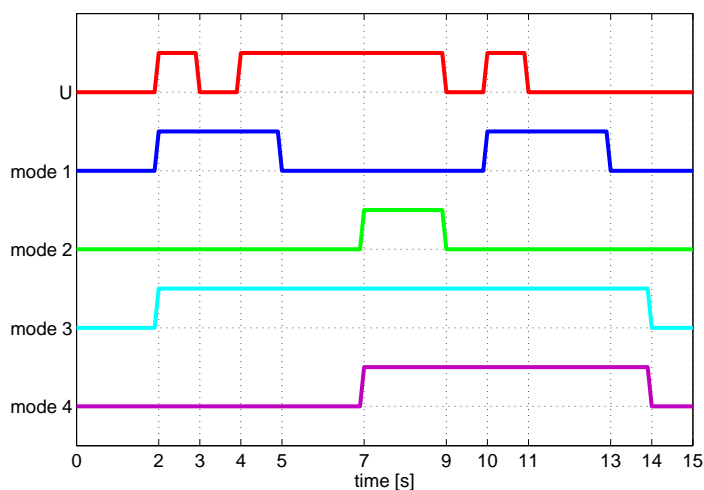
Licence: [STANDARD](#)



### 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

<code>mode</code>	Režim činnosti časovače	⊙1	<code>long</code>
	1 ..... generovaný pulz – na výstupu je pulz délky <code>pt</code> sekund, který začíná náběžnou hranou na vstupu <code>U</code> ; další náběžné hrany na vstupu <code>U</code> během trvání pulzu jsou ignorovány		
	2 ..... zpožděné zapnutí – signál ze vstupu <code>U</code> je kopírován na výstup <code>Q</code> tak, že začátek impulzu na výstupu je opožděn o <code>pt</code> sekund proti začátku pulzu na vstupu; pulzy kratší než <code>pt</code> sekund se na výstupu neobjeví		
	3 ..... zpožděné vypnutí – signál ze vstupu <code>U</code> je kopírován na výstup <code>Q</code> tak, že konec impulzu na výstupu je opožděn o <code>pt</code> sekund proti konci pulzu na vstupu; pokud je mezera mezi vstupními pulzy kratší než <code>pt</code> sekund, výstup je trvale aktivní		
	4 ..... zpožděná změna – výstupní signál <code>Q</code> se přepne na hodnotu vstupu <code>U</code> až tehdy, když je vstup po dobu <code>pt</code> sekund neměnný		
<code>pt</code>	Doba časování [s]	⊙1.0	<code>double</code>





## Kapitola 9

# TIME – Bloky pro práci s časem

### Obsah

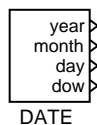
---

DATE_ – Aktuální datum . . . . .	258
DATETIME – Čtení, nastavování a konverze času . . . . .	259
TIME – Aktuální čas . . . . .	262
WSCH – Týdenní časovač . . . . .	263

---

**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**

<b>year</b>	Rok	long
<b>month</b>	Měsíc	long
<b>day</b>	Den	long
<b>dow</b>	Den v týdnu, první den je neděle (1)	long

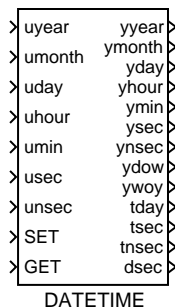
**Parametr**

<b>tz</b>	Časové pásmo	⊙1	long
	1 ..... lokální čas		
	2 ..... UTC		

## 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

<b>uyear</b>	Vstup pro nastavení roku	long
<b>umonth</b>	Vstup pro nastavení měsíce	long
<b>uday</b>	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	long
	1 ..... čas OS		
	2 ..... vstupy bloku		
	3 ..... vstup unsec		
	4 ..... vstup usec		
	5 ..... vstup unsec relativně		
igetmode	Zdroj ze kterého přečíst čas pro nastavení či konverzi	⊙6	long
	1 ..... čas OS		
	2 ..... vstupy bloku		
	3 ..... vstup unsec		
	4 ..... vstup usec		
	5 ..... vstup uday		
	6 ..... čas systému REX		
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		bool
	off ... týden začíná pondělím		
	on .... týden začíná nedělí		

tz

Časové pásmo

1 ..... lokální čas

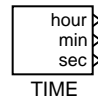
2 ..... UTC

⊙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

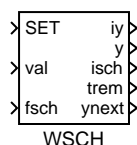
### Parametr

tz	Časové pásmo	⊙1	long
	1 ..... lokální čas		
	2 ..... UTC		

## 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

<code>SET</code>	Nastavení výstupů <code>y</code> a <code>iy</code> pomocí náběžné hrany	<code>bool</code>
<code>val</code>	Hodnota pro dočasné nastavení výstupů <code>y</code> a <code>iy</code>	<code>double</code>
<code>fsch</code>	Vynucený denní program	<code>long</code>
	0 ..... provoz dle týdenního programu	
	1 ..... pondělí	
	2 ..... úterý	
	..... ..	
	7 ..... neděle	
	8 a více další denní programy dle tabulky <code>wst</code>	

## Výstupy

<code>iy</code>	Celočíselná výstupní hodnota	<code>long</code>
<code>y</code>	Výstupní hodnota	<code>double</code>
<code>isch</code>	Identifikace denního programu	<code>long</code>
<code>trem</code>	Čas zbývajících v aktuálním intervalu [s]	<code>double</code>
<code>ynext</code>	Výstupní hodnota v dalším intervalu	<code>double</code>

## Parametry

<code>tz</code>	Časové pásmo	$\odot 1$	<code>long</code>
	1 ..... lokální čas		
	2 ..... UTC		
<code>nmax</code>	Velikost alokovaných polí	$\downarrow 10 \uparrow 1000000 \odot 100$	<code>long</code>
<code>wst</code>	Tabulka týdenního programu (seznam trojic <i>den-hodina-hodnota</i> )		<code>double</code>
	$\odot [1 \ 0.01 \ 18.0; 2 \ 6.0 \ 22.0; 2 \ 18.0 \ 18.0; 3 \ 6.0 \ 22.0; 3 \ 18.0 \ 18.0; 4 \ 6.0 \ 22.0; 4$		
<code>specdays</code>	Seznam speciálních dnů (seznam dvojic <i>datum-denní program</i> )		<code>long</code>
	$\odot [20150406 \ 1; 20151224 \ 1; 20151225 \ 1; 20151226 \ 1; 20160101 \ 1; 20160328 \ 1; 20170$		



## Kapitola 10

# ARC – Archivace dat

### Obsah

<b>10.1</b>	<b>Funkce archivačního subsystému</b>	<b>266</b>
<b>10.2</b>	<b>Generování alarmů u a událostí</b>	<b>267</b>
	ALB, ALBI – Alarmy pro logickou hodnotu	267
	ALN, ALNI – Alarmy pro číselnou hodnotu	269
<b>10.3</b>	<b>Záznam trendů</b>	<b>271</b>
	ACD – Archivní komprese s použitím delta kritéria	271
	TRND – Záznam trendů v reálném čase	273
	TRNDV – Záznam trendů v reálném čase (vektorová forma)	276
	TRNDLF – * Záznam trendů v reálném čase (lock-free)	278
	TRNDVLF – * Záznam trendů v reálném čase (pro vektory, lock-free)	280
<b>10.4</b>	<b>Správa archivů</b>	<b>281</b>
	AFLUSH – Vynucené zapsání archivu	281

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 řídicího 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.

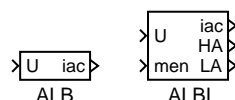
Daná hardwarová platforma nemusí podporovat všechny druhy archivů. Příznaky podporovaných druhů archivů jsou součástí verze řídicího systému cílového zařízení a lze je zjistit v programu **RexView** po kliknutí na jméno (IP adresu) cílového zařízení ve stromu exekutivy. Nachází se na kartě **Target** v levé spodní části.

## 10.2 Generování alarmů u a událostí

### ALB, ALBI – Alarmy pro logickou hodnotu

Symbole 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 lv1. Pokud je  $1 \leq lv1 \leq 127$ , jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah  $128 \leq lv1 \leq 255$  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

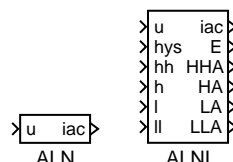
## Parametry

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 <a href="#">ARC</a> . 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). $\odot 1$	word
lvl	Úroveň (závažnost) alarmu, určující, zda jde o skutečný alarm či jen o událost. $\downarrow 1 \odot 1$	byte
Desc	Řetězec blíže specifikující daný alarm či událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REX. $\odot$ 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 *lv1*. Pokud je  $1 \leq lv1 \leq 127$ , jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah  $128 \leq lv1 \leq 255$  je určen pro události, u nichž se zapisuje pouze okamžik, kdy daná událost nastala.

### Vstupy

<i>u</i>	Analogový vstupní signál, podle jehož hodnoty se generují alarmy	double
<i>hys</i>	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
<i>hh</i>	Mez pro druhý horní alarm. Musí být větší než mez <i>h</i> .	double
<i>h</i>	Mez pro horní alarm. Musí být větší než mez <i>l</i> .	double
<i>l</i>	Mez pro dolní alarm. Musí být větší než mez <i>ll</i> .	double
<i>ll</i>	Mez pro druhý dolní alarm	double
<i>iACK</i>	Potvrzení alarmů	byte
	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.	

V případě, že stačí daným blokem generovat jen jednoúrovňové alarmy, stačí nastavit *lv12=0*. Alternativně je možné druhou horní mez *hh* nastavit na větší a druhou dolní

mez 11 na menší hodnotu, než může vstup u dosáhnout.

## Výstupy

iac	Kód aktuálního stavu alarmového bloku	long
	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.	
E	Příznak chyby uspořádání alarmových mezí	bool
	off ... bez chyby                      on .... nastala chyba	
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	byte
	1 ..... Bool            4 ..... Long            7 ..... Float		
	2 ..... Byte            5 ..... Word            8 ..... Double		
	3 ..... Short          6 ..... DWord          9 ..... Time		
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 <a href="#">ARC</a> . 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
lv11	Úroveň (závažnost) prvních horních a dolních alarmů (HA a LA), určující, zda jde o skutečný alarm či jen o událost	↓1 ⊙1	byte
lv12	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.		string
	⊙Alarm Description		

## 10.3 Záznam trendů

### ACD – Archivní komprese s použitím delta kritéria

Symbol bloku

Licence: [STANDARD](#)

#### Popis funkce

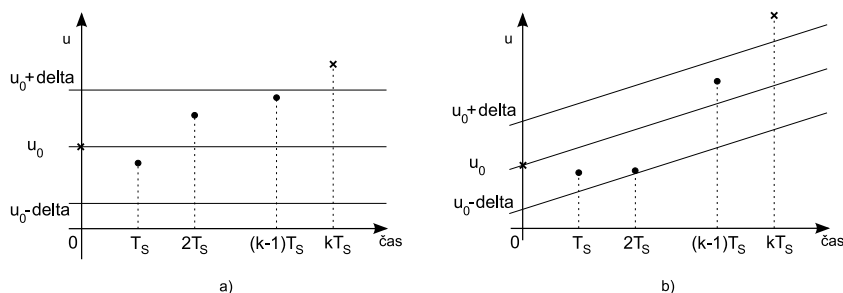
Blok ACD (Archive Compression using Delta criterion) je určen pro ukládání komprimovaných analogových signálů do archivu pomocí archivních událostí.

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  $\langle t_{\min}, t_{\max} \rangle$  sekund (doby jsou zaokrouhleny na nejbližší násobek periody vzorkování). Pokud se hodnota signálu „hodně“ mění, ukládá se signál jednou za čas  $t_{\min}$ , pokud se hodnota signálu mění „málo“ nebo je konstantní, ukládá se signál jednou za čas  $t_{\max}$ . Po spuštění bloku se vždy uloží první hodnota vstupu  $u$ , označme ji  $u_0$ . Přesná pravidla ukládání dalších vzorků jsou určena vstupem  $\delta$  a parametrem  $TR$ .

Je-li  $TR=off$ , testuje se podmínka  $|u - u_0| > \delta$ . Pokud je splněna a od minulého uložení uplynul alespoň čas  $t_{\min}$  uloží se tato hodnota  $u$  do archivu a nastaví se  $u_0 = u$ . Je-li podmínka splněna dříve než za čas  $t_{\min}$  od posledního uložení nastaví se chybový výstup  $E$  na 1 a počká se s uložením na první vzorek po uplynutí času  $t_{\min}$ , 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  $\times$ .



#### Vstupy

$u$	Komprimovaně ukládaný signál	double
$\delta$	Práh pro ukládání signálu do archivu	$\downarrow 0.0 \uparrow 10000000000.0$ 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	bool
	off ... bez chyby                      on .... nastala chyba	

## Parametry

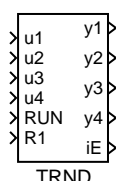
acls	Třída alarmu, určující typ proměnné, která bude do archivu ukládána	byte
	⊙8	
	1 ..... Bool      4 ..... Long      7 ..... Float	
	2 ..... Byte      5 ..... Word      8 ..... Double	
	3 ..... Short      6 ..... DWord      9 ..... Time	
arc	Seznam archivů, kam budou události ukládány. Zadává se ve tvaru např. 1,3..5,8. Událost bude uložena do všech uvedených archivů (detaily o číslování archivů viz blok <a href="#">ARC</a> . 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).	word
	⊙1	
tmin	Nejkratší čas (v sekundách) mezi dvěma uloženími hodnoty vstupu u do archivu	double
	↓0.001 ↑1000000.0 ⊙1.0	
tmax	Nejdelší čas (v sekundách) mezi dvěma uloženími hodnoty vstupu u do archivu	double
	↓1.0 ↑1000000.0 ⊙1000.0	
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.	bool
	⊙on	
	off ... vyhodnocuje se odchylka od poslední uložené hodnoty	
	on .... vyhodnocuje se odchylka od trendu posledně uložené hodnoty	
Desc	Řetězec blíže specifikující danou událost. Tento řetězec je zobrazován v diagnostických nástrojích řídicího systému REX.	string
	⊙Value Description	



## 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 1 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 **p1** až **p4**. 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  $s \cdot n \cdot 1$ , kde  $s$  je velikost proměnné daného typu v bytech. Přednastavený typ **Double** zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů  $n = 4$ , délka každého trendu  $1 = 1000$ , 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 14.

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

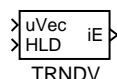
n	Počet signálů (bufferů) v trendu	↓1 ↑4 ⊙4	long
l	Počet vzorků vyhrazený v paměti pro každý buffer trendu	↓0 ↑268435000 ⊙1000	long
btype	Typ všech použitých bufferů trendu	⊙8	long
	1 ..... Bool      4 ..... Long      7 ..... Float		
	2 ..... Byte      5 ..... Word      8 ..... Double		
	3 ..... Short      6 ..... DWord      10 ..... Large		
ptype <sub>i</sub>	Způsob zpracování signálu u <sub>i</sub> , i = 1...4. Zvolený způsob se aplikuje na posledních pfac vzorků a výsledek se uloží do i-tého trendového bufferu	⊙1	long
	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		

<b>pfac</b>	Násobek periody spouštění bloku pro uložení zpracovaných hodnot do trendových bufferů. Pokud je vstup <code>RUN = on</code> , ukládají se zpracovaná data do trendu s periodou $\text{pfac} \cdot T_S$ , kde $T_S$ je perioda spouštění bloku ve vteřinách. ↓1 ↑1000000 ⊙1	long
<b>afac</b>	Archivační faktor je číslem udávajícím po kolika uložených vzorcích do trendu se mají ukládané hodnoty navíc uložit do archivů zadaných příznaky <code>arc</code> . Je-li <code>afac = 0</code> , neukládají se trendy do žádného archivu, jinak se ukládají s periodou $\text{afac} \cdot \text{pfac} \cdot T_S$ , kde $T_S$ je perioda spouštění bloku ve vteřinách. ↓0 ↑1000000	long
<b>arc</b>	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 <a href="#">ARC</a> . Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	word
<b>id</b>	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
<b>Title</b>	Text hlavičky trendu pro zobrazení v diagnostických nástrojích systému REX, např. v programu RexView ⊙Trend Title	string
<b>timesrc</b>	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 <code>CORETIMER</code> – 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 <code>RTC</code> . Ostatní volby jsou určeny pouze pro ladící nebo speciální účely. ⊙1 1 ..... <code>CORETIMER</code> – technologický čas – aktuální tick 2 ..... <code>CORETIMER-PRECISE</code> – technologický čas – při spuštění bloku 3 ..... <code>RTC</code> – reálný čas z operačního systému – aktuální tick 4 ..... <code>RTC-PRECISE</code> – reálný čas z operačního systému – při spuštění bloku 4 ..... <code>PFC</code> – 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 1 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  $s \cdot n \cdot 1$ , kde *s* je velikost proměnné daného typu v bytech. Přednastavený typ **Double** zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů **n** = 4, délka každého trendu **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 [14](#).

Číst, zobrazovat a exportovat průběžně ukládaná data umožňuje diagnostický program **RexView**.

### Vstupy

<b>uVec</b>	Vektorový signál určený k uložení	<b>reference</b>
<b>HLD</b>	Pozastavení ukládání dat do cyklických bufferů, při HLD = on se neukládají žádná data	<b>bool</b>

## Výstup

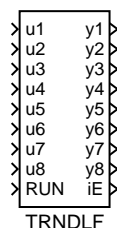
iE	Kód chyby	error
i	..... obecná chyba systému REX	

## 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	long
	1 ..... Bool      4 ..... Long      7 ..... Float		
	2 ..... Byte      5 ..... Word      8 ..... Double		
	3 ..... Short      6 ..... DWord      10 ..... Large		
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 $\text{pfac} \cdot T_S$ , 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 $\text{afac} \cdot \text{pfac} \cdot T_S$ , 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

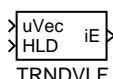
### Výstupy

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

<b>arc</b>	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 <a href="#">ARC</a> . Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitovou maskou – pro uvedený příklad tedy 157, binárně 10011101.	<b>word</b>
------------	---	-------------



## Kapitola 11

# STRING – Bloky pro práci s řetězcí

### Obsah

---

CNS – * Textová konstanta . . . . .	284
CONCAT – * Spojení stringů (podle vzoru) . . . . .	285
FIND – * Nalezení textu . . . . .	286
LEN – * Délka textu . . . . .	287
MID – * Výřez textu . . . . .	288
PJROCT – * Získání číselných hodnot z textu ve formátu JSON .	289
PJSOCT – * Získání textových hodnot z textu ve formátu JSON .	291
REGEXP – Regular expresion parser . . . . .	293
REPLACE – * Náhrada textu . . . . .	294
RTOS – * Konverze čísla na text . . . . .	295
SELSOCT – * Výběr textu z několika vstupů . . . . .	296
STOR – * Koverze textu na číslo . . . . .	297

---

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

<code>scv</code>	Textová hodnota	<code>string</code>
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520 <code>long</code>

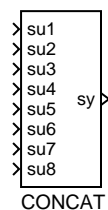
Výstup

<code>sy</code>	Výstupní textová hodnota	<code>string</code>
-----------------	--------------------------	---------------------

## 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

<code>su</code>	Vstupní textová hodnota	<code>string</code>
-----------------	-------------------------	---------------------

### Parametr

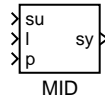
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520	<code>long</code>
-------------------	-------------------------------	-----------	-------------------

### Výstup

<code>len</code>	Délka vstupního textu	<code>long</code>
------------------	-----------------------	-------------------

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

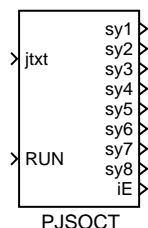
y1	Výstup bloku	double
y2	Výstup bloku	double
y3	Výstup bloku	double
y4	Výstup bloku	double
y5	Výstup bloku	double
y6	Výstup bloku	double

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

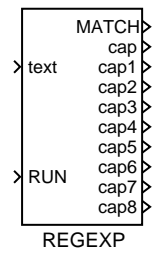
sy1	Výstupní textová hodnota	string
sy2	Výstupní textová hodnota	string
sy3	Výstupní textová hodnota	string
sy4	Výstupní textová hodnota	string
sy5	Výstupní textová hodnota	string
sy6	Výstupní textová hodnota	string
sy7	Výstupní textová hodnota	string

<code>sy8</code>	Výstupní textová hodnota	<code>string</code>
<code>iE</code>	Kód chyby	<code>error</code>

## REGEXP – Regular expression parser

Symbol bloku

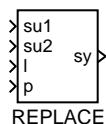
Licence: [ADVANCED](#)



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

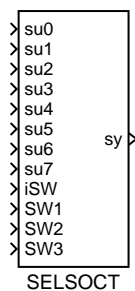
prec	Přesnost (počet cifer)	↓0 ↑20	long
mode	Formát výstupního textu	⊙1	long
	1 ..... nejvhodnější		
	2 ..... normální		
	3 ..... exponenciální		

### 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

<code>su</code>	Vstupní textová hodnota	<code>string</code>
-----------------	-------------------------	---------------------

### Parametr

<code>yerr</code>	Náhradní hodnota pro případ chyby	<code>double</code>
-------------------	-----------------------------------	---------------------

### Výstupy

<code>y</code>	Analogový výstupní signál	<code>double</code>
<code>E</code>	Příznak chyby	<code>bool</code>



## Kapitola 12

# PARAM – Bloky pro manipulaci s parametry

### Obsah

---

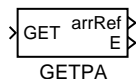
GETPA – Blok pro vzdálené získání vektorového parametru . . . .	300
GETPR, GETPI, GETPB – Bloky pro vzdálené získání parametru . . .	302
GETPS – * Blok pro vzdálené získání parametru typu string . . .	304
PARA – Blok s vektorovým parametrem nastavitelným ze vstupu	305
PARR, PARI, PARB – Bloky s nastavitelným parametrem ze vstupu	306
PARS – * Blok s parametrem typu string nastavitelným ze vstupu	308
SETPA – Blok pro vzdálené nastavování vektorového parametru .	309
SETPR, SETPI, SETPB – Bloky pro vzdálené nastavování parametru	311
SETPS – * Blok pro vzdálené nastavování parametru typu string	313
SGSLP – Nastavování, čtení, ukládání a načítání parametrů . . . .	314
SILO – Uložení vstupního signálu, načtení výstupního signálu . .	318
SILOS – Uložení vstupního řetězce, načtení výstupního řetězce .	319

---

## 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

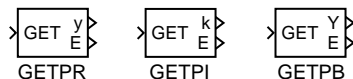
## Parametry

<code>sc</code>	Jméno vzdáleného parametru	<code>string</code>
<code>GETF</code>	Načtení parametru pouze po vyžádání	<code>bool</code>
	<code>off ...</code> režim průběžného čtení parametru	
	<code>on ....</code> režim jednorázového přečtení parametru po náběžné hraně na vstupu <code>GET</code>	
<code>nmax</code>	Maximální velikost vektoru (pole)	⊙256 <code>long</code>

## GETPR, GETPI, GETPB – Bloky pro vzdálené získání parametru

Symbody 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	bool
	off ... bez chyby	
	on .... nastala chyba	

## Parametry

sc	Jméno vzdáleného parametru podle výše uvedených pravidel	string
GETF	Zapnutí manuálního čtení vzdáleného parametru	bool
	off ... režim průběžného čtení parametru	
	on .... režim jednorázového přečtení parametru po náběžné hraně na vstupu GET	

## 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	bool
	off ... režim průběžného čtení parametru	
	on .... režim jednorázového přečtení parametru	
nmax	Rezervovaná paměť pro řetězec	long

### Výstupy

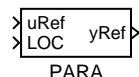
sy	Hodnota parametru	string
E	Příznak chyby	bool
	off ... bez chyby	
	on .... nastala chyba	



## 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 umožňuje 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

<b>uRef</b>	Odkaz na pole (vektor nebo matice)	<b>reference</b>
<b>LOC</b>	Aktivace lokálního režimu	<b>bool</b>
	<b>off</b> ... parametr je ovládán vstupním signálem	
	<b>on</b> .... lokální režim aktivován	

### Výstup

<b>yRef</b>	Odkaz na pole (vektor nebo matice)	<b>reference</b>
-------------	------------------------------------	------------------

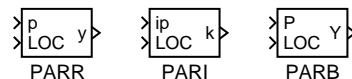
### Parametry

<b>SETS</b>	Nastavení velikosti pole. Použijte tento příznak pro úpravu velikosti pole při nastavování vektorového parametru.	<b>bool</b>
<b>apar</b>	Počáteční hodnota parametru	$\odot$ [0.0 1.0 2.0 3.0 4.0 5.0] <b>double</b>

## PARR, PARI, PARB – Bloky s nastavitelným parametrem ze vstupu

Symbody 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

<b>p</b>	Hodnota parametru (blok PARR)	<b>double</b>
<b>ip</b>	Hodnota parametru (blok PARI)	<b>long</b>
<b>P</b>	Hodnota parametru (blok PARB)	<b>bool</b>
<b>LOC</b>	Aktivace lokálního režimu	<b>bool</b>
	<b>off</b> ... parametr je ovládán vstupním signálem	
	<b>on</b> .... lokální režim aktivován	

### Výstup

<b>y</b>	Analogový výstupní signál bloku PARR	<b>double</b>
<b>k</b>	Celočíselný výstupní signál bloku PARI	<b>long</b>
<b>Y</b>	Logický výstupní signál bloku PARB	<b>bool</b>

### Parametry

<b>par</b>	Počáteční hodnota parametru bloku PARR	©1.0 <b>double</b>
------------	--	--------------------

<code>ipar</code>	Počáteční hodnota parametru bloku PARI	$\odot 1$	<code>long</code>
<code>PAR</code>	Počáteční hodnota parametru bloku PARB	$\odot \text{on}$	<code>bool</code>
<code>SATF</code>	Omezení výstupu pro bloky PARR a PARI <code>off ...</code> signál není omezen <code>on ....</code> saturační meze jsou aktivní		<code>bool</code>
<code>hilim</code>	Horní saturační mez (bloky PARR a PARI)	$\odot 1.0$	<code>double</code>
<code>lolim</code>	Dolní saturační mez (bloky PARR a PARI)	$\odot -1.0$	<code>double</code>

**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

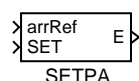
Výstup

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

<b>arrRef</b>	Odkaz na pole (vektor nebo matice)	<b>reference</b>
<b>SET</b>	Vstup pro jednorázový zápis parametru	<b>bool</b>

### Výstup

<b>E</b>	Příznak chyby	<b>bool</b>
----------	---------------	-------------

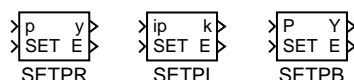
## Parametry

sc	Jméno vzdáleného parametru	string
SETF	Nastavení parametru pouze na vyžádání	bool
	off ... režim průběžného nastavování parametru	
	on .... režim jednorázového nastavení parametru po náběžné hraně na vstupu SET	
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 parametru 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

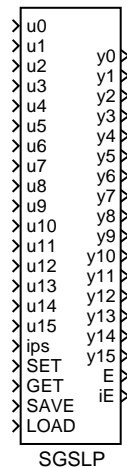
### Výstupy

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 **sci** prázdný (nezadaný), není žádný parametr specifikován, jinak parametry **sci** 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  $i$ ,  $i = 0 \dots 15$  volit různý způsob zadání **sci**, 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 **sci**. V případě, že je **broot** = '.', je výsledek stejný, jako by parametr obsahoval cestu k subsystému, do něž 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 **sci** 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 modifikovanou 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  $sci$ ,  $i = 0, \dots, m$ , kde  $m < 16$  je maximální číslo, pro něž je parametr  $scm$  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. Poznámenejme, 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  $sci$  obsahuje jedinou dvojici <blok>:<param> (viz 1. syntaxe výše). Tato skutečnost umožňuje přecházet mezi oběma syntaxemi parametrů  $sci$ , aniž by musel být soubor upravován.

Při ukládání malého počtu hodnot můžete rovněž využít blok **SIL0**.

## Vstupy

$ui$	$i$ -tý analogový vstupní signál, $i = 0, \dots, 15$	double
<b>ips</b>	Číslo sady parametrů (číslováno od 0)	long
<b>SET</b>	Přečtení vstupů $ui$ a nastavení parametrů sady <b>ips</b> na jejich hodnoty	bool
<b>GET</b>	Přečtení parametrů sady <b>ips</b> a nastavení výstupů $yi$ na jejich hodnoty	bool
<b>SAVE</b>	Uložení parametrů sady <b>ips</b> do souboru na disk cílového zařízení	bool
<b>LOAD</b>	Načtení parametrů sady <b>ips</b> ze souboru na disku cílového zařízení	bool

## Výstupy

$yi$	$i$ -tý analogový výstupní signál, $i = 0, \dots, 15$	double
<b>E</b>	Příznak chyby	bool
	off ... bez chyby	
	on .... nastala chyba, viz výstup <b>iE</b>	

<b>iE</b>	Chybový nebo varovný výstup poslední operace	<b>long</b>
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 <b>ips</b> 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 <b>GET</b> (neexistující parametr?)	
13	..... hodnotu parametru nelze nastavit operací <b>SET</b> (neexistující parametr?)	
14	..... překročení času při získávání/nastavování parametru (timeout)	
15	..... připojenou hodnotu (parametr) není dovoleno zapisovat	
16	..... číslo sady <b>ips</b> je mimo přípustný rozsah	

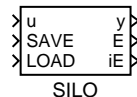
## Parametry

<b>nps</b>	Počet sad parametrů	↓1 ↑16 ⊙1	<b>long</b>
<b>ips0</b>	Číslo sady parametrů, která se načte ze souboru při inicializaci bloku. Je-li <b>ips0</b> < 0 nebo <b>ips0</b> ≥ <b>nps</b> , načte se při inicializaci žádná sada	↓-1 ↑15	<b>long</b>
<b>iprec</b>	Počet platných číslic pro zápis hodnoty typu <b>double</b> do souboru	↓2 ↑15 ⊙12	<b>long</b>
<b>icolw</b>	Šířka sloupce v souboru. Je-li skutečná šířka menší, je doplněna zprava mezerami. Pokud je <b>icolw</b> < <b>iprec</b> , nebudou žádné mezery přidávány.	↓0 ↑22	<b>long</b>
<b>fname</b>	Jméno souboru, do kterého se ukládají parametry příkazem <b>SAVE</b> a ze kterého se načítají příkazem <b>LOAD</b>	⊙ <b>status</b>	<b>string</b>
<b>broot</b>	Cesta k subsystému, přidávaná na začátek specifikace bloků v parametrech <b>sci</b> , viz popis v textu výše	⊙.	<b>string</b>
<b>sci</b>	Řetězce specifikující připojení vstupů <b>u<sub>i</sub></b> a výstupů <b>y<sub>i</sub></b> , <i>i</i> = 0, ..., 15 k požadovaným parametrům, viz popis v textu výše		<b>string</b>

## 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

<b>u</b>	Vstupní signál	<b>double</b>
<b>SAVE</b>	Uložení vstupní hodnoty do souboru	<b>bool</b>
<b>LOAD</b>	Načtení hodnoty výstupu ze souboru	<b>bool</b>

### Parametry

<b>fname</b>	Jméno souboru pro ukládání/načítání parametrů	<b>string</b>
<b>CSF</b>	Příznak pro průběžné ukládání	<b>bool</b>
<b>CLF</b>	Příznak pro průběžné načítání	<b>bool</b>
<b>yerr</b>	Náhradní hodnota pro případ chyby	<b>double</b>

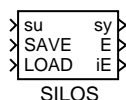
### Výstupy

<b>y</b>	Výstupní signál	<b>double</b>
<b>E</b>	Příznak chyby	<b>bool</b>
<b>iE</b>	Kód chyby operačního systému	<b>long</b>

## SILOS – Uložení vstupního řetězce, načtení výstupního řetězce

Symbol bloku

Licence: [STANDARD](#)



### Popis funkce

Blok **SILOS** je určen pro export nebo import jednoho řetězce do nebo ze souboru. Řetězec je uložen při náběžné hraně (**off**→**on**) na vstupu **SAVE** a po úspěšném uložení je nastavena také na výstup **sy**. Načtení hodnoty probíhá při startu a při náběžné hraně (**off**→**on**) na vstupu **LOAD**.

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.

### Vstupy

<b>su</b>	Vstupní řetězec	⊙0	<b>string</b>
<b>SAVE</b>	Uložení vstupního řetězce do souboru		<b>bool</b>
<b>LOAD</b>	Načtení řetězce ze souboru		<b>bool</b>

### Parametry

<b>fname</b>	Jméno souboru pro ukládání/načítání parametrů	<b>string</b>
<b>CSF</b>	Příznak pro průběžné ukládání	<b>bool</b>
<b>CLF</b>	Příznak pro průběžné načítání	<b>bool</b>

### Výstupy

<b>sy</b>	Výstupní řetězec	<b>string</b>
<b>E</b>	Příznak chyby	<b>bool</b>
	<b>off</b> ... bez chyby	
	<b>on</b> .... nastala chyba	
<b>iE</b>	Kód chyby operačního systému	<b>long</b>





## Kapitola 13

# MODEL – Simulace dynamických systémů

### Obsah

---

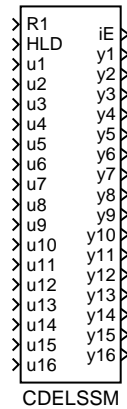
CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním . . . . .	322
CSSM – Stavový model spojitého lineárního systému . . . . .	325
DDELSSM – Stavový model diskrétního lineárního systému s dopravním zpožděním . . . . .	328
DSSM – Stavový model diskrétního lineárního systému . . . . .	330
FMUCS – Import modelu FMU CS (pro Co-Simulation) . . . . .	332
FMUINFO – Informace o importovaném modelu FMU . . . . .	334
FOPDT – Model systému 1. řádu s dopravním zpožděním . . . . .	336
MDL – Model procesu . . . . .	337
MDLI – Model procesu s proměnnými parametry . . . . .	338
MVD – Motorizovaný pohon ventilu . . . . .	339
SOPDT – Model systému 2. řádu s dopravním zpožděním . . . . .	340

---

## 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

$$\begin{aligned}\frac{dx(t)}{dt} &= A_c x(t) + B_c u(t - del), \quad x(0) = x_0 \\ y(t) &= C_c x(t) + D_c u(t),\end{aligned}$$

kde  $x(t) \in \mathbb{R}^n$  je vektor stavu,  $x_0 \in \mathbb{R}^n$  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  $x_0$  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

$$\begin{aligned}x((k+1)T) &= A_d x(kT) + B_{d1} u((k-d)T) + B_{d2} u((k-d+1)T), \quad x(0) = x_0 \\ y(kT) &= C_c x(kT) + D_c u(kT),\end{aligned}$$

kde  $k \in \{1, 2, \dots\}$  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 nezadáva, 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 diskretních matic  $A_d$ ,  $B_{d1}$  a  $B_{d2}$  je založen na metodě popsané v [6], využívající Padéových aproximací maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskretního stavového modelu uvedeného výše.

## Vstupy

R1	Resetovací signál, je-li R1 = on, je stavový vektor <b>x</b> nastaven na počáteční hodnotu <b>x0</b> . 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 <b>Bc</b> .	double

## Výstupy

iE	Kód chyby bloku	error
	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 <a href="#">B</a>
y1..y16	Výstupy simulovaného systému. Pro danou simulaci se používá prvních $p$ výstupů, kde $p$ je počet řádků matice <b>Cc</b> .	double

## Parametry

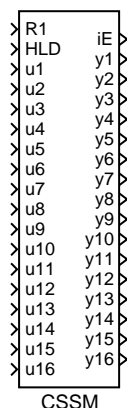
UD	Příznak použití matice <b>Dc</b> . Pokud je UD=off, matice <b>Dc</b> se při simulaci nepoužívá (chová se jako by byla nulová).	bool
del	Dopravní zpoždění modelu [s].	↓0.0 double

<b>is</b>	Stupeň Padéovy aproximace maticové exponenciály pro výpočet matic diskretizovaného systému.	↓0 ↑4 ⊙2	<b>long</b>
<b>eps</b>	Požadovaná přesnost Padéovy aproximace.	↓0.0 ↑1.0 ⊙0.0	<b>double</b>
<b>Ac</b>	Matice (typu [n,n]) dynamiky spojitého lineárního systému.		<b>double</b>
<b>Bc</b>	Vstupní matice (typu [n,m]) spojitého lineárního systému.		<b>double</b>
<b>Cc</b>	Výstupní matice (typu [p,n]) spojitého lineárního systému.		<b>double</b>
<b>Dc</b>	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.		<b>double</b>
<b>x0</b>	Počáteční hodnota vektoru stavu (typu [n]) spojitého lineárního systému.		<b>double</b>

## 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

$$\begin{aligned}\frac{dx(t)}{dt} &= A_c x(t) + B_c u(t), \quad x(0) = x_0 \\ y(t) &= C_c x(t) + D_c u(t),\end{aligned}$$

kde  $x(t) \in \mathbb{R}^n$  je vektor stavu,  $x_0 \in \mathbb{R}^n$  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  $x_0$  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

$$\begin{aligned}x((k+1)T) &= A_d x(kT) + B_d u(kT), \quad x(0) = x_0 \\ y(kT) &= C_c x(kT) + D_c u(kT),\end{aligned}$$

kde  $k \in \{1, 2, \dots\}$  je krok simulace,  $T$  je perioda spouštění bloku v [s]. Perioda  $T$  se v bloku nezadá, 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 diskretních matic  $A_d$  a  $B_d$  je založen na metodě popsané v [6], využívající Padéových aproximací maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskretního stavového modelu uvedeného výše.

### Vstupy

R1	Resetovací signál, je-li R1 = on, je stavový vektor $x$ nastaven na počáteční hodnotu $x_0$ . 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 $B_c$ .	double

### Výstupy

iE	Kód chyby bloku	error
	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 blízka singulární matici)	
	xxx ... chybový kód xxx systému REX, více viz přílohu B	
y1..y16	Výstupy simulovaného systému. Pro danou simulaci se používá prvních $p$ výstupů, kde $p$ je počet řádků matice $C_c$ .	double

### Parametry

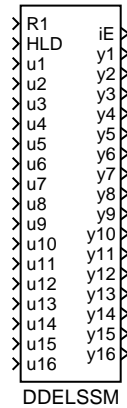
UD	Příznak použití matice $D_c$ . Pokud je UD=off, matice $D_c$ se při simulaci nepoužívá (chová se jako by byla nulová).	bool
is	Stupeň Padéovy aproximace maticové exponenciály pro výpočet matic diskretizovaného systému.	long
	↓0 ↑4 ⊙2	
eps	Požadovaná přesnost Padéovy aproximace.	double
	↓0.0 ↑1.0 ⊙0.0	
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

<b>Dc</b>	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr <b>UD=on</b> . Je-li <b>UD=off</b> , rozměry matice <b>Dc</b> se nekontrolují.	<b>double</b>
<b>x0</b>	Počáteční hodnota vektoru stavu (typu [n]) spojitého lineárního systému.	<b>double</b>

## 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

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k-d), \quad x(0) = x_0 \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned}$$

kde  $k$  je krok simulace,  $x(k) \in \mathbb{R}^n$  je vektor stavu,  $x_0 \in \mathbb{R}^n$  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  $x_0$  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 = <b>on</b> , je stavový vektor <b>x</b> nastaven na počáteční hodnotu <b>x0</b> . Simulace se znovu spustí sestupnou hranou signálu R1 ( <b>on</b> → <b>off</b> ).	bool
HLD	Zmrazení simulace po dobu, kdy je HLD= <b>on</b> .	bool
u1..u16	Vstupy simulovaného systému. Pro danou simulaci se používá prvních <i>m</i> vstupů, kde <i>m</i> 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 <i>p</i> výstupů, kde <i>p</i> je počet řádků matice Cd.	double

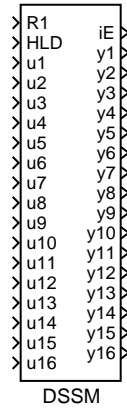
## Parametry

UD	Příznak použití matice Dd. Pokud je UD= <b>off</b> , 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= <b>on</b> . Je-li UD= <b>off</b> , 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

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k), \quad x(0) = x_0 \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned}$$

kde  $k$  je krok simulace,  $x(k) \in \mathbb{R}^n$  je vektor stavu,  $x_0 \in \mathbb{R}^n$  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  $x_0$  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 **bool** počáteční hodnotu **x0**. Simulace se znovu spustí sestupnou hranou signálu **R1 (on→off)**.

HLD	Zmrazení simulace po dobu, kdy je HLD= <b>on</b> .	bool
u1..u16	Vstupy simulovaného systému. Pro danou simulaci se používá prvních $m$ vstupů, kde $m$ je počet sloupců matice B <i>d</i> .	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 <a href="#">B</a>	error
y1..y16	Výstupy simulovaného systému. Pro danou simulaci se používá prvních $p$ výstupů, kde $p$ je počet řádků matice C <i>d</i> .	double

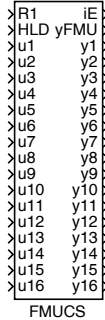
## Parametry

UD	Příznak použití matice D <i>d</i> . Pokud je UD= <b>off</b> , matice D <i>d</i> se při simulaci nepoužívá (chová se jako by byla nulová).	bool
A <i>d</i>	Matice (typu [n,n]) dynamiky diskrétního lineárního systému.	double
B <i>d</i>	Vstupní matice (typu [n,m]) diskrétního lineárního systému.	double
C <i>d</i>	Výstupní matice (typu [p,n]) diskrétního lineárního systému.	double
D <i>d</i>	Matice (typu [p,m]) přímého působení vstupu na výstup. Matice se v modelu používá jen pokud je parametr UD= <b>on</b> . Je-li UD= <b>off</b> , rozměry matice D <i>d</i> se nekontrolují.	double
x0	Počáteční hodnota vektoru stavu (typu [n]) diskrétního lineárního systému.	double

## FMUCS – Import modelu FMU CS (pro Co-Simulation)

Symbol bloku

Licence: [FMI/FMU](#)



### Popis funkce

Blok **FMUCS** umožňuje simulovat dané FMU ve formátu Co-Simulation (tj. včetně FMU, obsahujícího kromě modelu i simulační solver).

V parametru **FMUPath** je uložena úplná cesta k souboru modelu s příponou **.FMU** na počítači, kde je tento blok konfigurován. Tento blok předpokládá, že načtený model má maximálně 16 vstupů (symbolicky označených **u1..u16**), 16 výstupů (**y1..y16**) a 16 parametrů (**p1..p16**), které jsou po načtení modelu namapovány na jim pořadím odpovídajícím vstupům, výstupům a parametrům modelu FMU, které lze zjistit pomocí informačního bloku **FMUINFO**. Protože FMU může mít velký počet parametrů, lze jich pro mapování zvolit jen několik pomocí seznamu **SelPars**, jehož význam je popsán v bloku **FMUINFO**. Všechny vstupy **u1..u16**, výstupy **y1..y16** i parametry **p1..p16** jsou typu **double**, který odpovídá typu **fmu2Real** ze specifikace [7]. Pokud má některý z těchto signálů v dané FMU typ **fmu2Integer** nebo **fmu2Boolean**, je v tomto bloku na tento typ automaticky zkonvertován. Signály typu **fmu2String** nejsou v tomto bloku podporovány.

První dva vstupy řídí běh simulace. Logický vstup **R1** umožňuje resetovat (inicializovat) simulaci do počátečních podmínek. Náběžnou hranou ( $0 \rightarrow 1$ ) tohoto vstupu se simulace zastaví, sestupnou hranou ( $1 \rightarrow 0$ ) se simulace spustí z počátečních podmínek od simulačního času 0. Hodnota logického vstupu **HLD** = 1 umožňuje pozastavit simulaci, je-li **HLD** = 0 simulace běží (tj. i v případě, že vstup **HLD** není připojen).

Výstup **iE** indikuje výskyt chyby simulace, pokud jeho hodnota je nenulová. Vyskytneli se při simulaci chyba, simulace se ukončí až do nového startu simulačního systému. Druhý výstup **yFMU** obsahuje odkaz na načtenou FMU. Tento odkaz může být připojen na vstup bloku **FMUINFO**, který umožní získat počáteční informace o FMU. Odpojením tohoto odkazu od bloku **FMUINFO** se žádné informace nevyhodnocují, což může start simulace zrychlit.

Zbylé parametry bloku **FMUCS** slouží pro řízení běhu simulace. Pokud je logický vstup **TSTOPDEF** = 1, bude experiment ukončen v čase **tstop**, po němž se hodnoty výstupů

nemění (jako by byla nastaven vstup  $HLD = 1$ ) až do následujícího resetu vstupem **R1**. Parametr **eps** určuje přesnost simulace.

Parametr **loglevel** určuje závažnost výpisů, které budou ukládány při běhu simulačního systému do systémového logu. Je třeba dát pozor na to, že velké množství výpisů během probíhající simulace tuto simulaci značně zpomaluje, a proto je vhodné množství výpisů ukládaných do systémového logu vizuálně zkontrolovat!

FMI 2.0 zavádí, kromě obyčejných parametrů, ještě tzv. *laditelné parametry* (tunable parameters). Obyčejné parametry lze nastavit ve fázi inicializace modelu (před zahájením vlastní simulace), kdežto laditelné parametry lze měnit i mezi jednotlivými simulačními kroky. Tento požadavek specifikace FMI 2.0 však některé simulační systémy nerespektují, např. OpenModelica. Proto byl v tomto bloku zaveden logický parametr **TUNEALLP**. Pokud je jeho hodnota nulová, nastavují se před každým simulačním krokem pouze laditelné parametry. Je-li **TUNEALLP = 1**, považují se všechny vybrané parametry (pomocí parametru **SelPars**) za laditelné, a proto se blok před každým simulačním krokem pokouší nastavit všechny.

## Vstupy

<b>R1</b>	Reset bloku	<b>bool</b>
<b>HLD</b>	Podržení aktuálního stavu modelu	<b>bool</b>
<b>u1..u16</b>	Vstupní signály daného FMU	<b>double</b>

## Výstupy

<b>iE</b>	Kód chyby	<b>error</b>
<b>yFMU</b>	Výstupní odkaz na instanci FMU	<b>reference</b>
<b>y1..y16</b>	Výstupní signály daného FMU	<b>double</b>

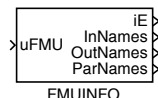
## Parametry

<b>tstop</b>	Koncový čas simulace	$\downarrow 0.000001 \odot 1.0$	<b>double</b>
<b>eps</b>	Přesnost aproximace	$\downarrow 0.0 \uparrow 1.0 \odot 0.000001$	<b>double</b>
<b>loglevel</b>	Úroveň protokolování knihovny FMI do systémového logu	$\downarrow 0 \uparrow 7 \odot 2$	<b>long</b>
	0 ..... Nic	4 ..... Info	
	1 ..... Fatální chyba	5 ..... Podrobný	
	2 ..... Chyba	6 ..... Ladění	
	3 ..... Varování	7 ..... Všechno	
<b>SelPars</b>	Seznam vybraných parametrů FMU		<b>string</b>
<b>TUNEALLP</b>	Považuj všechny vybrané parametry za laditelné parametry		<b>bool</b>
<b>p1..p16</b>	Vybrané parametry modelu FMU podle nastavení parametru <b>SelPars</b>		<b>double</b>

## FMUINFO – Informace o importovaném modelu FMU

Symbol bloku

Licence: [FMI/FMU](#)



### Popis funkce

Blok `FMUINFO` je pomocným blokem, který poskytuje informace o daném modelu ve formátu FMU, aniž by musel být uživatelem analyzován vnitřek souboru s příponou `.FMU`.

Na vstup `uFMU` bloku je připojen odkaz na instanci simulačního modelu FMU, která je výstupem výkonného simulačního bloku (např. výstup `yFMU` bloku `FMUCS`). Na výstupech `InNames`, `OutNames` a `ParNames` jsou řetězce obsahující seznamy názvů vstupů, výstupů a parametrů dané FMU v pořadí, v jakém je poskytují funkce použité knihovny FMI. Jednotlivé názvy jsou odděleny řetězcem uloženým v parametru `Separ` tohoto bloku. Protože typické modely obsahují velké množství parametrů (kromě parametrů modelu v nejvyšší úrovni hierarchie též parametry všech modelů v nižších úrovních hierarchie), umožňuje parametr `SelPars` zvolit požadované parametry, které se objeví ve výstupu `ParNames` (detaily viz tabulku parametrů níže).

### Vstup

<code>uFMU</code>	Vstupní odkaz na instanci FMU	<code>reference</code>
-------------------	-------------------------------	------------------------

### Výstupy

<code>iE</code>	Kód chyby	<code>error</code>
<code>InNames</code>	Seznam jmen vstupů FMU oddělených parametrem <code>Separ</code>	<code>string</code>
<code>OutNames</code>	Seznam jmen výstupů FMU oddělených parametrem <code>Separ</code>	<code>string</code>
<code>ParNames</code>	Seznam jmen vybraných parametrů FMU oddělených parametrem <code>Separ</code> . Do seznamu jsou zařazeny pouze takové parametry, které odpovídají nastavení parametru <code>SelPars</code> , viz níže	<code>string</code>

### Parametry

<code>SelPars</code>	Seznam vybraných parametrů FMU, které budou zařazeny do výstupu <code>ParNames</code> . Pokud je tento řetězec prázdný (neobsahuje žádný znak), budou do seznamu zařazeny všechny parametry z nejvyšší úrovně hierarchie modelu v daném FMU. V případě, že je tento řetězec neprázdný, obsahuje seznam parametrů, které se mají objevit na výstupu <code>ParNames</code> , oddělených znakem <code>,</code> (čárka)	<code>string</code>
----------------------	---	---------------------

**Separ**      Oddělovač jmen v řetězcových výstupech (nejlépe složený z jediného znaku), kterým budou oddělena jména vstupů, výstupů a parametrů ve výstupních řetězcích `InNames`, `OutNames` a `ParNames`      `string` ☺,

## FOPDT – Model systému 1. řádu s dopravním zpožděním

Symbol bloku

Licence: [STANDARD](#)

## Popis funkce

Blok FOPDT realizuje diskretní simulátor lineárního systému prvního řádu s přídavným dopravním zpožděním, který je popsán následující přenosovou funkcí:

$$P(s) = \frac{k_0}{(\tau \cdot s + 1)} \cdot e^{-\text{del} \cdot s}$$

Diskretní 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 diskretní 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 <b>del</b> (na tolik hodnot se alokuje paměť)	↓1 ↑100000000 ⊙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

<b>u</b>	Analogový vstupní signál	double
<b>k0</b>	Statické zesílení	double
<b>del</b>	Dopravní zpoždění [s]	double
<b>tau1</b>	První časová konstanta	double
<b>tau2</b>	Druhá časová konstanta	double

## Výstup

<b>y</b>	Analogový výstupní signál	double
----------	---------------------------	--------

## Parametry

<b>nmax</b>	Délka vyrovnávací paměti pro dopravní zpoždění <b>del</b> (na tolik hodnot se alokuje paměť)	long ↓1 ↑10000000 ⊕10
-------------	--	--------------------------

## 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  $1/t_v$ , kde  $t_v$  je parametr bloku. Při UP = on (DN = on) otevírání probíhá až do úplného otevření  $y = \text{hilim}$  (úplného zavření  $y = \text{lolim}$ ) ventilu. Krajiní poloha otevření (zavření) je signalizována koncovým spínačem HS (LS). Počáteční poloha ventilu po spuštění je  $y = y_0$ . 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

$y_0$	Počáteční poloha ventilu	double
$t_v$	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 diskretní simulátor lineárního systému druhého řádu s přídavným dopravním zpožděním, který je alternativně popsán, v závislosti na parametru `itf`, následujícími přenosovými funkcemi:

$$\begin{aligned} \text{itf} = 1: \quad P(s) &= \frac{\text{pb1} \cdot s + \text{pb0}}{s^2 + \text{pa1} \cdot s + \text{pa0}} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 2: \quad P(s) &= \frac{k0 (\text{tau} \cdot s + 1)}{(\text{tau1} \cdot s + 1) (\text{tau2} \cdot s + 1)} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 3: \quad P(s) &= \frac{k0 \cdot \text{om}^2 \cdot (\text{tau}/\text{om} \cdot s + 1)}{(s^2 + 2 \cdot \text{xi} \cdot \text{om} \cdot s + \text{om}^2)} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 4: \quad P(s) &= \frac{k0 (\text{tau} \cdot s + 1)}{(\text{tau1} \cdot s + 1) s} \cdot e^{-\text{del} \cdot s} \end{aligned}$$

Diskretní 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

<code>itf</code>	Tvar přenosové funkce	⊙1	long
	1 ..... obecný tvar		
	2 ..... reálné póly ve jmenovateli		
	3 ..... komplexní póly (kmitavý systém)		
	4 ..... systém s integrátorem		
<code>k0</code>	Statické zesílení	⊙1.0	double
<code>tau</code>	Časová konstanta v čitateli		double

<code>tau1</code>	První časová konstanta ve jmenovateli	⊙1.0	double
<code>tau2</code>	Druhá časová konstanta ve jmenovateli	⊙1.0	double
<code>om</code>	Vlastní frekvence	⊙1.0	double
<code>xi</code>	Relativní koeficient tlumení	⊙1.0	double
<code>pb0</code>	Koeficient čitatele: $s^0$	⊙1.0	double
<code>pb1</code>	Koeficient čitatele: $s^1$	⊙1.0	double
<code>pa0</code>	Koeficient jmenovatele: $s^0$	⊙1.0	double
<code>pa1</code>	Koeficient jmenovatele: $s^1$	⊙1.0	double
<code>del</code>	Dopravní zpoždění [s]		double
<code>nmax</code>	Délka vyrovnávací paměti pro dopravní zpoždění <code>del</code> (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) . . . . .	344
RTOV – Vektorový multiplexer . . . . .	345
SWVMR – Přepínač vektorového/maticového/odkazovacího signálu	347
VTOR – Vektorový demultiplexer . . . . .	348

---

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

<b>filename</b>	Datový soubor s hodnotami oddělenými čárkou	<b>string</b>
<b>TRN</b>	Transponuj načtenou matici	<b>bool</b>
<b>nmax</b>	Rezervovaná paměť pro pole	↓2 ↑100000000 ⊙100 <b>long</b>
<b>etype</b>	Typ prvků	⊙8 <b>long</b>
	1 ..... Bool	
	2 ..... Byte	
	3 ..... Short	
	4 ..... Long	
	5 ..... Word	
	6 ..... DWord	
	7 ..... Float	
	8 ..... Double	
	-- .....	
	10 .... Large	
<b>acn</b>	Počáteční hodnota pole	⊙[0 1 2 3] <b>double</b>

## Výstup

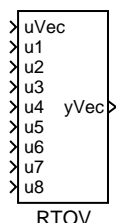
<b>vec</b>	Odkaz na vektor/matici dat	<b>reference</b>
------------	----------------------------	------------------



## 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

<b>uVec</b>	Vektorový signál	<b>reference</b>
<b>u1</b>	Analogový vstupní signál	<b>double</b>
<b>u2</b>	Analogový vstupní signál	<b>double</b>
<b>u3</b>	Analogový vstupní signál	<b>double</b>
<b>u4</b>	Analogový vstupní signál	<b>double</b>
<b>u5</b>	Analogový vstupní signál	<b>double</b>
<b>u6</b>	Analogový vstupní signál	<b>double</b>
<b>u7</b>	Analogový vstupní signál	<b>double</b>
<b>u8</b>	Analogový vstupní signál	<b>double</b>

### Výstup

<b>yVec</b>	Vektorový signál	<b>reference</b>
-------------	------------------	------------------

### Parametry

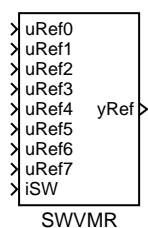
<b>nmax</b>	Alokovaná velikost vektoru	↓0 ⊙8	long
<b>offset</b>	Index prvního vstupu ve vektoru	↓0	long

<b>n</b>	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

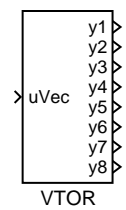
### Výstup

yRef	Vektorový signál	reference
------	------------------	-----------

VTOR – Vektorový demultiplexer

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok VTOR rozděljuje 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

---

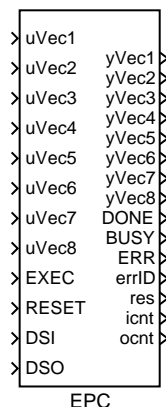
EPC – Blok pro spouštění externích programů . . . . .	350
HTTP – * Blok pro generování požadavků HTTP GET a POST .	353
SMTP – * Blok pro odesílání e-mailových oznámení přes SMTP .	355
RDC – Komunikační blok . . . . .	357
REXLANG – Volně programovatelný blok . . . . .	362

---

## EPC – Blok pro spouštění externích programů

Symbol bloku

Licence: [ADVANCED](#)



### Popis funkce

Tento blok v okamžiku náběžné hrany ( $\text{off} \rightarrow \text{on}$ ) 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 řetěžit, 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

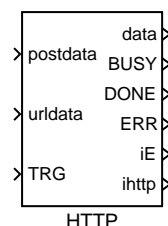
- Spuštěný skript má stejnou prioritu, jako task, který ji spustil. Ta je (implicitně) hodně velká (v některých případech dokonce vyšší, než tasky zpracovávající interrupty v kernelu operačního systému). Pokud je toto nežádoucí (tj. zejména pokud externí skript trvá dlouho), je potřeba prioritu externího 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	long
	1 ..... GET		
	2 ..... POST		
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

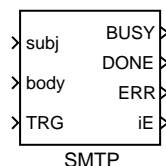
## Výstupy

<code>data</code>	Data z odpovědi	<code>string</code>
<code>BUSY</code>	Odesílání HTTP požadavku	<code>bool</code>
<code>DONE</code>	HTTP požadavek byl zpracován	<code>bool</code>
<code>ERROR</code>	Příznak chyby	<code>bool</code>
<code>errId</code>	Kód chyby	<code>error</code>
<code>hterror</code>	HTTP odpověď	<code>long</code>

## 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

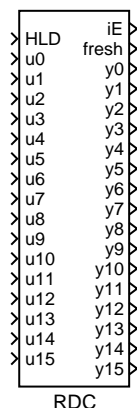
## Výstupy

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 přenesou `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

<code>HLD</code>	Vstup pozastavující činnost bloku. Pokud je <code>HLD = on</code> , blok nevysílá ani nepřijímá žádné pakety.	<code>bool</code>
<code>u0..u15</code>	Hodnoty, které se předávají/zapisují na hodnoty <code>y0</code> až <code>y15</code> spolupracujícího bloku <code>RDC</code>	<code>double</code>

## Výstupy

<code>iE</code>	Zobrazuje kód poslední chyby. Použitá čísla jsou v následující tabulce:	<code>long</code>
	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)

<code>fresh</code>	Udává počet sekund od přijetí posledního paketu. Má význam pro detekci chyby protilehlého bloku.	double
<code>y0..y15</code>	Signál přijatý ze vzdáleného bloku RDC – hodnoty naposledy přijatého paketu	double

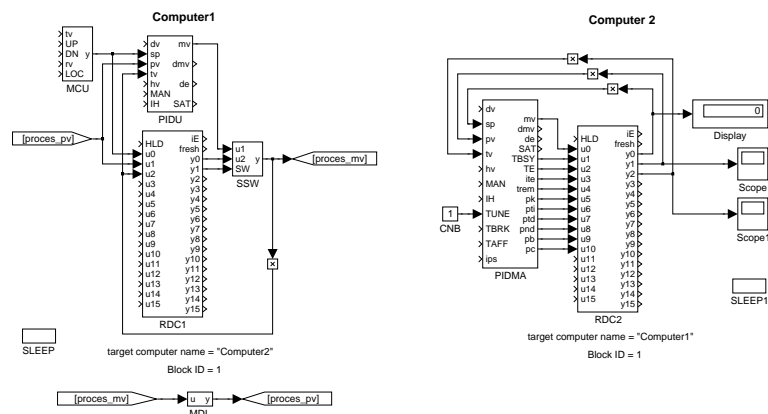
## Parametry

<code>target</code>	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
<code>rport</code>	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
<code>lport</code>	Místní port (tzv. local port), význam podobný jako u parametru <code>rport</code> . Remote port platí pro počítač, kam je paket poslán, local port platí pro počítač, ze kterého je paket poslán. ☉1288	word

<b>id</b>	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	<b>long</b>
<b>period</b>	Perioda v sekundách, určující nejkratší dobu, po které se vysílají a případně čtou došlé pakety. V případě hodnoty $\text{period} \leq 0$ , 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 <b>Variable step</b> ).	<b>double</b>

## 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

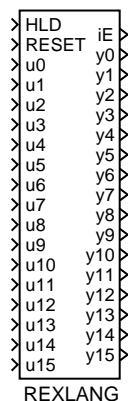
**<pin>** je název signálu nabývající jedné z hodnot:

- **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 řídicího 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 řídicího systému mohou být implementovány funkce `init()` (volá se při startu), `exit()` (volá se při ukončení řídicího 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  $y = a_n * x^n + \dots + a_1 * x + a_0$ . Funkce má nepovinný první parametr, který určuje počet prvků.

`double poly(n, x, vec)`

Vypočte hodnotu polynomu  $y = vec[n] * x^n + \dots + vec[1] * x + vec[0]$ .

`double scal(n, vec1, vec2)`

Vypočte skalární součin  $y = vec1[0] * vec2[0] + \dots + vec1[n-1] * vec2[n-1]$ .

`double scal(n, vec1, vec2, skip1, skip2)`

Vypočte skalární součin  $y = vec1[0] * vec2[0] + vec1[skip1] * vec2[skip2] + \dots + vec1[(n-1) * skip1] * vec2[(n-1) * skip2]$ . 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  $y = vec1[0] * vec2[n-1] + vec1[1] * vec2[n-1] + \dots + vec1[n-1] * vec2[0]$ .

`double sum(n, vec)`

Sečte prvky vektoru, tj.  $y = vec[0] + vec[1] + \dots + vec[n-1]$ .

`double sum([n,] val1, ..., valn)`

Sečte prvky, tj.  $y = val1 + val2 + \dots + valn$ . 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. Návrátový 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`. Návrátový 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 indexu 0).

- **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.

`strreplace(str,pattern,substr)`

Najde všechny výskyty stringu `pattern` ve stringu `str` a nahradí je stringem `substr`. Nový string je uložen do `str`.

`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 desetinné 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 kompatibility s ANSI C. Lze použít konstrukci `dest=dest+src` se stejným výsledkem.

`strcmp(str1,str2)`

Porovnání stringů. Funkce je zavedena z důvodu kompatibility 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 uzavřeným 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  $(0, 1)$ . 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`, `VOU`). 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 (zadáva 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/strukturu 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 - viz SetOptions
void SetOptions(long handle, long params[])
    Nastaví parametry sériové linky nebo socketu. Pole musí být dostatečně
    dlouhé - aktuálně je 22 parametrů pro sériovou linku, 2 pro soubor (1.
    prvek je režim: 1=seek begin, 2=seek current, 3=seek end, 4=set file end,
    2. prvek je offset pro seek), 3 pro SPI (1. prvek je SPI mode, 2. prvek
    je bits per word, 3. prvek je max speed Hz), 5 pro I2C (1. prvek je slave
    address, 2. prvek je 10bits address flag, 3. prvek je Packet Error Checking
    flag, 4. prvek je number of retries, 5. prvek je timeout)
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, že 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 logování)
- 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 ReadLine(long handle, string data)`

Přečte jednu řádku z (textového) souboru sériové linky nebo socketu; přečtené znaky jsou v proměnné `data` až do velikosti stringu; vrací skutečnou délku řádky 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`.*

`long DeleteFile(string filename)`

Smaže soubor. Vrací 0 pokud je soubor smazán; záporné číslo znamená chybu.

`long RenameFile(string filename, string newfilename)`

Přejmenuje soubor. Vrací 0 pokud je soubor přejmenován; záporné číslo znamená chybu.

## Ladění kódu, debugging

Pro ladění kódu je k dispozici příkaz `Trace`, viz výše. Dále lze použít výstupy bloku, které se noužívají 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 různé mezivýsledky pomocí funkce `SetExt`.

## Poznámky

- Typ vstupů `u0..u15`, výstupů `y0..y15` a parametrů `p0..p15` se určuje až při překladači 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)
  - Návrátové adresy funkcí
  - Parametry předávané funkcím (včetně „vestavěných“)
  - Lokální proměnné funkcí
  - Návrátová 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

HLID	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 <code>Init()</code> )	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 <code>main()</code> popř. <code>init()</code> -1 .... provádění programu skončilo příkazem <code>Suspend()</code> , 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	long
	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.		
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 <b>bez kontroly</b> se doporučuje nepoužívat (může vést až k pádu aplikace na cílové platformě při nesprávně napsaném kódu).	⊙3	long
	1 ..... bez kontroly 2 ..... základní kontrola 3 ..... úplná kontrola		
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
```

## Kapitola 16

# MC\_SINGLE – Řízení pohybu v jedné ose

### Obsah

RM_Axis – Osa pro řízení pohybu . . . . .	382
MC_AccelerationProfile, MCP_AccelerationProfile – Generování trajektorie (zrychlení) . . . . .	388
MC_Halt, MCP_Halt – Zastavení pohybu (přerušitelné) . . . . .	392
MC_HaltSuperimposed, MCP_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné) . . . . .	394
MC_Home, MCP_Home – Nalezení výchozí polohy . . . . .	395
MC_MoveAbsolute, MCP_MoveAbsolute – Pohyb do pozice (absolutní souřadnice) . . . . .	397
MC_MoveAdditive, MCP_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	401
MC_MoveRelative, MCP_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění) . . . . .	404
MC_MoveSuperimposed, MCP_MoveSuperimposed – Pohyb do pozice (přídavný pohyb) . . . . .	407
MC_MoveContinuousAbsolute, MCP_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice) . . . . .	410
MC_MoveContinuousRelative, MCP_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu) . . . . .	414
MC_MoveVelocity, MCP_MoveVelocity – Pohyb konstantní rychlostí . . . . .	418
MC_PositionProfile, MCP_PositionProfile – Generování trajektorie (poloha) . . . . .	421
MC_Power – Aktivace osy . . . . .	425
MC_ReadActualPosition – Skutečná poloha osy . . . . .	426
MC_ReadAxisError – Chyba osy . . . . .	427
MC_ReadBoolParameter – Čtení parametru (logická hodnota) . . . .	428

MC_ReadParameter – Čtení parametru (číselná hodnota) . . . . .	429
MC_ReadStatus – Stav osy . . . . .	431
MC_Reset – Nulování chyb osy . . . . .	433
MC_SetOverride, MCP_SetOverride – Nastavení násobivých faktorů na ose . . . . .	434
MC_Stop, MCP_Stop – Zastavení pohybu . . . . .	436
MC_TorqueControl, MCP_TorqueControl – Řízení síly/momentu . . . . .	438
MC_VelocityProfile, MCP_VelocityProfile – Generování trajektorie (rychlost) . . . . .	441
MC_WriteBoolParameter – Nastavení parametru (logická hodnota) . . . . .	445
MC_WriteParameter – Nastavení parametru (číselná hodnota) . . . . .	446
RM_AxisOut – Výstupní blok osy . . . . .	448
RM_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení) . . . . .	450
RM_Track – Sledování a krokování . . . . .	452

Tato kategorie bloků zahrnuje bloky pro řízení jedné osy (jednoho motoru), tak jak jsou definovány ve specifikaci PLCopen. Proto zde nejsou dodrženy konvence pro pojmenovávání zavedené v systému REX. Protože PLCopen používá pro všechny bloky v názvu prefix MC\_, je tento princip dodržen i zde. Pro řízení pohybu je vhodné, spíše však nutné, bloky dle PLCopen doplnit o další bloky (jedná se o vše, co je v normě označeno jako *vendor specific*, a různá rozšíření). Takové bloky mají v názvu prefix RM\_. Dále je potřeba si uvědomit, že PLCopen (respektive norma IEC 61131-3, ze které vychází) nerozlišuje vstupy od parametrů (vše na značce bloku vypadá jako vstup). Řídicí systém REX však parametry a vstupy zpracovává odlišně, zejména z důvodu lepší přehlednosti. Proto skoro všechny bloky existují ve dvojím provedení: s prefixem MC\_, které jsou plně kompatibilní s PLCopen a mají tedy parametry na vstupech, a bloky s prefixem MCP\_, které mají své vnitřní parametry jako ostatní bloky řídicího systému REX. Parametry, které bloky používají, ale nejsou definovány v PLCopen (tzv. *vendor specific* parametry), jsou i u bloků s prefixem MC\_ zadávány jako parametry a výše popsaným změnám nepodléhají.

Specifikace PLCopen říká, že pro činnost bloku je důležitá hodnota parametrů/vstupů při náběžné hraně na vstupu **Execute**. Předtím a potom se mohou parametry/vstupy libovolně měnit a na funkci bloku to nemá vliv. V systému REX se předpokládá, že parametry se mění jen občas a proto parametry (na rozdíl od vstupů) nejsou „zapamatovány“ při náběžné hraně na vstupu **Execute**. Je proto potřeba dodržovat pravidlo, že parametry bloku se nesmí měnit, během jeho činnosti, tj. pokud je výstup **Busy** ve stavu zapnuto.

Popis bloků v této příručce je dostatečný pro jejich použití, nicméně nejsou zde vysvětleny hlubší souvislosti a motivace. Proto je doporučeno před použitím bloků pro řízení pohybu prostudovat i specifikaci PLCopen.

PLCopen definuje některé signály jako vstupně-výstupní. Je to zejména odkaz na osu (signál **Axis**) a odkazy na další, obvykle *vendor specific*, struktury. Řídicí systém REX vstupně-výstupní signály nepodporuje. Proto všechny bloky používají místo signálu **Axis**

vstup **uAxis** a výstup **yAxis**. Blok na výstup **yAxis** vždy kopíruje hodnotu (odkaz na strukturu) vstupu **uAxis**. Pro správnou funkci je nutné, aby vstup **uAxis** byl spojen s výstupem **axisRef** příslušného bloku **RM\_Axis**, a to buď přímo nebo prostřednictvím výstupu **yAxis** jiného bloku. Výstup **yAxis** není pro bloky důležitý, ale použitá koncepce umožňuje bloky řetězit a tím určit pořadí jejich vykonávání. Ostatní vstupně-výstupní signály (odkazy) používají stejný princip, popřípadě jsou definovány pouze jako vstupní.

PLCopen definuje výstupy **Busy**, **Active**, **CommandAborted** jako nepovinné téměř u všech bloků. V řídicím systému REX jsou tyto signály zavedeny stejným způsobem, ačkoliv v současné implementaci se u některých bloků nepoužívají. Tento způsob byl zvolen z důvodu snadné budoucí rozšiřitelnosti.

Jednotky použité pro polohu a vzdálenost si uživatel může zvolit libovolně. Mohou to být metry, milimetry, pulzy z čidla polohy, úhlové stupně (v případě rotační osy) nebo cokoliv jiného. Je však nutné zvolenou jednotku používat ve všech blocích připojených k příslušné ose a dále je potřeba si uvědomit, že rychlost se ve všech blocích musí zadávat ve zvolených jednotkách polohy za sekundu a analogicky zrychlení a jerk. Časové údaje jsou vždy v sekundách.

Řídicí systém REX používá více vláken pro spouštění bloků. Normálně se o to uživatel nemusí starat, nicméně použití odkazů (vstupy typu **reference**) může narušit synchronizační mechanismus. Aby toto nenastalo, je nutné mít blok **RM\_Axis** a blok, který je k němu připojen vstupem typu **reference**, ve stejné úloze, tj. všechny bloky příslušející k jedné ose musí být v jedné úloze řídicího systému REX (viz bloky **EXEC**, **TASK**, **IOTASK** apod.). Některé bloky již mají synchronizaci ošetřenou, u těch je potřeba se řídit pokyny v této příručce. Momentálně se však jedná pouze o blok **RM\_AxisSpline**.

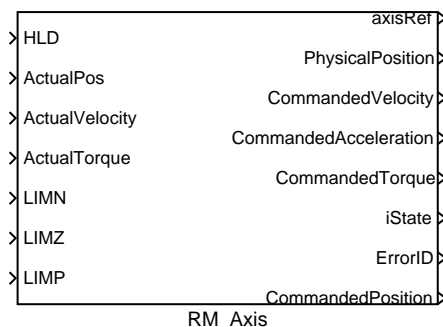
Některé vstupy a parametry jsou typu výběr ze seznamu. V takovém případě jsou v seznamu vždy všechny hodnoty pro daný typ signálu. Pro některé bloky však určité hodnoty nedávají smysl (například blok **MC\_MoveVelocity** nepodporuje **Direction = shortest\_way**). Platné hodnoty jsou uvedeny v této příručce v popisu jednotlivých bloků.

U všech pohybových bloků platí pravidlo, že pokud parametr určující maximální rychlost, zrychlení, zpomalení nebo jerk není zadán, respektive má hodnotu 0, použije se hodnota nastavená v připojeném bloku **RM\_Axis**.

## RM\_Axis – Osa pro řízení pohybu

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok **RM\_AXIS** je základní blok osy pro řízení pohybu. Představuje sdílenou strukturu, kde jsou uloženy všechny stavy a parametry osy. Algoritmus tohoto bloku se stará o kontrolu nastavených mezí, havarijní zastavení v případě potřeby a přepočty všech stavů a výstupů pro případ, že žádný blok není aktivní, ale osa (motor) má nenulovou rychlost. Výstupem tohoto bloku jsou pouze požadované hodnoty polohy, rychlosti, popřípadě zrychlení a momentu. Pokud nejde o virtuální osu a osa je spojena s reálným motorem, je nutné realizovat ještě regulátor polohy a rychlosti, který není součástí tohoto bloku – zpětnovazební signály jsou použity pouze pro kontrolu správné činnosti regulátoru, například odchylka nesmí překročit stanovenou mez, viz dále. Dále je mohou využívat některé speciální bloky, například bloky pro nalezení výchozí polohy.

Parametry tohoto bloku jsou stejné, jaké vyžaduje **PLCopen** pro osu. Pokud jsou zadány parametry nesprávně (nekonzistentně), výstup **errorID** je nastaven na hodnotu -700 (neplatný parametr) a všechny ostatní bloky navázané na osu skončí s chybou -720 (obecná chyba).

Implicitní hodnoty parametrů (zejména limity na rychlost a zrychlení) jsou záměrně nastaveny na 0, což je nedovolená hodnota. Všechny parametry tak musí nastavit uživatel podle skutečných možností připojeného motoru a stroje.

### Vstupy

HLD	Pozastavení	bool
	off ... pohyb je povolen	
	on .... osa je zablokována a žádný pohyb není možný	
ActualPos	Skutečná poloha osy	double
ActualVelocity	Skutečná rychlost osy	double
ActualTorque	Skutečný moment/síla osy	double

LIMN	Koncový spínač v záporném směru	bool
LIMZ	Koncový spínač pro nulovou polohu	bool
LIMP	Koncový spínač v kladném směru	bool

## Výstupy

axisRef	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	reference
CommandedPosition	Požadovaná poloha osy	double
CommandedVelocity	Požadovaná rychlost osy	double
CommandedAcceleration	Požadované zrychlení osy	double
CommandedTorque	Požadovaný moment/síla	double
iState	Stav osy pro řízení pohybu	long
	0 ..... Disabled (osa blokována)	
	1 ..... Stand still (osa připravena)	
	2 ..... Homing (hledání výchozí polohy)	
	3 ..... Discrete motion (jednorázový pohyb)	
	4 ..... Continuous motion (trvalý pohyb)	
	5 ..... Synchronized motion (pohyb synchronní s jinou osou)	
	6 ..... Coordinated motion (osa řízena blokem pro koordinovaný pohyb)	
	7 ..... Stopping (zastavování nebo dočasné blokování)	
	8 ..... Error stop (zastavování nebo blokování osy po chybě)	
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	

## Parametry

AxisType	Druh osy	⊙1	long
	1 ..... lineární osa		
	2 ..... cyklická osa s cyklickým polohovým senzorem		
	3 ..... cyklická osa s lineárním polohovým senzorem		
EnableLimitPos	Příznak kontroly maximální polohy v kladném směru		bool
SWLimitPos	Maximální hodnota v kladném směru (pro kontrolu)		double
MaxPosSystem	Maximální hodnota v kladném směru (fyzický doraz)		double
EnableLimitNeg	Příznak kontroly maximální polohy v záporném směru		bool
SWLimitNeg	Maximální hodnota v záporném směru (pro kontrolu)		double
MinPosSystem	Maximální hodnota v záporném směru (fyzický doraz)		double
EnablePosLagMonitor	Příznak kontroly skluzu (rozdílu mezi požadovanou a skutečnou polohou)		bool
MaxPositionLag	Maximální hodnota skluzu		double
MaxVelocitySystem	Maximální hodnota rychlosti (havarijní)		double
MaxVelocityAppl	Maximální hodnota rychlosti (pracovní)		double
MaxAccelerationSystem	Maximální hodnota zrychlení (havarijní)		double
MaxAccelerationAppl	Maximální hodnota zrychlení (pracovní)		double
MaxDecelerationSystem	Maximální hodnota brždění (havarijní)		double

MaxDecelerationAppl

MaxJerk

MaxTorque

kta

ReverseLimit

Maximální hodnota brždění (pracovní)

Maximální povolená změna zrychlení [unit/s<sup>3</sup>]

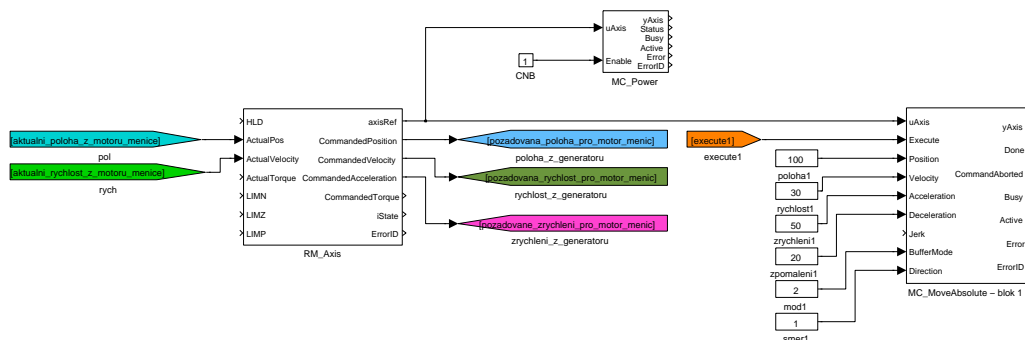
Maximální hodnota momentu/síly

Poměr momentu a zrychlení. Pokud moment osy není určen jiným způsobem (např. pomocí bloku [MC\\_TorqueControl](#)), určuje moment ze zrychlení pomocí tohoto koeficientu.

Negace významu logických vstupů LIMN, LIMZ a LIMP

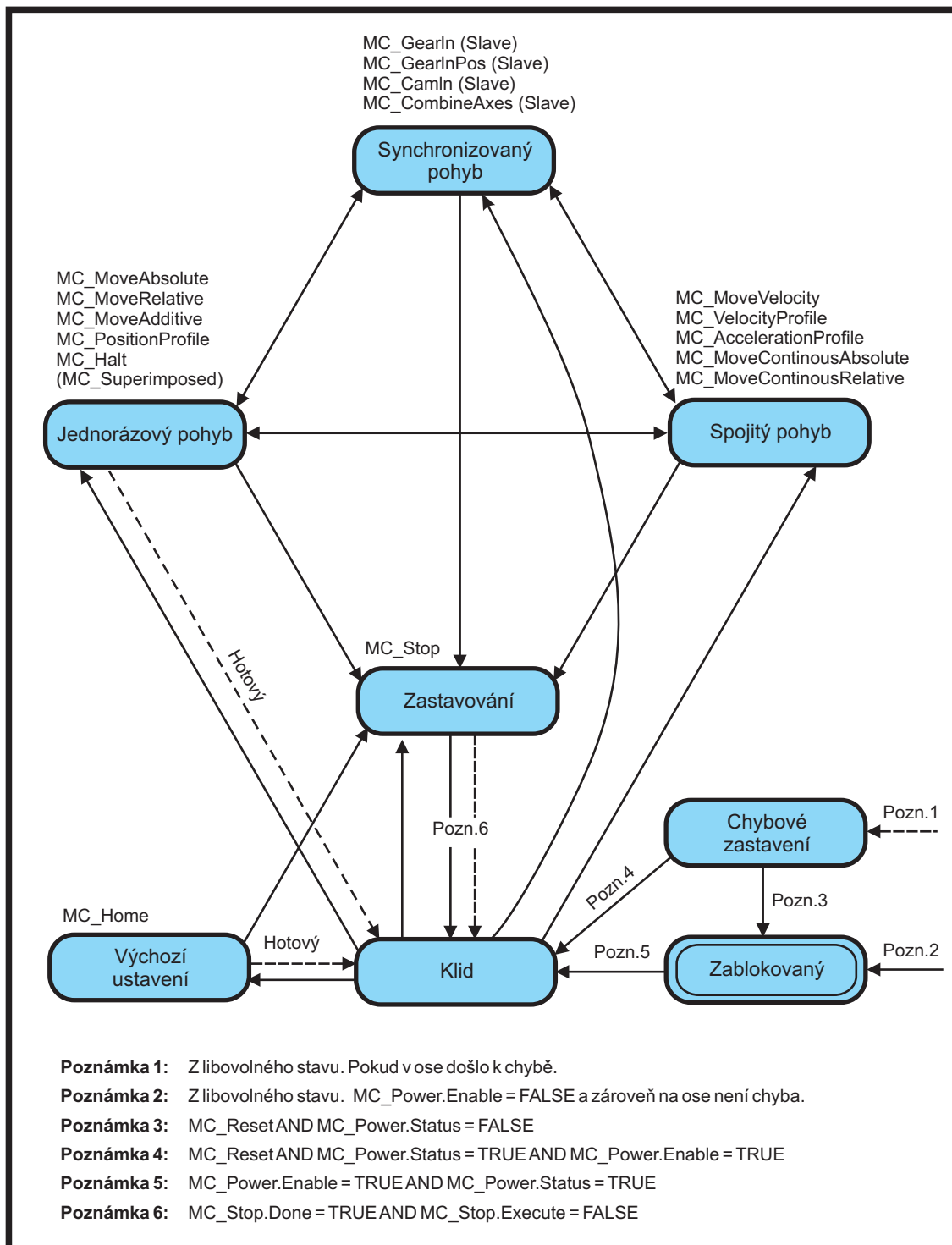
## Příklad

Následující schéma znázorňuje základní zapojení bloků z knihovny Motion Control v minimální realizaci nutné pro zahájení pohybu. Fyzická osa je reprezentována blokem **RM\_Axis** a jí příslušnou datovou strukturou. Po nastavení omezení na požadovaný průběh pohybu lze na vstupy bloku připojit signál aktuální polohy, rychlosti a momentu (využíváno pro kontrolu skluzu) popřípadě logické signály koncových spínačů pro nalezení referenční výchozí polohy. Výstupní signál **axisRef** se připojuje na vstup všech bloků pracujících s danou osou. Před zahájením pohybu je třeba osu aktivovat blokem **MC\_Power**, čímž dojde k přechodu ze zablokovaného do klidového stavu. V tuto chvíli je již možné zahájit libovolný jednorázový, spojitý nebo synchronizovaný pohyb spuštěním příslušného funkčního bloku. Generovaná trajektorie pohybu, tedy požadovaný průběh polohy, rychlosti a zrychlení je k dispozici na výstupech bloku **RM\_Axis**. Odtud může být přiveden do regulační struktury pohonu, která existuje buďto lokálně v systému REX ve stejném nebo jiném tasku nebo je implementována přímo v řídicí jednotce motoru (zesilovač, frekvenční měnič), kam může být ze systému REX předána s použitím některé ze standardních sériových komunikací. V případě zadání chybných parametrů, volání nepřipustného příkazu nebo při překročení maximální hodnoty skluzu dochází k nouzovému zastavení, osa přechází do režimu chyby, kterou indikuje na výstupu **ErrorID**. Pro opětovné zahájení pohybu je třeba chybu kvitovat blokem **MC\_Reset**. Přechod mezi jednotlivými režimy osy je znázorněn na stavovém diagramu.





## Stavový diagram osy

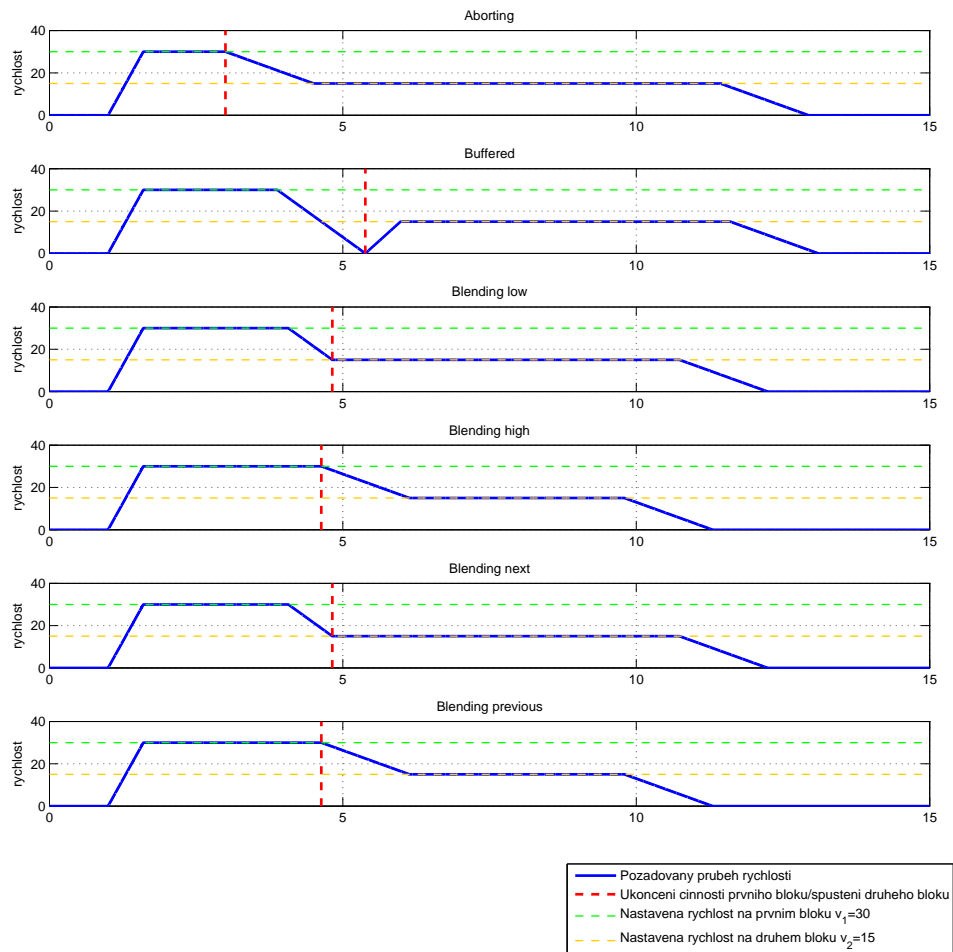


## Míchání pohybů

Norma PLCOpen definuje pro bloky spouštějící pohyb osy vstupní parametr **BufferMode**, který určuje chování osy v případě, že je za běhu zavolán nový příkaz ve formě jiného funkčního bloku. Tento přechod mezi různými pohyby (míchání, "Blending") lze řešit několika způsoby. Následující tabulka podává stručné vysvětlení funkce jednotlivých režimů míchání pohybu a jejich vliv na tvar výsledné generované trajektorie. Detailní popis lze nalézt ve specifikaci PLCOpen.

<b>Aborting</b>	nový pohyb okamžitě přerušuje původní příkaz
<b>Buffered</b>	nový pohyb je vykonán po dokončení původního příkazu, neprobíhá žádné míchání
<b>Blending low</b>	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí určenou jako nejnižší hodnota ze dvou limitů pro maximální rychlost zadanou na vstupu dvou bloků, které přebírají řízení osy
<b>Blending high</b>	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí určenou jako nejvyšší hodnota ze dvou limitů pro maximální rychlost zadanou na vstupu dvou bloků, které přebírají řízení osy
<b>Blending previous</b>	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí danou limitem pro maximální rychlost na vstupu prvního bloku
<b>Blending next</b>	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí danou limitem pro maximální rychlost na vstupu druhého bloku

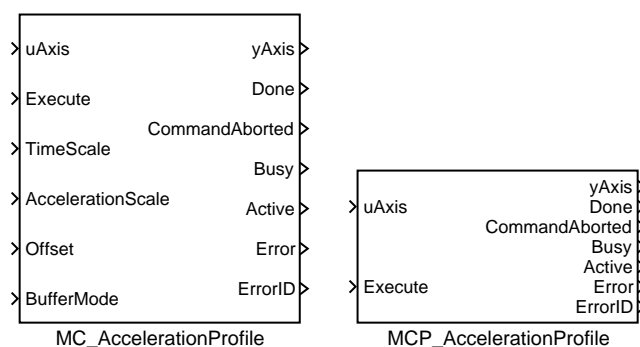
# Ilustrace míchání pohybů



## MC\_AccelerationProfile, MCP\_AccelerationProfile – Generování trajektorie (zrychlení)

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_AccelerationProfile a MCP\_AccelerationProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

### Popis funkce

Blok **MC\_AccelerationProfile** generuje takovou trajektorii, aby zrychlení byla požadovaná funkce času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a zrychlení. Mezi jednotlivými časy se hodnota zrychlení interpoluje lineárně. Hodnoty času (v sekundách) se zadávají do pole/parametru **times**, příslušné hodnoty zrychlení do pole/parametru **values**. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli **times**. Polynom pro každý interval je ve tvaru  $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ , přičemž na začátku časového intervalu je  $x = 0$ , a na konci  $x = 1$ . Koefficienty  $a_i$  jsou uloženy v poli **values** ve vzestupném pořadí (tj. pole **values** obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli **times** jen počáteční (obvykle 0) a koncový čas.

Poznámka 1: Přestože vstup/odkaz `TimePosition` je v `PLCopen` označen jako povinný, není zde použit, protože všechna potřebná data jsou uložena v parametrech bloku.

Poznámka 2: Parametr `values` musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru nebo čárku).

Poznámka 3: Nesprávný parametr `cSeg` (větší než skutečná velikost polí `times` a/nebo `values`) vede k nedefinovanému chování, v krajním případě nestandardnímu ukončení aplikace (závisí to na mnoha okolnostech, použití tohoto bloku v `SIMULINKu` vede k pádu `MATLABu`).

Poznámka 4: V režimu zadání funkce polynomem je hodnota polynomu poloha a polynom je vždy pátého řádu a nelze to nijak změnit. `AccelerationScale` a `Offset` je samozřejmě pro zrychlení. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka 5: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací `PLCopen`).

## Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<code>reference</code>
<code>Execute</code>	Náběžná hrana aktivuje blok	<code>bool</code>
<code>TimeScale</code>	Konstanta násobení pro přepočet časové osy profilu	<code>double</code>
<code>AccelerationScale</code>	Konstanta násobení pro přepočet hodnotové osy profilu	<code>double</code>
<code>Offset</code>	Aditivní konstanta pro přepočet hodnotové osy profilu	<code>double</code>
<code>BufferMode</code>	Režim převzetí osy	<code>long</code>
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

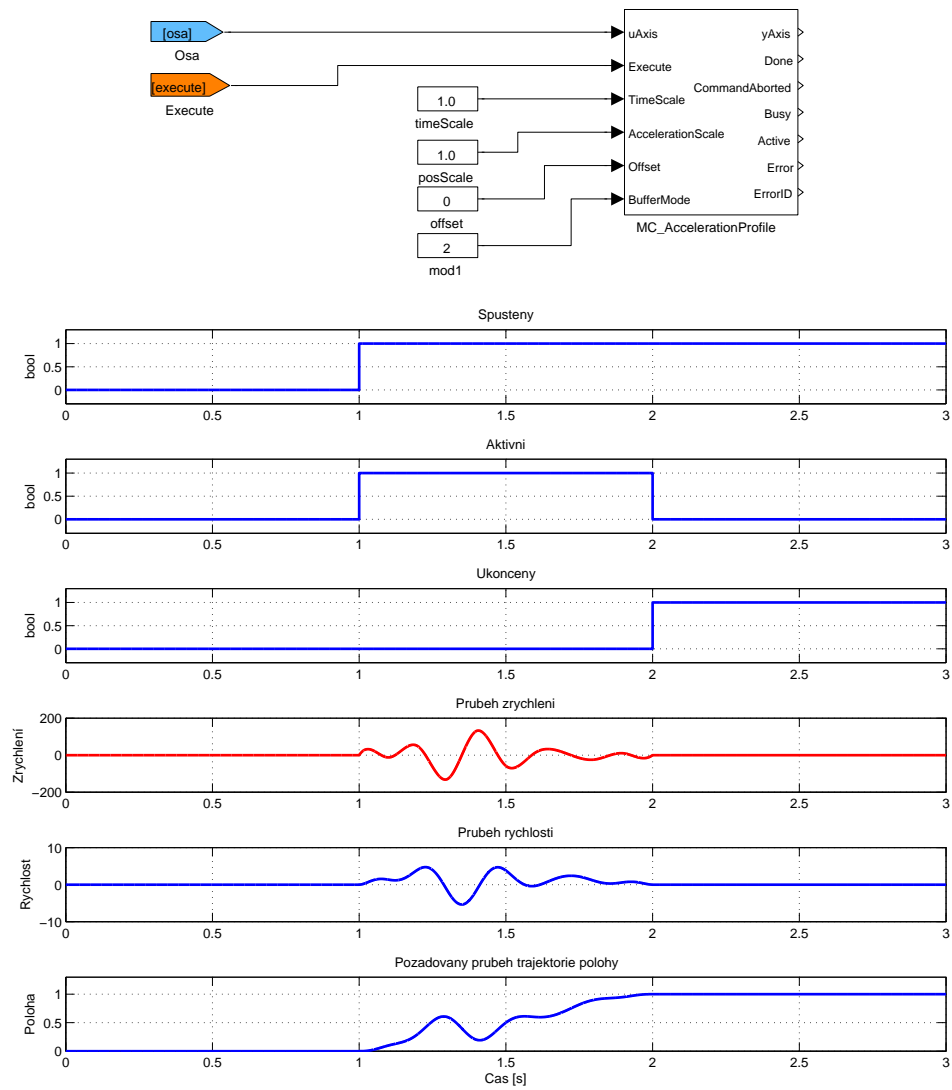
<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<code>reference</code>
<code>Done</code>	Příznak dokončení algoritmu	<code>bool</code>
<code>CommandAborted</code>	Příznak přerušení funkce bloku	<code>bool</code>
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	<code>bool</code>

<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i . . . . . obecná chyba systému REX	

## Parametry

<b>alg</b>	Typ interpolace	⊙2	<b>long</b>
	1 . . . . . tabulka čas/hodnota		
	2 . . . . . hodnoty ve stejném intervalu		
	3 . . . . . aproximace polynomy		
	4 . . . . . polynomy s ekvidistantními intervaly		
<b>cSeg</b>	Počet segmentů profilu	⊙3	<b>long</b>
<b>times</b>	Posloupnost hraničních časů jednotlivých segmentů	⊙[0 30]	<b>double</b>
<b>values</b>	Hodnoty veličiny nebo koeficienty interpolačních polynomů (a0, a1, a2, ...)	⊙[0 100 100 50]	<b>double</b>

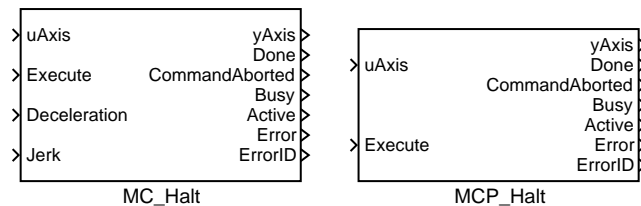
## Příklad



## MC\_Halt, MCP\_Halt – Zastavení pohybu (přerušitelné)

Symbole bloků

Licence: [MOTION CONTROL](#)



## Popis funkce

*Bloky MC\_Halt a MCP\_Halt mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_Halt zahajuje řízené zastavení pohybu. Osa se přesune do stavu DiscreteMotion, dokud není rychlost nulová. Společně s nastavením výstupu "Done" je stav změněn na "Standstill".

Poznámka 1: Blok MC\_Halt se používá k zastavení osy za normálních provozních podmínek. V non-buffered režimu je možné zadat další pohybový příkaz při zpomalení osy, který zruší MC\_Halt a bude ihned proveden.

Poznámka 2: Je-li tento příkaz aktivní, další příkaz může být aktivován (spuštěn). Např. vozidlo bez řidiče detekuje překážku a potřebuje zastavit. MC\_Halt je aktivován. Před dosažením stavu "Standstill" je překážka odstraněna a pohyb může pokračovat nastavením dalšího pohybového příkazu, aby vozidlo nemuselo zastavit.

## Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	reference
Execute	Náběžná hrana aktivuje blok	bool
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double

## Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	reference
Done	Příznak dokončení algoritmu	bool
CommandAborted	Příznak přerušení funkce bloku	bool

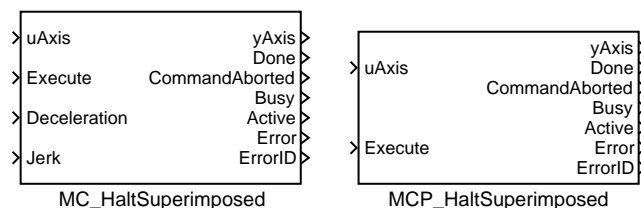


<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i ..... obecná chyba systému REX	

## MC\_HaltSuperimposed, MCP\_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné)

Symbody bloků

Licence: [MOTION CONTROL](#)



## Popis funkce

*Bloky MC\_HaltSuperimposed a MCP\_HaltSuperimposed mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_HaltSuperimposed zahajuje řízené zastavení přídavného pohybu, který generuje například blok [MC\\_MoveSuperimposed](#).

## Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	reference
Execute	Náběžná hrana aktivuje blok	bool
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
JerK	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double

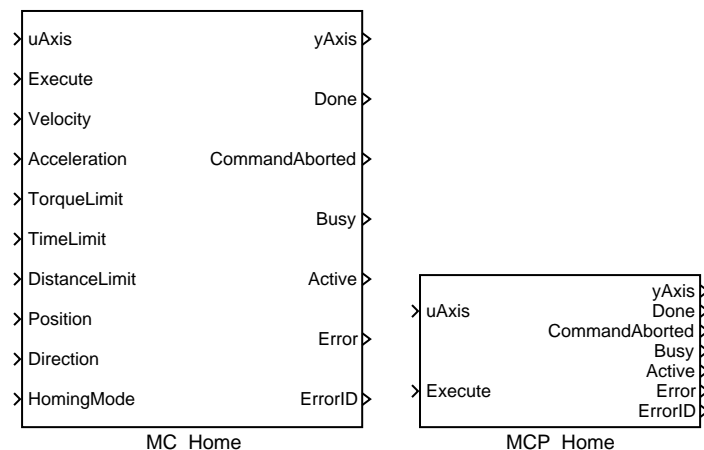
## Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	reference
Done	Příznak dokončení algoritmu	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	

## MC\_Home, MCP\_Home – Nalezení výchozí polohy

### Symbyly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_Home a MCP\_Home mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_Home provede algoritmus hledání výchozího bodu. Detaily závisí na parametrech bloku (zejména **HomingMode**). Podrobnější popis lze nalézt v **PLCopen**. Vstup **Position** je poloha, která je nastavena v okamžiku dosažení výchozí polohy. Po úspěšném ukončení algoritmu tohoto bloku je osa ve stavu „StandStill“.

Poznámka 1: Parametr či vstup **BufferMode** není podpořen - vždy je **BufferMode = Aborting**. To je nevýznamné omezení, protože tento blok se vykonává vždy jako první.

Poznámka 2: Tento blok vyžaduje připojené vstupy bloku **RM\_Axis**. Podle zvolené metody může být potřeba **ActualPos**, **ActualTorque**, **LimP**, **LimZ**, **LimN**. Předpokládá se, že metoda je zvolena dopředu podle konstrukce stroje, proto není zvlášť vstup na nulový koncový spínač a referenční pulz - jeden z nich se připojí k **LimZ** a zvolí se **HomingMode=3**(nulový pulz).

Poznámka 3: **HomingMode=4**(uživatelská hodnota) pouze nastaví aktuální polohu. Proto také není implementován blok MC\_SetPosition. **HomingMode=5** (absolutní snímač polohy) pouze přepne osu do stavu **StandStill**.

Poznámka 4: Tento blok nepodporuje jerk (limit na derivaci zrychlení) a také není použit zvlášť limit na zrychlení a brždění. Pokud je potřeba nastavit výchozí polohu

velmi přesně, doporučuje se spustit blok MC\_Home dvakrát. Poprvé s velkou rychlostí pro najetí blízko výchozí polohy a podruhé s malou rychlostí pro přesné nastavení pozice.

Poznámka 5: Pro HomingMode=6 (najíetí na mechanickou překážku) se požadovaná poloha pozná podle toho, že moment překročil nastavenou mez (tj. v tomto režimu to není chyba procedury) nebo že "position lag" překročil nastavenou mez (druhá podmínka se kontroluje jen pokud parametr MaxPositionLag v příslušném bloku RM\_Axis je kladný).

## Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	reference
Execute	Náběžná hrana aktivuje blok	bool
Velocity	Maximální povolená rychlost [unit/s]	double
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
TorqueLimit	Maximální povolený moment/síla	double
TimeLimit	Maximální povolený čas pro celý algoritmus bloku [s]	double
DistanceLimit	Maximální povolená vzdálenost pro celý algoritmus bloku [unit]	double
Position	Požadovaná poloha [unit]	double
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	long
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	
HomingMode	Algoritmus hledání výchozí pozice	long
	1 ..... nulový spínač	
	2 ..... koncové spínače	
	3 ..... nulový pulz	
	4 ..... uživatelská hodnota	
	5 ..... absolutní snímač polohy	
	6 ..... mechanický doraz	

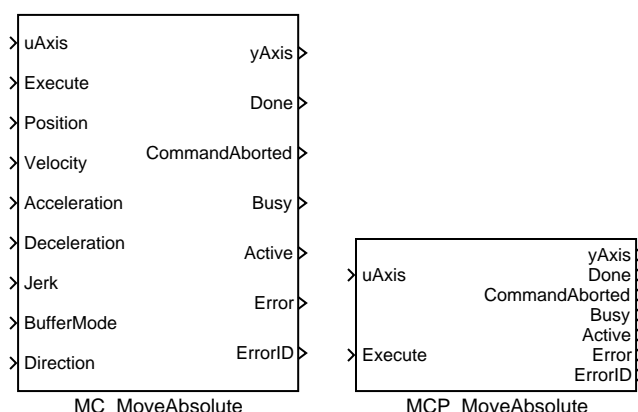
## Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	reference
Done	Příznak dokončení algoritmu	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	

## MC\_MoveAbsolute, MCP\_MoveAbsolute – Pohyb do pozice (absolutní souřadnice)

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveAbsolute a MCP\_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveAbsolute** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby blending módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.

5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

## Vstupy

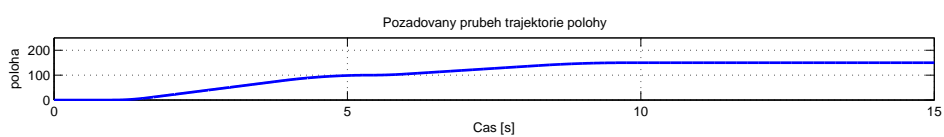
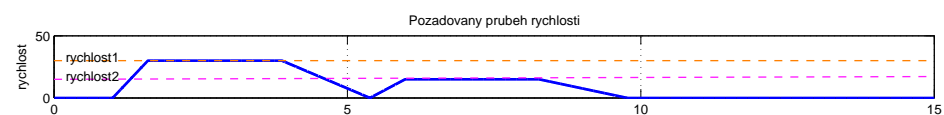
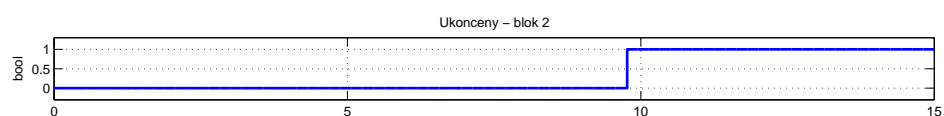
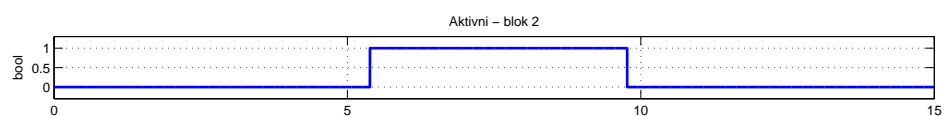
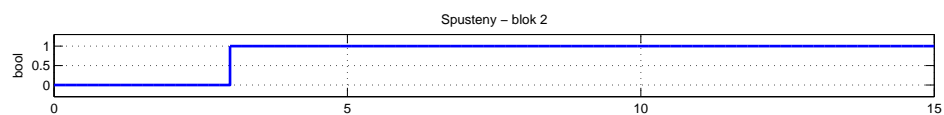
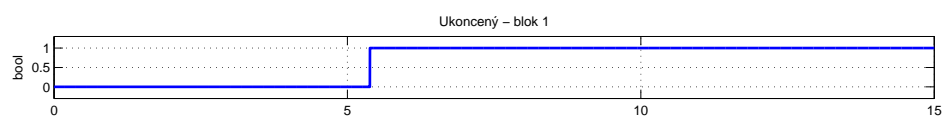
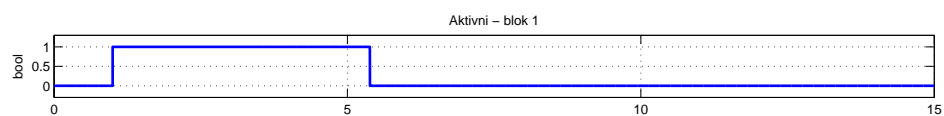
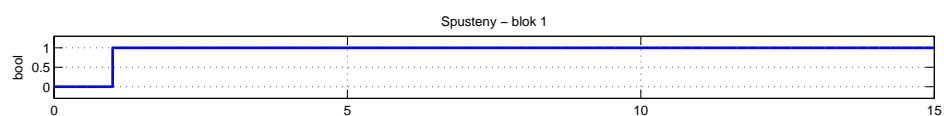
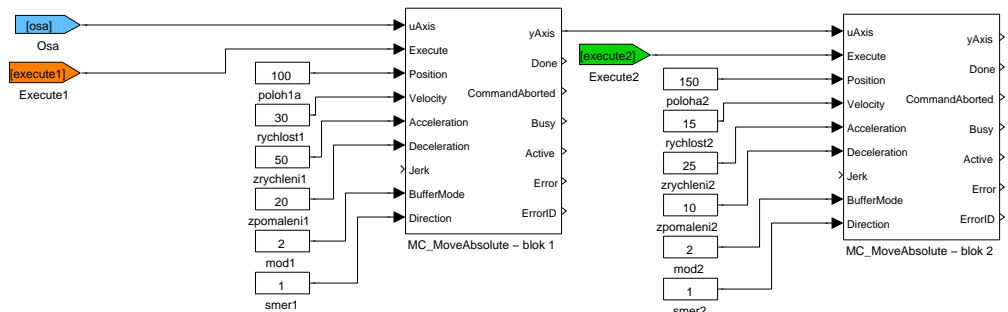
<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>Position</b>	Požadovaná poloha [unit]	<b>double</b>
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	<b>double</b>
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	<b>double</b>
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>Direction</b>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	<b>long</b>
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>

ErrorID	Kód chyby	error
i .....	obecná chyba systému REX	

## Příklad

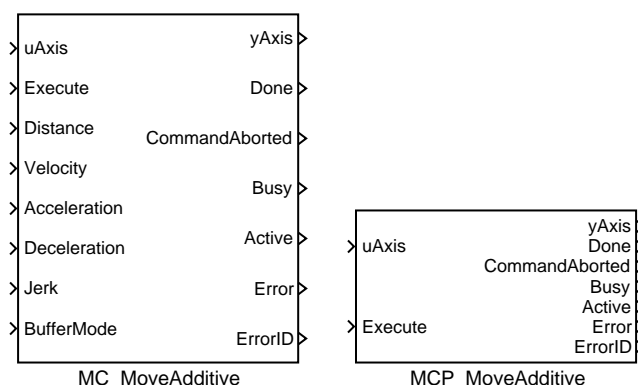




## MC\_MoveAdditive, MCP\_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbyly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveAbsolute a MCP\_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveAdditive** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku převzetí řízení osy přičte hodnota parametru **Distance**. Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mód buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.

5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

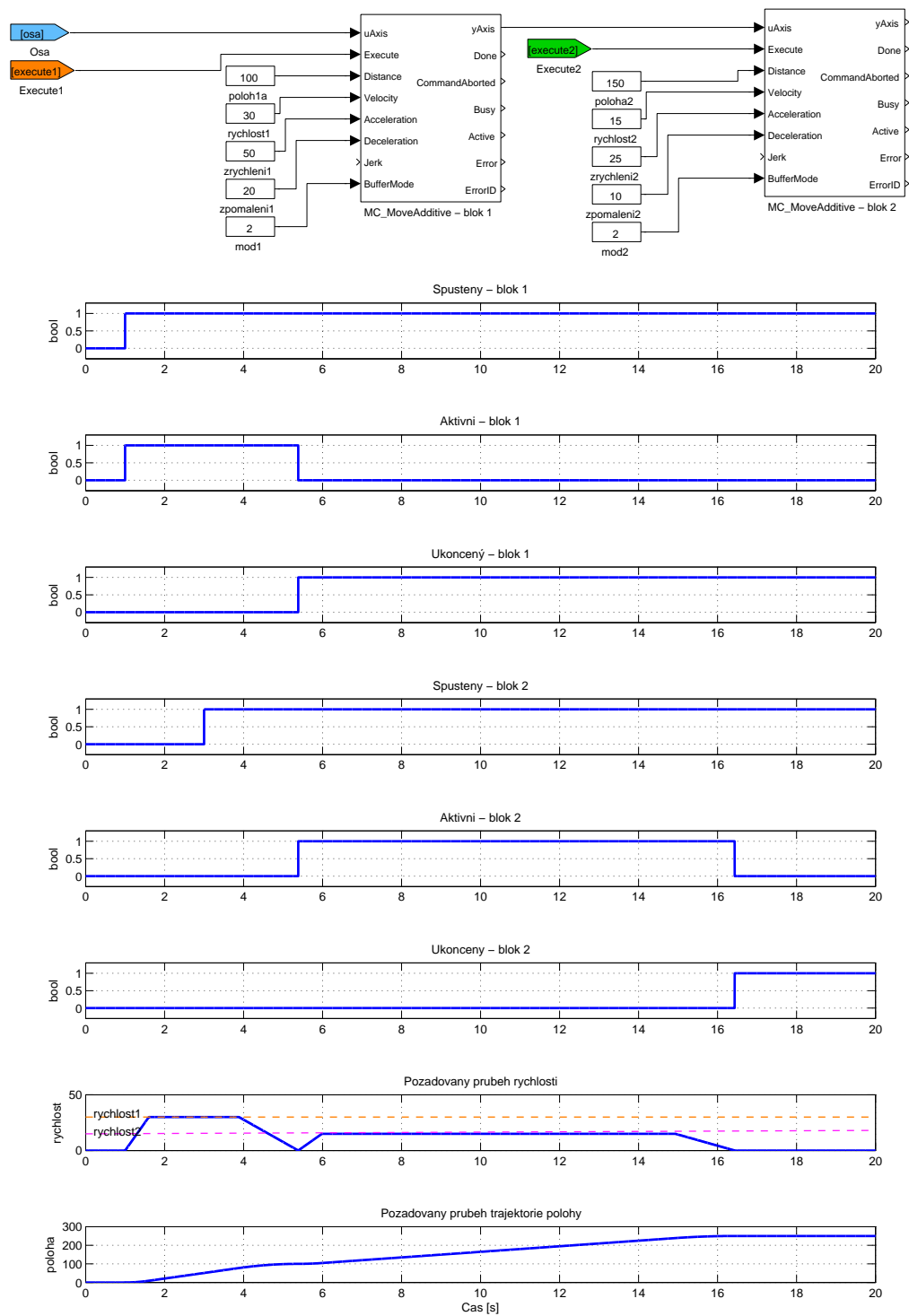
## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>Distance</b>	Požadovaná vzdálenost (od okamžiku začátku řízení osy blokem) [unit]	<b>double</b>
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	<b>double</b>
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	<b>double</b>
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i ..... obecná chyba systému REX	

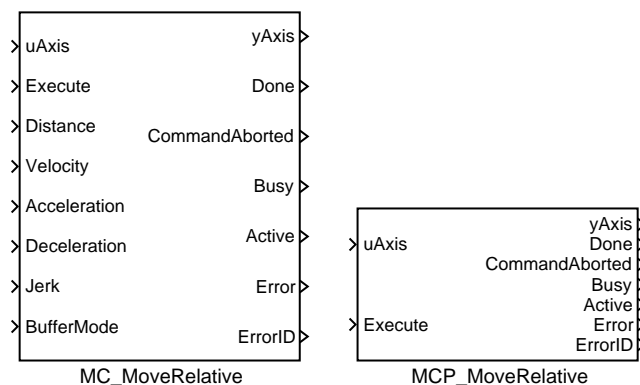
## Příklad



## MC\_MoveRelative, MCP\_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění)

Symbyly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveRelative a MCP\_MoveRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveRelative** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu **Execute**) přičte hodnota parametru **Distance**. Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde

v cílové poloze a s požadovanou rychlostí.

5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

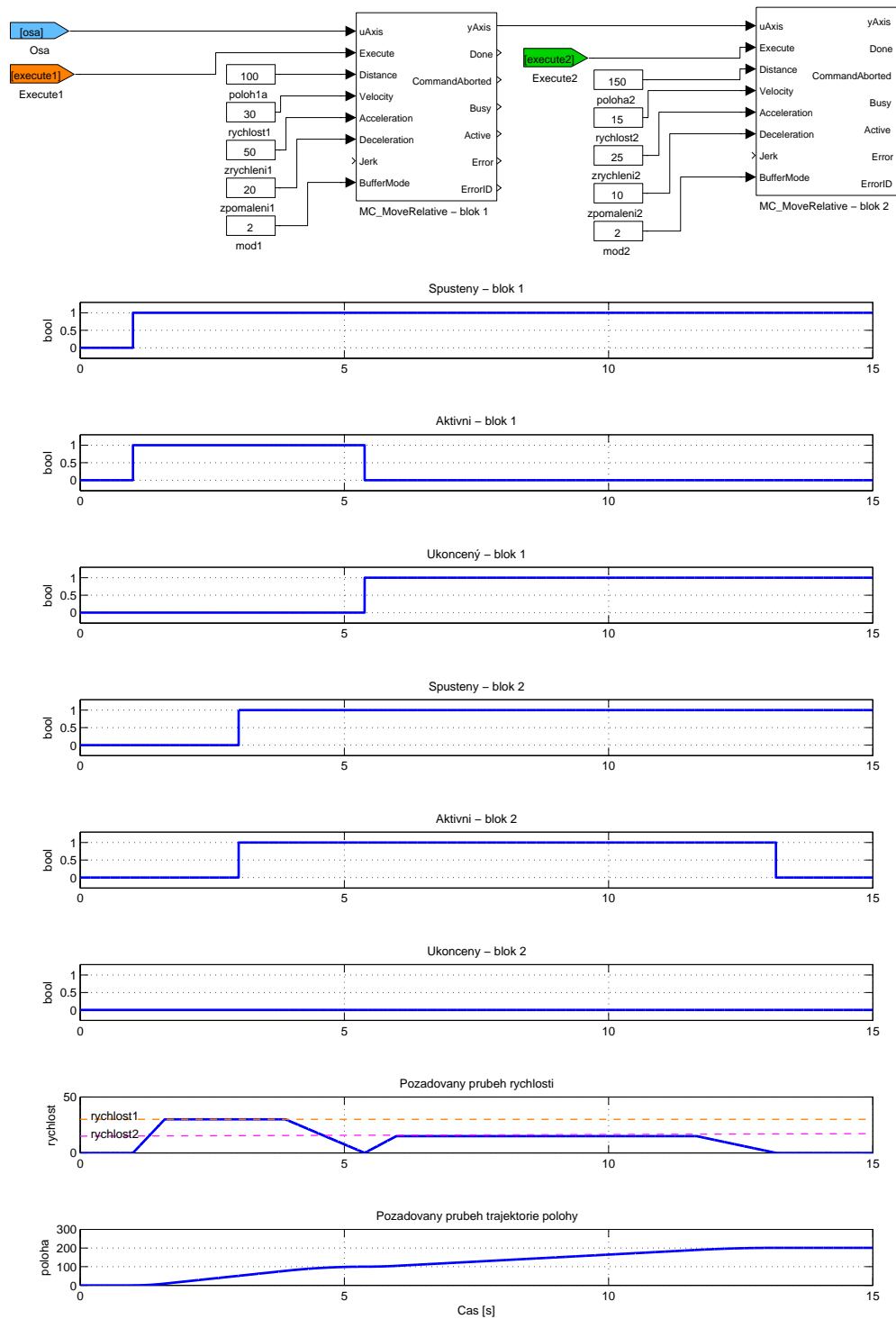
## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>Distance</b>	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	<b>double</b>
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	<b>double</b>
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	<b>double</b>
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i ..... obecná chyba systému REX	

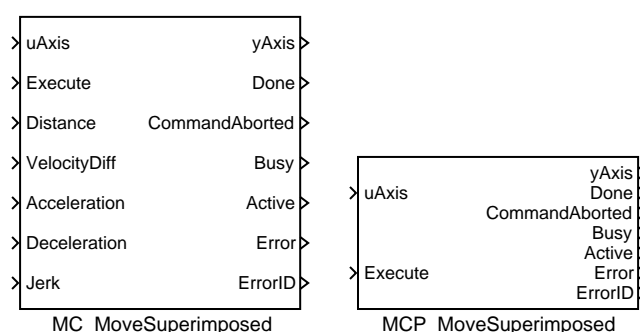
## Příklad



## MC\_MoveSuperimposed, MCP\_MoveSuperimposed – Pohyb do pozice (přídavný pohyb)

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveSuperimposed a MCP\_MoveSuperimposed mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveSuperimposed** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha je hodnota parametru **Distance**, přičemž počáteční poloha se považuje za nulovou. Pokud již nějaký blok běží, původní blok běží dále a hodnoty (poloha, rychlost, zrychlení) superimposed bloku se přičítají k hodnotám generovaným původním blokem. Pokud žádný blok neběží, tento blok se chová stejně jako [MC\\_MoveRelative](#).

Poznámka: Tento blok nemá parametr **BufferMode**, protože v superimposed režimu je to irrelevantní. Pokud v okamžiku spuštění (náběžná hrana na vstupu **Execute**) je již nějaký blok v režimu superimposed aktivní, dojde k okamžitému spuštění nového bloku (tj. analogicky režimu aborting).

### Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
Execute	Náběžná hrana aktivuje blok	bool
Distance	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	double
VelocityDiff	Maximální povolená rychlost [unit/s]	double
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double

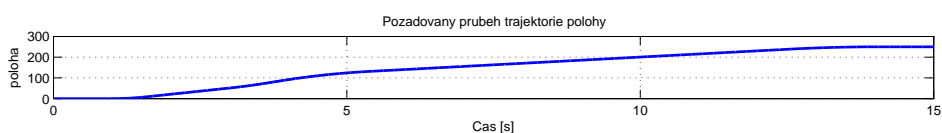
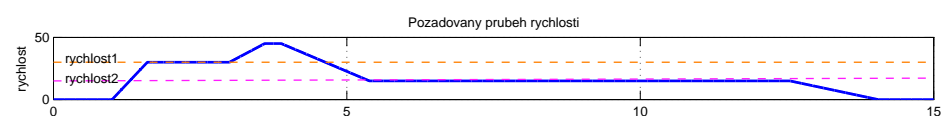
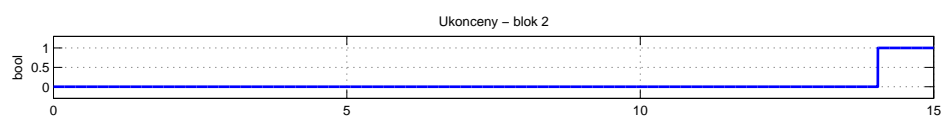
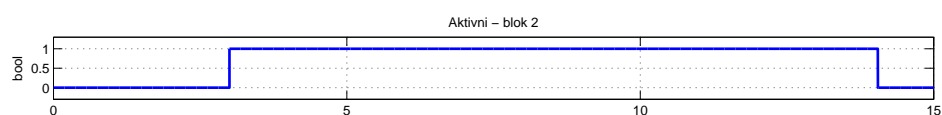
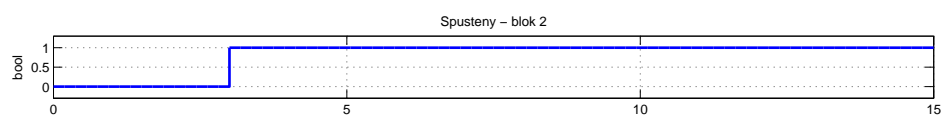
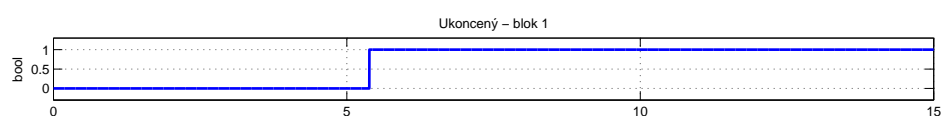
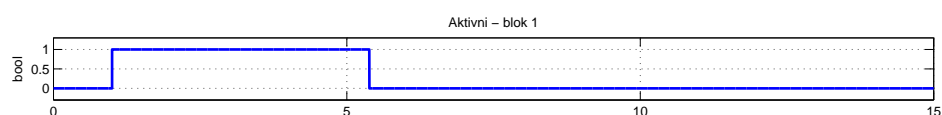
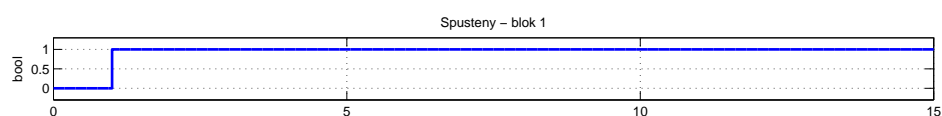
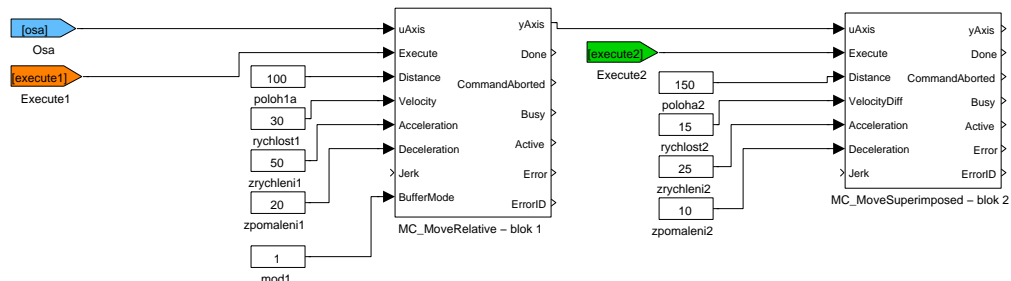
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	<b>double</b>
-------------	---	---------------

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby i ..... obecná chyba systému REX	<b>error</b>



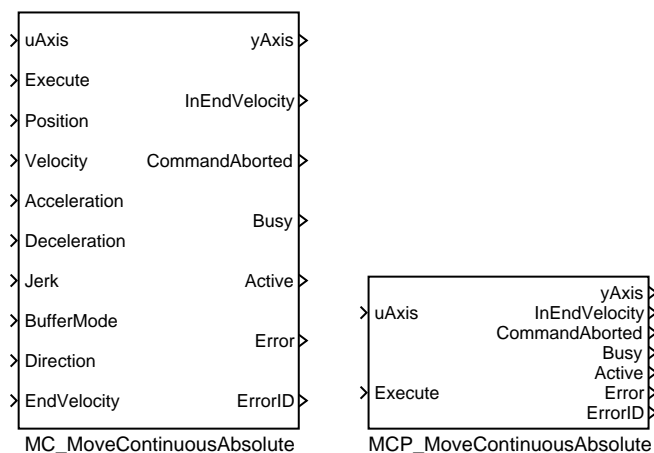
## Příklad



## MC\_MoveContinuousAbsolute, MCP\_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice)

Symbyl bloků

Licence: [MOTION CONTROL](#)



## Popis funkce

*Bloky MC\_MoveAbsolute a MCP\_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveContinuousAbsolute** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Pokud není aktivován další blok, koncová rychlost zůstává konstantní na hodnotě parametru **EndVelocity**, osa se tedy nezastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby blending módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti.

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde

v cílové poloze a s požadovanou rychlostí.

5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

Poznámka 1: Jestliže je nastaven parametr **EndVelocity** na nulovou hodnotu, pak se chová blok stejným způsobem jako **MC\_MoveAbsolute**.

Poznámka 2: Pokud dojde ke spuštění dalšího bloku dříve, než je dosaženo požadované polohy, blok se opět chová stejným způsobem jako **MC\_MoveAbsolute**.

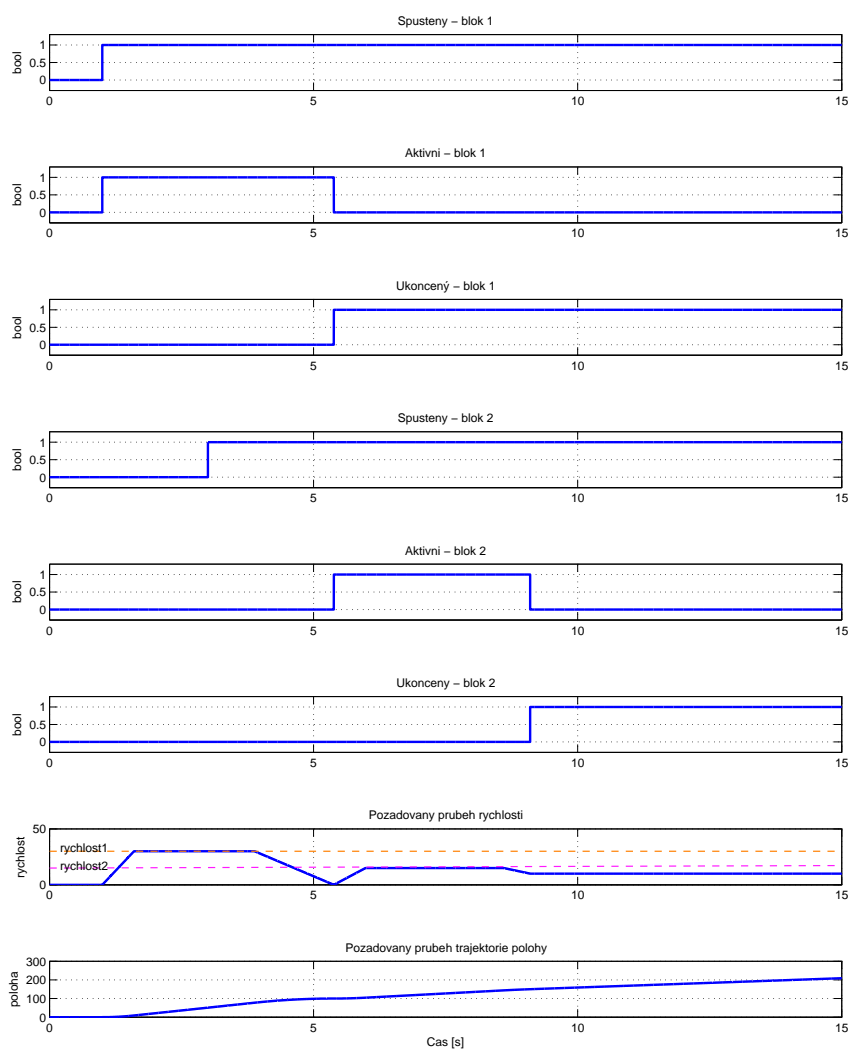
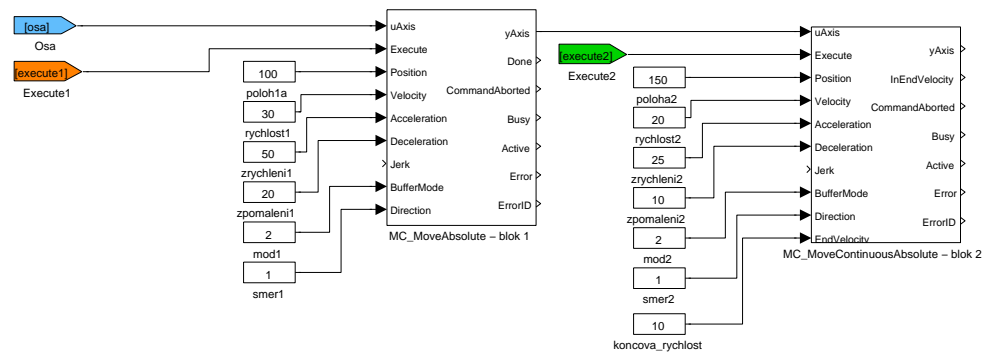
## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>Position</b>	Požadovaná poloha [unit]	double
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	double
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
<b>BufferMode</b>	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>Direction</b>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	long
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	
<b>EndVelocity</b>	Koncová rychlost	double

## Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	reference
InEndVelocity	Příznak dokončení algoritmu	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	

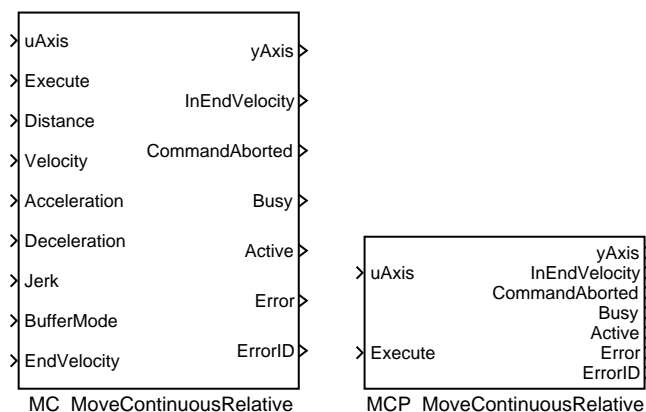
## Příklad



## MC\_MoveContinuousRelative, MCP\_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbole bloků

Licence: [MOTION CONTROL](#)



## Popis funkce

*Bloky MC\_MoveContinuousRelative a MCP\_MoveContinuousRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveContinuousRelative** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu **Execute**) přičte hodnota parametru **Distance**. Pokud není aktivován další blok, koncová rychlost zůstává konstantní na hodnotě parametru **EndVelocity**, osa se tedy nezastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.

3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.

4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.

5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

Poznámka 1: Jestliže je nastaven parametr `EndVelocity` na nulovou hodnotu, pak se chová blok stejným způsobem jako `MC_MoveRelative`.

Poznámka 2: Pokud dojde ke spuštění dalšího bloku dříve, než je dosaženo požadované polohy, blok se opět chová stejným způsobem jako `MC_MoveRelative`.

## Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<code>Execute</code>	Náběžná hrana aktivuje blok	bool
<code>Distance</code>	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	double
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	double
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
<code>Deceleration</code>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
<code>Jerk</code>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
<code>BufferMode</code>	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<code>EndVelocity</code>	Koncová rychlost	long

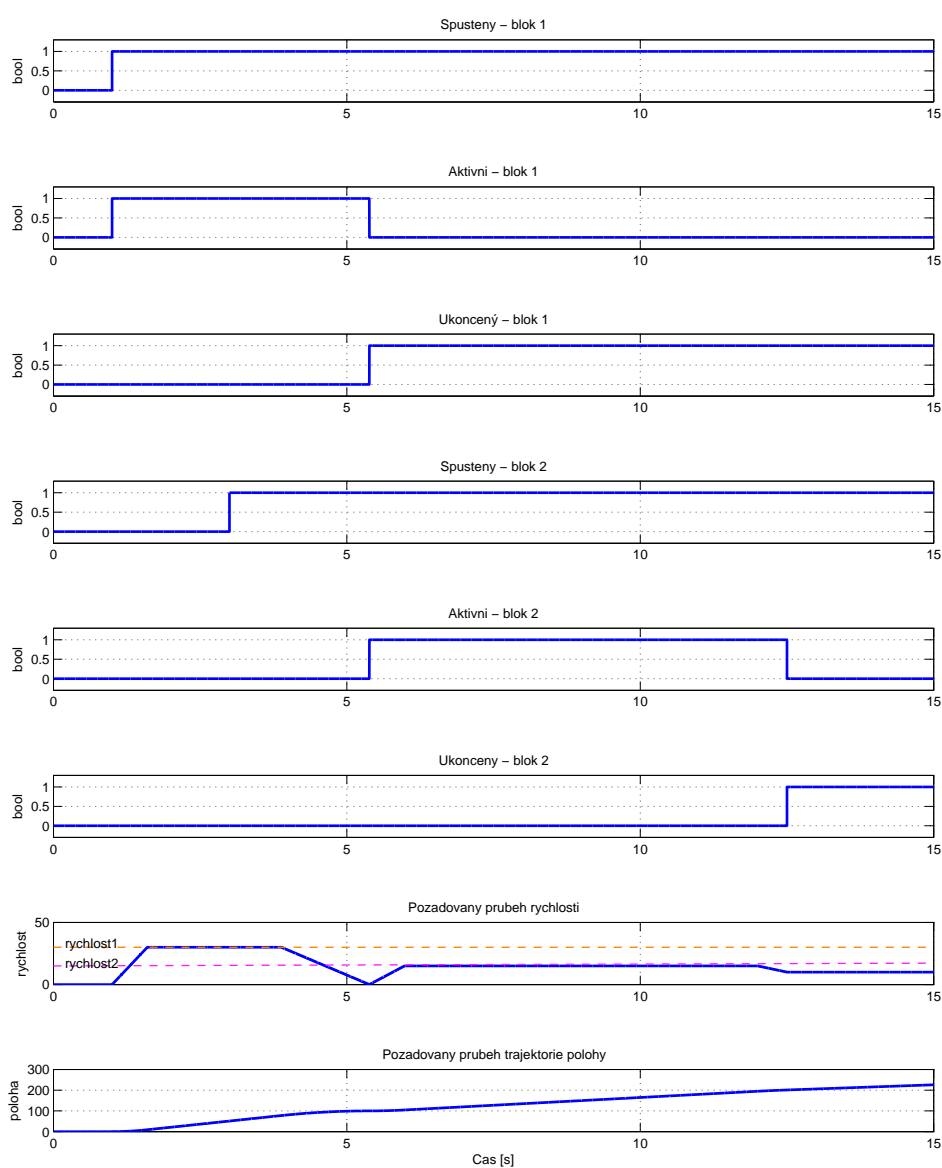
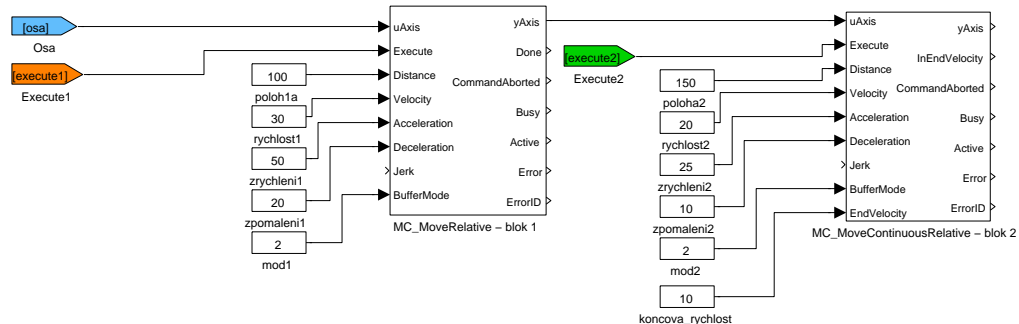
## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<code>InEndVelocity</code>	Příznak dokončení algoritmu	bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	bool

Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i . . . . . obecná chyba systému REX	



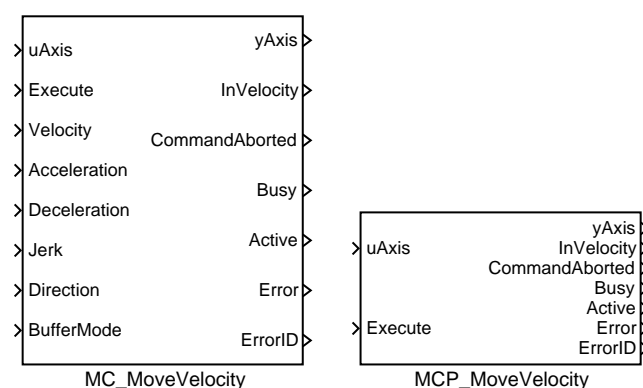
## Příklad



## MC\_MoveVelocity, MCP\_MoveVelocity – Pohyb konstantní rychlostí

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveVelocity a MCP\_MoveVelocity mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_MoveVelocity změní rychlost osy na požadovanou hodnotu za nejkratší možný čas s respektováním omezení na zrychlení a popřípadě jerk. Rychlost pak zůstává konstantní, dokud není aktivován jiný blok.

### Vstupy

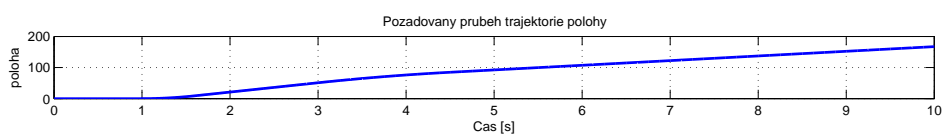
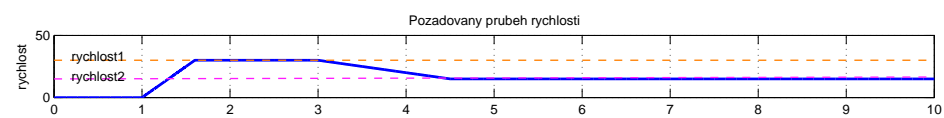
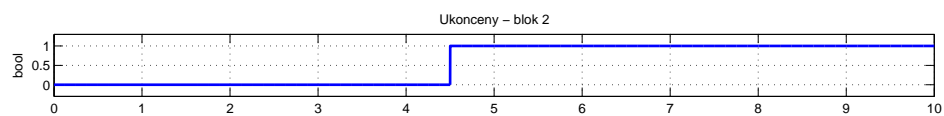
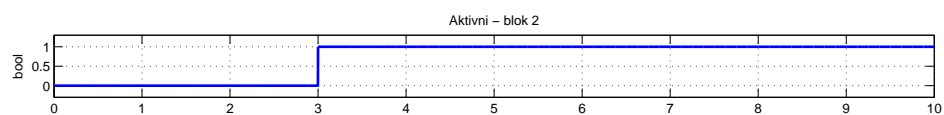
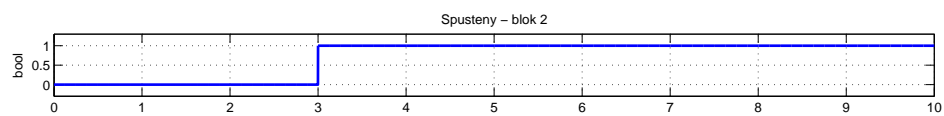
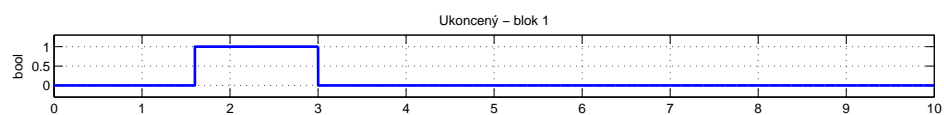
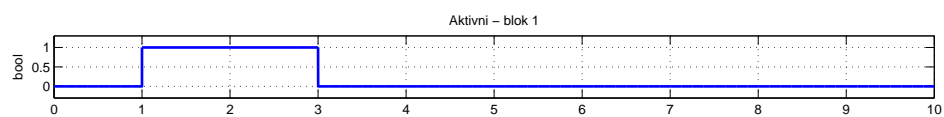
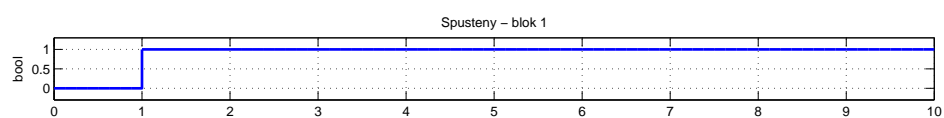
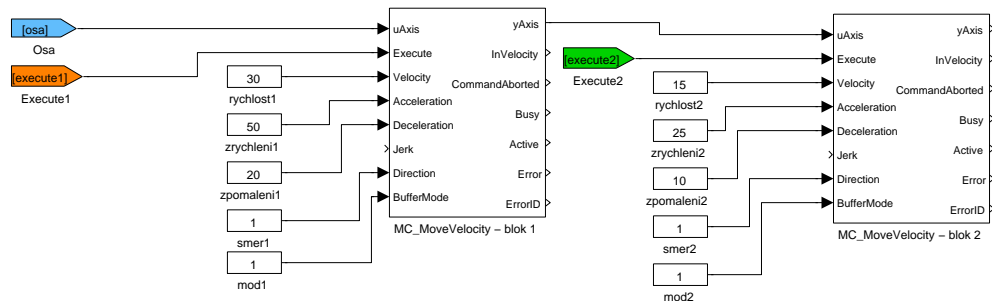
uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	reference
Execute	Náběžná hrana aktivuje blok	bool
Velocity	Maximální povolená rychlost [unit/s]	double
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	long
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	

<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>InVelocity</b>	Příznak dosažení požadované rychlosti	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i ..... obecná chyba systému REX	

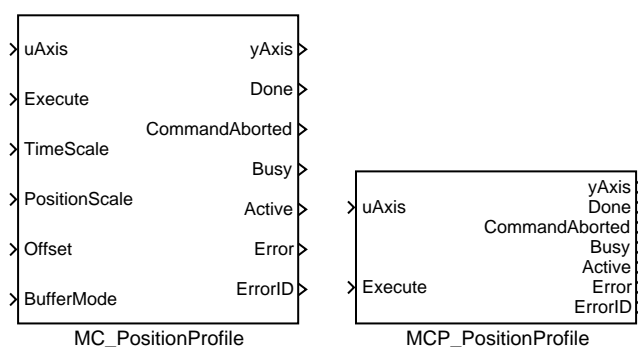
## Příklad



## MC\_PositionProfile, MCP\_PositionProfile – Generování trajektorie (poloha)

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_PositionProfile a MCP\_PositionProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_PositinProfile** generuje takovou trajektorii, aby poloha byla požadovaná funkce času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a poloha. Mezi jednotlivými časy se poloha interpoluje polynomem třetího řádu (lineární interpolace není v tomto případě vhodná, protože na okrajích intervalu by byl skok v rychlosti). Hodnoty času (v sekundách) se zadávají do pole/parametru **times**, příslušné hodnoty polohy do pole/parametru **values**. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli **times**. Polynom pro každý interval je ve tvaru  $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ , přičemž na začátku časového intervalu je  $x = 0$ , a na konci  $x = 1$ . Koeficienty  $a_i$  jsou uloženy v poli **values** ve vzestupném pořadí (tj. pole **values** obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli **times** jen počáteční (obvykle 0) a koncový čas.

Poznámka 1: Přestože vstup/odkaz **TimePosition** je v PLCopen označen jako po-

vinný, není zde použit, protože všechna potřebná data jsou uložena v parametrech bloku.

Poznámka 2: Parametr **values** musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru nebo čárku).

Poznámka 3: Nesprávný parametr **cSeg** (větší než skutečná velikost polí **times** a/nebo **values**) vede k nedefinovanému chování, v krajním případě nestandardnímu ukončení aplikace (závisí to na mnoha okolnostech, použití tohoto bloku v SIMULINKu vede k pádu MATLABu).

Poznámka 4: V režimu zadání funkce polynomem je pátého řádu a nelze to nijak měnit. Polynomy na sebe musí hladce navazovat, jinak dochází ke skokům v rychlosti. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka 5: Block neobsahuje tzv. ramp-in mode. Pokud tedy rychlost nebo poloha osy v okamžiku spuštění profilu neodpovídá počáteční rychlosti a poloze profilu, blok skončí s chybou -707 (skok v rychlosti nebo poloze). Tomuto problému s rychlostí lze předejít, pokud se použije **BufferMode=BlendingNext**. Skok v poloze se musí řešit správně nastaveným parametrem **Offset**.

Poznámka 6: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací **PLCopen**).

## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>TimeScale</b>	Konstanta násobení pro přepočet časové osy profilu	<b>double</b>
<b>PositionScale</b>	Konstanta násobení pro přepočet hodnotové osy profilu	<b>double</b>
<b>Offset</b>	Aditivní konstanta pro přepočet hodnotové osy profilu	<b>double</b>
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

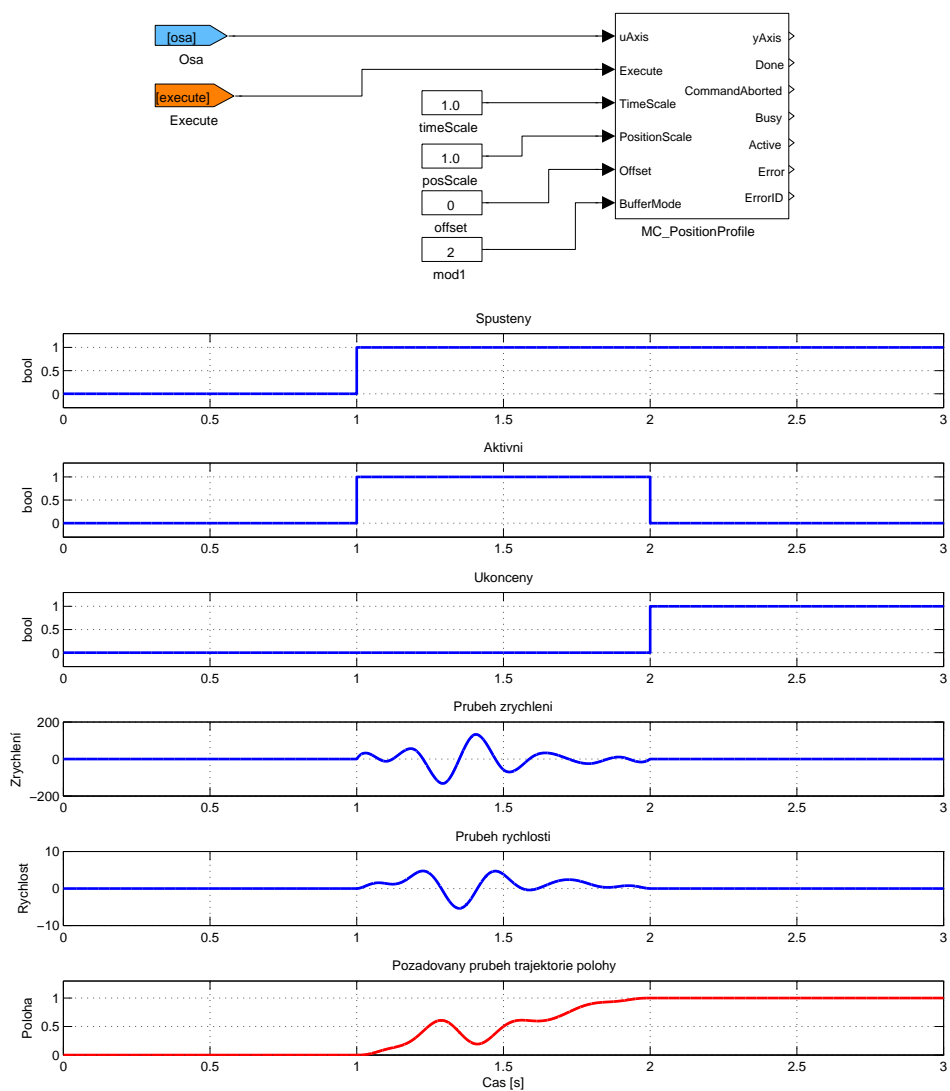
## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<code>Done</code>	Příznak dokončení algoritmu	bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	bool
<code>Active</code>	Příznak, že blok řídí osu	bool
<code>Error</code>	Příznak chyby	bool
<code>ErrorID</code>	Kód chyby	error
	<code>i</code> ..... obecná chyba systému REX	

## Parametry

<code>alg</code>	Typ interpolace	⊙2	long
	1 ..... tabulka čas/hodnota		
	2 ..... hodnoty ve stejném intervalu		
	3 ..... aproximace polynomy		
	4 ..... polynomy s ekvidistantními intervaly		
<code>cSeg</code>	Počet segmentů profilu	⊙3	long
<code>times</code>	Posloupnost hraničních časů jednotlivých segmentů	⊙[0 30]	double
<code>values</code>	Hodnoty veličiny nebo koeficienty interpolačních polynomů ( <code>a0, a1, a2, ...</code> )	⊙[0 100 100 50]	double

## Příklad

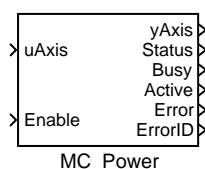




## MC\_Power – Aktivace osy

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok **MC\_Power** musí být použit s každou osou. Je to jediný blok, který převádí osu ze stavu **Disabled** do stavu **StandStill** (tj. aktivní režim). Vstup **Enable** musí být nastaven po celou dobu práce s osou. Výstup **Status** může být použit pro blokování motoru/měniče.

Pokud je osa vypnuta (nastavením vstupu **Enable** na nulu) při aktivním bloku (je nenulová rychlost), je nejprve aktivována zastavovací sekvence a teprve po jejím skončení je osa nastavena na stav vypnuto (disabled).

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	bool

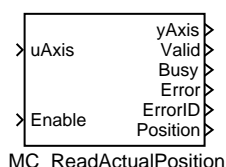
### Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<b>Status</b>	Skutečný stav napájení osy	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby i ..... obecná chyba systému REX	error

## MC\_ReadActualPosition – Skutečná poloha osy

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Blok `MC_ReadActualPosition` zpřístupňuje na výstupu `Position` aktuální polohu připojené osy. Hodnota je platná jen pokud je výstup `Valid` nenulový, čehož se dosáhne nastavením vstupu `Enable` na nenulovou hodnotu.

Blok zobrazuje logickou polohu, tj. hodnoty, které se do všech MC bloků zadávají jako poloha. Pokud není použit absolutní snímač polohy nebo pokud se poloha nastaví jinak (např. blokem `MC_Home`) je na výstupu `CommandedPosition` bloku `RM_Axis` odlišná hodnota.

## Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	bool

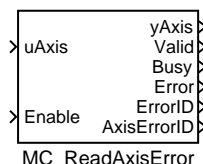
## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	bool
<code>Error</code>	Příznak chyby	bool
<code>ErrorID</code>	Kód chyby i ..... obecná chyba systému REX	error
<code>Position</code>	Aktuální poloha osy	double

## MC\_ReadAxisError – Chyba osy

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok **MC\_ReadAxisError** zpřístupňuje na výstupu **AxisErrorID** aktuální chybový kód připojené osy. Pokud osa není ve stavu chyby, hodnota tohoto výstupu je 0. Hodnota je platná jen pokud je výstup **Valid** nenulový, čehož se dosáhne nastavením vstupu **Enable** na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť zobrazuje stejnou veličinu, která je přístupná i na výstupu **ErrorID** bloku [RM\\_Axis](#).

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef–uAxis</b> nebo <b>yAxis–uAxis</b> )	reference
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	bool

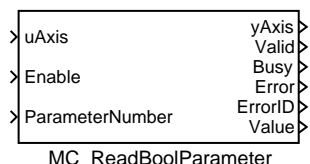
### Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef–uAxis</b> nebo <b>yAxis–uAxis</b> )	reference
<b>Valid</b>	Příznak platnosti výstupní hodnoty	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby i ..... obecná chyba systému REX	error
<b>AxisErrorID</b>	Chybový kod přečtený z osy i ..... obecná chyba systému REX	error

## MC\_ReadBoolParameter – Čtení parametru (logická hodnota)

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Blok **MC\_ReadBoolParameter** zpřístupňuje na výstupu **Value** aktuální hodnotu parametru připojené osy. Číslo požadovaného parametru musí být na vstupu **ParameterNumber**. Hodnota je platná jen pokud je výstup **Valid** nenulový, čehož se dosáhne nastavením vstupu **Enable** na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť zobrazuje parametry bloku **RM\_Axis**.

## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	reference
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	bool
<b>ParameterNumber</b>	Číslo požadovaného parametru	long
	4 ..... kontrola kladného omezení polohy	
	5 ..... kontrola záporného omezení polohy	
	6 ..... kontrola regulační odchylky (poloha)	

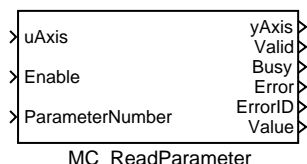
## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	reference
<b>Valid</b>	Příznak platnosti výstupní hodnoty	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby	error
	i ..... obecná chyba systému REX	
<b>Value</b>	Hodnota parametru	bool

## MC\_ReadParameter – Čtení parametru (číselná hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok **MC\_ReadParameter** zpřístupňuje na výstupu **Value** aktuální hodnotu parametru připojené osy. Číslo požadovaného parametru musí být na vstupu **ParameterNumber**. Hodnota je platná jen pokud je výstup **Valid** nenulový, čehož se dosáhne nastavením vstupu **Enable** na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť zobrazuje parametry a výstupy bloku [RM\\_Axis](#).

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef</b> – <b>uAxis</b> nebo <b>yAxis</b> – <b>uAxis</b> )	<b>reference</b>
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	<b>bool</b>
<b>ParameterNumber</b>	Číslo požadovaného parametru	<b>long</b>
	1 ..... požadovaná poloha	
	2 ..... kladné omezení polohy	
	3 ..... záporné omezení polohy	
	7 ..... maximální odchylka polohy	
	8 ..... maximální rychlost (systém)	
	9 ..... maximální rychlost (bloky)	
	10 .... skutečná rychlost	
	11 .... požadovaná rychlost	
	12 .... maximální zrychlení (systém)	
	13 .... maximální zrychlení (bloky)	
	14 .... maximální zpomalení (systém)	
	15 .... maximální zpomalení (bloky)	
	16 .... maximální změna zrychlení (jerk)	
	1000 .. skutečná poloha	
	1001 .. maximální síla/moment	
	1003 .. skutečná síla/moment	
	1004 .. požadovaná síla/moment	

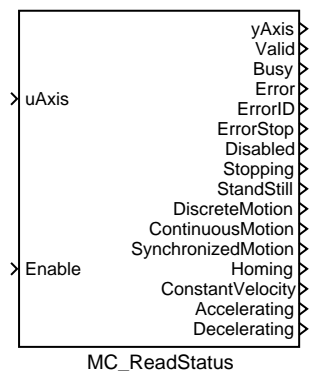
## Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	reference
Valid	Příznak platnosti výstupní hodnoty	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby i . . . . . obecná chyba systému REX	error
Value	Hodnota parametru	double

## MC\_ReadStatus – Stav osy

Symbol bloku

Licence: [MOTION CONTROL](#)



## Popis funkce

Blok **MC\_ReadStatus** indikuje na svých výstupech různé stavy připojené osy jako logickou hodnotu. indikovaný stav je zřejmý z názvu výstupu, popřípadě z jeho popisu. Hodnota je platná jen pokud je výstup **Valid** nenulový, čehož se dosáhne nastavením vstupu **Enable** na nenulovou hodnotu.

## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	<b>bool</b>

## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<b>reference</b>
<b>Valid</b>	Příznak platnosti výstupní hodnoty	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby i ..... obecná chyba systému REX	<b>error</b>
<b>ErrorStop</b>	Osa je ve stavu CHYBA	<b>bool</b>
<b>Disabled</b>	Osa je ve stavu VYPNUTO	<b>bool</b>
<b>Stopping</b>	Osa je ve stavu STOP	<b>bool</b>
<b>StandStill</b>	Osa je ve stavu PŘIPRAVEN	<b>bool</b>

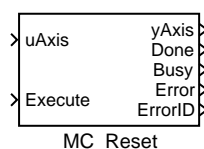
DiscreteMotion	Osa je ve stavu JEDNORÁZOVÝ POHYB	bool
ContinuousMotion	Osa je ve stavu TRVALÝ POHYB	bool
SynchronizedMotion	Osa je ve stavu SYNCHRONIZOVANÝ POHYB	bool
Homing	Osa je ve stavu HLEDÁNÍ VÝCHOZÍ POLOHY	bool
ConstantVelocity	Osa se pohybuje konstantní rychlostí	bool
Accelerating	Osa zrychluje	bool
Decelerating	Osa brzdí/zpomaluje	bool



## MC\_Reset – Nulování chyb osy

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok **MC\_Reset** převede připojenou osu ze stavu „ErrorStop“ do stavu „StandStill“ a vynuluje v ose všechny příznaky chyby. Je to v podstatě jediný blok, který ve stavu „ErrorStop“ nehlásí chybu.

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool

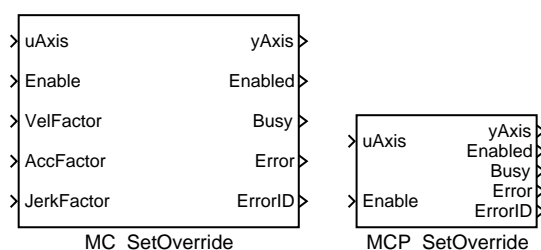
### Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby i ..... obecná chyba systému REX	error

## MC\_SetOverride, MCP\_SetOverride – Nastavení násobivých faktorů na ose

Symbody bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_SetOverride a MCP\_SetOverride mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_SetOverride** nastavuje násobivé faktory které se projeví ve všech blocích pracujících s osou. Hodnoty rychlosti, zrychlení a jerku ve všech blocích je potřeba vynásobit faktorem z tohoto bloku, abychom dostali hodnotu, se kterou blok skutečně pracuje. Toto se netýká limitních hodnot zadaných v [RM\\_Axis](#) a administrativních bloků.

Tento blok není aktivován hranou, ale pokud je vstup **Enable** nenulový, tak se hodnoty trvale aktualizují. Pokud je aktivní blok typu [MC\\_MoveAbsolute](#), vede to na neustálé přepočítávání trajektorie, což je výpočetně (a tím i časově) náročná operace a navíc se kumulují zaokrouhlovací chyby. Proto je zavedena necitlivost (parametr **diff**) a přepočet trajektorie je proveden až když se některý z faktorů změní více, než je tato necitlivost.

Poznámka: Všechny faktory musí být kladné. Faktory větší než 1 jsou možné, ale často vedou k překročení mezí nastavených na ose a k selhání pohybu (blok hlásí chybu **errorID** = -700 - neplatný parametr) nebo dokonce k havarijnímu zastavení osy (blok pak hlásí chybu **errorID** = -701 - hodnota mimo rozsah).

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <a href="#">RM_Axis.axisRef</a> – <b>uAxis</b> nebo <b>yAxis</b> – <b>uAxis</b> )	<b>reference</b>
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	<b>bool</b>
<b>VelFactor</b>	Faktor násobení pro rychlost	<b>double</b>
<b>AccFactor</b>	Faktor násobení pro zrychlení	<b>double</b>
<b>JerkFactor</b>	Faktor násobení pro změnu zrychlení	<b>double</b>

## Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<code>reference</code>
<code>Enabled</code>	Povolení funkce bloku (aktivace výstupů)	<code>bool</code>
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	<code>bool</code>
<code>Error</code>	Příznak chyby	<code>bool</code>
<code>ErrorID</code>	Kód chyby i ..... obecná chyba systému REX	<code>error</code>

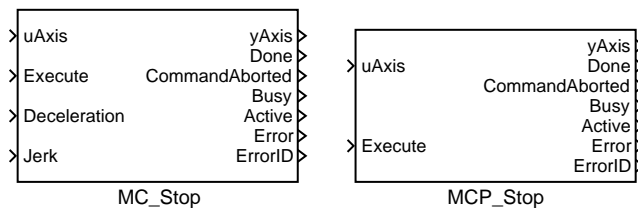
## Parametr

<code>diff</code>	Pásmo necitlivosti (pro přepočet trajektorie)	$\downarrow 0.0 \uparrow 1.0 \odot 0.1$	<code>double</code>
-------------------	---	---	---------------------

## MC\_Stop, MCP\_Stop – Zastavení pohybu

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_Stop a MCP\_Stop mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Block MC\_Stop provede zastavovací sekvenci a převede osu do stavu **Stopping**. V tomto stav není možné spustit žádný pohyb a osa v něm zůstává dokud je vstup **Execute** nenulový.

Poznámka1: Blok nemá parametr **BufferMode**. Mód je vždy **Aborting**.

Poznámka2: Protože selhání příkazu k zastavení může být nebezpečné, blok generuje chybu jen v naprosto fatálních případech (např. nezapojený vstup **uAxis**) a snaží se co nejkorektněji zastavit (např. při nekorektních parametrech použije nastavení osy nebo vyvolá sekvenci pro chybové zastavení).

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double

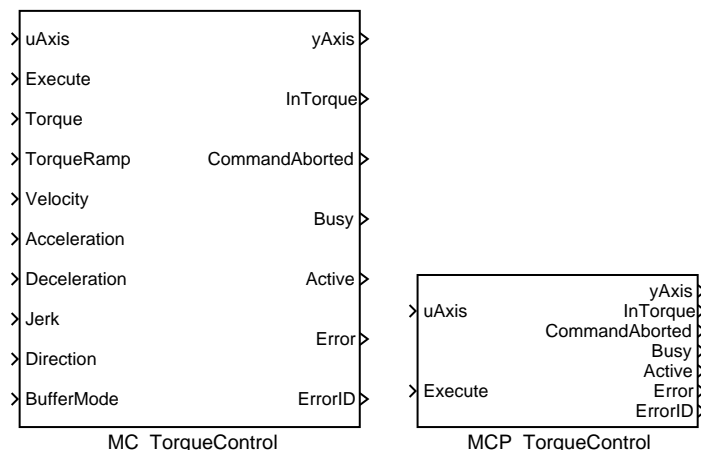
### Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool

Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	

## MC\_TorqueControl, MCP\_TorqueControl – Řízení síly/momentu

Symboly bloků

Licence: [MOTION CONTROL](#)

## Popis funkce

*Bloky MC\_TorqueControl a MCP\_TorqueControl mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_TorqueControl generuje požadovaný moment/sílu nejprve s konstantním nárůstem (parametr TorqueRamp) a po dosažení maximální hodnoty (parametr Torque) je již moment/síla konstantní. Pohyb osy je řízen podle požadovaného momentu tak, aby nebyly překročeny maximální hodnoty rychlosti, zrychlení/zpomalení a případně jerku.

## Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	reference
Execute	Náběžná hrana aktivuje blok	bool
Torque	Maximální povolený moment/síla	double
TorqueRamp	Maximální povolená změna momentu/síly	double
Velocity	Maximální povolená rychlost [unit/s]	double
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double

<b>Direction</b>	Směr pohybu (jen pro cyklické osy nebo speciální případy)	long
	1 ..... kladný	
	2 ..... nejkratší	
	3 ..... záporný	
	4 ..... aktuální	
<b>BufferMode</b>	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

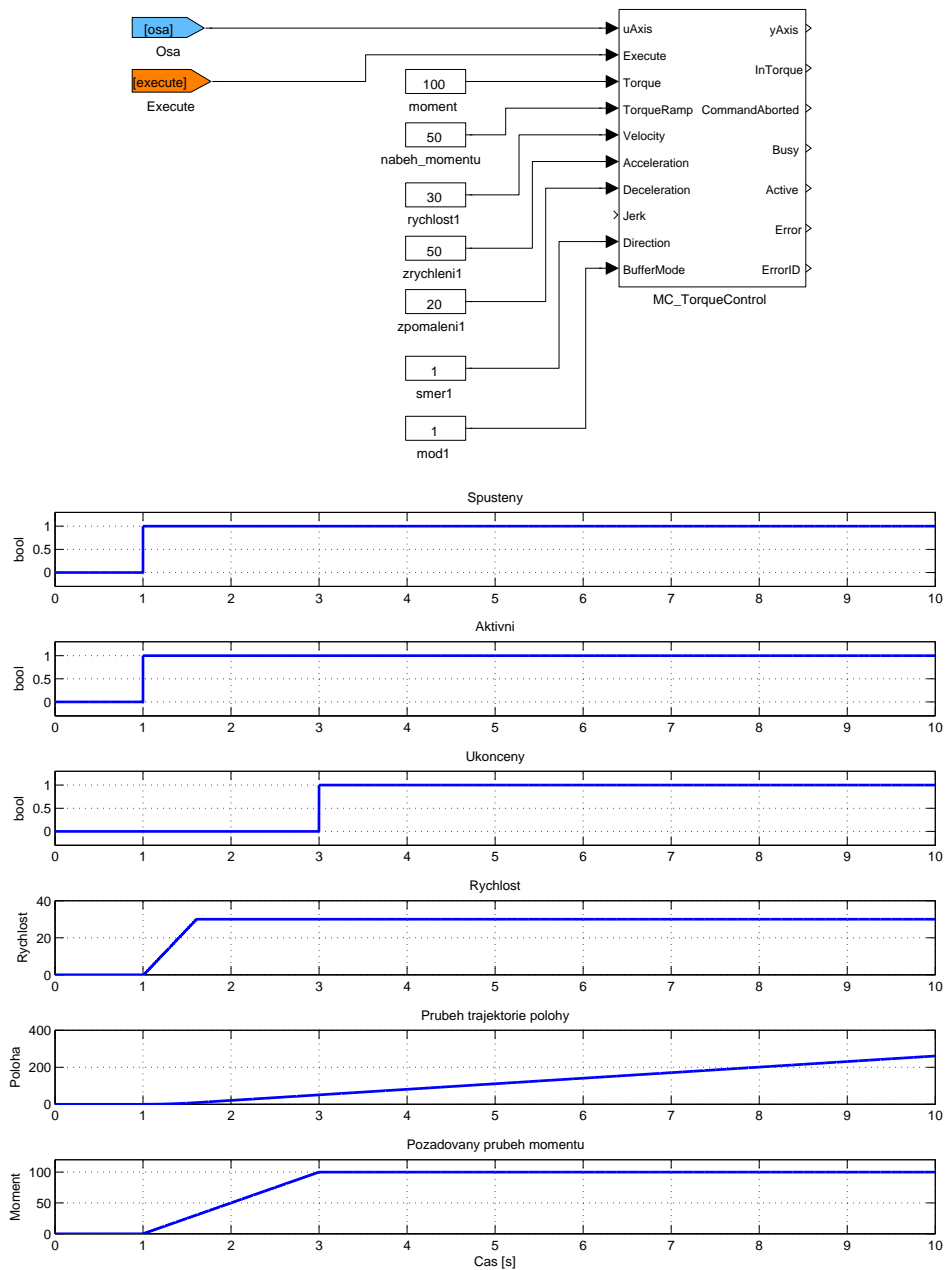
## Výstupy

<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<b>InTorque</b>	Příznak dosažení požadovaného momentu/síly	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby	error
	i ..... obecná chyba systému REX	

## Parametr

<b>kma</b>	Poměr mezi silou/momentem a zrychlením	double
------------	--	--------

## Příklad

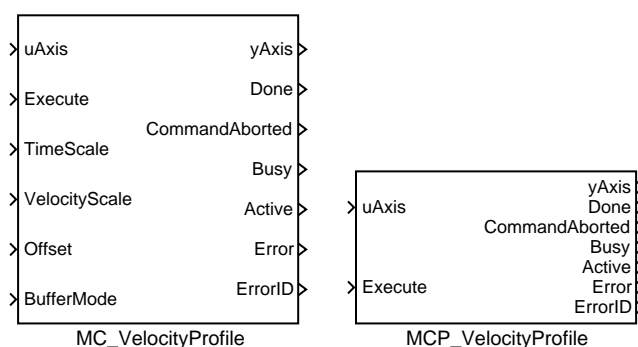




## MC\_VelocityProfile, MCP\_VelocityProfile – Generování trajektorie (rychlost)

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_VelocityProfile a MCP\_VelocityProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_VelocityProfile** generuje takovou trajektorii, aby rychlost byla požadovaná funkce času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a rychlost. Mezi jednotlivými časy se hodnota rychlosti interpoluje lineárně. Hodnoty času (v sekundách) se zadávají do pole/parametru **times**, příslušné hodnoty rychlosti do pole/parametru **values**. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli **times**. Polynom pro každý interval je ve tvaru  $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ , přičemž na začátku časového intervalu je  $x = 0$ , a na konci  $x = 1$ . Koeficienty  $a_i$  jsou uloženy v poli **values** ve vzestupném pořadí (tj. pole **values** obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli **times** jen počáteční (obvykle 0) a koncový čas.

Poznámka1: Přestože vstup/odkaz **TimePosition** je v **PLCopen** označen jako povinný, není zde použit, protože všechna potřebná data jsou uložena v parametrech bloku.

Poznámka2: Parametr **values** musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru nebo čárku).

Poznámka3: Nesprávný parametr **cSeg** (větší než skutečná velikost polí **times** a/nebo **values**) vede k nedefinovanému chování, v krajním případě nestandardnímu ukončení aplikace (závisí to na mnoha okolnostech, použití tohoto bloku v SIMULINKu vede k pádu MATLABu).

Poznámka4: V režimu zadání funkce polynomem je hodnota polynomu poloha a polynom je vždy pátého řádu a nelze to nijak měnit. **VelocityScale** a **Offset** je samozřejmě pro rychlost. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka5: Block neobsahuje tzv. ramp-in mode. Pokud tedy rychlost osy v okamžiku spuštění profilu neodpovídá počáteční rychlosti profilu, blok skončí s chybou -707 (skok v rychlosti nebo poloze). Tomuto problému lze předejít, pokud se použije **BufferMode=BlendingNext** nebo je potřeba správně nastavit parametr **Offset**.

Poznámka6: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací **PLCopen**).

## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>TimeScale</b>	Konstanta násobení pro přepočet časové osy profilu	<b>double</b>
<b>VelocityScale</b>	Konstanta násobení pro přepočet hodnotové osy profilu	<b>double</b>
<b>Offset</b>	Aditivní konstanta pro přepočet hodnotové osy profilu	<b>double</b>
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

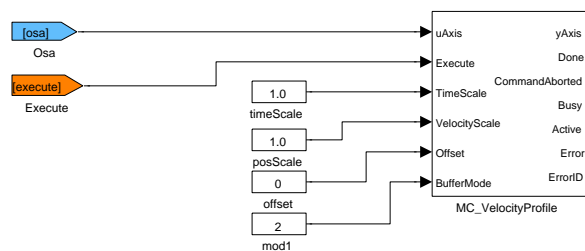
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>

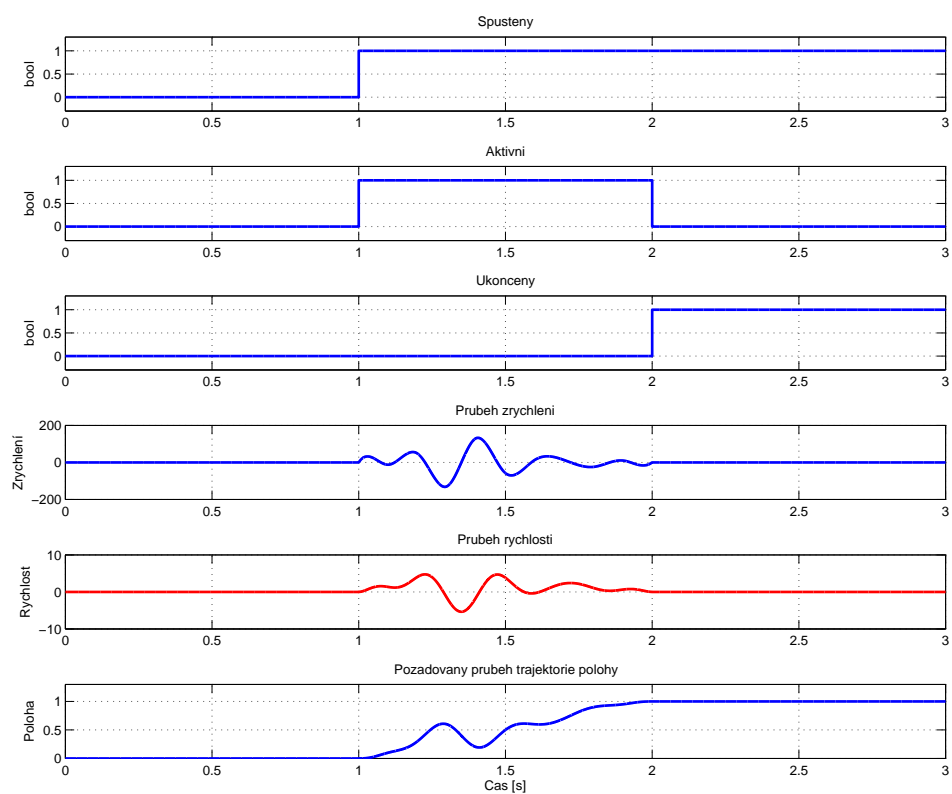
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby	error
	i ..... obecná chyba systému REX	

## Parametry

<b>alg</b>	Typ interpolace	⊙1	long
	1 ..... tabulka čas/hodnota		
	2 ..... hodnoty ve stejném intervalu		
	3 ..... aproximace polynomy		
	4 ..... polynomy s ekvidistantními intervaly		
<b>cSeg</b>	Počet segmentů profilu	⊙3	long
<b>times</b>	Posloupnost hraničních časů jednotlivých segmentů	⊙[0 15 25 30]	double
<b>values</b>	Hodnoty veličiny nebo koeficienty interpolačních polynomů (a0, a1, a2, ...)	⊙[0 100 100 50]	double

## Příklad

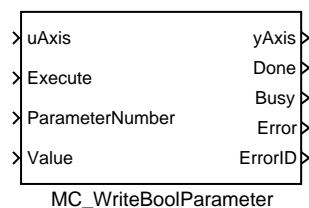




## MC\_WriteBoolParameter – Nastavení parametru (logická hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok `MC_WriteBoolParameter` změní hodnotu parametru připojené osy na hodnotu danou vstupem `Value`. Číslo požadovaného parametru musí být přivedeno na vstup `ParameterNumber`.

Tento blok je implementován z důvodu kompatibility s `PLCopen` neboť nastavuje parametry bloku [RM\\_Axis](#), což umožňuje i blok `SETPB`.

### Vstupy

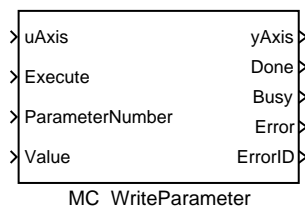
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<code>Execute</code>	Náběžná hrana aktivuje blok	bool
<code>ParameterNumber</code>	Číslo požadovaného parametru	long
	4 ..... kontrola kladného omezení polohy	
	5 ..... kontrola záporného omezení polohy	
	6 ..... kontrola regulační odchylky (poloha)	
<code>Value</code>	Hodnota parametru	bool

### Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	reference
<code>Done</code>	Příznak dokončení algoritmu	bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	bool
<code>Error</code>	Příznak chyby	bool
<code>ErrorID</code>	Kód chyby	error
	i ..... obecná chyba systému REX	

## MC\_WriteParameter – Nastavení parametru (číselná hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)

## Popis funkce

Blok **MC\_WriteParameter** změní hodnotu parametru připojené osy na hodnotu danou vstupem **Value**. Číslo požadovaného parametru musí být na vstupu **ParameterNumber**.

Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť nastavuje parametry bloku [RM\\_Axis](#), což umožňuje i blok **SETPR**.

## Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>ParameterNumber</b>	Číslo požadovaného parametru	long
	2 ..... kladné omezení polohy	
	3 ..... záporné omezení polohy	
	7 ..... maximální odchylka polohy	
	8 ..... maximální rychlost (systém)	
	9 ..... maximální rychlost (bloky)	
	13 ..... maximální zrychlení (bloky)	
	15 ..... maximální zpomalení (bloky)	
	16 ..... maximální změna zrychlení (jerk)	
	1001 .. maximální síla/moment	
<b>Value</b>	Hodnota parametru	double

## Výstupy

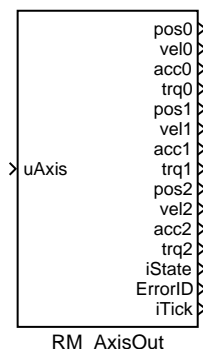
<b>yAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Error</b>	Příznak chyby	bool

ErrorID	Kód chyby	error
i .....	obecná chyba systému REX	

## RM\_AxisOut – Výstupní blok osy

Symbol bloku

Licence: [MOTION CONTROL](#)



## Popis funkce

Blok `RM_AxisOut` zpřístupňuje důležité stavy bloku `RM_Axis`. Některé z výstupů jsou přímo na bloku `RM_Axis`, ale ty jsou o krok zpožděné. Aby nebyly výstupy zpožděné i na tomto bloku, je potřeba jej zařadit jako poslední.

Poznámka 1: Řídící systém REX řadí bloky podle toku signálu a druhotně podle jména bloku (abecedně, vzestupně). Blok je proto vhodné pojmenovat „zzzAxis1“ a podobně. Pro kontrolu pořadí spouštění bloků je možné použít `RexView`, kde jsou bloky řazeny v pořadí, jak se vykonávají.

Poznámka 2: Téměř všechny bloky nepracují s momentem, proto je na příslušném výstupu 0. Obvykle je tento signál používán jako dopředná vazba pro regulátor rychlosti, tak to nepředstavuje problém.

## Vstup

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>reference yAxis-uAxis</code> )
--------------------	---

## Výstupy

<code>pos0</code>	Požadovaná poloha osy (aktuální krok)	<code>double</code>
<code>vel0</code>	Požadovaná rychlost osy (aktuální krok)	<code>double</code>
<code>acc0</code>	Požadované zrychlení osy (aktuální krok)	<code>double</code>
<code>trq0</code>	Požadovaný moment/síla osy (aktuální krok)	<code>double</code>
<code>pos1</code>	Požadovaná poloha osy (následující krok)	<code>double</code>
<code>vel1</code>	Požadovaná rychlost osy (následující krok)	<code>double</code>
<code>acc1</code>	Požadované zrychlení osy (následující krok)	<code>double</code>

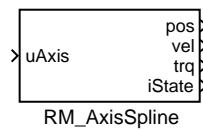


<b>trq1</b>	Požadovaný moment/síla osy (následující krok)	<b>double</b>
<b>pos2</b>	Požadovaná poloha osy (2. následující krok)	<b>double</b>
<b>vel2</b>	Požadovaná rychlost osy (2. následující krok)	<b>double</b>
<b>acc2</b>	Požadované zrychlení osy (2. následující krok)	<b>double</b>
<b>trq2</b>	Požadovaný moment/síla osy (2. následující krok)	<b>double</b>
<b>iState</b>	Stav osy pro řízení pohybu	<b>long</b>
	0 ..... Disabled (osa blokována)	
	1 ..... Stand still (osa připravena)	
	2 ..... Homing (hledání výchozí polohy)	
	3 ..... Discrete motion (jednorázový pohyb)	
	4 ..... Continuous motion (trvalý pohyb)	
	5 ..... Synchronized motion (pohyb synchronní s jinou osou)	
	6 ..... Coordinated motion (osa řízena blokem pro koordinovaný pohyb)	
	7 ..... Stopping (zastavování nebo dočasné blokování)	
	8 ..... Error stop (zastavování nebo blokování osy po chybě)	
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i ..... obecná chyba systému REX	
<b>iTick</b>	Aktuální číslo periody (pro interpolátor)	<b>long</b>

## RM\_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení)

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Mnoho bloků pro řízení pohybu obsahuje výpočetně náročný algoritmus. To vede na relativně velké vzorkovací periody (typicky mezi 10 a 200ms). Naproti tomu regulátor motoru vyžaduje vzorkovací periodu malou (typicky do 1ms), aby nedocházelo k trhavému pohybu. Tyto protichůdné požadavky řeší blok **RM\_AxisSpline**, který může běžet v jiné úloze (s kratší periodou vzorkování) než blok [RM\\_Axis](#) a provádí interpolaci hodnot, tak aby výsledná křivka byla spojitá a pokud možno hladká.

Pro napojení na blok [RM\\_Axis](#) lze použít dvě možnosti: buď se musí předávat všechny potřebné hodnoty (výstupy bloku [RM\\_AxisOut](#)) nebo se předá odkaz a zajistí správná synchronizace. Blok **RM\_AxisSpline** používá předávání hodnot odkazem. Pro správnou synchronizaci musí blok [RM\\_Axis](#) být spouštěn jako první, následují všechny ostatní bloky (spojené s touto osou) a nakonec blok [RM\\_AxisOut](#).

Poznámka 1: Pro interpolaci polohy je použit polynom třetího řádu  $p(t)$ , přičemž  $p_s(0) = pos0, p_s(t_s) = pos1, \frac{dp_s(t)}{dt} \big|_{t=0} = vel0, \frac{dp_s(t)}{dt} \big|_{t=t_s} = vel1$ . Pro interpolaci rychlosti je také použit polynom třetího řádu  $p_v(t)$ , kde  $p_v(0) = vel0, p_v(t_s) = vel1, \frac{dp_v(t)}{dt} \big|_{t=0} = acc0, \frac{dp_v(t)}{dt} \big|_{t=t_s} = acc1$ . Pro interpolaci momentu/síly je použita pouze lineární interpolace.

Poznámka 2: Protože doba vykonávání bloků pro řízení pohybu značně kolísá, dostává interpolátor nové hodnoty neekvidistantně, přičemž výstup interpolátoru musí být souvislý. Proto se podle potřeby pro interpolaci používá aktuální hodnota, jednokroková predikce nebo dvoukroková predikce a tomu odpovídají stavy `interval0`, `interval1`, `interval2`. Důležité je pravidlo, že stavy odpovídající kladným číslům jsou normální činnost a stavy odpovídající záporným číslům představují chybu.

Poznámka 3: Řídící systém REX řadí bloky podle toku signálu a druhotně podle jména bloku (abecedně, vzestupně). Blok je proto vhodné pojmenovat „zzz“ a podobně. Pro kontrolu pořadí spouštění bloků je možné použít `RexView`, kde jsou bloky řazeny v pořadí, jak se vykonávají.

## Vstup

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code> )	<code>reference</code>
--------------------	---	------------------------

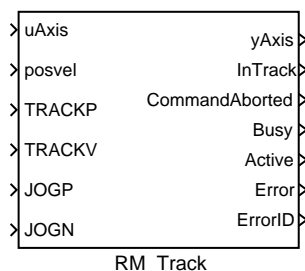
## Výstupy

<code>pos</code>	Požadovaná (interpolovaná) poloha	<code>double</code>
<code>vel</code>	Požadovaná (interpolovaná) rychlost	<code>double</code>
<code>trq</code>	Požadovaný (interpolovaný) moment/síla	<code>double</code>
<code>iState</code>	Stav interpolátoru	<code>long</code>
	0 . . . . . neaktivní	
	1 . . . . . interval0	
	2 . . . . . interval1	
	3 . . . . . interval2	
	5 . . . . . změna 1. predikce	
	-1 . . . . změna aktuální hodnoty	
	-2 . . . . zpoždění <code>RM_Axis</code>	
	-3 . . . . přetížení tasku	
	-4 . . . . velká perioda	

## RM\_Track – Sledování a krokování

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok **RM\_Track** sdružuje několik užitečných doplňkových funkcí.

Pokud je aktivní (tj. nenulový) vstup **TRACK**, blok se snaží dosáhnout zadané polohy nebo rychlosti (vstup **posvel**) s respektováním omezení na rychlost, zrychlení/zpomalení a jerk. Rozdíl oproti bloku [MC\\_MoveAbsolute](#) je ten, že tento blok v každém kroku aktualizuje cílovou polohu nebo rychlost a přepočítává trajektorii. Tento režim je vhodný pro sledování trajektorie polohy nebo rychlosti generované mimo motion control subsystém. Pokud je trajektorie známa předem, je mnohem vhodnější a přesnější použít blok [MC\\_PositionProfile](#).

Pokud je aktivní (tj. nenulový) vstup **JOGP**, blok jede maximální dovolenou rychlostí v kladném směru s respektováním maximálního zrychlení a jerku při rozjezdu. Po deaktivaci signálu blok zastaví pohyb (s respektováním maximálního zpomalení a případně jerku) a vzdá se řízení osy. Tento režim je vhodný pro najetí na výchozí polohu operátorem pomocí tlačítek vpřed a vzad.

Vstup **JOGN** má stejnou funkci jako **JOGP**, jen je pohyb v negativním směru.

Pokud je na ose již nějaký pohyb aktivní, je tímto blokem přerušen (chová se tedy jako **BufferMode=aborting**). Pokud je sepnuto více funkcí, vykonává se ta s nejvyšší prioritou a ostatní jsou ignorovány. Pořadí je **TRACK**, **JOGP**, **JOGN**. Takovém režimu je však vhodné se vyhnout, protože chování není dostatečně intuitivní a výsledek je pak nečekaný.

### Vstupy

<b>uAxis</b>	Odkaz na osu (přípustné je jen spojení <b>RM_Axis.axisRef-uAxis</b> nebo <b>yAxis-uAxis</b> )	<b>reference</b>
<b>posvel</b>	Požadovaná poloha nebo rychlost [unit, unit/s]	<b>double</b>
<b>TRACKP</b>	Zapnutí režimu sledování polohy	<b>bool</b>
<b>TRACKV</b>	Zapnutí režimu sledování rychlosti	<b>bool</b>
<b>JOGP</b>	Zapnutí režimu posunu v kladném směru	<b>bool</b>

JOGN	Zapnutí režimu posuvu v záporném směru	bool
------	--	------

## Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	reference
InTrack	Příznak dosažení požadované polohy v režimu sledování polohy	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby i ..... obecná chyba systému REX	error

## Parametry

pv	Maximální povolená rychlost [unit/s]	double
pa	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
pd	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
pj	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
iLen	Počet kroků pro odhad rychlosti	⊙10 long



## Kapitola 17

# MC\_MULT – Řízení pohybu více os

### Obsah

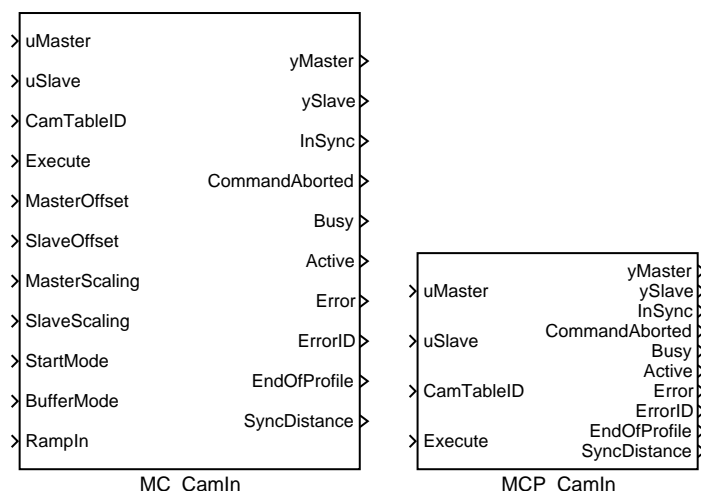
MC_CamIn, MCP_CamIn – Zapnutí vačky . . . . .	456
MC_CamOut – Vypnutí vačky . . . . .	460
MCP_CamTableSelect – Definice vačky . . . . .	462
MC_CombineAxes, MCP_CombineAxes – Kombinace pohybu dvou os do třetí . . . . .	464
MC_GearIn, MCP_GearIn – Zapnutí konstantního převodového poměru . . . . .	467
MC_GearInPos, MCP_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici . . . . .	470
MC_GearOut – Vypnutí konstantního převodového poměru . . . . .	475
MC_PhasingAbsolute, MCP_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice) . . . . .	477
MC_PhasingRelative, MCP_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění) . . . . .	480

Tato kategorie bloků zahrnuje bloky pro synchronizovaný pohyb více os, jak jsou definovány ve specifikaci PLCopen. Jde zejména o tzv. elektronické vačky a elektronické převodovky. Pro bloky této kategorie platí stejné obecné zásady, jaké byly uvedeny v kapitole 16 (knihovna MC\_SINGLE, bloky pro řízení jedné osy).

## MC\_CamIn, MCP\_CamIn – Zapnutí vačky

Symboly bloků

Licence: MOTION CONTROL



## Popis funkce

*Bloky MC\_CamIn a MCP\_CamIn mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_CamIn zapíná režim, kdy je podřízená osa (tj. ta, která je připojena ke vstupu uSlave) řízena tak, že její poloha je závislá na poloze hlavní osy (tj. ta, která je připojena ke vstupu uMaster), přičemž převodní funkce je určena blokem MCP\_CamTableSelect připojeným ke vstupu CamTableID. Pokud převodní funkci označíme  $Cam(\cdot)$ , polohu hlavní osy  $PosM$  a polohu podřízené osy  $PosS$ , pak platí (pro absolutní režim, bez fázování):

$$PosS = Cam((PosM - MasterOffset) / MasterScaling) * SlaveScaling + SlaveOffset$$

Tento režim osy je často nazýván elektronická vačka.

Režim vačky lze ukončit zapnutím jiného pohybu na podřízené ose v režimu **aborting** nebo spuštěním bloku MC\_CamOut. Pokud vačka (její definiční funkce - viz popis bloku není cyklická MCP\_CamTableSelect), dojde k ukončení režimu vačky také při vyjetí hlavní osy z definičního oboru funkce vačky. Toto je signalizováno výstupem EndOfProfile.

Při aktivaci funkce vačky (tj. v okamžiku, kdy blok MC\_CamIn převeze řízení osy) nemusí poloha a rychlost (popř. i zrychlení, ale současná implementace při zapnutí vačky



jerk neuvažuje a připouští skok ve zrychlení) odpovídat požadovaným hodnotám, tj. poloze a rychlosti hlavní osy a profilu vačky. V takovém případě záleží na hodnotě parametru **RampIn**. Pokud má parametr hodnotu 0, režim vačky se nezapne a je signalizována chyba -707 (skok v poloze nebo rychlosti). Pokud je parametr kladný, nastává přechodový děj, kdy poloha ještě neodpovídá poloze podle definice vačky - tzv. RampIn režim. Parametr **RampIn** přibližně odpovídá rychlosti, kterou by se podřízená osa pohybovala během RampIn režimu, kdyby hlavní osa stála. Pokud se hlavní osa pohybuje, pak je výsledný pohyb určen součtem pohybu daného definicí vačky a pohybu daného RampIn režimem se stojící hlavní osou. Pokud je potřeba RampIn režim použít, volíme parametr **RampIn** přibližně 0,1 až 0,5 maximální rychlosti podřízené osy a pokud dojde k chybě překročení maximální rychlosti nebo zrychlení během RampIn režimu, tak jej zmenšíme.

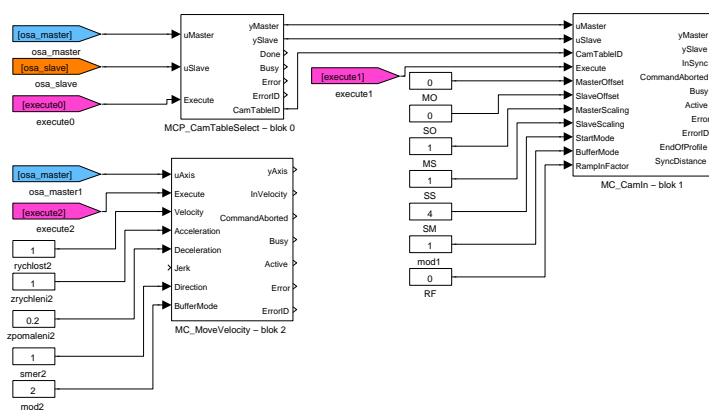
## Vstupy

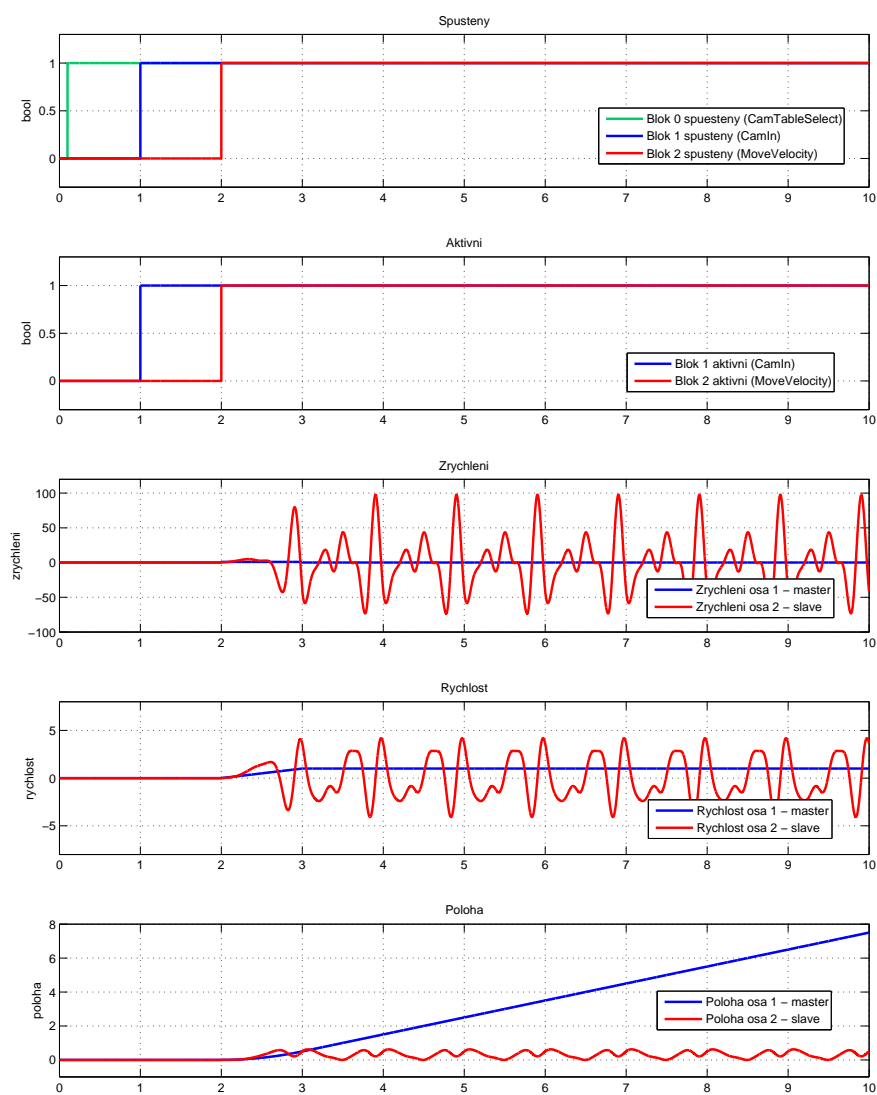
<b>uMaster</b>	Odkaz na hlavní osu	reference
<b>uSlave</b>	Odkaz na podřízenou osu	reference
<b>CamTableID</b>	Odkaz na vačku (spojit s <code>MCP_CamTableSelect.CamTableID</code> )	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>MasterOffset</b>	Posunutí v definici vačky na straně hlavní osy [unit]	double
<b>SlaveOffset</b>	Posunutí v definici vačky na straně podřízené osy [unit]	double
<b>MasterScaling</b>	Měřítko pro definici vačky (strana hlavní osy)	double
<b>SlaveScaling</b>	Měřítko pro definici vačky (strana podřízené osy)	double
<b>StartMode</b>	Volba absolutního nebo relativního profilu vačky	long
	1 ..... hlavní osa relativní podřízená osa absolutní	
	2 ..... hlavní osa absolutní podřízená osa relativní	
	3 ..... obě osy relativní	
	4 ..... obě osy absolutní	
<b>BufferMode</b>	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>RampIn</b>	RampIn faktor (0 = RampIn režim se nepoužívá); odpovídá přibližně rychlosti synchronizace [unit/s] podřízené osy na polohu vačky	double

## Výstupy

yMaster	Odkaz na hlavní osu	reference
ySlave	Odkaz na podřízenou osu	reference
InSync	Příznak dosažení profilu vačky podřízenou osou	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
i ..... obecná chyba systému REX		
EndOfProfile	Příznak dosažení konce profilu vačky (jen necyklické vačky)	bool
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	double

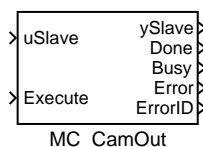
## Příklad





## MC\_CamOut – Vypnutí vačky

Symbol bloku

Licence: [MOTION CONTROL](#)

## Popis funkce

Blok **MC\_CamOut** ukončuje režim vačky zapnutý blokem **MC\_CamIn**. Pokud žádná vačka není aktivní, blok nemá žádnou funkci (a ani nehlásí chybu).

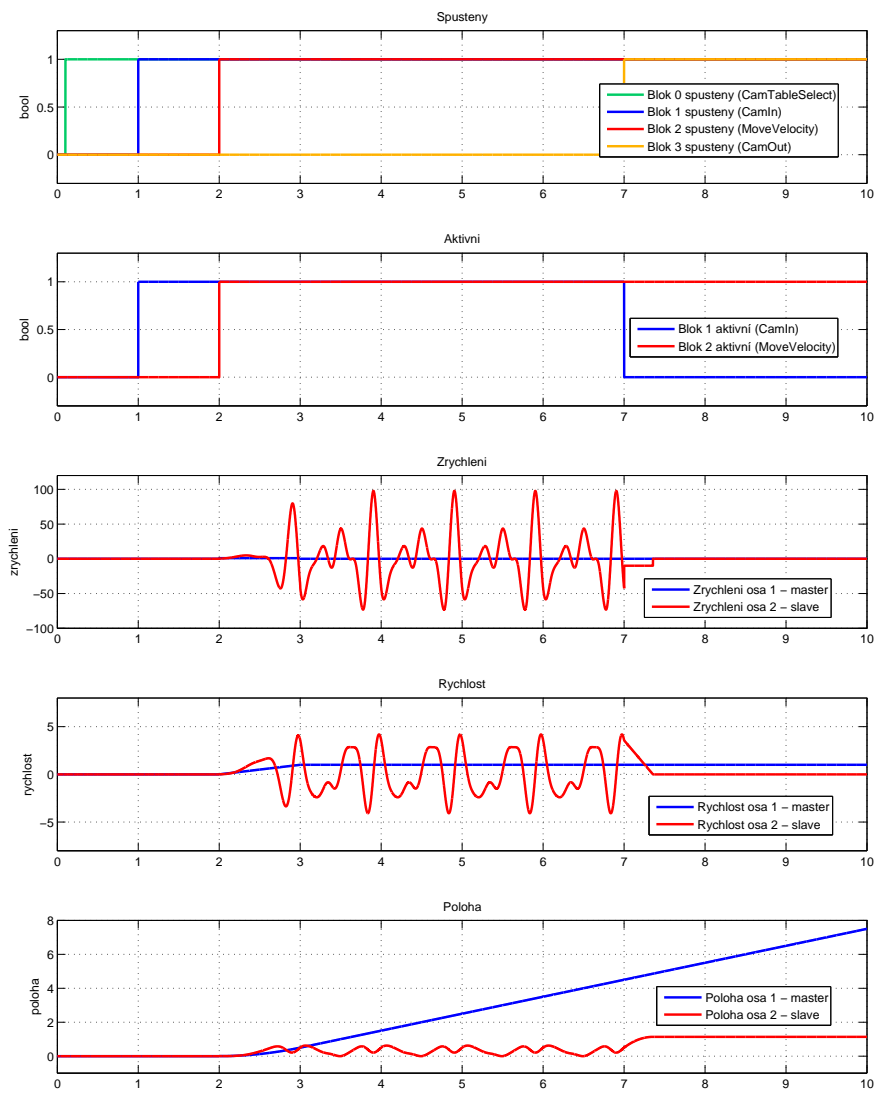
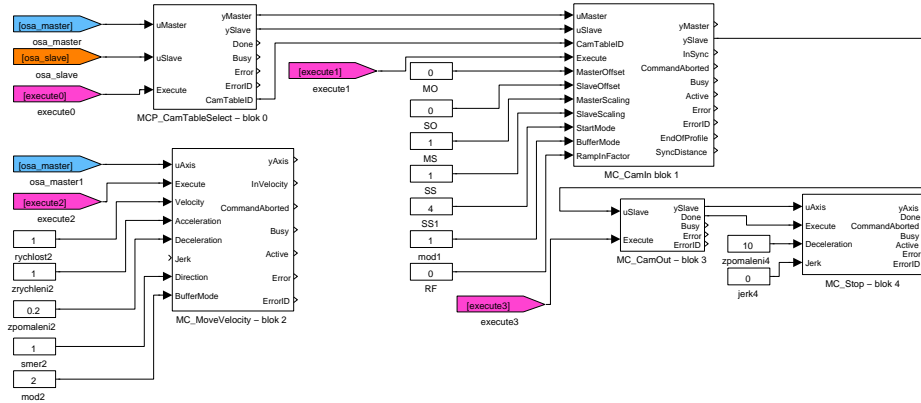
## Vstupy

<b>uSlave</b>	Odkaz na podřízenou osu	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>

## Výstupy

<b>ySlave</b>	Odkaz na podřízenou osu	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i . . . . . obecná chyba systému REX	

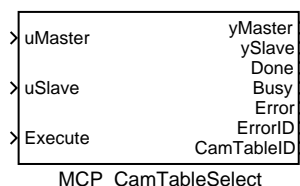
## Příklad



## MCP\_CamTableSelect – Definice vačky

Symbol bloku

Licence: MOTION CONTROL



## Popis funkce

Blok `MCP_CamTableSelect` spolupracuje s blokem `MCP_CamIn` a definuje vačku jako spojitou funkci jedné proměnné. Možnosti definování této funkce jsou analogické, jako v bloku `MC_PositionProfile`, tj. máme dvě možnosti:

1. tabulkou: zadávají se dvojice čísel poloha hlavní osy a poloha podřízené osy. Mezi jednotlivými časy se poloha interpoluje polynomem třetího řádu (lineární interpolace není v tomto případě vhodná, protože na okrajích intervalu by byl skok v rychlosti). Hodnoty polohy hlavní osy se zadávají do pole/parametru `mvalues`, příslušné hodnoty polohy podřízené osy do pole/parametru `svalues`. Posloupnost hodnot `mvalues` musí být stoupající.

2. polynomy: celá funkce se v hlavní ose (tj. v nezávislé proměnné) rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Polohy hlavní osy a tím i příslušné intervaly se definují jako v předchozím případě v poli `mvalues`. Polynom pro každý interval je ve tvaru  $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ , přičemž na začátku intervalu je  $x = 0$ , a na konci  $x = 1$ . Koeficienty  $a_i$  jsou uloženy v poli `svalues` ve vzestupném pořadí (tj. pole `values` obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly, pak je v poli `mvalues` jen počáteční a koncová poloha.

## Vstupy

<code>uMaster</code>	Odkaz na hlavní osu	reference
<code>uSlave</code>	Odkaz na podřízenou osu	reference
<code>Execute</code>	Náběžná hrana aktivuje blok	bool

## Výstupy

<code>yMaster</code>	Odkaz na hlavní osu	reference
----------------------	---------------------	-----------

ySlave	Odkaz na podřízenou osu	reference
Done	Příznak dokončení algoritmu	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	
CamTableID	Odkaz na vačku (spojit s <a href="#">MC_CamIn</a> .CamTableID)	reference

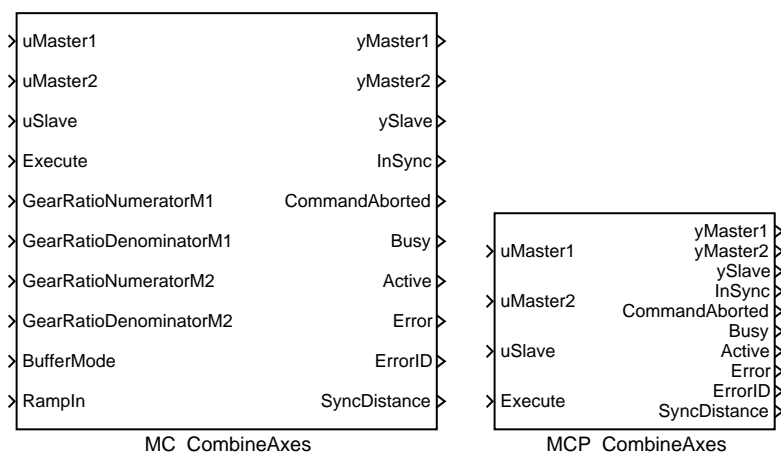
## Parametry

alg	Typ interpolace	⊙2	long
	1 ..... tabulka čas/hodnota		
	2 ..... hodnoty ve stejném intervalu		
	3 ..... aproximace polynomy		
	4 ..... polynomy s ekvidistantními intervaly		
cSeg	Počet segmentů profilu	⊙3	long
Periodic	Příznak cyklické vačky (konec navazuje na začátek)	⊙on	bool
camname	Jméno souboru, kam si speciální editor ukládá data (pokud je parametr prázdný, zvolí se automaticky podle jména bloku)		string
mvalues	Posloupnost hraničních pozic jednotlivých segmentů na hlavní ose		double
		⊙[0 30]	
svalues	Posloupnost poloh řízené osy nebo koeficienty interpolačních polynomů (a0, a1, a2, ...)		double
		⊙[0 100 100 0]	

## MC\_CombineAxes, MCP\_CombineAxes – Kombinace pohybu dvou os do třetí

Symboły bloků

Licence: [MOTION CONTROL](#)



## Popis funkce

*Bloky MC\_CombineAxes a MCP\_CombineAxes mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_CombineAxes kombinuje pohyb dvou os do třetí. V podstatě se jedná o výpočet nové žádané pozice na základě dvou poloh. Platí, že

$$\text{SlavePosition} = \text{Master1Position} \cdot \frac{\text{GearRatioNumeratorM1}}{\text{GearRatioDenominatorM1}} + \text{Master2Position} \cdot \frac{\text{GearRatioNumeratorM2}}{\text{GearRatioDenominatorM2}}$$

Blok umožňuje zadat do parametru GearRatio... záporné číslo a výsledný pohyb podřízené osy může být rozdíl poloh obou hlavních os.

## Vstupy

uMaster1	Odkaz na první hlavní osu	reference
uMaster2	Odkaz na druhou hlavní osu	reference
uSlave	Odkaz na podřízenou osu	reference
Execute	Náběžná hrana aktivuje blok	bool

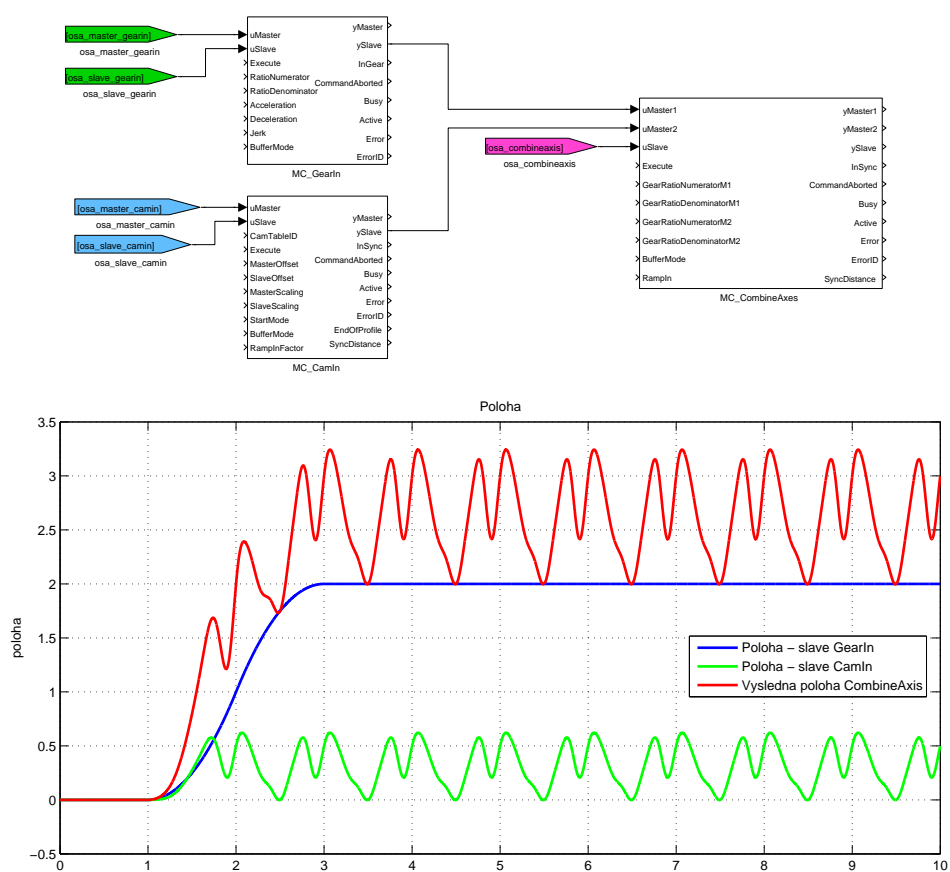


GearRatioNumeratorM1	Čítatel převodového poměru master osy 1	long
GearRatioDenominatorM1	Jmenovatel převodového poměru master osy 1	long
GearRatioNumeratorM2	Čítatel převodového poměru master osy 2	long
GearRatioDenominatorM2	Jmenovatel převodového poměru master osy 2	long
BufferMode	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
RampIn	RampIn faktor (0 = RampIn režim se nepoužívá). Odpovídá přibližně rychlosti synchronizace [unit/s] podřízené osy na polohu vačky.	double

## Výstupy

yMaster1	Odkaz na první hlavní osu	reference
yMaster2	Odkaz na druhou hlavní osu	reference
ySlave	Odkaz na podřízenou osu	reference
InSync	Příznak dosažení profilu vačky podřízenou osou	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	double

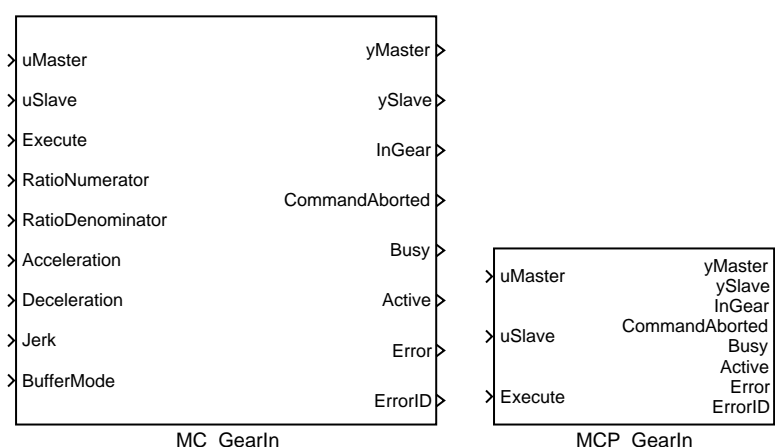
## Příklad



## MC\_GearIn, MCP\_GearIn – Zapnutí konstantního převodového poměru

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_GearIn a MCP\_GearIn mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_GearIn zapíná režim, kdy je podřízená osa (tj. ta, která je připojena ke vstupu uSlave) řízena tak, že její poloha je závislá na poloze hlavní osy (tj. ta, která je připojena ke vstupu uMaster), přičemž poměr rychlostí obou os je dán parametry RatioNumerator a RatioDenominator. Pokud označíme rychlost hlavní osy  $VelM$  a rychlost podřízené osy  $VelS$ , pak platí (bez fázování):

$$VelS = VelM \cdot \frac{RatioNumerator}{RatioDenominator}$$

Tento režim osy je často nazýván elektronická převodovka. Poloha a zrychlení podřízené osy je dopočítávána konzistentně s uvedenou rychlostí.

Režim převodovky lze ukončit zapnutím jiného pohybu na podřízené ose v režimu aborting nebo spuštěním bloku MC\_GearOut.

Při aktivaci funkce převodovky (tj. v okamžiku, kdy blok MC\_GearIn převeze řízení osy) nemusí rychlost (a popřípadě i zrychlení pokud jej požadujeme spojitě, tj.  $jerk <> 0$ ) odpovídat požadované hodnotě rychlosti (a popř. i zrychlení) hlavní osy a převodovému

poměru. V takovém případě nastává přechodový děj, kdy rychlost ještě neodpovídá převodovému poměru - tzv. RampIn režim. V tomto RampIn režimu jsou použity parametry *Acceleration*, *Deceleration*, *Jerk* a blok řídí podřízenou osu tak, aby se co nejdříve (s respektováním omezení na zrychlení a popř. jerk) dostala do synchronního stavu.

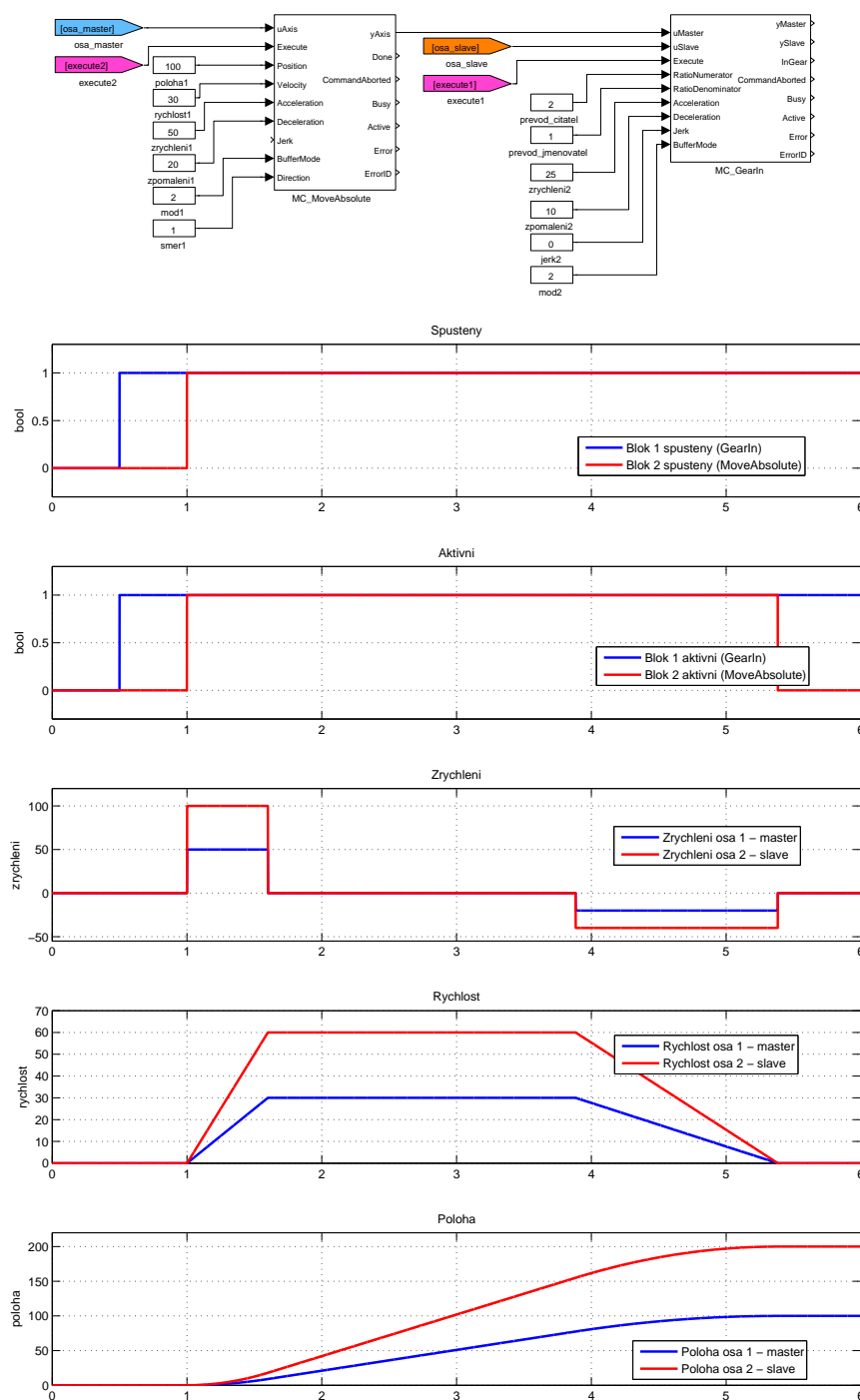
## Vstupy

<i>uMaster</i>	Odkaz na hlavní osu	<i>reference</i>
<i>uSlave</i>	Odkaz na podřízenou osu	<i>reference</i>
<i>Execute</i>	Náběžná hrana aktivuje blok	<i>bool</i>
<i>RatioNumerator</i>	Převodový poměr - číselník (podřízená osa)	<i>long</i>
<i>RatioDenominator</i>	Převodový poměr - jmenovatel (hlavní osa)	<i>long</i>
<i>Acceleration</i>	Maximální povolené zrychlení [ $\text{unit}/\text{s}^2$ ]	<i>double</i>
<i>Deceleration</i>	Maximální povolené zpomalení [ $\text{unit}/\text{s}^2$ ]	<i>double</i>
<i>Jerk</i>	Maximální povolená změna zrychlení [ $\text{unit}/\text{s}^3$ ]	<i>double</i>
<i>BufferMode</i>	Režim převzetí osy	<i>long</i>
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

## Výstupy

<i>yMaster</i>	Odkaz na hlavní osu	<i>reference</i>
<i>ySlave</i>	Odkaz na podřízenou osu	<i>reference</i>
<i>InGear</i>	Příznak dosažení požadované rychlosti řízenou osou	<i>bool</i>
<i>CommandAborted</i>	Příznak přerušení funkce bloku	<i>bool</i>
<i>Busy</i>	Příznak, že algoritmus ještě neskončil	<i>bool</i>
<i>Active</i>	Příznak, že blok řídí osu	<i>bool</i>
<i>Error</i>	Příznak chyby	<i>bool</i>
<i>ErrorID</i>	Kód chyby	<i>error</i>
	i ..... obecná chyba systému REX	

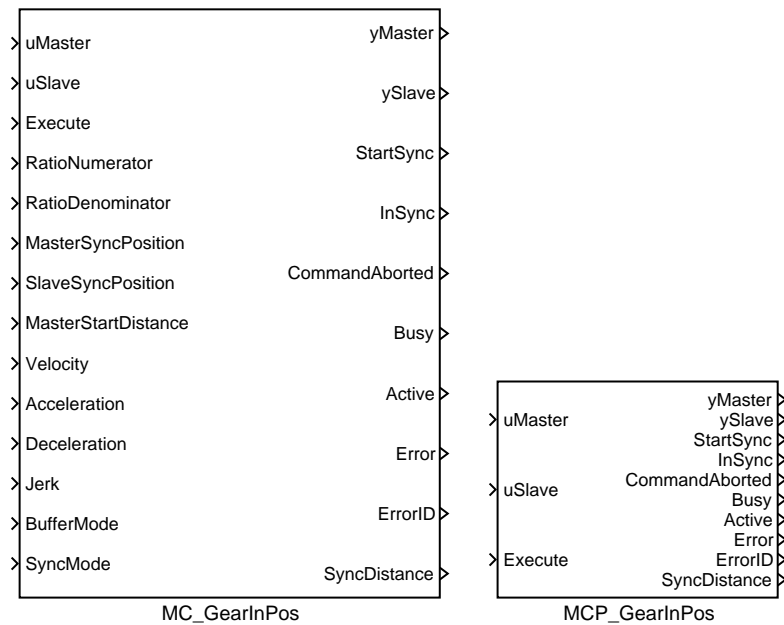
## Příklad



## MC\_GearInPos, MCP\_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici

Symbody bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_GearInPos a MCP\_GearInPos mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_GearInPos** zapne řízení podřízené osy tak, že poměr rychlostí hlavní a podřízené osy je v poměru **RatioNumerator:RatioDenominator**. Na rozdíl od bloku **MC\_GearIn** je u tohoto bloku zachováván i poměr vzdáleností a je určena i poloha obou os v okamžiku zasynchronizování, tzn. platí

$$\frac{SlavePosition - SlaveSyncPosition}{MasterPosition - MasterSyncPosition} = \frac{RatioNumerator}{RatioDenominator}$$

Pokud v okamžiku aktivace bloku (tj. v okamžiku náběžné hrany na vstupu **Execute** a v případě buffered režimu ještě po ukončení předchozích bloků) neodpovídá poloha podřízené osy poloze hlavní osy (dle výše uvedeného vzorce), je spuštěn proces synchronizace (indikováno výstupem **StartSync**). Během tohoto procesu je generována trajektorie tak,

aby došlo co nejrychleji k synchronizaci a nebyly překročeny maximální hodnoty rychlosti, zrychlení a změny zrychlení dané parametry Velocity, Acceleration, Deceleration, Jerk. Po dokončení synchronizace se již tyto limity neuplatňují. Pokud je parameter **MasterStartDistance=0**, proces synchronizace se spouští hned po aktivaci bloku (vstupem Execute). V opačném případě se proces synchronizace spouští ve chvíli, kdy hlavní osa dosáhne polohy v intervalu **MasterSyncPosition ± MasterStartDistance**.

Poznámky:

1. Proces synchronizace používá dva algoritmy: I. Algoritmus shodný s blokem **MC\_MoveAbsolute**, přičemž trajektorie je v každém kroku přepočítána tak, aby koncová rychlost odpovídala aktuální rychlosti hlavní osy. II. Poloha (i rychlost i zrychlení) se generuje jako v případě synchronního pohybu, ale k hodnotám je po určitou dobu přičítána hodnota vhodného polynomu 5. stupně, tak aby nenastal skok v poloze, v rychlosti a v případě zadaného jerku ani ve zrychlení a aby na konci doby přičítání měl polynom nulovou hodnotu. Doba je volena tak, aby polynom nepřekročil požadované meze rychlosti, zrychlení a jerku. První metoda nevede k úspěšné synchronizaci, pokud se hlavní osa pohybuje se zrychlením, druhá metoda negarantuje dodržení limitů na rychlost, zrychlení a jerk, navíc vyžaduje ponechat určitou rychlost a zrychlení na synchronní část, takže je obecně delší. Proto se oba algoritmy vhodně kombinují.

2. Parametry bloku (ať už start procesu synchronizace nebo limity na rychlost a zrychlení) je nutné volit tak, aby podřízená osa byla v poloze **SlaveSyncPosition** (přibližně) ve stejný okamžik, jako hlavní osa v poloze **MasterSyncPosition**. V opačné případě může nastat neočekávaný pohyb podřízené osy nebo porušení zadaných limitů. Pohyb hlavní osy může být libovolný, ale v konkrétní aplikaci bývá obvykle dobře definován. Správným nastavením parametrů je tedy možné zajistit vhodný průběh synchronizace.

## Vstupy

<b>uMaster</b>	Odkaz na hlavní osu	<b>reference</b>
<b>uSlave</b>	Odkaz na podřízenou osu	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>RatioNumerator</b>	Převodový poměr - číselník (podřízená osa)	<b>long</b>
<b>RatioDenominator</b>	Převodový poměr - jmenovatel (hlavní osa)	<b>long</b>
<b>MasterSyncPosition</b>	Poloha hlavní osy v okamžiku zasynchronizování [unit]	<b>double</b>
<b>SlaveSyncPosition</b>	Poloha podřízené osy v okamžiku zasynchronizování	<b>double</b>
<b>MasterStartDistance</b>	Definuje polohu hlavní osy pro spuštění procesu synchronizace	<b>double</b>
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	<b>double</b>
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	<b>double</b>

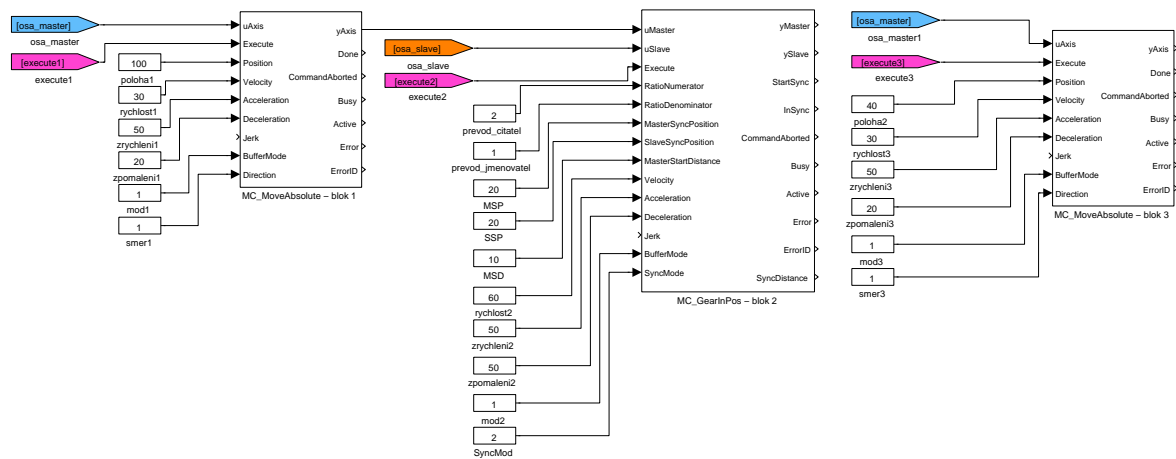
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
1 .....	Aborting (nový blok se spustí okamžitě)	
2 .....	Buffered (nový blok se spustí po dokončení předchozího)	
3 .....	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4 .....	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5 .....	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6 .....	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>SyncMode</b>	Režim synchronizace (pouze cyklické osy)	<b>long</b>
1 .....	synchronizovat zrychlením	
2 .....	synchronizovat na nejbližší bod	
3 .....	synchronizovat zpomalením	

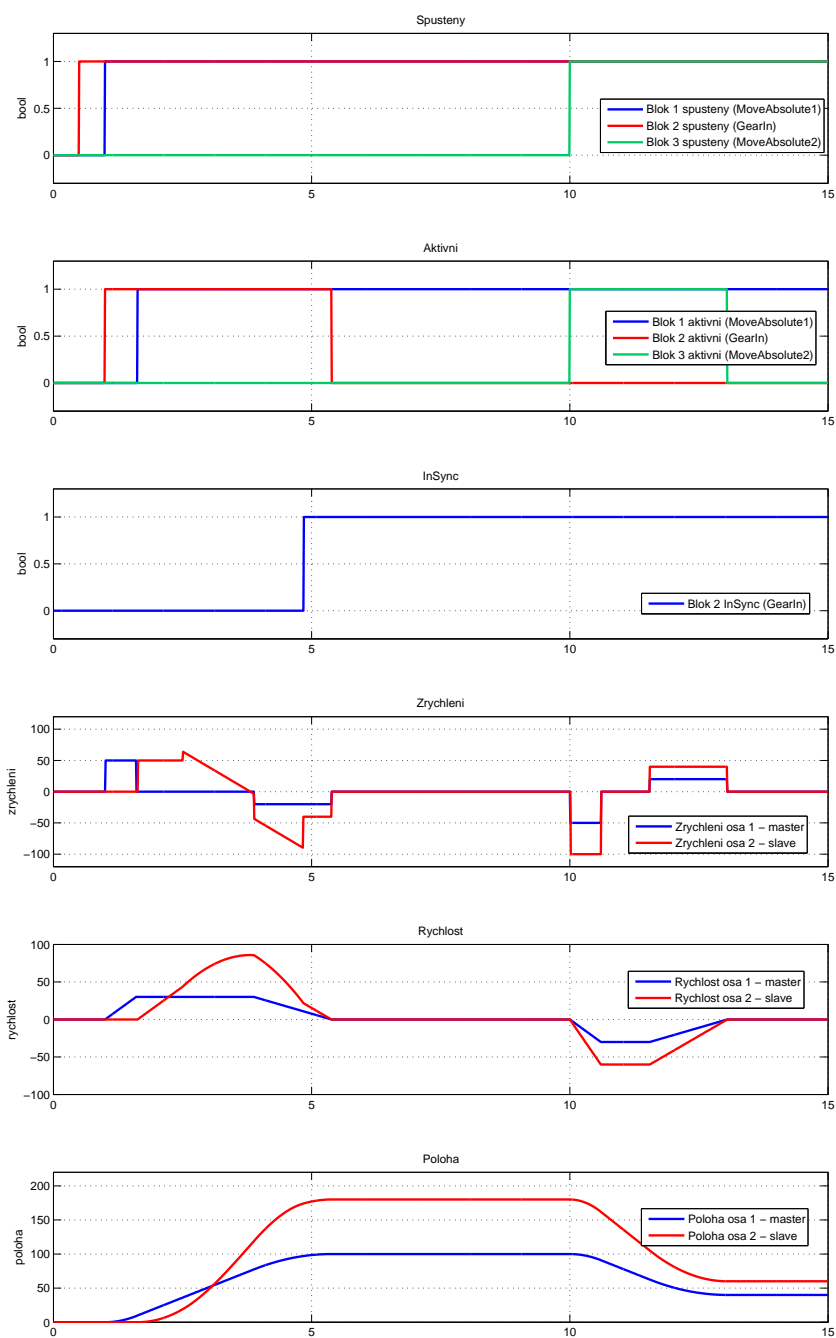
## Výstupy

<b>yMaster</b>	Odkaz na hlavní osu	<b>reference</b>
<b>ySlave</b>	Odkaz na podřízenou osu	<b>reference</b>
<b>StartSync</b>	Začíná synchronizace pohybu	<b>bool</b>
<b>InSync</b>	Příznak dosažení profilu vačky podřízenou osou	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i .....	obecná chyba systému REX
<b>SyncDistance</b>	Odchylka v poloze podřízené osy od synchronizované polohy [unit/s]	<b>double</b>



## Příklad

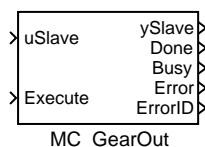




## MC\_GearOut – Vypnutí konstantního převodového poměru

Symbol bloku

Licence: [MOTION CONTROL](#)



### Popis funkce

Blok **MC\_GearOut** ukončuje režim elektronické převodovky zapnutý blokem **MC\_GearIn**. Pokud žádný blok **MC\_GearIn** není aktivní, blok nemá žádnou funkci (a ani nehlásí chybu).

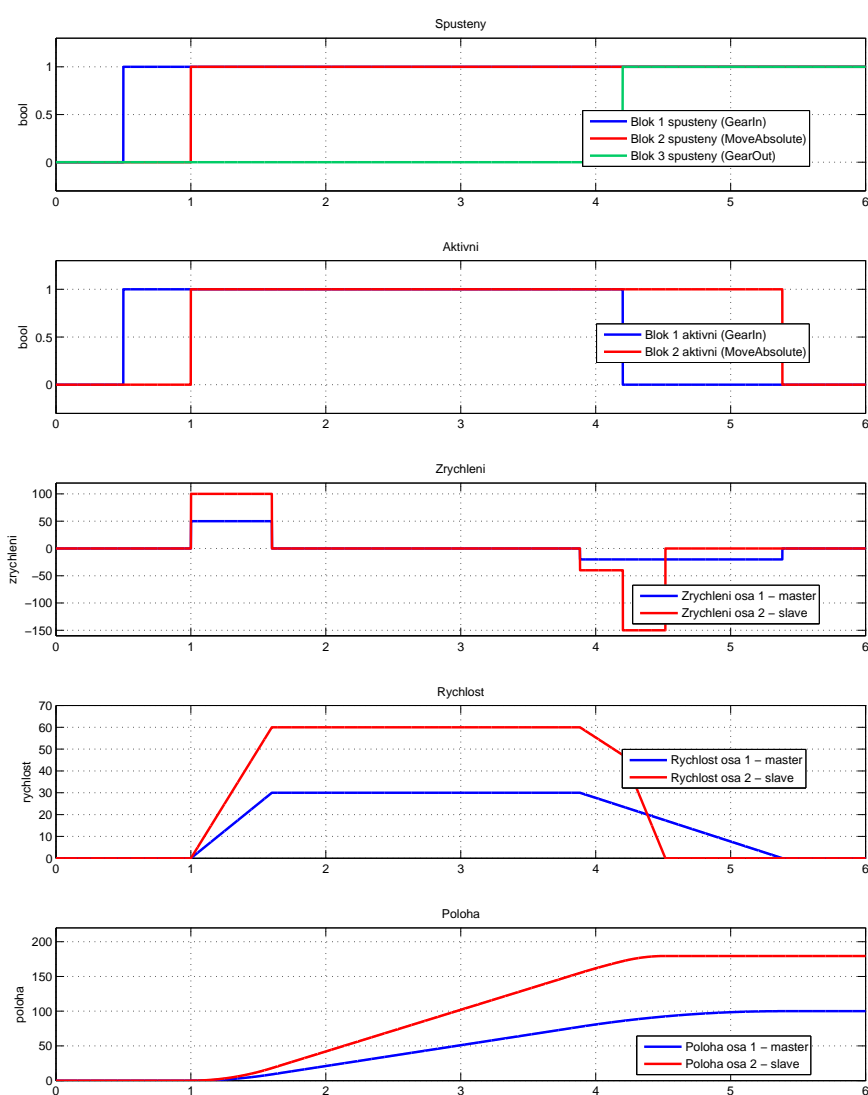
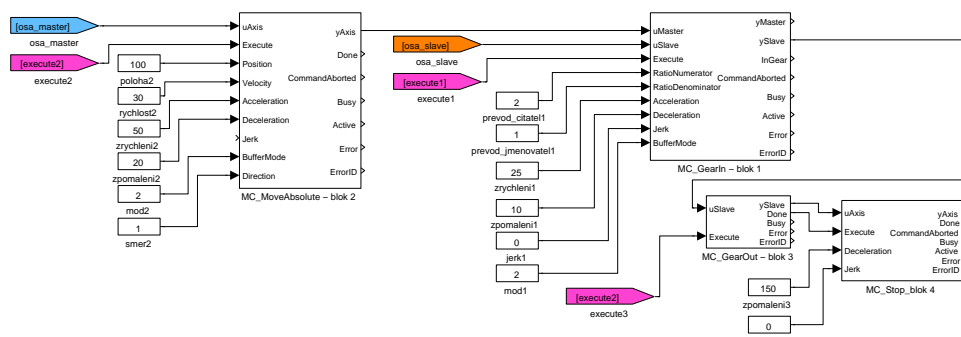
### Vstupy

<b>uSlave</b>	Odkaz na podřízenou osu	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>

### Výstupy

<b>ySlave</b>	Odkaz na podřízenou osu	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i . . . . . obecná chyba systému REX	

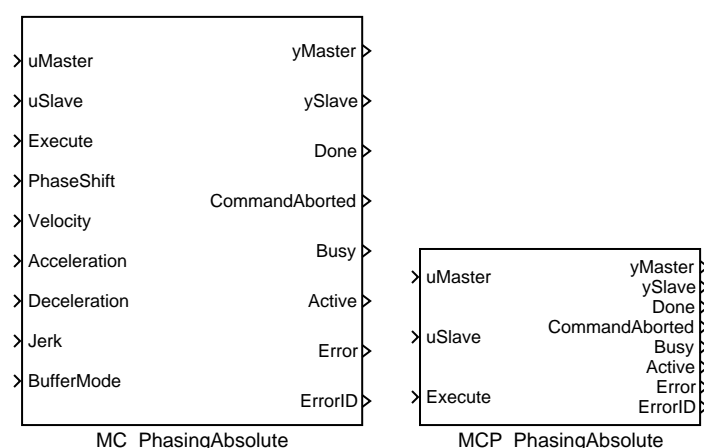
## Příklad



## MC\_PhasingAbsolute, MCP\_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice)

Symboly bloků

Licence: [MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_PhasingAbsolute a MCP\_PhasingAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_PhasingAbsolute** zavádí další posunutí na hlavní ose pro vačku (blok **MC\_CamIn**) a převodovku (blok **MC\_GearIn**). Blok funguje velice podobně bloku **MC\_MoveSuperimposed** (tj. generuje pohyb z bodu 0 do bodu **PhaseShift** s respektováním omezení na rychlost, zrychlení a popřípadě jerk, tak aby pohyb trval co nejkratší dobu) s tím rozdílem, že generovaná poloha/rychlost/zrychlení se nepřičítá ke skutečné poloze hlavní osy, ale přičítá se k ní jen z pohledu bloku **MC\_CamIn** a **MC\_GearIn**.

Poznámka 1: Tento blok je analogie natočení mechanické vačky na hřídeli o úhel **PhaseShift**.

### Vstupy

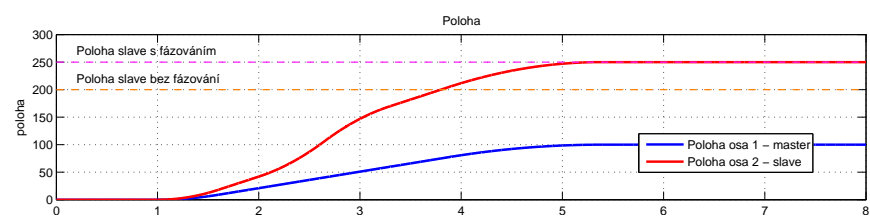
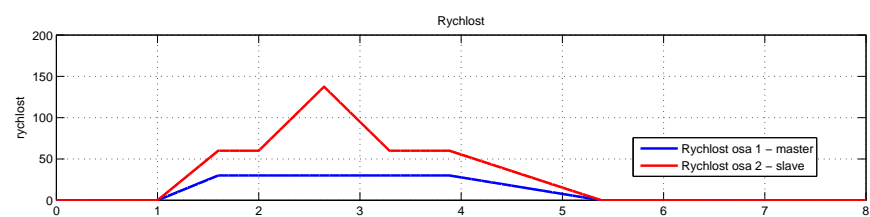
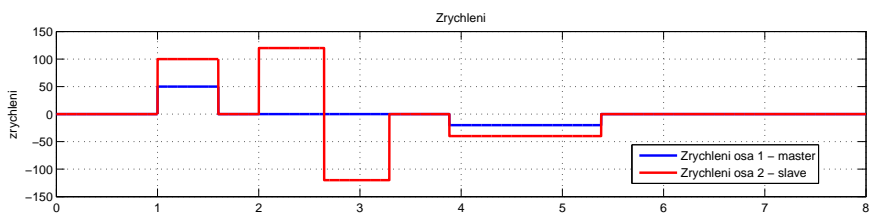
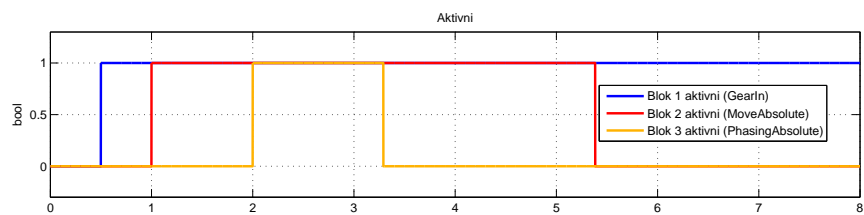
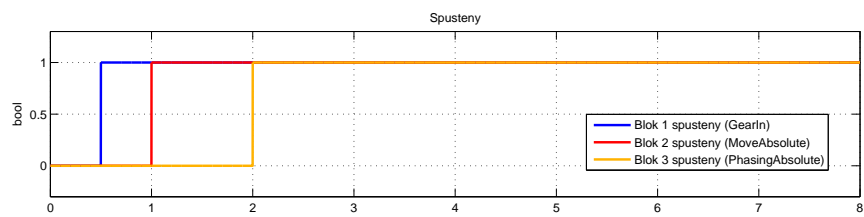
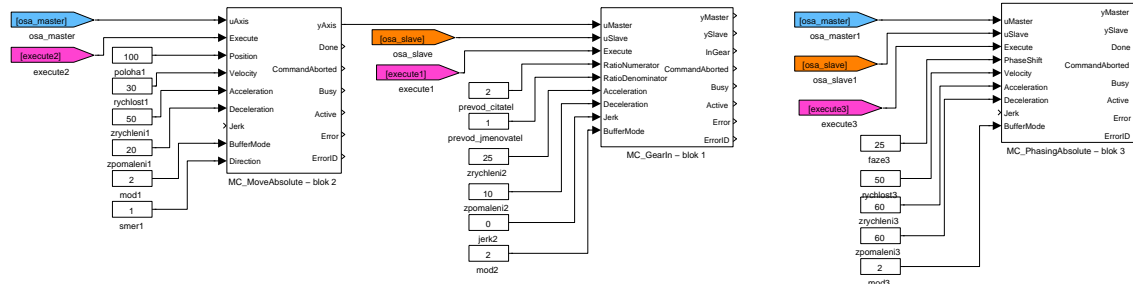
<b>uMaster</b>	Odkaz na hlavní osu	reference
<b>uSlave</b>	Odkaz na podřízenou osu	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>PhaseShift</b>	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	double
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	double
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double

<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	<b>double</b>
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	<b>double</b>
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
	1 ..... aborting	
	2 ..... buffered	

## Výstupy

<b>yMaster</b>	Odkaz na hlavní osu	<b>reference</b>
<b>ySlave</b>	Odkaz na podřízenou osu	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i ..... obecná chyba systému REX	

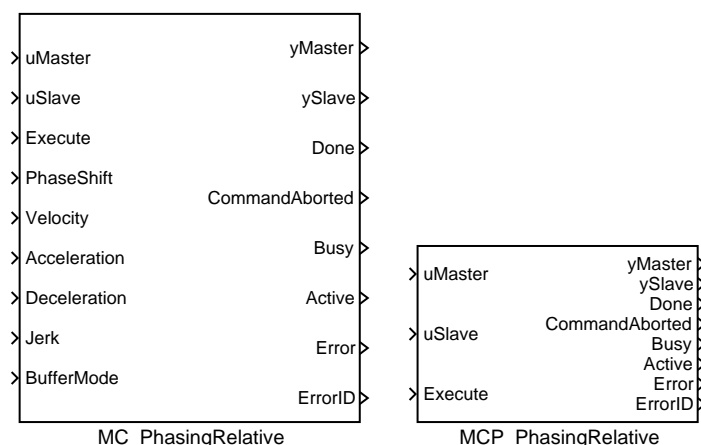
## Příklad



## MC\_PhasingRelative, MCP\_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění)

Symbody bloků

Licence: [MOTION CONTROL](#)



## Popis funkce

*Bloky MC\_PhasingRelative a MCP\_PhasingRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_PhasingRelative** zavádí další posunutí na hlavní ose pro vačku (blok **MC\_CamIn**) a převodovku (blok **MC\_GearIn**). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu **Execute**) přičte hodnota parametru **PhaseShift**. Blok funguje velice podobně bloku **MC\_MoveSuperimposed** (tj. generuje pohyb z bodu 0 do bodu **PhaseShift** s respektováním omezení na rychlost, zrychlení a popřípadě jerk, tak aby pohyb trval co nejkratší dobu) s tím rozdílem, že generovaná poloha/rychlost/zrychlení se nepřičítá ke skutečné poloze hlavní osy, ale přičítá se k ní jen z pohledu bloku **MC\_CamIn** a **MC\_GearIn**.

Poznámka 1: Tento blok je analogie natočení mechanické vačky na hřídeli o úhel **PhaseShift**.

## Vstupy

uMaster	Odkaz na hlavní osu	reference
uSlave	Odkaz na podřízenou osu	reference
Execute	Náběžná hrana aktivuje blok	bool
PhaseShift	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	double

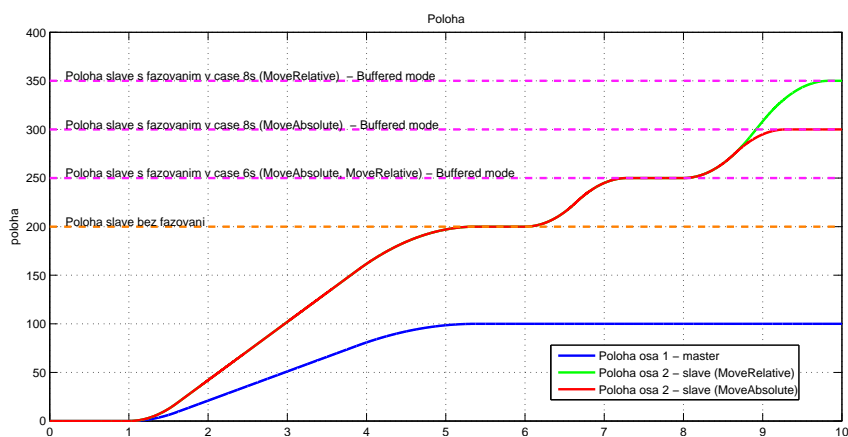


Velocity	Maximální povolená rychlost [unit/s]	double
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
BufferMode	Režim převzetí osy	long
	1 ..... aborting	
	2 ..... buffered	

## Výstupy

yMaster	Odkaz na hlavní osu	reference
ySlave	Odkaz na podřízenou osu	reference
Done	Příznak dokončení algoritmu	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	

## Příklad





## Kapitola 18

# MC\_COORD – Koordinované řízení pohybu

### Obsah

RM_AxesGroup – Skupina os pro koordinované řízení pohybu . . .	489
RM_Feed – * MC krmič ??? . . . . .	493
RM_Gcode – * CNC řízení pohybu . . . . .	494
MC_AddAxisToGroup – Přidání osy do skupiny os . . . . .	496
MC_UngroupAllAxes – Odebrání všech os ze skupiny . . . . .	497
MC_GroupEnable – Převedení skupiny do stavu GroupStandby . .	498
MC_GroupDisable – Převedení skupiny do stavu GroupDisabled .	499
MC_SetCartesianTransform, MCP_SetCartesianTransform – Kartézská transformace . . . . .	500
MC_ReadCartesianTransform – Přčtení použité kartézské transformace . . . . .	503
MC_GroupSetPosition, MCP_GroupSetPosition – Nastavení polohového offsetu skupiny os . . . . .	504
MC_GroupReadActualPosition – Aktuální poloha skupiny os . . . .	506
MC_GroupReadActualVelocity – Aktuální rychlost skupiny os . . .	507
MC_GroupReadActualAcceleration – Aktuální zrychlení skupiny os	508
MC_GroupStop, MCP_GroupStop – Zastavení koordinovaného pohybu	509
MC_GroupHalt, MCP_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné) . . . . .	512
MC_GroupInterrupt, MCP_GroupInterrupt – Přerušení pohybu skupiny os . . . . .	517
MC_GroupContinue – Pokračování v přerušeném pohybu . . . . .	518
MC_GroupReadStatus – Stav skupin os . . . . .	519
MC_GroupReadError – Chyby ve skupině os . . . . .	521
MC_GroupReset – Nulování chyb os ve skupině . . . . .	522

MC_MoveLinearAbsolute, MCP_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice) . . . . .	523
MC_MoveLinearRelative, MCP_MoveLinearRelative – Pohyb do pozice po přímkách (relativní souřadnice) . . . . .	527
MC_MoveCircularAbsolute, MCP_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice) . . . . .	531
MC_MoveCircularRelative, MCP_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice) . . . . .	536
MC_MoveDirectAbsolute, MCP_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice) . . . . .	541
MC_MoveDirectRelative, MCP_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice) . . . . .	545
MC_MovePath, MCP_MovePath – Generování obecné trajektorie v prostoru . . . . .	549
MC_GroupSetOverride, MCP_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině . . . . .	552

---

Tato kategorie bloků zahrnuje bloky pro koordinovaný pohyb více os, jak jsou definovány ve specifikaci PLCopen. Pro bloky této kategorie platí stejné obecné zásady, jaké byly uvedeny v kategorii MC\_SINGLE (bloky pro řízení pohybu v jedné ose). Dále platí, že sdílená struktura AXES\_GROUP\_REF popisovaná v PLCopen je reprezentována blokem [RM\\_AxesGroup](#).

Pohybové bloky jsou opět ve variantě s parametry na vstupech a v parametrech, tj. s prefixem MC\_ a MCP\_. Protože nelze dopředu říci, kolik bude souřadnic (je to parametr bloku [RM\\_AxesGroup](#)), musí být údaje o poloze vektorové parametry, respektive vstupy (předpokládá to i norma PLCopen). Pro variantu MC\_ se na příslušný vstup připojí blok [RTOV](#) (popřípadě řetězec těchto bloků, pokud je vektor delší), který umožňuje složit vektor ze skalárních vstupů. Pro výstupní vektory se analogicky použije blok [VTOR](#). Oba tyto bloky se nacházejí v knihovně MATRIX. V případě MCP\_ varianty se příslušný vektor zapisuje přímo do bloku jako jeho parametr.

Rychlost, zrychlení, zpomalení a jerk se zadává skalárně (jedno číslo, nikoliv vektor) a je to vždy ve smyslu tečném k výsledné trajektorii. Hodnota se kontroluje jen při spuštění bloku (zda není větší, než zadané maximum v bloku [RM\\_AxesGroup](#)), takže v některých případech může být rychlost vyšší.

## Transformace a souřadné systémy

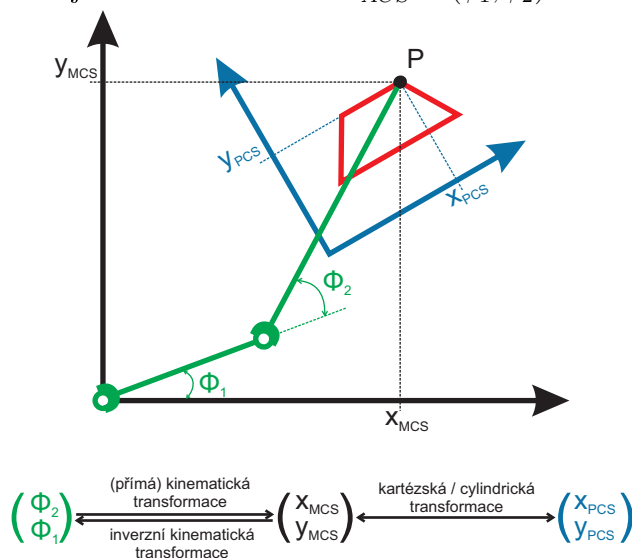
Transformace mezi polohou v kartézských souřadnicích a polohou motorů je obecně nelineární a není tak jednoduše možné určit, zda naplánovaná trajektorie nepřekračuje limity na rychlost a zrychlení v jednotlivých osách (motorech). Proto se provede vygenerování trajektorie dle zadaných parametrů a její spuštění. Pokud dojde k překročení limitu polohy, rychlosti nebo zrychlení v ose, skupina přejde do chybového stavu a spustí se zastavovací sekvence po původní trajektorii. Pokud přesto dojde k překročení systémového

limitu polohy, rychlosti nebo zrychlení na některé z os, jsou všechny osy skupiny brzděny s maximální možnou intenzitou samostatně a tedy již ne po požadované trajektorii.

Norma PLCopen dále rozlišuje tři souřadné systémy. To, jaký souřadný systém se použije (tj. ve kterém je zadávána poloha) určuje parametr **CoordSystem** příslušného pohybového bloku. Možnosti jsou:

- **ACS** (Axis Coordinate System) – V tomto souřadném systému jsou jednotlivé souřadnice přímo polohy jednotlivých motorů/připojených os.
- **MCS** (Machine Coordinate System) – Pravoúhlý souřadný systém spojený se strojem. Předpokládá se, že tento souřadný systém obsahuje tři (popř. dvě pro planární stroje) na sebe navzájem kolmé osy pro určení polohy (tzv. kartézský souřadný systém; obvykle se používá pravotočivý) a dále až 3 souřadnice pro určení natočení koncového efektoru stroje. Pro reprezentaci směru/úhlu natočení se obvykle používá směrový vektor (tři souřadnice polohy, přičemž směr od bodu  $[0, 0, 0]$  k zadanému bodu určuje směr koncového efektoru) nebo tzv. eulerovy úhly (obdoba zeměpisné šířky, zeměpisné délky a azimutu na zeměkouli). Vazbu mezi MCS a ACS určuje kinematická transformace (tj. některý z bloků **MC\_SetKinTransform\_XXX**), která tím pak také určuje jak je MCS definován.
- **PCS** (Product Coordinate System) – Pravoúhlý souřadný systém spojený s výrobkem. Je to vlastně posunutý a otočený MCS. Někdy se PCS oproti MCS pohybuje. Vazbu mezi MCS a PCS zajišťuje (pro statický případ) blok **MC\_SetCartesianTransform**

Níže uvedený obrázek ukazuje reprezentaci bodu P (ležícím na obrobku – červený lichoběžník) v různých souřadných systémech. V PCS je dán polohou  $P_{PCS} = (x_{PCS}, y_{PCS})$ . Translací a rotací lze převést do MCS jako  $P_{MCS} = (x_{MCS}, y_{MCS})$ . Nakonec může být popsán v ACS jako dvojice úhlů natočení os  $P_{ACS} = (\phi_1, \phi_2)$



## Druhy pohybů

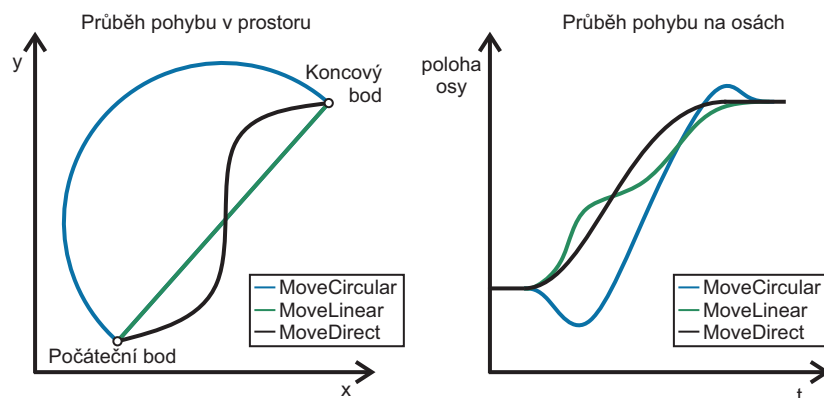
Generované pohyby lze rozdělit na dva základní druhy:

**Pohyb z bodu do bodu (Point - to - Point movements)** – Při tomto pohybu se chceme dostat do zadané pozice co nejrychleji. To se provede tak, že se pomocí inverzní kinematiky spočte požadované natočení pohonu, které je nutné pro dosažení zadané pozice. Každý pohon se pak bude do této pozice řídit nezávisle na ostatních a to s maximální rychlostí a zrychlením nastavenými jako limity na osách. Pokud se budou takto pohybovat všechny osy, tak výsledná trajektorie mezi počátečním a koncovým bodem v MCS není definována. Je tedy nutné dát pozor, jestli je bezpečné tento blok použít. Použitelné bloky pro tento pohyb jsou:

- `MC_MoveDirectAbsolute`
- `MC_MoveDirectRelative`

**Pohyb po dané trajektorii (Cartesian Path Movement)** – Podporované trajektorie jsou pohyb po přímce, po kružnici a po spline křivce. Trajektorie je vygenerována obecně v prostoru a následně převedena inverzní kinematikou do souřadnicového systému pohonů ACS. Takto přepočtenou trajektorii je pak již možné přivést jako požadovanou hodnotu na regulátory pohonů. Použitelné bloky pro tento pohyb jsou:

- `MC_MoveLinearAbsolute`
- `MC_MoveLinearRelative`
- `MC_MoveCircularAbsolute`
- `MC_MoveCircularRelative`
- `MC_MovePath`



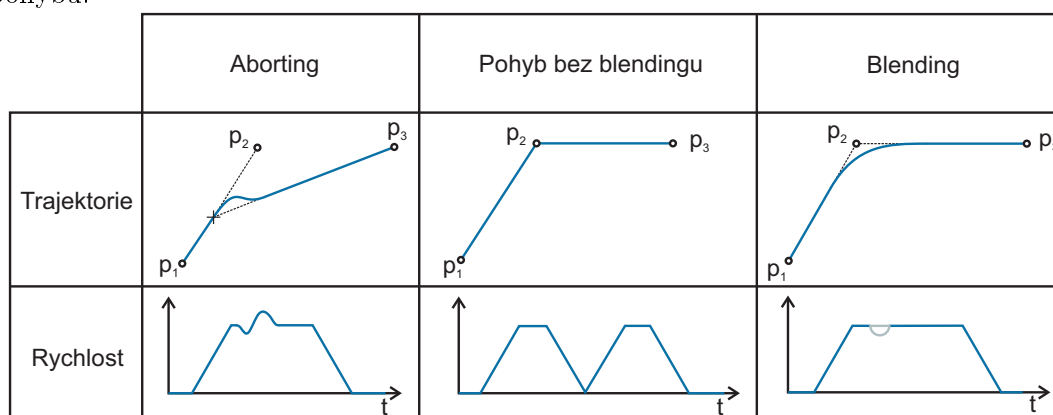
## Míchání pohybů

Důležitá část při generování pohybu je navazování po sobě jdoucích pohybových příkazů na skupinu os (blending). Bez zapnutí blending módů provede skupina os vždy pohyb do

požadované pozice, kde zastaví přesně na cílovém místě a uvede se do klidu. Následující pohyb se do této doby nespustí. Po zastavení je tedy nutné se opět znovu rozjet. V mnoha aplikacích může být toto neustálé zastavování a rozjíždění nežádoucí a pohyby je nutné nějak prokládat. Blending je vhodné použít například pro:

- Zrychlení výroby.
- Generování hladších trajektorií a tím zmenšení mechanického namáhání.
- Některé aplikace vyžadují pohyby s konstantní rychlostí (např. nanášení lepidla, barvení, svařování, ...)

Všechny výše uvedené příklady lze vyřešit použitím různých druhů blendingů. Společné pro všechny druhy je modifikace původní trasy v hladkou trajektorii bez rohů (zastavování). Na obrázku níže jsou uvedeny příklady některých možných navazování pohybu.



## Přehled režimů převzetí osy

Tabulka režimů převzetí osy:

režim	funkce
1: Aborting	Start následujícího bloku okamžitě.
2: Buffered	Start následujícího bloku ihned po dokončení předcházejícího (dojde k zastavení pohybu).
3: BlendingLow	Rychlost je proložena s menší z rychlostí obou bloků.
4: BlendingHigh	Rychlost je proložena s větší z rychlostí obou bloků.
5: BlendingPrevious	Rychlost je proložena s rychlostí prvního bloku.
6: BlendingNext	Rychlost je proložena s rychlostí druhého bloku.

## Přehled režimů míchání pohybu

Tabulka režimů míchání pohybu:

režim	funkce
1: TMNone	Nevkládá žádnou přechodovou křivku.
2: TMStartVelocity	Přechod s danou počáteční rychlostí.
3: TMConstantVelocity	Přechod s danou konstantní rychlostí.
4: TMCornerDistance	Přechod s daným corner distance.
5: TMMaxCornerDeviation	Přechod s daným maximálním corner deviation.
11: Milanův režim	Nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku.

### Synchronizace

Pro správnou synchronizaci je nutné (podrobněji viz. popis skupiny MC\_SINGLE), aby všechny bloky přiřazené do jedné skupiny (k jednomu bloku) **RM\_AxesGroup** byly buď v jedné úloze/výkresu/tasku nebo v několika úlohách se stejnou periodou vzorkování. Blok **RM\_AxesGroup** by měl být vykonán jako první, po všech k němu přiřazených blocích **RM\_Axis**. Pokud se na některé ose využívá interpolace (blok **RM\_AxisSpline**), je nutné, aby se blok **RM\_AxesGroup** vykonával mezi bloky **RM\_Axis** a **RM\_AxisOut** všech k němu přiřazených os (respektive všech těch, kde se využívá **RM\_AxisSpline**).



## RM\_AxesGroup – Skupina os pro koordinované řízení pohybu

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

Blok **RM\_AxesGroup** je základní blok skupiny os pro koordinovaný pohyb. Představuje sdílenou strukturu, kde jsou uloženy všechny stavy a parametry skupiny. Algoritmus tohoto bloku představuje kontrolu nastavených mezí, havarijní zastavení v případě potřeby, předávání dat do a z bloků **RM\_Axis** podřízených os, přepočítání všech stavů a výstupů pro případ, že žádný blok není aktivní, ale je potřeba generovat pohyb (např. zastavení v PCS a částečně i algoritmus pro generování požadované trajektorie). Výstupem tohoto bloku jsou pouze stavové a pomocné signály (viz dále).

Implicitní hodnoty parametrů (zejména limity na rychlost a zrychlení) jsou záměrně nastaveny na 0, což je nedovolená hodnota. Všechny parametry tak musí nastavit uživatel podle skutečných možností připojeného motoru a stroje.

### Parametry

<b>McsCount</b>	Počet os v MCS	↓1 ↑6 ⊖6	long
<b>AcsCount</b>	Počet os v ACS	↓1 ↑16 ⊖6	long
<b>PosCount</b>	Počet polohových os	↓1 ↑6 ⊖3	long
<b>Velocity</b>	Maximální povolená rychlost [unit/s]		double
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]		double
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]		double

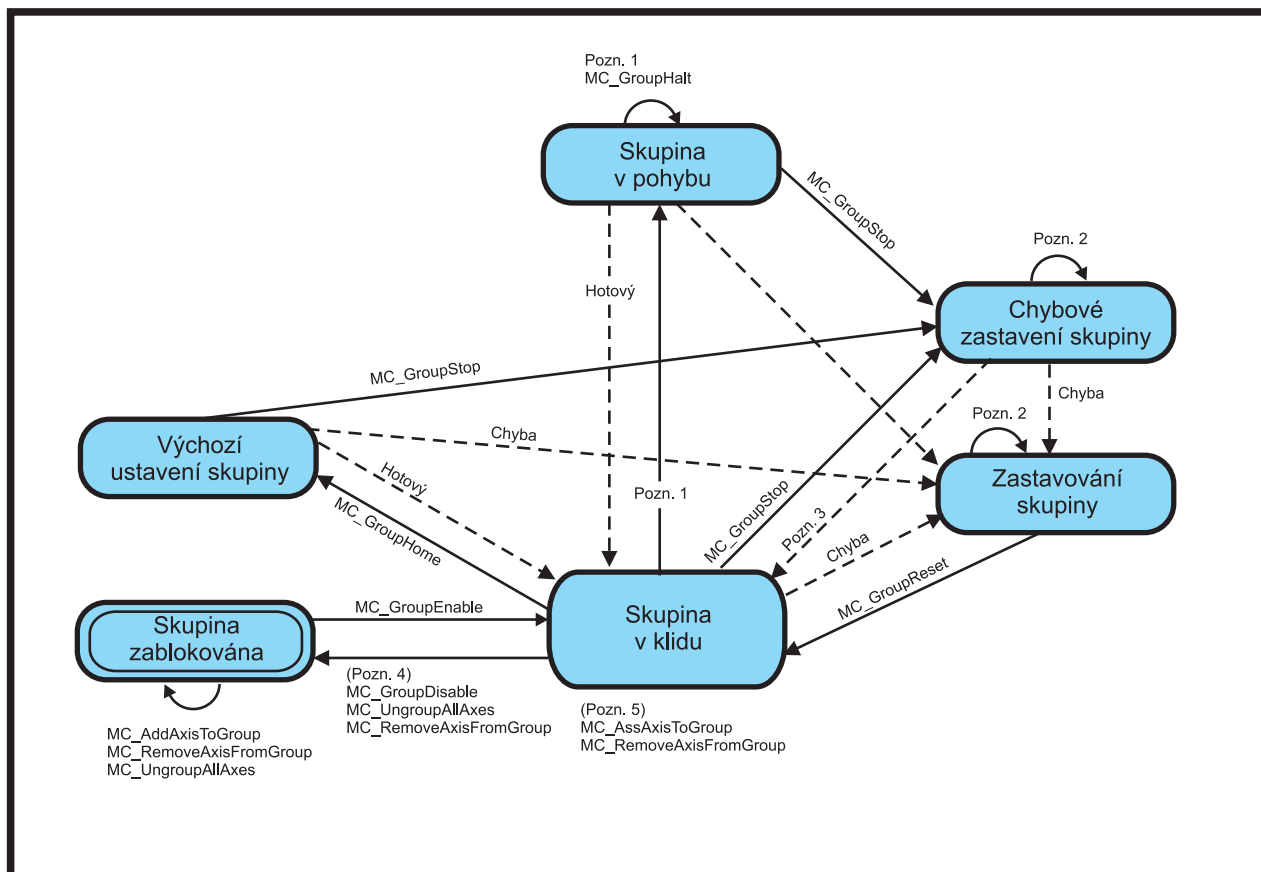
### Výstupy

<b>refGroup</b>	Odkaz na skupinu os	reference
<b>refPos</b>	Vektor poloh, rychlostí a zrychlení	reference
<b>iState</b>	Stav skupiny	long
	0 ..... Disabled (skupina blokována)	
	1 ..... Standby (skupina připravena)	
	2 ..... Homing (hledání výchozí polohy)	
	6 ..... Moving (koordinovaný pohyb)	
	7 ..... Stopping (zastavování nebo dočasné blokování)	
	8 ..... Error stop (zastavování nebo blokování skupiny po chybě)	

ErrorID      Kód chyby  
i        obecná chyba systému REX

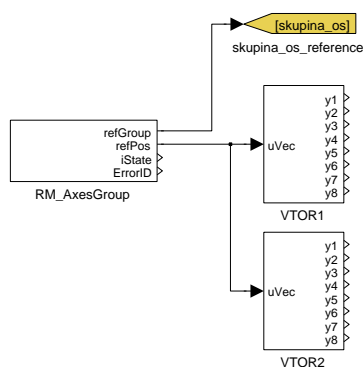
error

### Stavový diagram skupiny os



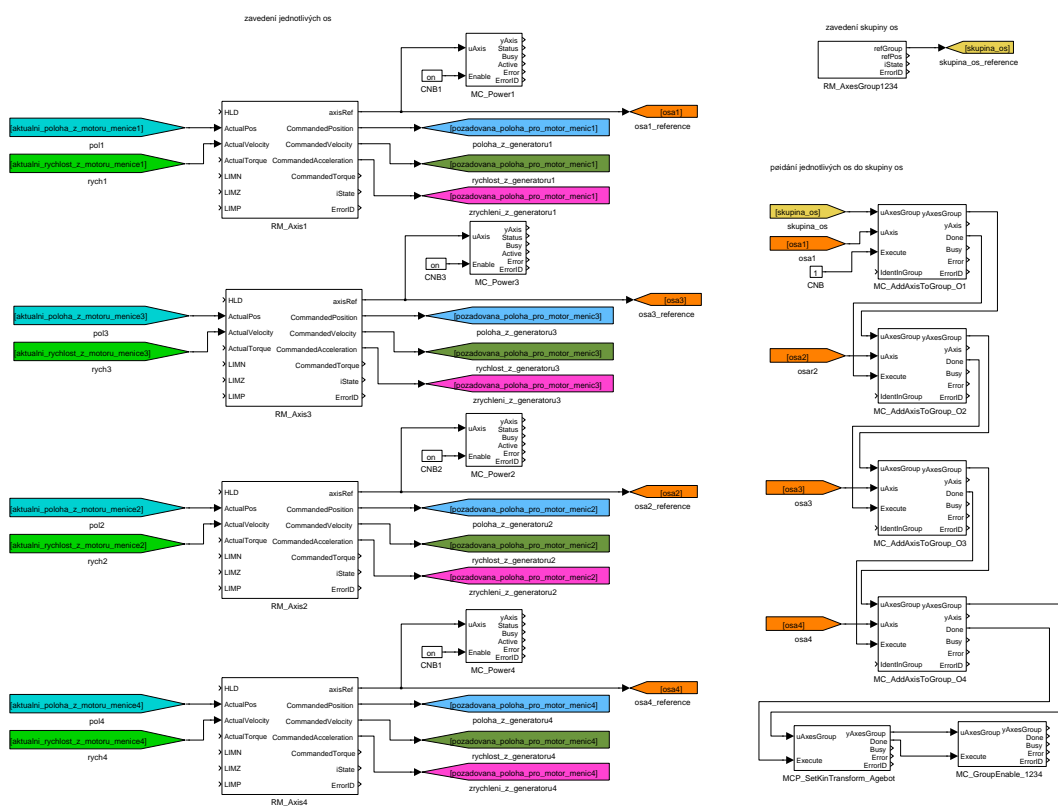
## Čtení z výstupu refPos

Z výstupu `refPos` bloku `RM_AxesGroup` lze číst aktuální polohu, rychlost a zrychlení (stejně jako u bloků `MC_GroupReadActualPosition`, `MC_GroupReadActualVelocity` a `MC_GroupReadActualAcceleration`). Výstup `refPos` je vektorový, pro získání konkrétních hodnot je nutné použít blok `VTOR` z knihovny `MATRIX`. Tento blok má 8 výstupů. Pokud je souřadnic více než 8, vyřeší se to paralelním připojením dalšího bloku `VTOR` (viz. Obr. níže) do kterého je nutné zadat jako parametr správný offset (od jakého indexu se mají dát data na výstup). Např. robot AGEBOT (jeho sestavení jako skupiny os viz. níže) má čtyři osy. Bude mít tedy čtyři souřadnice pro polohu, rychlost a zrychlení. Jedním blokem `VTOR` bychom získali pouze informaci o poloze a rychlosti (výstupy  $y_1 - y_4$  a  $y_5 - y_8$  bloku `VTOR1`), proto je nutné použití druhého bloku `VTOR`. Pokud mu nastavíme offset na hodnotu 8, tak na výstupech  $y_1 - y_4$  bloku `VTOR2` budou informace o zrychlení.



## Sestavení skupiny os

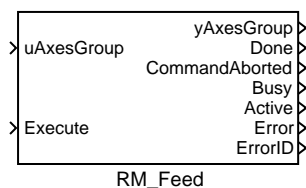
Pro správné spuštění bloku generujícího víceosý pohyb je nutné definovat jednotlivé osy `RM_Axis` a postupně je všechny přidat do skupiny os pomocí bloku `MC_AddAxisToGroup`. Dále je třeba nastavit kinematickou transformaci (dle typu stroje jeden z bloků `MC_SetKinTransform_XXX`). Kinematickou transformaci není třeba nastavovat pro pravoúhlý stroj a pro pohyby v ACS. Nakonec je třeba aktivovat skupinu os blokem `MC_GroupEnable`. Pro pohyby v PCS je nutné ještě přidat kartézskou transformaci (blok `MC_SetCartesianTransforms`). Zpět do výchozího stavu (tj. před přiřazení os do skupiny) se skupina dostane spuštěním bloku `MC_UngroupAllAxis`. Příklad sestavení osy je uveden na obrázku níže.



## RM\_Feed — \* MC krmič ???

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



## 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

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>

## Parametry

<b>Filename</b>	0		<b>string</b>
<b>VelFactor</b>	0	↓0.01 ↑100.0 ⊖1.0	<b>double</b>
<b>Relative</b>	0		<b>bool</b>
<b>CoordSystem</b>	0	↓1 ↑3 ⊖2	<b>long</b>
<b>BufferMode</b>	0	↓1 ↑6 ⊖1	<b>long</b>
<b>TransitionMode</b>	0	↓0 ↑15 ⊖1	<b>long</b>
<b>TransitionParameter</b>	0		<b>double</b>

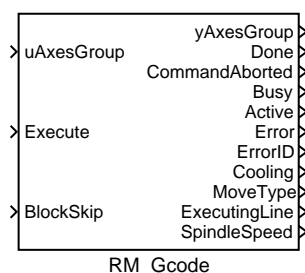
## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
<b>Aux</b>	0	<b>double</b>

## RM\_Gcode – \* CNC řízení pohybu

Symbol bloku

Licence: COORDINATED MOTION CONTROL



## 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

uAxesGroup	Odkaz na skupinu os	reference
Execute	Náběžná hrana aktivuje blok	bool
BlockSkip	Přeskočení bloku v G-kódu	bool

## Parametry

BaseDir	Adresář se soubory G-kódu	string
MainFile	Číslo hlavního programu	long
CoordSystem	0	↓1 ↑3 ⊙3 long
BufferMode	Režim převzetí osy	⊙1 long
	1 ..... aborting	
	2 ..... buffered	
	3 ..... blending low	
	4 ..... blending high	
	5 ..... blending previous	
	6 ..... blending next	
TransitionMode	Režim míchání pohybu	⊙1 long
	1 ..... TMNone	
	2 ..... TMStartVelocity	
	3 ..... TMConstantVelocity	
	4 ..... TMCornerDistance	
	5 ..... TMMaxCornerDeviation	
	11 ..... Smooth	

<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	double
<b>workOffsets</b>	Sady počátečních souřadnic $\odot[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$	double
<b>toolOffsets</b>	Sady korekcí nástroje $\odot[0\ 0\ 0]$	double
<b>cutterOffsets</b>	Poloměry nástroje $\odot[0\ 0\ 0]$	double

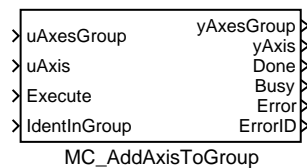
## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby i ..... obecná chyba systému REX	error
<b>Cooling</b>	Chlazení zapnuto	bool
<b>MoveType</b>	Vykonávaný příkaz	long
<b>ExecutingLine</b>	Právě prováděná řádka G-kódu	long
<b>SpindleSpeed</b>	Rychlost otáčení vřetene	long

## MC\_AddAxisToGroup – Přidání osy do skupiny os

Symbol bloku

Licence: COORDINATED MOTION CONTROL



## Popis funkce

Blok MC\_AddAxisToGroup přidá do skupiny os uAxesGroup osu uAxis. Skupina os uAxesGroup se založí pomocí bloku RM\_AxesGroup. Osa uAxis přiváděná na vstup bloku MC\_AddAxisToGroup musí být definována stejně jako jednotlivá osa RM\_Axis z knihovny MC\_SINGLE.

Poznámka 1: Každý IdentInGroup může být použit pouze jednou, jinak nastane chyba.

## Vstupy

uAxesGroup	Odkaz na skupinu os	reference
uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	reference
Execute	Náběžná hrana aktivuje blok	bool
IdentInGroup	Pořadí osy ve skupině (0=první nepřirazená)	long

## Výstupy

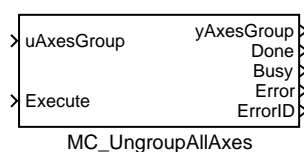
yAxesGroup	Odkaz na skupinu os	reference
yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	reference
Done	Příznak dokončení algoritmu	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	



## MC\_UngroupAllAxes – Odebrání všech os ze skupiny

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



Popis funkce

Blok **MC\_UngroupAllAxes** odebere všechny osy ze dané skupiny os. Po dokončení je změněn stav skupiny na „GroupDisabled“.

Poznámka 1: Pokud je blok spuštěn a skupina os není ve stavu „GroupDisabled“, „GroupStandBy“ nebo „GroupErrorStop“, tak je vyvolána chyba a blok není spuštěn.

Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>

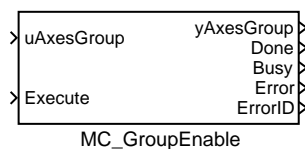
Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
i . . . . . obecná chyba systému REX		

## MC\_GroupEnable – Převedení skupiny do stavu GroupStandby

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



Popis funkce

Blok **MC\_GroupEnable** převede stav skupiny os ze stavu „GroupDisabled“ do stavu „GroupStandby“ (viz. stavový diagram u bloku [RM\\_AxesGroup](#)).

Poznámka 1: Příkaz neovlivňuje napájení žádné z os ve skupině os.

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>

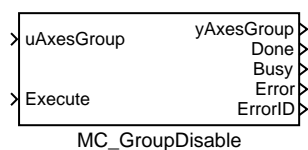
### Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i . . . . . obecná chyba systému REX	

## MC\_GroupDisable – Převedení skupiny do stavu GroupDisabled

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

Blok **MC\_GroupDisable** mění stav skupiny os na stav „GroupDisabled“. Pokud je skupina os v pohybu, tak je nejprve převedena do stavu „Stopping“. Zastavuje se s maximální možnou hodnotou zpomalení (nastavenou v bloku [RM\\_Axis](#)). Po zastavení je skupina os převedena do stavu „GroupDisabled“ (viz. stavový diagram u bloku [RM\\_AxesGroup](#)).

Poznámka 1: Příkaz neovlivňuje napájení žádné z os ve skupině os.

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>

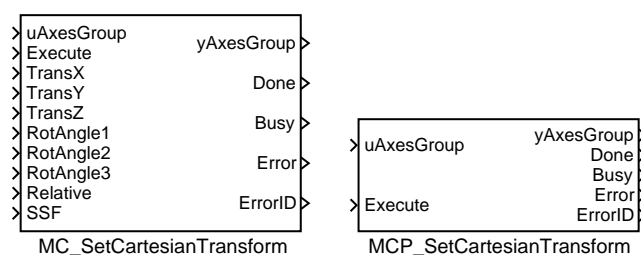
### Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i . . . . . obecná chyba systému REX	

## MC\_SetCartesianTransform, MCP\_SetCartesianTransform – Kartézská transformace

Symboly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_SetCartesianTransforms a MCP\_SetCartesianTransforms mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_SetCartesianTransform** nastavuje kartézskou transformaci mezi souřadným systémem MCS a PCS. Pomocí vstupů **Trans** se nastavuje posunutí v jednotlivých osách. Pomocí vstupů **RotAngle** se pak nastavuje rotace kolem daných os.

Poznámka 1: Na danou osu může působit více kartézských transformací současně.

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>TransX</b>	Posun v ose X	<b>double</b>
<b>TransY</b>	Posun v ose Y	<b>double</b>
<b>TransZ</b>	Posun v ose Z	<b>double</b>
<b>RotAngle1</b>	Úhel rotace podél osy X	<b>double</b>
<b>RotAngle2</b>	Úhel rotace podél osy Y	<b>double</b>
<b>RotAngle3</b>	Úhel rotace podél osy Z	<b>double</b>
<b>Relative</b>	Výběr absolutních (=false) nebo relativních (=true) souřadnic	<b>bool</b>

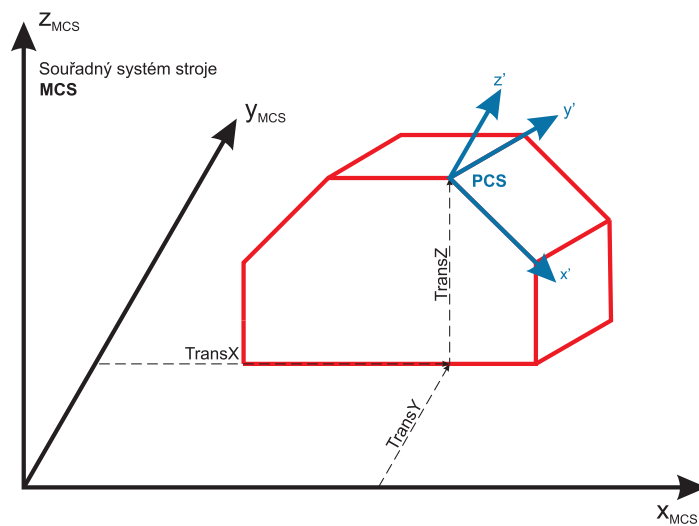
### Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>

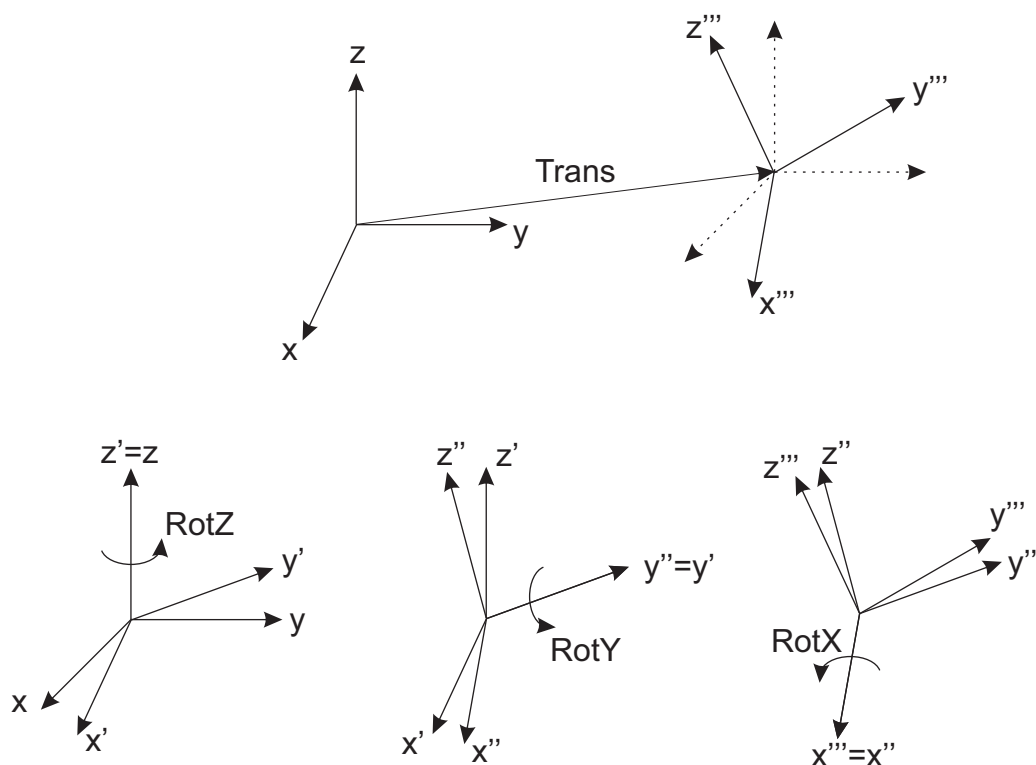
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
i	..... obecná chyba systému REX	

## Definice translace a rotace

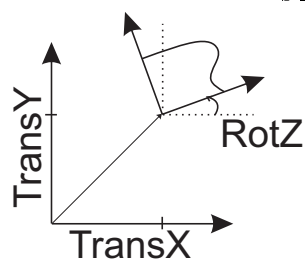
Na prvním obrázku je uveden příklad translace ze souřadného systému MCS do souřadného systému PCS.



Na druhém obrázku je pak uveden příklad rotace. Kdy celková rotace je docílena postupným provedením dílčích rotací kolem jednotlivých os.



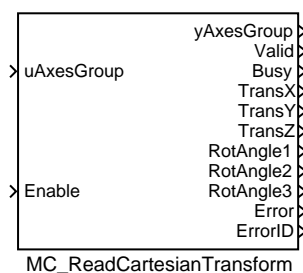
Ukázka zadání kartézské transformace (v rovině). Pro dosažení transformace na obrázku níže se spustí blok `MC_SetCartesianTransforms` s nastavenou transformací  $\{50,50,0,0,0,30\}$



## MC\_ReadCartesianTransform – Přechzení použité kartézské transformace

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

Blok `MC_ReadCartesianTransform` zpřístupňuje na výstup `TransX`, `TransY`, `TransZ`, `RotAngle1`, `RotAngle2` a `RotAngle3` aktivní kartézskou transformaci mezi souřadnými systémy MCS a PCS. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true. Pokud je aktivní více než jedna kartézská transformace, tak je na výstupu dána výsledná kartézská transformace.

### Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	bool

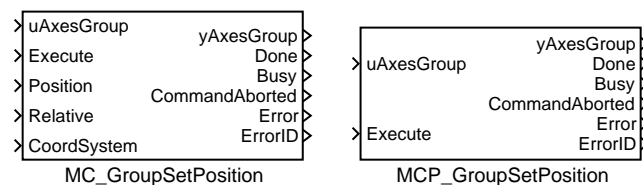
### Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	bool
<code>TransX</code>	Posun v ose X	double
<code>TransY</code>	Posun v ose Y	double
<code>TransZ</code>	Posun v ose Z	double
<code>RotAngle1</code>	Úhel rotace podél osy X	double
<code>RotAngle2</code>	Úhel rotace podél osy Y	double
<code>RotAngle3</code>	Úhel rotace podél osy Z	double
<code>Error</code>	Příznak chyby	bool
<code>ErrorID</code>	Kód chyby	error
i ..... obecná chyba systému REX		

## MC\_GroupSetPosition, MCP\_GroupSetPosition – Nastavení polohového offsetu skupiny os

Symbole bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_GroupSetPosition a MCP\_GroupSetPosition mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_GroupSetPosition** nastaví polohu všech os ve skupině bez jejich pohybu. Nové souřadnice jsou dány vstupem **Position**. Pomocí vstupu **CoordSystem** se nastaví, v jakém souřadnicovém systému se provede změna. Tato změna následně ovlivní i souřadnice ve vyšších souřadnicových systémech.

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>Position</b>	Pole souřadnic (pozic a orientací)	reference
<b>Relative</b>	Výběr absolutních (=false) nebo relativních (=true) souřadnic off ... absolute on .... relative	bool
<b>CoordSystem</b>	Volba souřadného systému 1 ..... ACM 2 ..... MCS 3 ..... PCS	long

### Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool

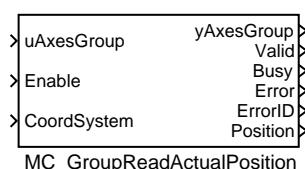


Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i ..... obecná chyba systému REX	

## MC\_GroupReadActualPosition – Aktuální poloha skupiny os

Symbol bloku

Licence: COORDINATED MOTION CONTROL



## Popis funkce

Blok `MC_GroupReadActualPosition` zpřístupňuje na výstup `Position` aktuální polohu skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true.

## Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	bool
<code>CoordSystem</code>	Volba souřadného systému	long
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

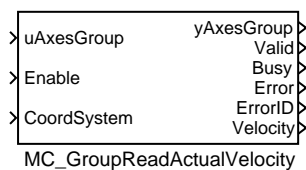
## Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	bool
<code>Error</code>	Příznak chyby	bool
<code>ErrorID</code>	Kód chyby	error
	i ..... obecná chyba systému REX	
<code>Position</code>	Pole aktuálních souřadnic (pozice a orientace) skupiny os	reference

## MC\_GroupReadActualVelocity – Aktuální rychlost skupiny os

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

Blok `MC_GroupReadActualVelocity` zpřístupňuje na výstup `Velocity` aktuální rychlost skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true.

### Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	bool
<code>CoordSystem</code>	Volba souřadného systému	long
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

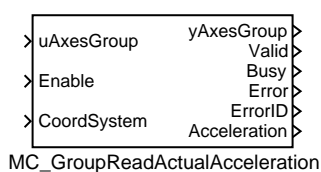
### Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	bool
<code>Error</code>	Příznak chyby	bool
<code>ErrorID</code>	Kód chyby	error
	i ..... obecná chyba systému REX	
<code>Velocity</code>	Pole aktuálních rychlostí skupiny os	reference

## MC\_GroupReadActualAcceleration – Aktuální zrychlení skupiny os

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

Blok `MC_GroupReadActualAcceleration` zpřístupňuje na výstup `Position` aktuální zrychlení skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu `Valid` `true`, čehož se dosáhne nastavením vstupu `Enable` na hodnotu `true`.

### Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	bool
<code>CoordSystem</code>	Volba souřadného systému	long
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

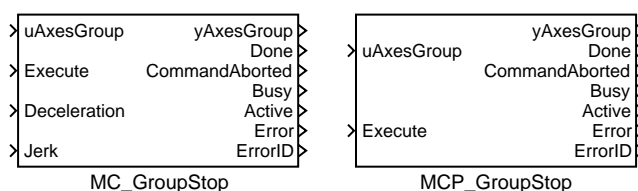
### Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	bool
<code>Error</code>	Příznak chyby	bool
<code>ErrorID</code>	Kód chyby	error
	i ..... obecná chyba systému REX	
<code>Acceleration</code>	Pole aktuálních zrychlení skupiny os	reference

## MC\_GroupStop, MCP\_GroupStop – Zastavení koordinovaného pohybu

Symboly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_GroupStop a MCP\_GroupStop mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_GroupStop zastaví pohyb. Režim je vždy aborted, tj. zastavování se spouští okamžitě. Blok se nejprve pokusí zastavit ve směru původní trajektorie. Pokud se to nepodaří, zastaví se pomocí errorstop sekvence na jednotlivých osách již nekoordinovaně. Dokud je na vstup **Execute** hodnota true nebo dokud se skupina pohybuje, nachází se skupina ve stavu „Stopping“ a není možné spouštět další bloky. Okamžitě po zastavení se nastaví na výstupu **Done** hodnota true (pokud nenastane chyba). Skupina os přejde do „Standby“ až po deaktivování vstupu **Execute**.

Poznámka 1: Blok nemá parametr **CoordSystem**, protože jej přejímá z právě běžícího bloku.

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double

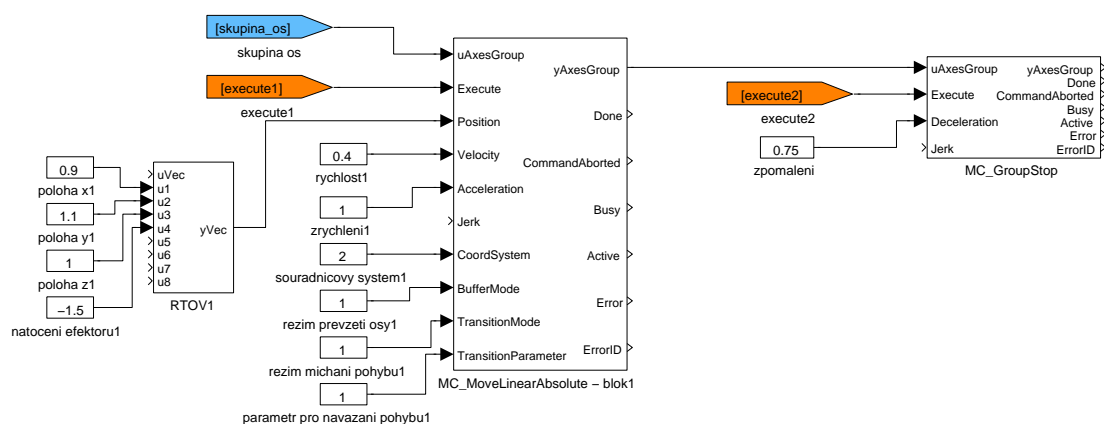
### Výstupy

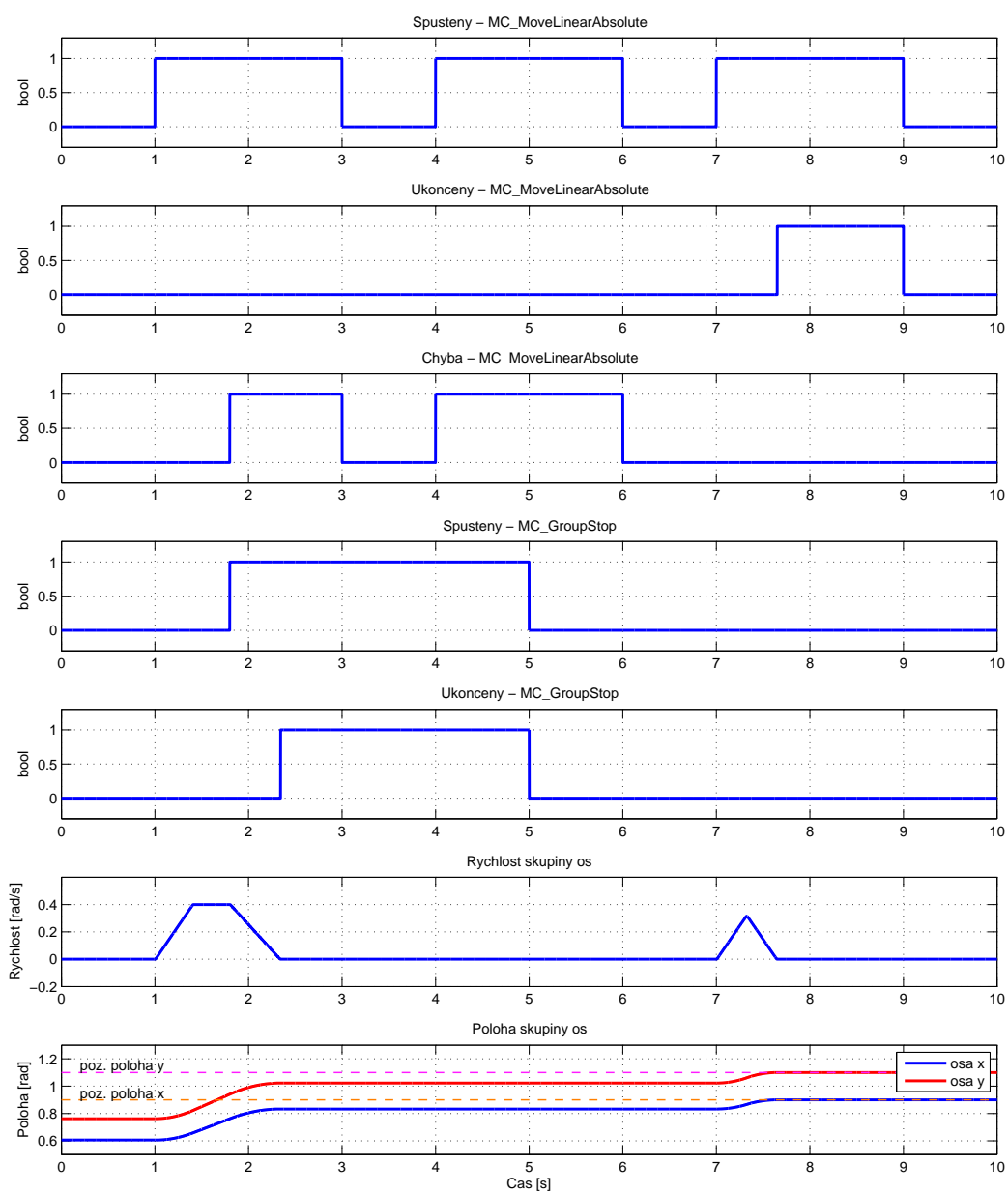
<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool

Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i . . . . . obecná chyba systému REX	

## Příklad

Na časovém diagramu níže, je uveden příklad na chování bloku **MC\_GroupStop**. Nejprve dojde ke spuštění bloku **MC\_MoveLinearAbsolute** a tím dojde k uvedení osy do pohybu. Následně je již spuštěn blok **MC\_GroupStop**, který způsobí zastavení skupiny os. Následně druhé spuštění exekutivy bloku **MC\_MoveLinearAbsolute** nic nezpůsobí a to i přes to, že je skupina již v klidu. To z toho důvodu, že je stále spuštěný blok **MC\_GroupStop**. Až třetí spuštění exekutivy bloku **MC\_MoveLinearAbsolute** dostane skupinu os do požadované polohy.

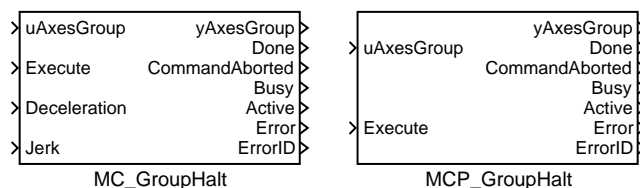




## MC\_GroupHalt, MCP\_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné)

Symbyly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_GroupHalt a MCP\_GroupHalt mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_GroupHalt** zahajuje řízené zastavení pohybu. Osa se přesune do stavu „GroupMoving“, dokud není rychlost nulová. Společně s nastavením výstupu **Done** je stav změněn na „GroupStandby“.

Poznámka 1: Blok **MC\_GroupHalt** se používá k zastavení skupiny os za normálních provozních podmínek. V non-buffered režimu je možné zadat další pohybový příkaz při zpomalení osy, který zruší **MC\_GroupHalt** a bude ihned proveden.

Poznámka 2: Je-li tento příkaz aktivní, další příkaz může být aktivován (spuštěn). Např. vozidlo bez řidiče detekuje překážku a potřebuje zastavit. **MC\_GroupHalt** je aktivován. Před dosažením stavu „GroupStandby“ je překážka odstraněna a pohyb může pokračovat nastavením dalšího pohybového příkazu, aby vozidlo nemuselo zastavit.

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>Deceleration</b>	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double

### Výstupy

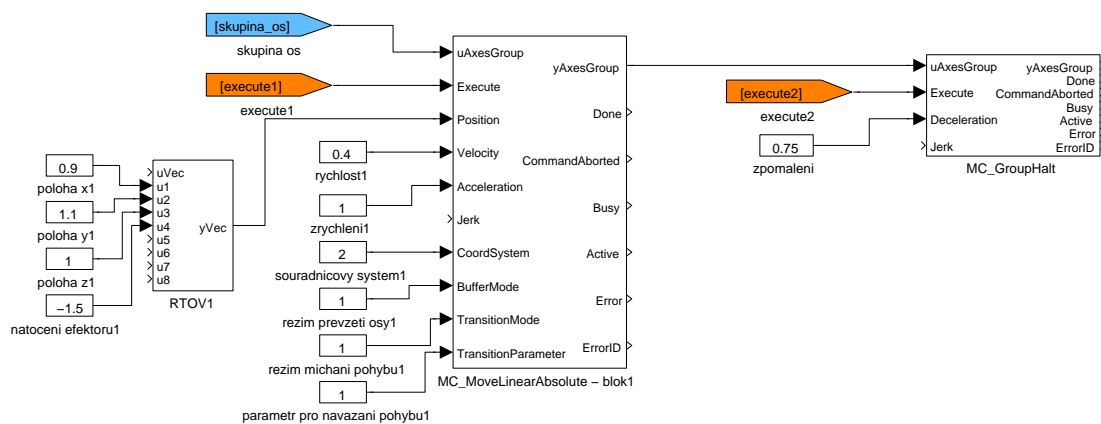
<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool

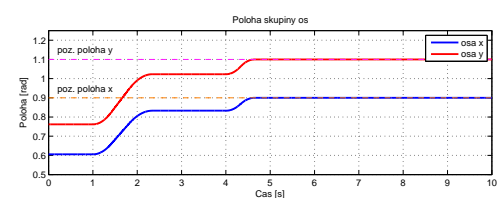
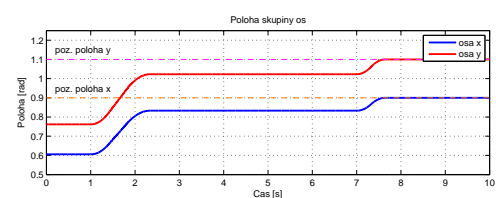
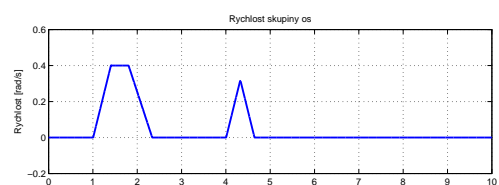
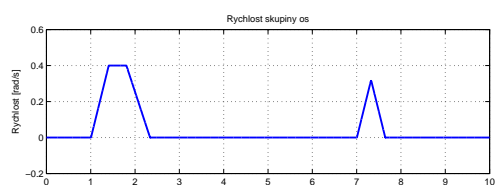
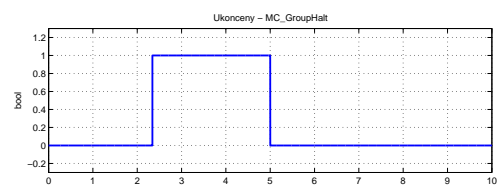
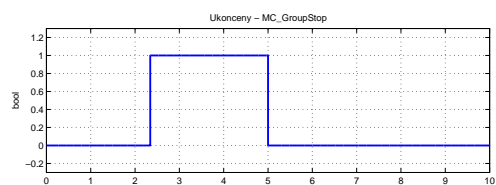
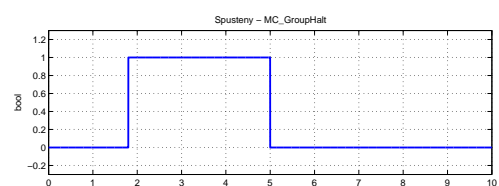
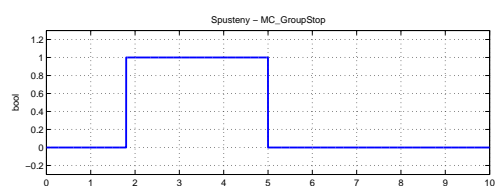
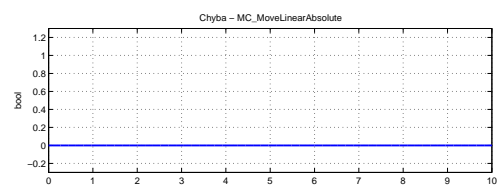
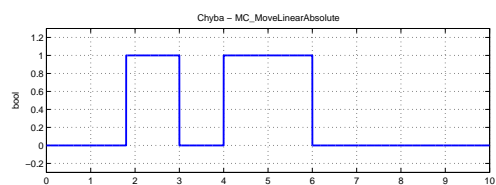
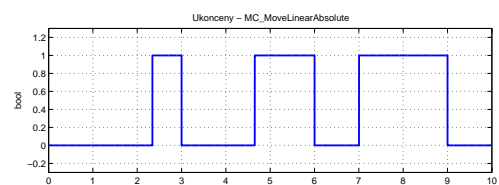
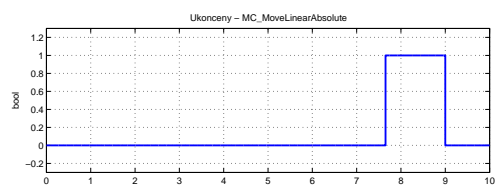
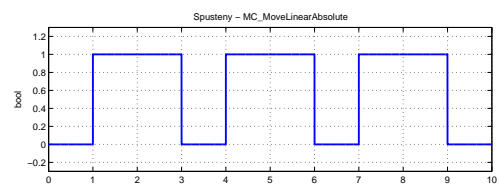
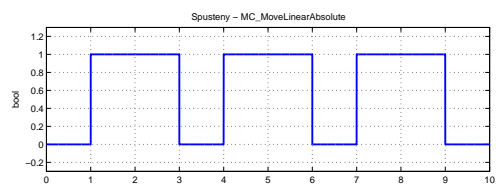


Active	Příznak, že blok řídí osu	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i . . . . . obecná chyba systému REX	

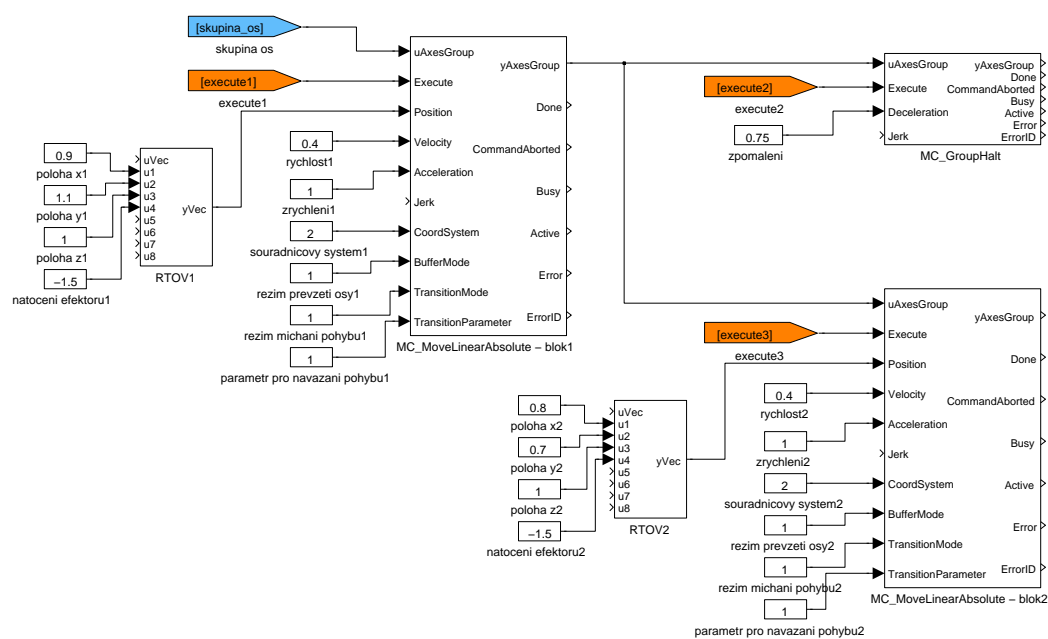
## Příklad

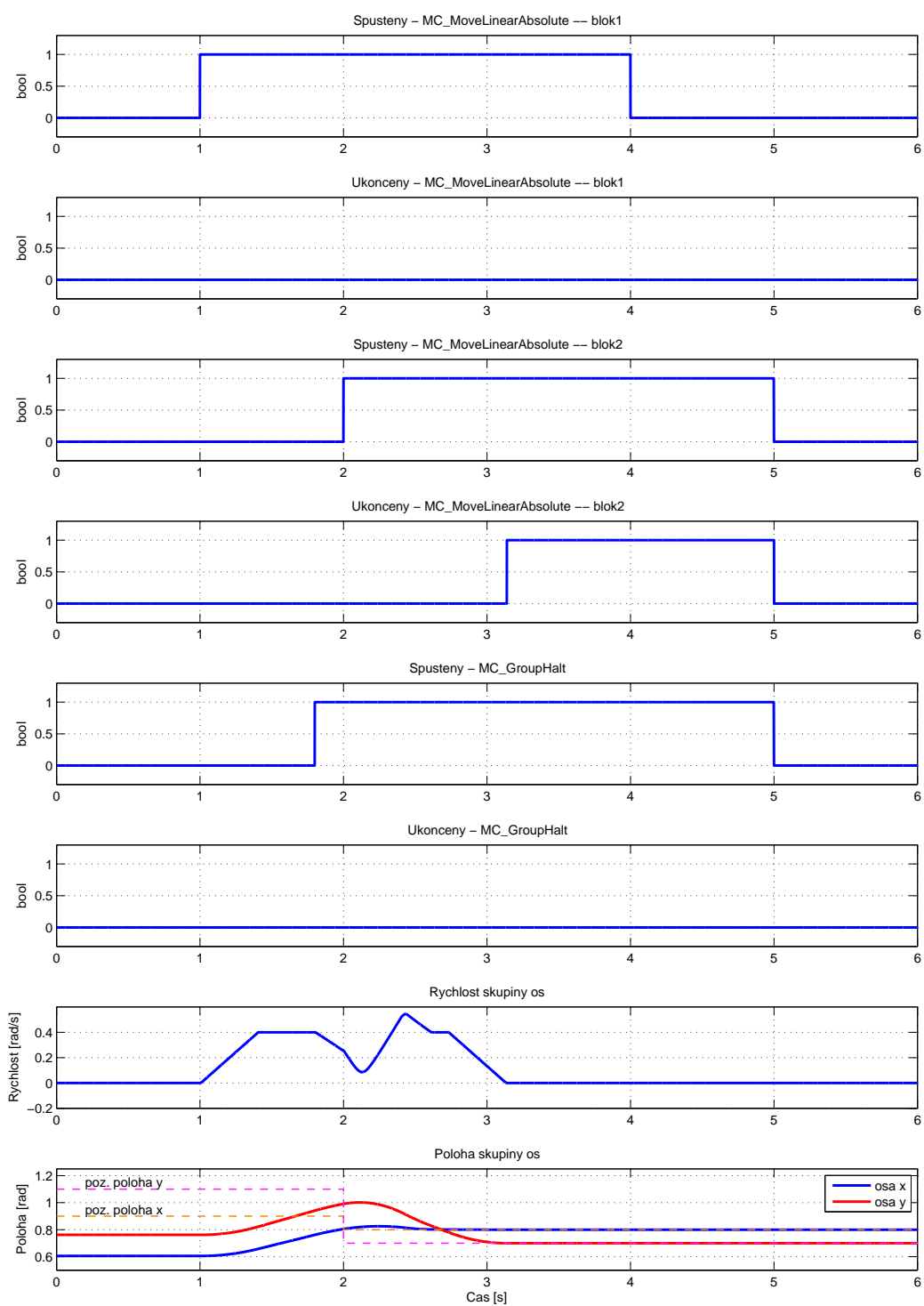
Pro porovnání je následující příklad totožný jako u bloku [MC\\_GroupStop](#). Časové průběhy v levém sloupci odpovídají bloku [MC\\_GroupStop](#). Průběhy v pravém sloupci bloku [MC\\_GroupHalt](#). Je vidět, že při použití bloku [MC\\_GroupHalt](#) dojde k dojetí do požadované polohy již na druhé spuštění exekutivy bloku [MC\\_MoveLinearAbsolute](#).





V druhém příkladě je spuštěn blok **MC\_MoveLinearAbsolute**, který je následně přerušen blokem **MC\_GroupHalt**. Ještě před zastavením je spuštěn druhý pohyb blokem dalším **MC\_MoveLinearAbsolute**. Příklad ilustruje to, že pro možnost spuštění dalšího pohybu, nemusí dojít k úplnému zastavení skupiny os.

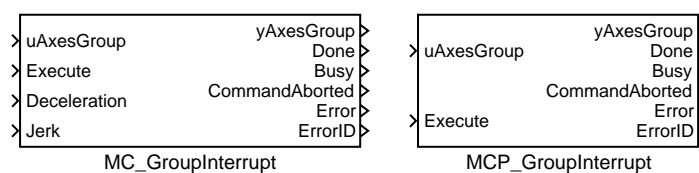




## MC\_GroupInterrupt, MCP\_GroupInterrupt – Přerušení pohybu skupiny os

Symboly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_GroupInterrupt a MCP\_GroupInterrupt mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_GroupInterrupt** přeruší právě prováděný pohyb a uvede skupinu os do klidu, nicméně tento příkaz nezruší přerušený pohyb (výstup **CommandAborted** nebude nastaven na true a **Busy** zůstane true). Informace o původním pohybu zůstane uložena. Skupina os zůstane v původním stavu i po zastavení, kdy je výstup **Done** nastaven na true.

Poznámka 1: Tento blok je spárován s blokem **MC\_GroupContinue**, jehož aktivací je skupina os vrácena do stavu, před spuštěním bloku **MC\_GroupInterrupt**.

### Vstupy

uAxesGroup	Odkaz na skupinu os	reference
Execute	Náběžná hrana aktivuje blok	bool
Deceleration	Maximální povolené zpomalení [unit/s <sup>2</sup> ]	double
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double

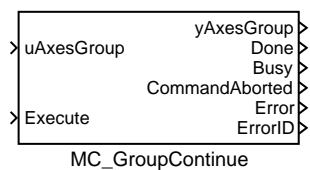
### Výstupy

yAxesGroup	Odkaz na skupinu os	reference
Done	Příznak dokončení algoritmu	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
i	..... obecná chyba systému REX	

## MC\_GroupContinue – Pokračování v přerušeném pohybu

Symbol bloku

Licence: COORDINATED MOTION CONTROL



## Popis funkce

Blok **MC\_GroupContinue** ruší působení bloku **MC\_GroupInterrupt**. Skupina os dokončí původně prováděný pohyb před přerušením. Výstup **Done** určuje, zda se pohyb vrátil do stavu před přerušením.

## Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool

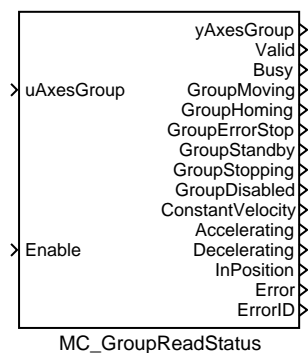
## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby	error
	i . . . . . obecná chyba systému REX	

## MC\_GroupReadStatus – Stav skupin os

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



## Popis funkce

Blok **MC\_GroupReadStatus** indikuje na svých výstupech různé stavy připojené skupiny os jako logickou hodnotu. Indikovaný stav je zřejmý z názvu výstupu, popřípadě z jeho popisu. Hodnota je platná jen pokud je na výstupu **Valid** true, čehož se dosáhne nastavením vstupu **Enable** na hodnotu true.

## Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	<b>bool</b>

## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Valid</b>	Příznak platnosti výstupní hodnoty	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>GroupMoving</b>	Stav GroupMoving	<b>bool</b>
<b>GroupHoming</b>	Stav GroupHoming	<b>bool</b>
<b>GroupErrorStop</b>	Stav ErrorStop	<b>bool</b>
<b>GroupStandby</b>	Stav Standby	<b>bool</b>
<b>GroupStopping</b>	Stav Stopping	<b>bool</b>
<b>GroupDisabled</b>	Stav Disabled	<b>bool</b>
<b>ConstantVelocity</b>	Pohyb konstantní rychlostí	<b>bool</b>
<b>Accelerating</b>	Zrychlování	<b>bool</b>
<b>Decelerating</b>	Zpomolování	<b>bool</b>

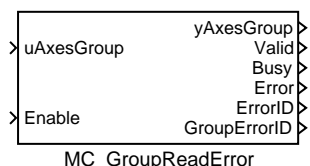
InPosition	Příznak dosažení zadané pozice	bool
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
	i . . . . . obecná chyba systému REX	



## MC\_GroupReadError – Chyby ve skupině os

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

Blok **MC\_GroupReadError** zpřístupňuje na výstupu **GroupErrorID** aktuální chybový kód připojené osy. Pokud osa není ve stavu chyby, hodnota tohoto výstupu je 0. Hodnota je platná jen pokud je na výstupu **Valid** true, čehož se dosáhne nastavením vstupu **Enable** na hodnotu true.

Poznámka 1: Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť zobrazuje stejnou veličinu, která je přístupná i na výstupu **ErrorID** bloku [RM\\_AxesGroup](#).

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	reference
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	bool

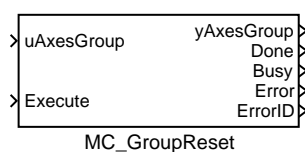
### Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Valid</b>	Příznak platnosti výstupní hodnoty	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby	error
	i ..... obecná chyba systému REX	
<b>GroupErrorID</b>	Kód chyby	error
	i ..... obecná chyba systému REX	

## MC\_GroupReset – Nulování chyb os ve skupině

Symbol bloku

Licence: [COORDINATED MOTION CONTROL](#)



Popis funkce

Blok **MC\_GroupReset** převede připojenou skupinu os ze stavu „GroupErrorStop“ do stavu „GroupStandBy“ a vynuluje ve skupině všechny příznaky chyby. Blok resetuje také všechny osy v dané skupině os stejně jako blok [MC\\_Reset](#) z knihovny MC\_SINGLE.

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>

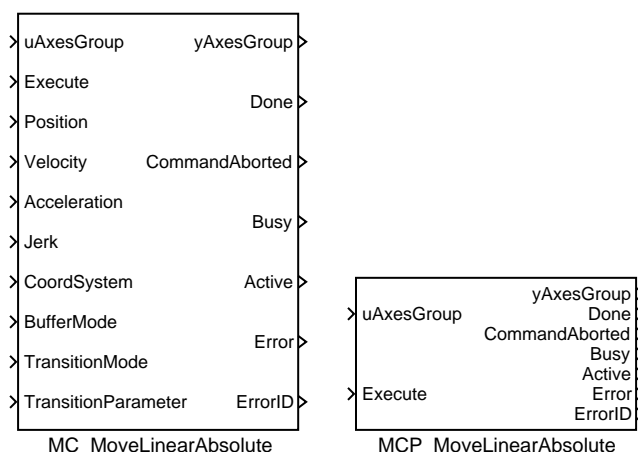
### Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
	i . . . . . obecná chyba systému REX	

## MC\_MoveLinearAbsolute, MCP\_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice)

Symbyoly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveLinearAbsolute a MCP\_MoveLinearAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveLinearAbsolute** slouží pro přesun koncového efektoru po přímce na zadanou pozici. Pozice se zadává absolutně v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **Position**. Tento parametr musí mít tolik prvků, kolik je nastaveno souřadnic v přidruženém bloku **RM\_GroupAxes**. V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

## Vstupy

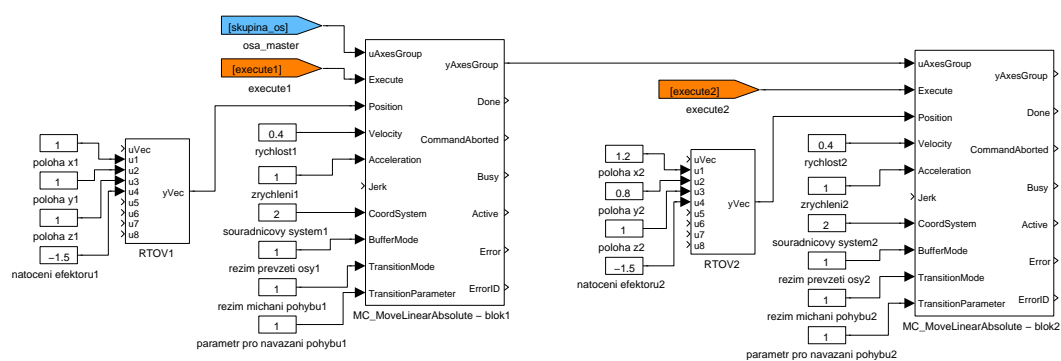
uAxesGroup	Odkaz na skupinu os	reference
Execute	Náběžná hrana aktivuje blok	bool
Position	Pole souřadnic (pozic a orientací)	reference
Velocity	Maximální povolená rychlost [unit/s]	double
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
CoordSystem	Volba souřadného systému	long
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
BufferMode	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionMode	Režim míchání pohybu	long
	1 ..... TMNone (xx)	
	2 ..... TMstart velocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstant Velocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	double

## Výstupy

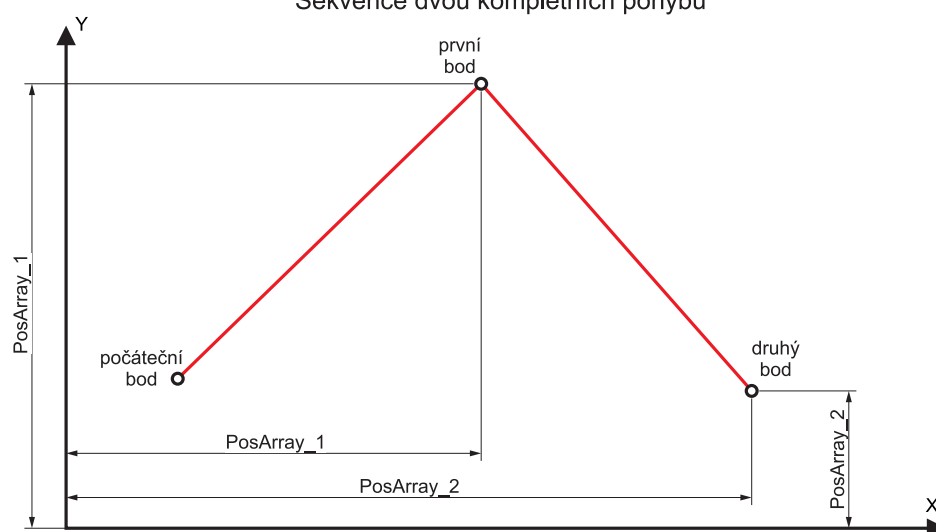
yAxesGroup	Odkaz na skupinu os	reference
Done	Příznak dokončení algoritmu	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool

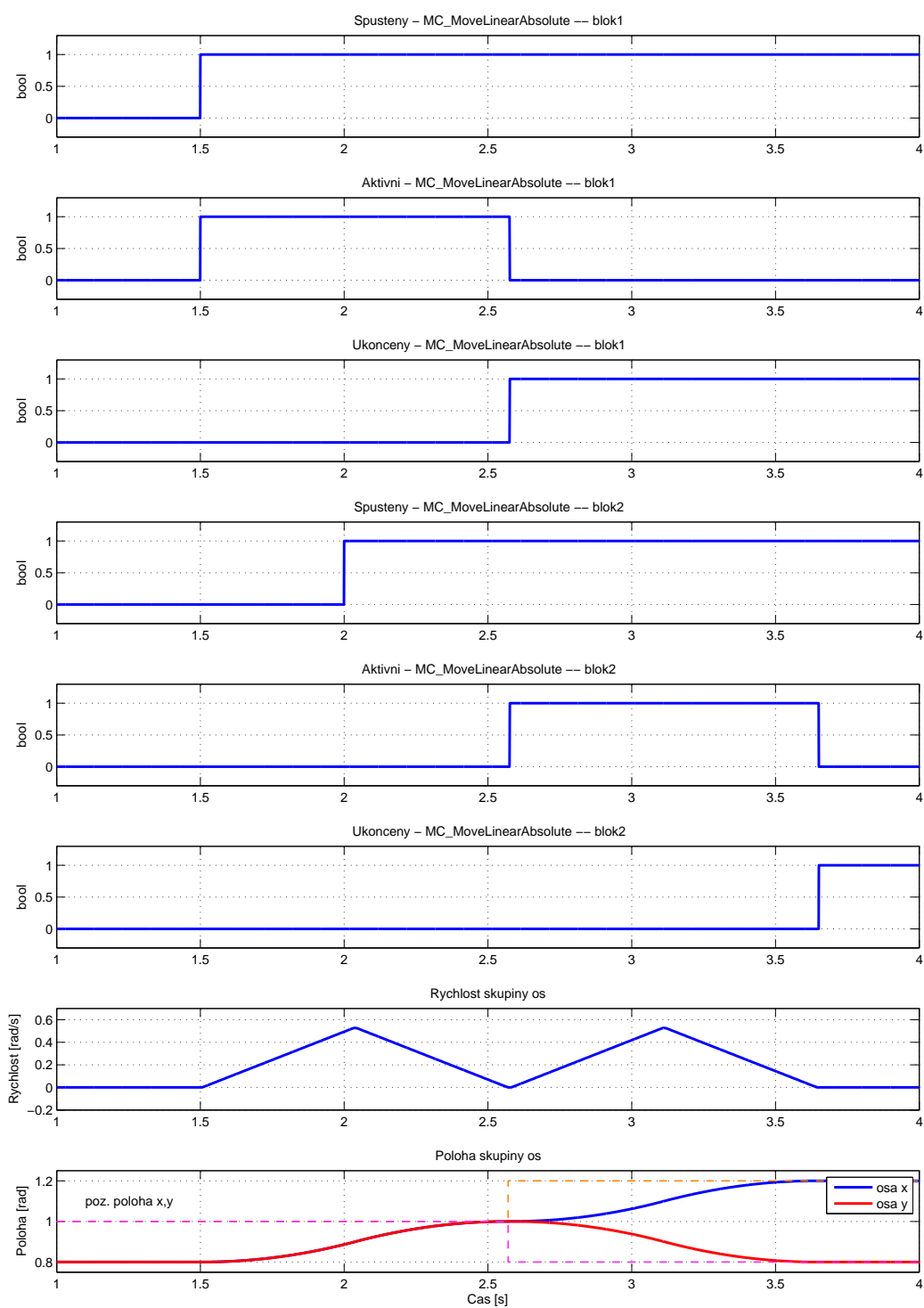
Error	Příznak chyby	bool
ErrorID	Kód chyby	error
i ..... obecná chyba systému REX		

## Příklad



Sekvence dvou kompletních pohybů

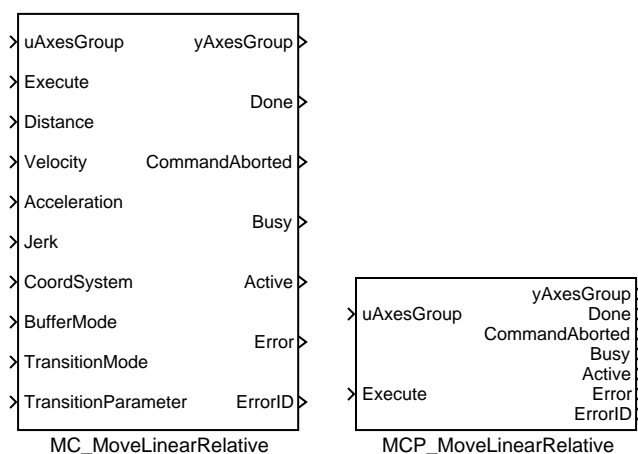




## MC\_MoveLinearRelative, MCP\_MoveLinearRelative – Pohyb do pozice po přímkách (relativní souřadnice)

Symboly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveLinearRelative a MCP\_MoveLinearRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveLinearAbsolute** slouží pro přesun koncového efektoru po přímce na zadanou pozici. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **Distance**. Tento parametr musí mít tolik prvků, kolik je nastaveno souřadnic v přidruženém bloku **RM\_GroupAxes**. V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

## Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>Distance</b>	Pole souřadnic (relativních pozic a orientací)	reference
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	double
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
<b>CoordSystem</b>	Volba souřadného systému	long
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
<b>BufferMode</b>	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	long
	1 ..... TMNone (xx)	
	2 ..... TMstart velocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstant Velocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	double

## Výstupy

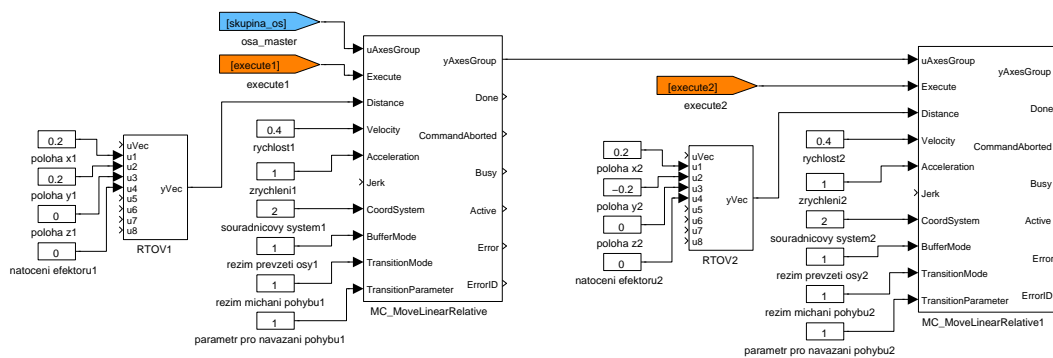
<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool



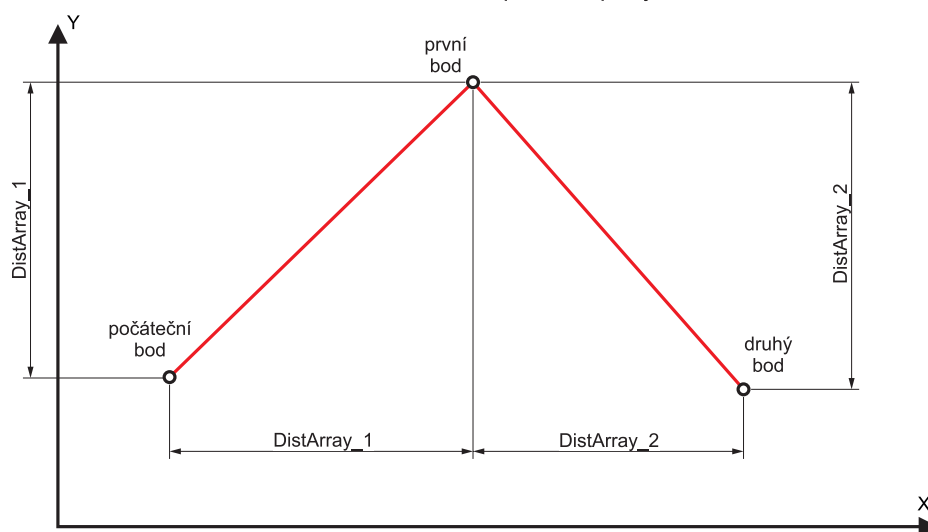
Error Příznak chyby  
 ErrorID Kód chyby  
 i ..... obecná chyba systému REX

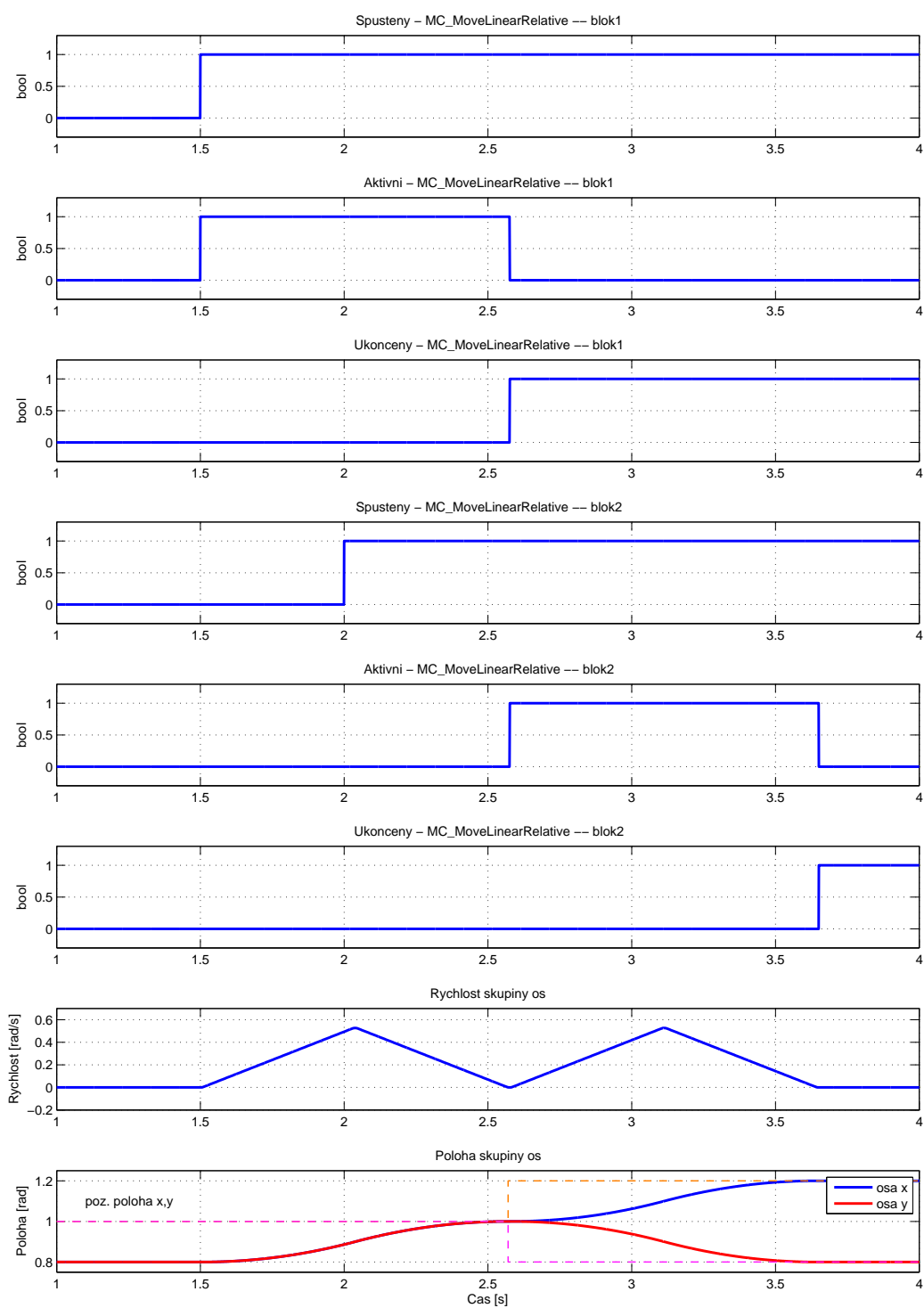
bool  
 error

## Příklad



Sekvence dvou kompletních pohybů

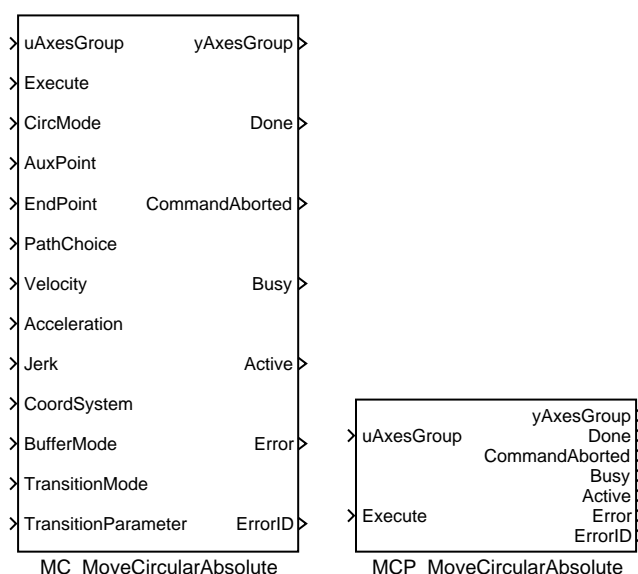




## MC\_MoveCircularAbsolute, MCP\_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice)

Symboly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveCircularAbsolute a MCP\_MoveCircularAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

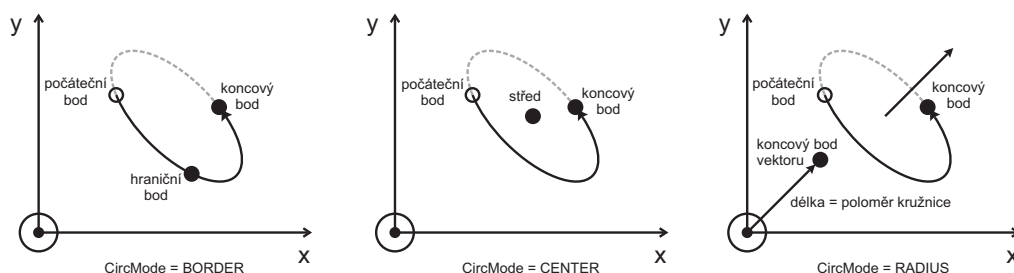
Blok **MC\_MoveCircularAbsolute** slouží pro přesun koncového efektoru po kružnici (resp. kruhovém oblouku) na zadanou pozici. Pozice se zadává absolutně v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **EndPoint**. Pomocný bod je určen vektorovým parametrem **AuxPoint**. Tyto parametry musí mít tolik prvků, kolik je nastaveno souřadnic v přidruženém bloku **RM\_GroupAxes**.

V opačném případě je signalizována chyba a pohyb se neprovede. Význam pomocného bodu je uveden níže. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**. K dispozici jsou následující způsoby zadání kružnice:

**BORDER** – Zadává se koncový bod a bod, kterým má kružnice procházet (vstup **AuxPoint**).

**CENTER** – Zadává se koncový bod a střed kružnice (vstup **AuxPoint**). Vstup **PathChoice** potom definuje, zda bude vygenerovaný pohyb po směru nebo proti směru hodinových ručiček.

**RADIUS** – Zadává se koncový bod a vektor kolmý k rovině ve které se má nacházet kružnice. Délka vektoru udává poloměr kružnice. Příklad: Vstup **AuxPoint** = (50,0,0) odpovídá kružnici v rovině y-z s poloměrem 50 a rotaci kolem osy paralelní s osou x odpovídající pravidlu pravé ruky (**CoordSystem** = MCS).



Poznámka 1: v režimu „RADIUS“ nemá pomocný bod význam polohy a proto se zadává vždy absolutně.

Poznámka 2: Každý ze způsobu zadávání (parametr **CircMode**) má některé výhody a nevýhody. Žádný z uvedených způsobů neumožňuje „projet“ celý kruh nebo dokonce několik „otáček“. Dále vzniká problém v případě půlkruhu, kdy v režimu zadání středu není definována rovina kružnice. Při zadání středu je kružnice přerušena a je nutné dát pozor, aby vzdálenost počátečního a koncového bodu byla stejná (v opačném případě blok skončí s chybou a trajektorie se negeneruje).

Poznámka 3: U kružnic v třírozměrném prostoru je problém definovat kladný směr. Algoritmus používá metodu vektorového součinu počátečního a koncového průvodiče. Tím dostaneme normálový vektor, který určuje „směr nahoru“ a pak již je kladný směr zřejmý. Metoda ovšem nefunguje pro půlkružnici a také je potřeba dát pozor při posunutí koncového nebo počátečního bodu, protože může dojít ke změně orientace (kladný směr se stane záporným).

## Vstupy

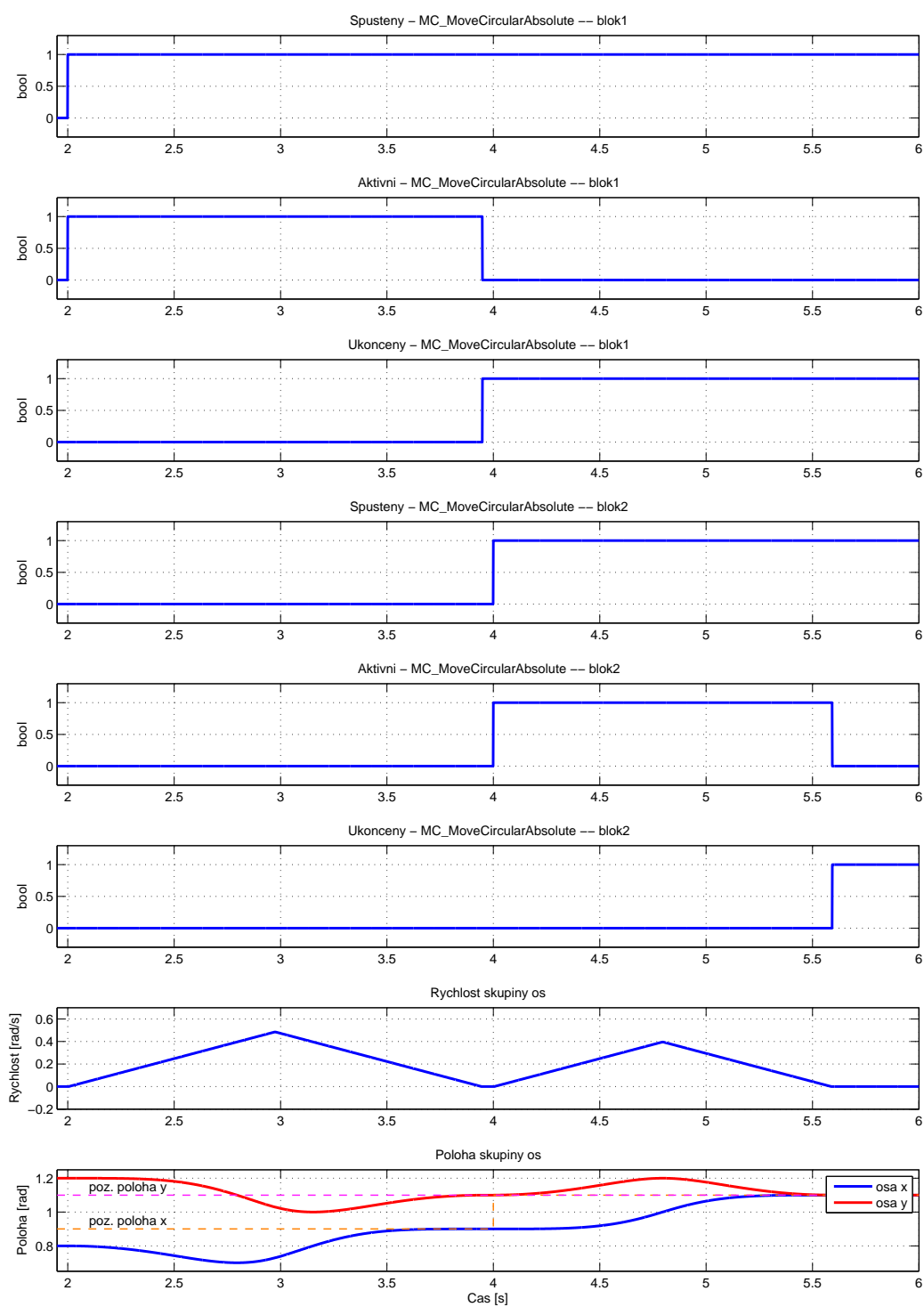
<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>CircMode</b>	Určuje význam vstupních signálů <b>AuxPoint</b> a <b>CircDirection</b>	<b>long</b>
	1 ..... BORDER	
	2 ..... CENTER	
	3 ..... RADIUS	

<b>AuxPoint</b>	Absolutně zadaná pozice	reference
<b>EndPoint</b>	Absolutně zadaná pozice koncového bodu	reference
<b>PathChoice</b>	Volba směru	long
	1 ..... Ve směru hodinových ručiček	
	2 ..... Protisměru hodinových ručiček	
<b>Velocity</b>	Maximální povolená rychlost [unit/s]	double
<b>Acceleration</b>	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
<b>Jerk</b>	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
<b>CoordSystem</b>	Volba souřadného systému	long
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
<b>BufferMode</b>	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	long
	1 ..... TMNone (xx)	
	2 ..... TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 ..... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	double

## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool

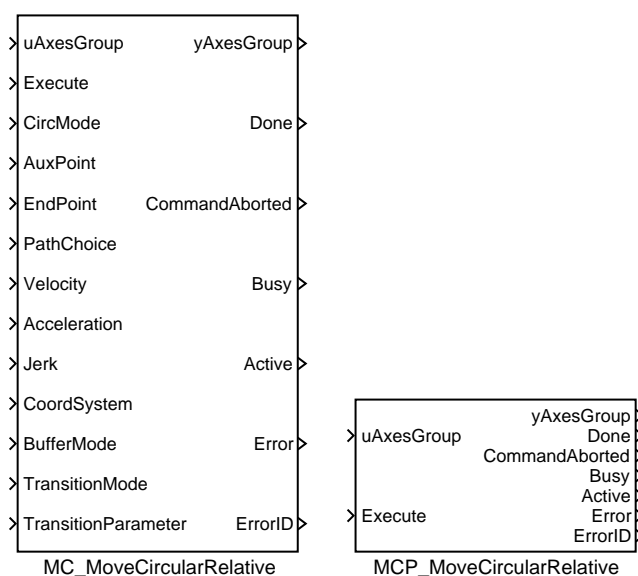




## MC\_MoveCircularRelative, MCP\_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice)

Symbylů bloků

Licence: [COORDINATED MOTION CONTROL](#)



## Popis funkce

*Bloky MC\_MoveCircularRelative a MCP\_MoveCircularRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveCircularRelative** slouží pro přesun koncového efektoru po kružnici (resp. kruhovém oblouku) na zadanou pozici. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **EndPoint**. Pomocný bod je určen vektorovým parametrem **AuxPoint**. Tyto parametry musí mít tolik prvků, kolik je nastaveno souřadnic v

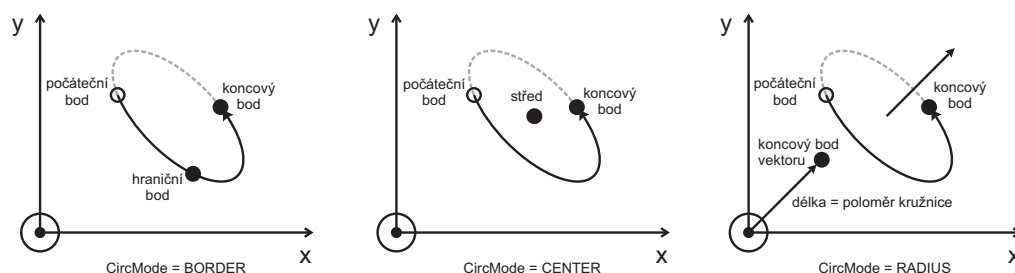


přidruženém bloku **RM\_GroupAxes**. V opačném případě je signalizována chyba a pohyb se neprovede. Význam pomocného bodu je uveden níže. Pohyb je spuštěn náběžnou hranou na vstup **Execute**. K dispozici jsou následující způsoby zadání kružnice:

**BORDER** – Zadává se koncový bod a bod, kterým má kružnice procházet (vstup **AuxPoint**).

**CENTER** – Zadává se koncový bod a střed kružnice (vstup **AuxPoint**). Vstup **PathChoice** potom definuje, zda bude vygenerovaný pohyb po směru nebo proti směru hodinových ručiček.

**RADIUS** – Zadává se koncový bod a vektor kolmý k rovině ve které se má nacházet kružnice. Délka vektoru udává poloměr kružnice. Příklad: Vstup **AuxPoint** = (50,0,0) odpovídá kružnici v rovině y-z s poloměrem 50 a rotaci kolem osy paralelní s osou x odpovídající pravidlu pravé ruky (**CoordSystem** = MCS).



Poznámka 1: v režimu „RADIUS“ nemá pomocný bod význam polohy a proto se zadává vždy absolutně.

Poznámka 2: Každý ze způsobu zadávání (parametr **CircMode**) má některé výhody a nevýhody. Žádný z uvedených způsobů neumožňuje „projet“ celý kruh nebo dokonce několik „otáček“. Dále vzniká problém v případě půlkruhu, kdy v režimu zadání středu není definována rovina kružnice. Při zadání středu je kružnice přeurčena a je nutné dát pozor, aby vzdálenost počátečního a koncového bodu byla stejná (v opačném případě blok skončí s chybou a trajektorie se negeneruje).

Poznámka 3: U kružnic v třírozměrném prostoru je problém definovat kladný směr. Algoritmus používá metodu vektorového součinu počátečního a koncového průvodiče. Tím dostaneme normálový vektor, který určuje „směr nahoru“ a pak již je kladný směr zřejmý. Metoda ovšem nefunguje pro půlkružnici a také je potřeba dát pozor při posunutí koncového nebo počátečního bodu, protože může dojít ke změně orientace (kladný směr se stane záporným).

## Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>CircMode</b>	Určuje význam vstupních signálů <b>AuxPoint</b> a <b>CircDirection</b>	<b>long</b>
	1 ..... <b>BORDER</b>	
	2 ..... <b>CENTER</b>	
	3 ..... <b>RADIUS</b>	

AuxPoint	Absolutně zadaná pozice	reference
EndPoint	Absolutně zadaná pozice koncového bodu	reference
PathChoice	Volba směru	long
	1 ..... Ve směru hodinových ručiček	
	2 ..... Protisměru hodinových ručiček	
Velocity	Maximální povolená rychlost [unit/s]	double
Acceleration	Maximální povolené zrychlení [unit/s <sup>2</sup> ]	double
Jerk	Maximální povolená změna zrychlení [unit/s <sup>3</sup> ]	double
CoordSystem	Volba souřadného systému	long
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
BufferMode	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionMode	Režim míchání pohybu	long
	1 ..... TMNone (xx)	
	2 ..... TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 ..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 ..... TMCornerDistance (xx)	
	5 ..... TMMaxCornerDeviation (xx)	
	11 .... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	double

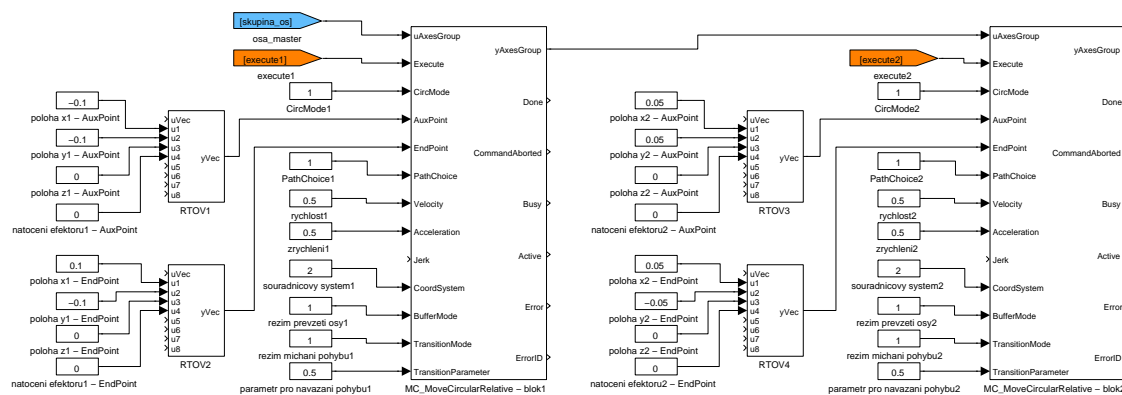
## Výstupy

yAxesGroup	Odkaz na skupinu os	reference
Done	Příznak dokončení algoritmu	bool
CommandAborted	Příznak přerušení funkce bloku	bool
Busy	Příznak, že algoritmus ještě neskončil	bool
Active	Příznak, že blok řídí osu	bool

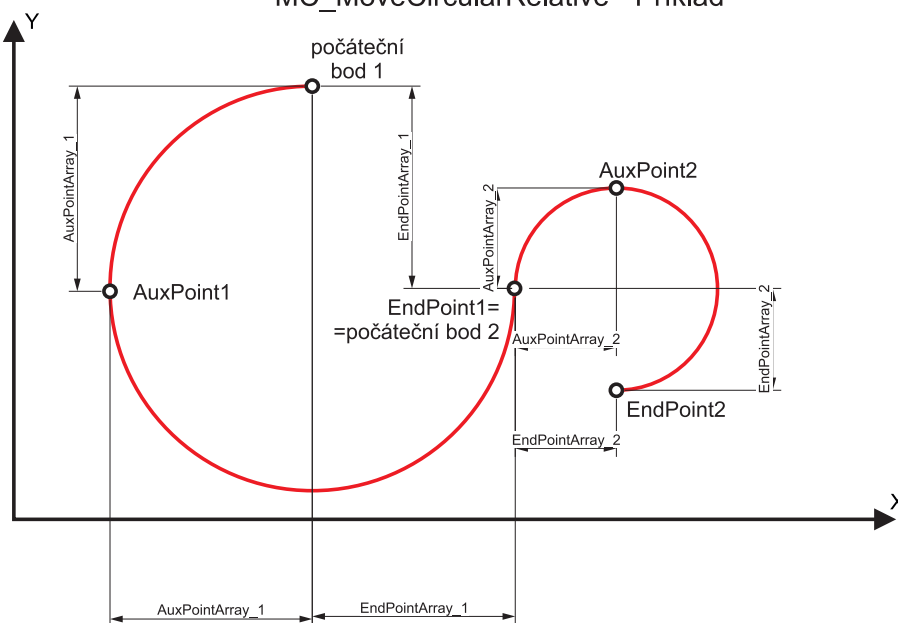
Error Příznak chyby  
 ErrorID Kód chyby  
 i ..... obecná chyba systému REX

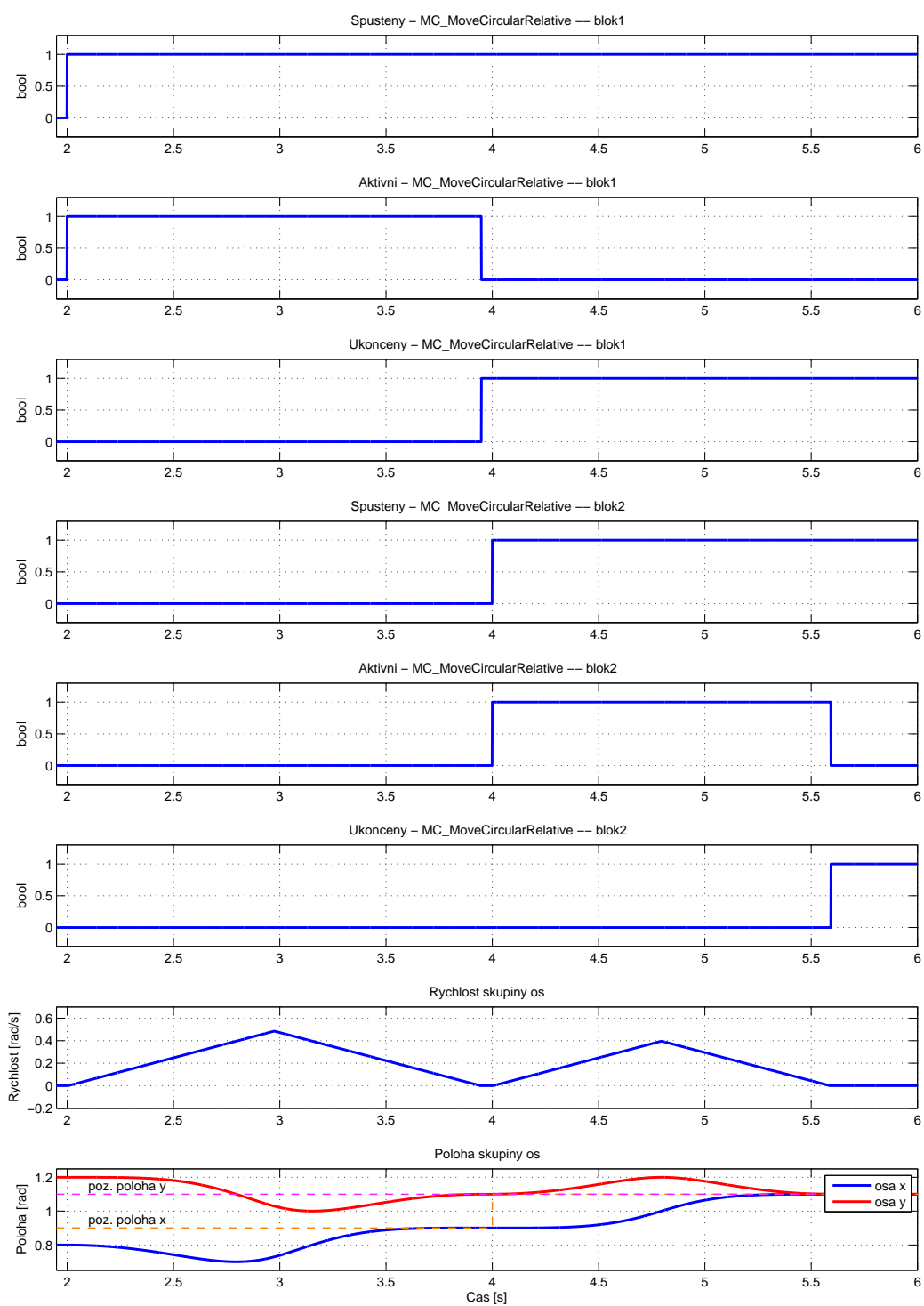
bool  
 error

## Příklad



MC\_MoveCircularRelative - Příklad

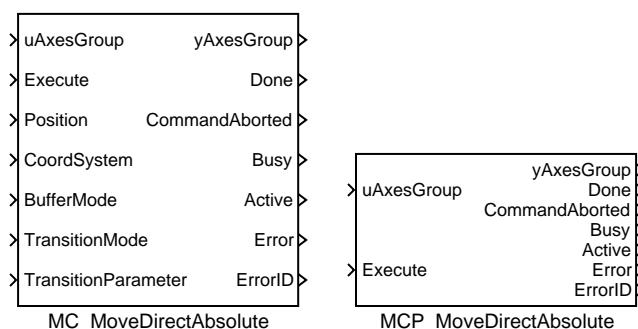




## MC\_MoveDirectAbsolute, MCP\_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice)

Symboly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveDirectAbsolute a MCP\_MoveDirectAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveDirectAbsolute** slouží pro co nejrychlejší přesun stroje (skupiny os) na zadanou pozici, přičemž nezáleží na přesné trajektorii. Tento blok proto nemá parametry určující rychlost a zrychlení. Pohybuje se s maximální rychlostí a zrychlením nastavenými na jednotlivých osách/motorech. Trajektorie je generována tak, aby všechny motory dokončili pohyb ve stejnou dobu, proto se některé motory mohou pohybovat pomaleji, než je jejich limit. Pozice se zadává v absolutně v souřadném systému zvoleném vstupem **CoordSystem**. Tento parametr musí mít tolik prvků, kolik je nastaveno souřadnic v přidruženém bloku **RM\_GroupAxes**. V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

### Vstupy

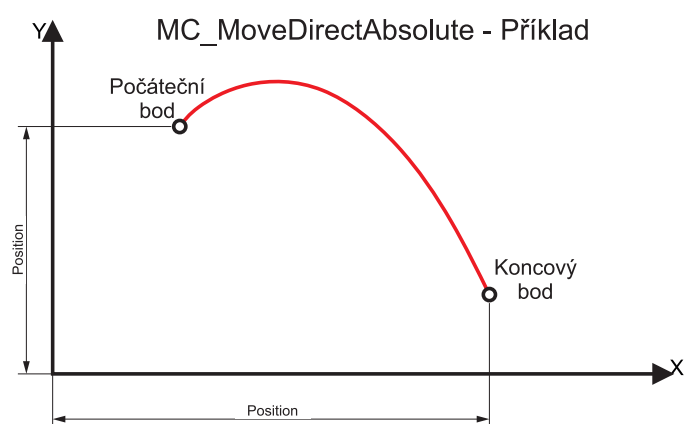
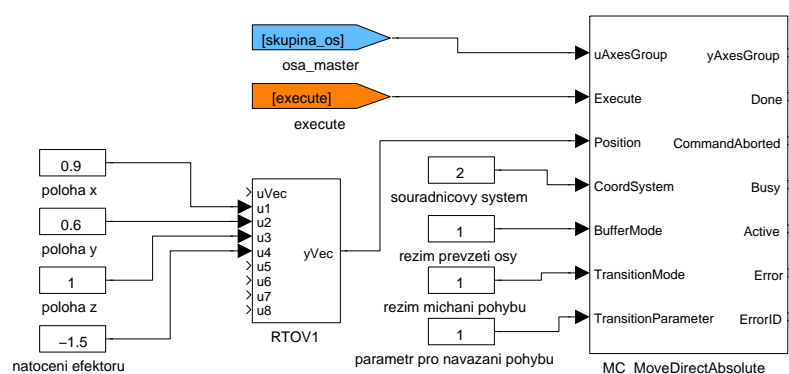
<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>Position</b>	Pole souřadnic (pozic a orientací)	<b>reference</b>
<b>CoordSystem</b>	Volba souřadného systému	<b>long</b>
1	..... ACM	
2	..... MCS	
3	..... PCS	

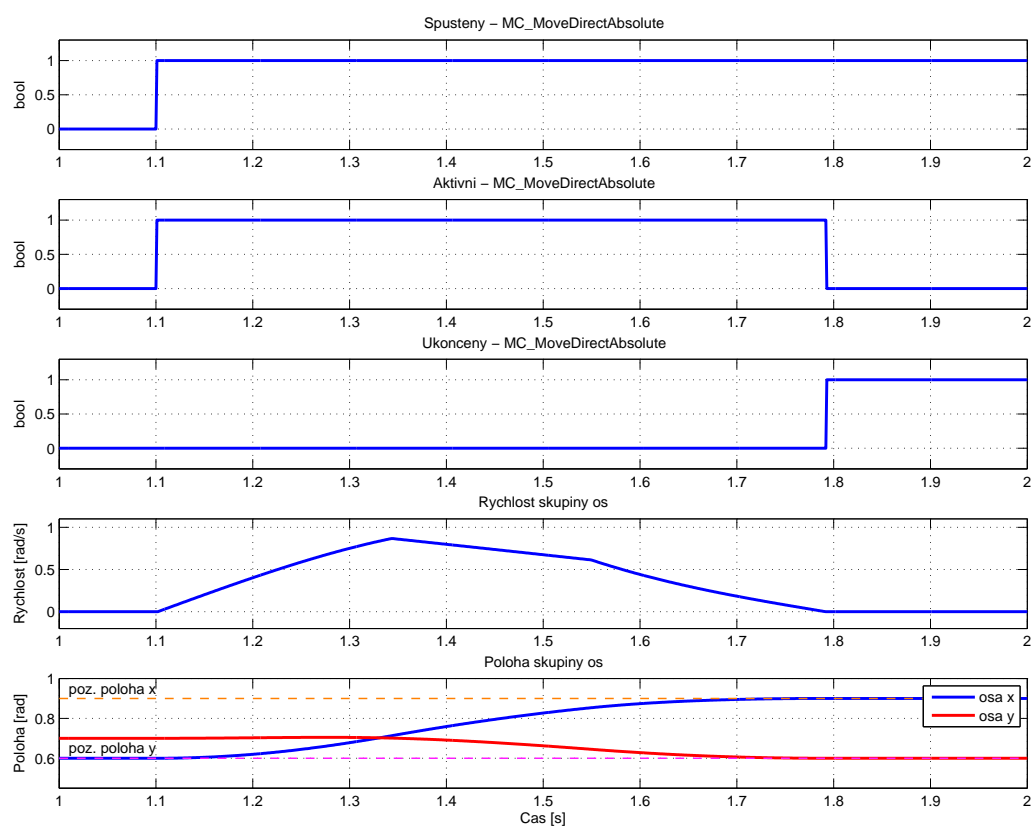
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	<b>long</b>
1	..... TMNone (xx)	
2	..... TMstartvelocity (proložení s danou počáteční rychlostí)	
3	..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
4	..... TMCornerDistance (xx)	
5	..... TMMaxCornerDeviation (xx)	
11	..... Smooth(nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	<b>double</b>

## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
i	..... obecná chyba systému REX	

## Příklad



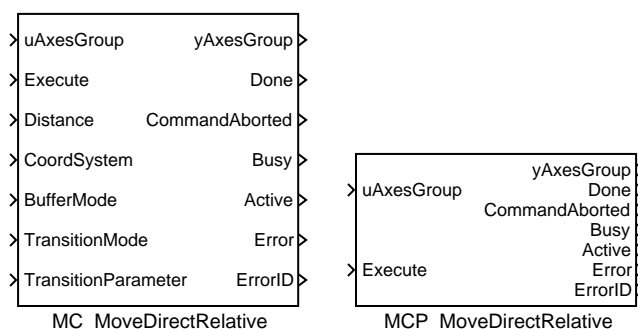




## MC\_MoveDirectRelative, MCP\_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice)

Symboly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MoveDirectRelative a MCP\_MoveDirectRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_MoveDirectAbsolute** slouží pro co nejrychlejší přesun stroje (skupiny os) na zadanou pozici, přičemž nezáleží na přesné trajektorii. Tento blok proto nemá parametry určující rychlost a zrychlení. Pohybuje se s maximální rychlostí a zrychlením nastavenými na jednotlivých osách/motorech. Trajektorie je generována tak, aby všechny motory dokončili pohyb ve stejnou dobu, proto se některé motory mohou pohybovat pomaleji, než je jejich limit. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem **CoordSystem**. Tento parametr musí mít tolik prvků, kolik je nastaveno souřadnic v přidruženém bloku **RM\_GroupAxes**. V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

### Vstupy

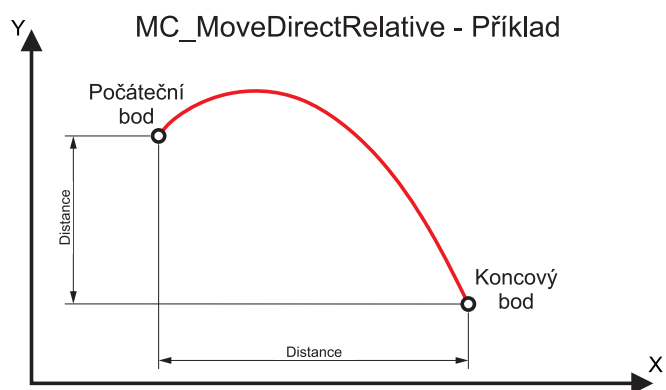
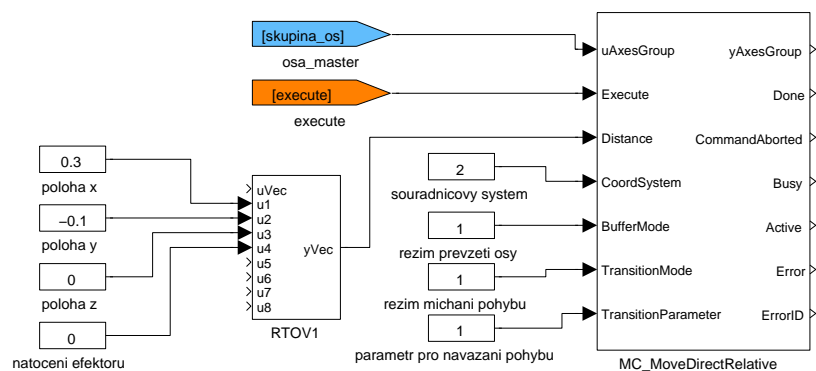
<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Execute</b>	Náběžná hrana aktivuje blok	<b>bool</b>
<b>Distance</b>	Pole souřadnic (relativních pozic a orientací)	<b>reference</b>
<b>CoordSystem</b>	Volba souřadného systému	<b>long</b>
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	

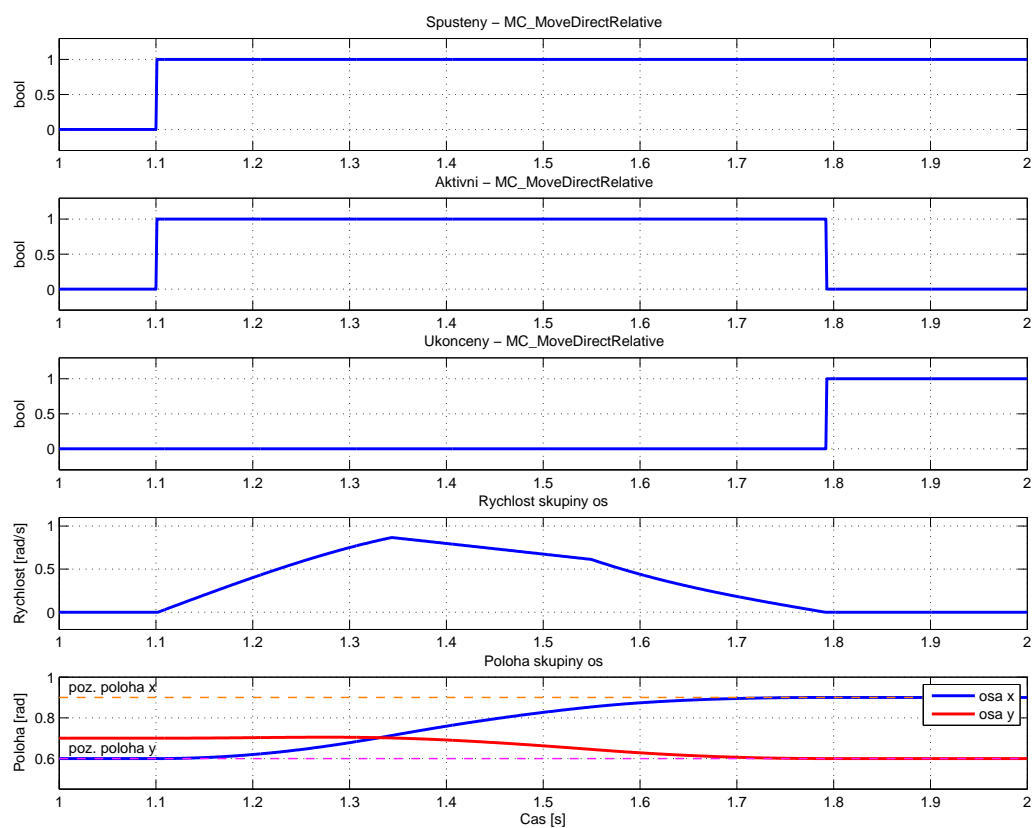
<b>BufferMode</b>	Režim převzetí osy	<b>long</b>
1	..... Aborting (nový blok se spustí okamžitě)	
2	..... Buffered (nový blok se spustí po dokončení předchozího)	
3	..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionMode</b>	Režim míchání pohybu	<b>long</b>
1	..... TMNone (xx)	
2	..... TMstartvelocity (proložení s danou počáteční rychlostí)	
3	..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
4	..... TMCornerDistance (xx)	
5	..... TMMaxCornerDeviation (xx)	
11	..... Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	<b>double</b>

## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Done</b>	Příznak dokončení algoritmu	<b>bool</b>
<b>CommandAborted</b>	Příznak přerušení funkce bloku	<b>bool</b>
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	<b>bool</b>
<b>Active</b>	Příznak, že blok řídí osu	<b>bool</b>
<b>Error</b>	Příznak chyby	<b>bool</b>
<b>ErrorID</b>	Kód chyby	<b>error</b>
i	..... obecná chyba systému REX	

## Příklad

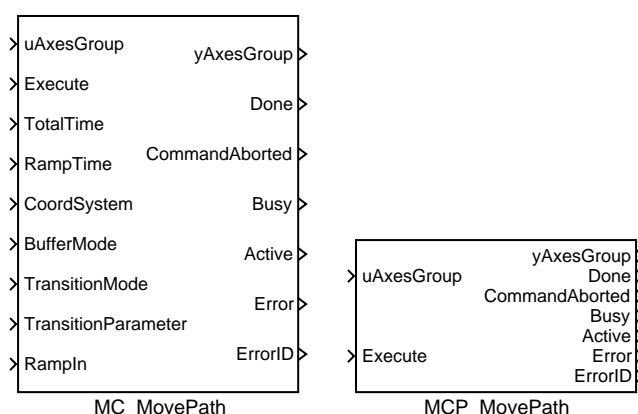




## MC\_MovePath, MCP\_MovePath – Generování obecné trajektorie v prostoru

Symboly bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_MovePath a MCP\_MovePath mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok MC\_MovePath slouží pro pohyb stroje (skupiny os) po definované křivce. Křivka je zadána pomocí NURBS aproximace pátého řádu. Křivka je vždy absolutní, proto je před spuštěním tohoto bloku nutné nastavit stroj do počátku křivky. V opačném případě je hlášena chyba -707 (skok v poloze). Blok umožňuje zadat parametry NURBS aproximace i ručně, ale to je prakticky neproveditelné (snad jen jako import z jiného programu). K tomuto bloku existuje speciální editor trajektorie (spouští se tlačítkem **SpecialEdit** v dialogu pro zadávání parametrů bloku), kde se křivka zadává/nakreslí pomocí bodů v prostoru, kterými má procházet a NURBS parametry jsou pak napočítány automaticky. NURBS parametrizace definuje pouze tvar křivky (pokud jistý parametr probíhá od 0 do 1 pak výstupní hodnota/vektor probíhá požadovanou křivku). Pro pohyb stroje musíme ještě definovat rychlost, s jakou křivku prochází. Blok podporuje 2 způsoby. Buď zadáme celkovou dobu jízdy a popřípadě dobu zrychlování na začátku a na konci pomocí parametrů **TotalTime** a **RampTime** nebo vytvoříme rychlostní profil pomocí parametrů **pv** a **pt**. Tento profil se skládá z několika intervalů a v každém se aproximuje polynomem 5. stupně. Ruční vytváření profilu je prakticky neproveditelné a editor trajektorií to zatím nepodporuje. Nouzově je možné použít editor bloku MC\_PositionProfile (kde mají data stejný formát), přičemž poloha musí začínat v 0 a končit v 1. Ve většině případů však

vystačíme s první metodou, protože editor trajektorií generuje NURBS tak, aby pro konstantní rychlost změny vstupního parametru byla přibližně konstantní rychlost pohybu po křivce. Teorie NURBS řeší aproximaci funkce jedné proměnné. Protože poloha je vektor, potřebujeme několik funkcí jedné proměnné (nezávisle proměnná je pro všechny tyto funkce parametr, který probíhá interval od 0 do 1). Teoreticky máme tedy pro každou souřadnici jednu NURBS funkci, nicméně použitá implementace má některé parametry společné pro všechny souřadnice (například řád, uzlové body). Ačkoliv v bloku lze použít souřadný systém ACS, grafický editor trajektorie předpokládá pravoúhlý souřadný systém. Taktéž jsou formálně dovoleny všechny buffered a transition režimy, ale vzhledem k povaze bloku je vhodné používat jen buffered nebo blending next (ostatní většinou vedou na chybu -707 - skok v poloze). Tento blok nemá vstup **PathData**, který vyžaduje specifikace **PLCopen**. V systému **REX** má tento blok všechna potřebná data (nebo odkazy na ně) uložena v parametrech. Z tohoto důvodu se také nepoužívá blok **MC\_PathSelect**.

## Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	reference
<b>Execute</b>	Náběžná hrana aktivuje blok	bool
<b>TotalTime</b>	Čas celého pohybu	double
<b>RampTime</b>	Čas [s] zrychlování/zpomalování	double
<b>CoordSystem</b>	Volba souřadného systému	long
	1 ..... ACM	
	2 ..... MCS	
	3 ..... PCS	
<b>BufferMode</b>	Režim převzetí osy	long
	1 ..... Aborting (nový blok se spustí okamžitě)	
	2 ..... Buffered (nový blok se spustí po dokončení předchozího)	
	3 ..... Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 ..... Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 ..... Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 ..... Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

<b>TransitionMode</b>	Režim míchání pohybu	long
1	..... TMNone (xx)	
2	..... TMstartvelocity (proložení s danou počáteční rychlostí)	
3	..... TMConstantVelocity (proložení s danou konstantní rychlostí)	
4	..... TMCornerDistance (xx)	
5	..... TMMaxCornerDeviation (xx)	
11	.... Smooth(nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
<b>TransitionParameter</b>	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	double
<b>RampIn</b>	RampIn faktor (0 = RampIn režim se nepoužívá)	double

## Parametry

<b>pc</b>	Matice řídicích bodů	double
	$\odot[0.0 \ 1.0 \ 2.0; \ 0.0 \ 1.0 \ 1.0; \ 0.0 \ 1.0 \ 0.0]$	
<b>pk</b>	Uzlový vektor	double
	$\odot[0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.5 \ 1.0 \ 1.0]$	
<b>pw</b>	Váhový vektor	double
	$\odot[1.0 \ 1.0 \ 1.0]$	
<b>pv</b>	Polynom pro korekci rychlosti posuvu	double
	$\odot[0.0 \ 0.05 \ 0.95; \ 0.0 \ 0.1 \ 0.1; \ 0.0 \ 0.0 \ 0.0; \ 0.1 \ 0.0 \ -0.1; \ -0.05 \ 0.0 \ 0.05; \ 0.0 \ 0.0 \ 0.0]$	
<b>pt</b>	Uzlové body pro korekci posuvu	double
	$\odot[0.0 \ 1.0 \ 10.0 \ 11.0]$	
<b>user</b>	Pouze pro special edit	double
	$\odot[0.0 \ 1.0 \ 2.0 \ 3.0]$	

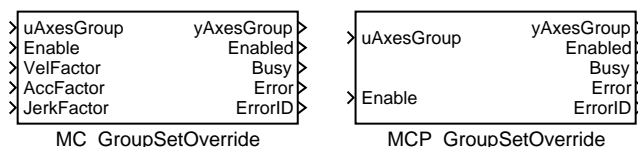
## Výstupy

<b>yAxesGroup</b>	Odkaz na skupinu os	reference
<b>Done</b>	Příznak dokončení algoritmu	bool
<b>CommandAborted</b>	Příznak přerušení funkce bloku	bool
<b>Busy</b>	Příznak, že algoritmus ještě neskončil	bool
<b>Active</b>	Příznak, že blok řídí osu	bool
<b>Error</b>	Příznak chyby	bool
<b>ErrorID</b>	Kód chyby	error
i	..... obecná chyba systému REX	

## MC\_GroupSetOverride, MCP\_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině

Symbole bloků

Licence: [COORDINATED MOTION CONTROL](#)



### Popis funkce

*Bloky MC\_GroupSetOverride a MCP\_GroupSetOverride mají naprosto shodnou funkci, jediným rozdílem je, že MCP\_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.*

Blok **MC\_GroupSetOverride** nastavuje násobivé faktory, které se projeví ve všech blocích pracujících se skupinou os. Hodnoty rychlosti, zrychlení a jerku ve všech blocích je nutné vynásobit faktorem z tohoto bloku, tím dostaneme hodnotu, se kterou blok skutečně pracuje. Toto se netýká limitních hodnot zadaných v [RM\\_Axis](#) a administrativních bloků. Tento blok není aktivován hranou, ale pokud je na vstupu **Enable** true, tak se hodnoty trvale aktualizují. Pokud je aktivní například blok typu [MC\\_MoveLinearAbsolute](#), vede to na neustálé přepočítávání trajektorie, což je výpočetně (a tím i časově) náročná operace a navíc se kumulují zaokrouhlovací chyby. Proto je zavedena necitlivost (parametr **diff**) a přepočet trajektorie je proveden, až když se některý z faktorů změní více, než je tato necitlivost.

Poznámka 1: Všechny faktory musí být kladné. Faktory větší než 1 jsou možné, ale často vedou k překročení mezí nastavených na ose a k selhání pohybu (blok hlásí chybu **errorID** = -700 - neplatný parametr) nebo dokonce k havarijnímu zastavení osy (blok pak hlásí chybu **errorID** = -701 - hodnota mimo rozsah).

### Vstupy

<b>uAxesGroup</b>	Odkaz na skupinu os	<b>reference</b>
<b>Enable</b>	Povolení funkce bloku (aktivace výstupů)	<b>bool</b>
<b>VelFactor</b>	Faktor násobení pro rychlost	<b>double</b>
<b>AccFactor</b>	Faktor násobení pro zrychlení	<b>double</b>
<b>JerkFactor</b>	Faktor násobení pro změnu zrychlení	<b>double</b>



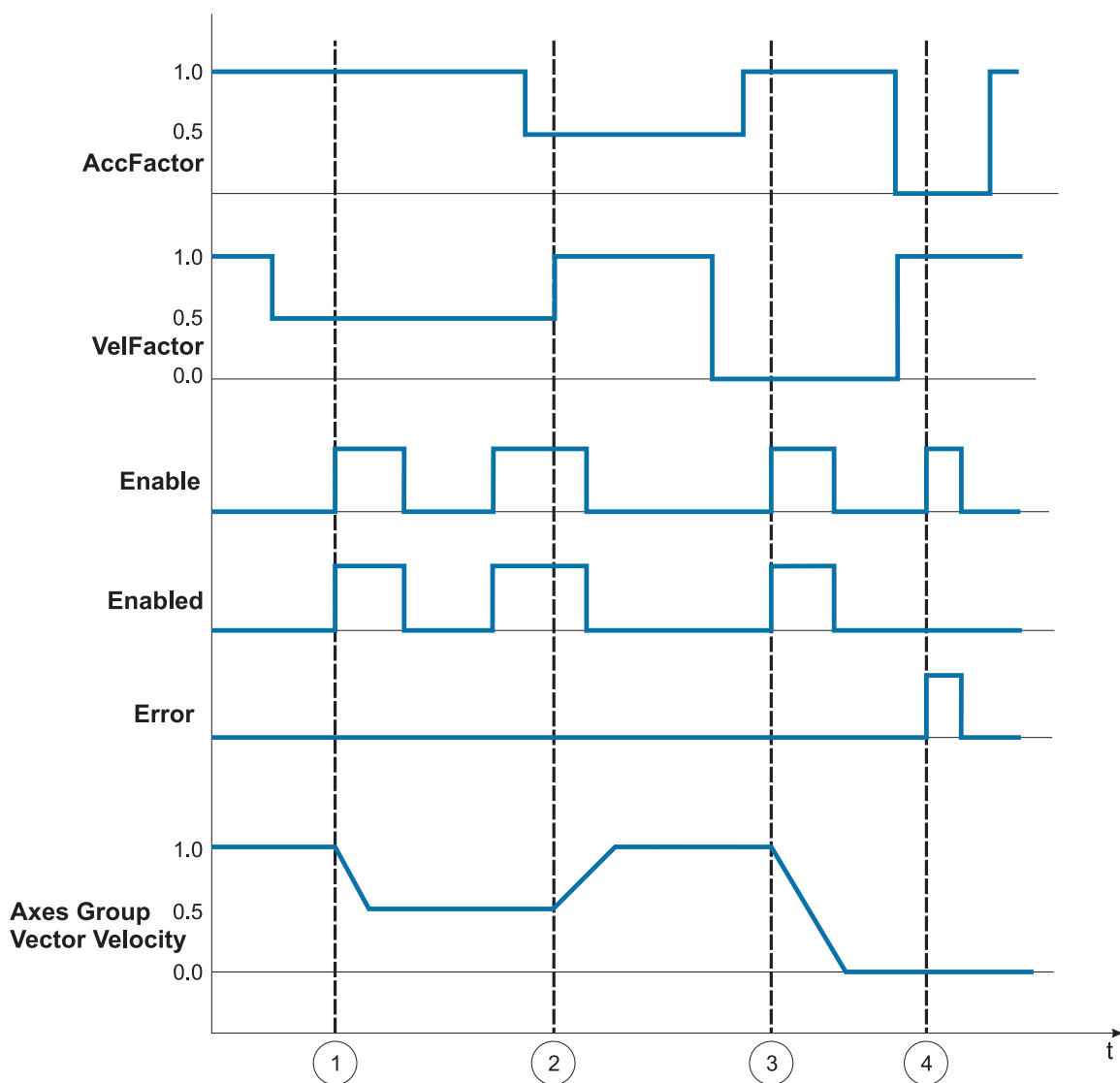
## Parametr

<code>diff</code>	Pásmo necitlivosti (pro přepoččet trajektorie)	$\odot 0.05$	<code>double</code>
-------------------	--	--------------	---------------------

## Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	<code>reference</code>
<code>Enabled</code>	Signalizuje úspěšné nastavení násobivých faktorů	<code>bool</code>
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	<code>bool</code>
<code>Error</code>	Příznak chyby	<code>bool</code>
<code>ErrorID</code>	Kód chyby	<code>error</code>
	<code>i . . . . .</code> obecná chyba systému REX	

## Příklad



- ① Rychlost skupiny os zmenšena na 50% se 100% zpomalením
- ② Rychlost skupiny os vrácena na 100% s 50% akcelerací
- ③ Rychlost skupiny os zmenšena na 0% se 100% zpomalením
- ④ Žádná změna, protože není povolen  $\text{AccFactor} = 0.0$ , je sepnuta chyba

## 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_	•	ADVANCED
ABSROT		
ACD	•	ADVANCED
ADD	•	
ADDHEXD	•	
ADDOCT	•	
ADDQUAD	•	
AFLUSH	•	
ALB	•	
ALBI	•	
ALN	•	
ALNI	•	
AND_	•	
ANDHEXD	•	
ANDOCT	•	
ANDQUAD	•	
ANLS	•	
ARC	•	
ARLY	•	
ASW		ADVANCED

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ATMT	•	
AVG	•	
AVS		ADVANCED
BDHEXD	•	
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	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
EXEC	•	
FIND	•	
FLCU		ADVANCED
FMUCS		ADVANCED
FMUINFO		ADVANCED
FNX	•	
FNXY	•	
FOPDT	•	
FRID		ADVANCED
From	•	
GAIN	•	
GETPA	•	
GETPB	•	
GETPI	•	
GETPR	•	
GETPS	•	
Goto	•	
GotoTagVisibility	•	
GRADS		ADVANCED
HMI	•	
HTTP		ADVANCED
HTTP2		ADVANCED
I3PM		ADVANCED
IADD	•	
IDIV	•	
IMOD	•	
IMUL	•	
INFO	•	
INHEXD	•	
INOCT	•	
Inport	•	
INQUAD	•	
INSTD	•	
INTE	•	
INTSM	•	
IODRV	•	
IOTASK	•	
ISSW	•	
ISUB	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ITOI	•	
KDER		ADVANCED
LC	•	
LEN	•	
LIN	•	
LLC	•	
LPBRK	•	
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

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
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
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

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
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
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_	•	

Seznam pokračuje na další stránce...



Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ORHEXD	•	
OROCT	•	
ORQUAD	•	
OSCALL	•	
OUTHEXD	•	
OUTOCT	•	
Outport	•	
OUTQUAD	•	
OUTRHEXD		ADVANCED
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

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
REL	•	
REPLACE	•	
REXLANG		REXLANG
RLIM	•	
RLY	•	
RM_AxesGroup		COORDINATED MOTION CONTROL
RM_Axis		MOTION CONTROL
RM_AxisOut		MOTION CONTROL
RM_AxisSpline		MOTION CONTROL
RM_Feed		COORDINATED MOTION CONTROL
RM_Gcode		COORDINATED MOTION CONTROL
RM_GroupTrack		COORDINATED MOTION CONTROL
RM_Track		MOTION CONTROL
RS	•	
RTOI	•	
RTOS	•	
RTOV	•	
S10F2		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	•	

*Seznam pokračuje na další stránce...*

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
SILO	•	
SILOS	•	
SINT	•	
SLEEP	•	
SMHCC		ADVANCED
SMHCCA		AUTOTUNING
SMTF		ADVANCED
SOPDT	•	
SPIKE		ADVANCED
SQR	•	
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
- 127 .... Zdroj nelze použít, neboť je uzamčen jiným modulem

#### 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 .... Vyrovňovací paměť pro příjem je příliš malá
- 440 .... Vyrovňovací paměť 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 generování 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

### Kódy spojené s knihovnou RexVision

- 1000 ... Výsledek není vyhodnocen
- 1001 ... Nelze nalézt hledaný objekt/vzor
- 1002 ... Zadanému kritériu hledání vyhovuje více objektů

### Kódy spojené se standardem FMI

- 1100 ... Nepodařilo se alokovat kontext FMI
- 1101 ... Nesprávná verze FMU
- 1102 ... Chyba parsování souboru XML pro FMI
- 1103 ... Vyžadován pouze druh FMI pro Model Exchange
- 1104 ... Vyžadován pouze druh FMI pro Co-Simulation
- 1105 ... Nepodařilo se zavést FMU
- 1106 ... Nepodařilo se vytvořit instanci FMU
- 1107 ... Nepodařilo se ukončit instanci FMU
- 1108 ... Selhal reset FMU
- 1109 ... Selhalo nastavení experimentu FMU
- 1110 ... Selhalo zahájení inicializačního módu FMU
- 1111 ... Selhalo ukončení inicializačního módu FMU
- 1112 ... Chyba získání seznamu proměnných FMU
- 1113 ... Chyba čtení reálné proměnné z FMU
- 1114 ... Chyba zápisu reálné proměnné do FMU
- 1115 ... Chyba čtení celočíselné proměnné z FMU
- 1116 ... Chyba zápisu celočíselné proměnné do FMU
- 1117 ... Chyba čtení booleovské proměnné z FMU
- 1118 ... Chyba zápisu booleovské proměnné do FMU
- 1119 ... Chyba provedení simulačního kroku FMU
- 1120 ... FMU má příliš mnoho vstupů
- 1121 ... FMU má příliš mnoho výstupů
- 1122 ... FMU má příliš mnoho parametrů

# 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.
- [7] Modelica Association Project FMI. *Functional Mock-up Interface for Model Exchange and Co-Simulation, version 2.0*, July 2014.



# Rejstřík

- úloha
  - rychlá, [30](#)
  - standardní, [35](#)
- čítání pulsů
  - obousměrné, [240](#)
- čítač řízený, [240](#)
- časovač, [254](#)
  - systémový, [26](#)
  - týdenní, [263](#)
- řízení
  - pohybu, [13](#), [108](#)
  - sekvenční, [234](#)
- šířka pásma, [121](#)
- šířková modulace, [201](#)
- TODO
  - SRTF DGLOG, [33](#)
  - X, [107](#), [113–116](#), [145](#), [149](#), [336–338](#), [341](#)
- ABS\_, [65](#), [555](#)
- absolutní
  - snímač polohy, [103](#)
- ABSR0T, [103](#), [555](#)
- ACD, [271](#), [555](#)
- ADD, [66](#), [67](#), [555](#)
- ADDHEXD, [67](#), [555](#)
- ADDOCT, [66](#), [67](#), [99](#), [555](#)
- ADDQUAD, [67](#), [555](#)
- AFLUSH, [281](#), [555](#)
- alarm
  - číselná hodnota, [269](#)
  - logická hodnota, [267](#)
- ALB, [267](#), [555](#)
- ALBI, [267](#), [555](#)
- ALN, [269](#), [555](#)
- ALNI, [269](#), [555](#)
- AND\_, [232](#), [233](#), [555](#)
- ANDHEXD, [233](#), [555](#)
- ANDOCT, [232](#), [233](#), [555](#)
- ANDQUAD, [233](#), [555](#)
- ANLS, [152](#), [555](#)
- aplikace
  - řídícího systému REX, [20](#)
- ARC, [18](#), [21](#), [268](#), [270](#), [272](#), [275](#), [277](#), [281](#), [555](#)
- architektura
  - otevřená, [28](#)
- archiv, [18](#), [266](#)
  - alarmů, [18](#)
  - konfigurace, [18](#)
  - na disku, [266](#)
  - trendů, [18](#)
  - událostí, [18](#)
  - v paměti RAM, [266](#)
  - v zálohované paměti, [266](#)
- archivace
  - delta kritérium, [271](#)
- ARLY, [165](#), [555](#)
- ASW, [105](#), [555](#)
- ATMT, [12](#), [234](#), [241](#), [302](#), [311](#), [315](#), [556](#)
- automat
  - pro sekvenční řízení, [234](#)
- automaton
  - finite-state, [241](#)
- AVG, [107](#), [556](#)
- AVS, [12](#), [108](#), [556](#)
- běh úloh, [32](#)
- BDHEXD, [237](#), [241](#), [556](#)
- BDOCT, [237](#), [241](#), [556](#)
- Besselův filtr, [121](#)
- binární číslo
  - transformace, [247](#)

- binární posloupnost
  - generátor, 154, 156
- BINS, 154, 556
- BIS, 156, 157, 556
- BITOP, 238, 556
- bitová operace, 238
- blok
  - formát popisu, 13
  - komunikační, 357
  - parametry, 13
  - popis funkce, 13
  - symbol, 13
  - výstupu, 13
  - volně programovatelný, 362
  - vstupy, 13
- bloky
  - generátory, 12
  - matematické, 12
  - maticové, 12
  - pro archivaci dat, 12
  - pro logické řízení, 12
  - pro modelování, 12
  - pro práci s parametry, 12
  - pro regulaci, 12
  - pro zpracování analogových signálů, 12
  - speciální, 13
  - vektorové, 12
  - vstupně-výstupní, 11
- BMHEXD, 239, 241, 556
- BMOCT, 239, 241, 556
- BPF, 109, 556
- Butterworthův filtr, 121
- CDELSSM, 322, 556
- celé číslo
  - transformace, 247
- celočíslný signál
  - přepínání, 246
- cesta
  - úplná, 32
- chyba
  - fatální, 30
- CMP, 110, 556
- CNA, 344, 556
- CNB, 68, 556
- CNDR, 111, 556
- CNE, 69, 556
- CNI, 70, 556
- CNR, 71, 556
- CNS, 284, 556
- CONCAT, 285, 556
- control
  - sequential, 241
- COUNT, 240, 556
- CSSM, 325, 556
- dělení
  - celočíslné, 89
  - dvou signálů, 73
  - rozšířené, 75
  - zbytek, 90
- DATE\_, 258, 259, 556
- DATETIME, 258, 259, 262, 556
- DDELSSM, 328, 556
- DEL, 113, 556
- DELM, 114, 556
- delta kritérium, 271
- demultiplexer
  - bitový, 237
- DER, 115, 556
- derivace, 115, 119
- detekce
  - hrany, 244
- DIF\_, 72, 556
- diference, 72
- Display, 42, 556
- DIV, 73, 556
- DLL knihovna, 28
- dopravní zpoždění, 114, 336, 340
  - s inicializací, 113
  - variantní, 145
- DSSM, 330, 556
- EAS, 74, 556
- EATMT, 241, 556
- EDGE\_, 137, 244, 556
- EMD, 75, 556
- EPC, 34, 350, 556

- EVAR, 116, 556
- EXEC, 18, 20, 24–26, 28, 30, 31, 35–37, 381, 557
- exekutiva
  - konfigurace, 11, 17
  - program RexCore, 11
  - reálného času, 20
- externí program, 350
- filtr
  - šířka pásma, 121
  - Besselův, 121
  - Butterworthův, 121
  - dolní propust', 121
  - nelineární, 141
  - pásmová propust', 109
  - pulzů, 141
  - vlečný průměr, 107
- filtrace, 115, 119
  - číslicová vstupních signálů, 30
- FIND, 286, 557
- finite-state machine, 241
- FLCU, 12, 166, 557
- FMUCS, 332, 557
- FMUINFO, 334, 557
- FNX, 76, 557
- FNXY, 78, 557
- FOPDT, 173
- FOPDT, 336, 557
- Fourierova transformace, 124
- frekvenční charakteristika, 176
- FRID, 168, 557
- From, 43, 45–47, 557
- funkce
  - dvou proměnných, 78
  - jedné proměnné, 76
  - operačního systému, 34
- GAIN, 80, 557
- generátor
  - časových funkcí, 195
  - binární posloupnosti, 154, 156
  - po částech lineární funkce, 152
  - signálu, 160
- GETPA, 300, 557
- GETPB, 302, 557
- GETPI, 302, 557
- GETPR, 302, 315, 557
- GETPS, 304, 557
- Goto, 43–45, 47, 557
- GotoTagVisibility, 46, 47, 557
- GRADS, 81, 557
- hierarchie, 50
- HMI, 22, 557
- hodnota
  - implicitní, 14
  - maximální, 14
  - minimální, 14
  - náhradní, 73, 75, 76, 78, 89, 90, 94, 98
  - převrácená, 94
  - polynomu, 93
  - střední, 116
- HTTP, 353, 557
- HTTP2, 557
- hystereze, 110
- I3PM, 170, 557
- IADD, 83, 557
- identifikace
  - modelu se třemi parametry, 170
- IDIV, 89, 557
- IMOD, 90, 557
- IMUL, 87, 557
- INFO, 23, 557
- INHEXD, 51, 557
- inicializace
  - pořadí modulů, 24
  - pořadí ovladačů, 24
  - rychlé úlohy, 30
- INOCT, 51, 557
- Inport, 48, 50, 557
- INQUAD, 51, 557
- INSTD, 43, 51, 53, 557
- INTE, 117, 140, 557
- integrátor
  - řízený, 117
  - jednoduchý, 140

- interpolace
  - lineární, 91
- INTSM, 245, 557
- IODRV, 21, 24, 43, 45, 557
- IOTASK, 26, 33, 37, 300, 302, 309, 311, 323, 326, 381, 557
- ISSW, 246, 557
- ISUB, 85, 557
- ITOI, 247, 558
- jednotka
  - rozběhová, 108
- jmenovatel, 75
- KDER, 119, 558
- klopný obvod
  - Reset-Set, 252
  - Set-Reset, 253
- komparátor, 110
- kompatibilia
  - REX a Simulink, 27
- kompensátor
  - derivační, 172
  - integračně-derivační, 173
  - jednoduché nelinearity, 123
  - složitě nelinearity, 111
- komprese, 271
- konfigurace
  - archivy, 20
  - moduly, 20
  - systému REX, 20
  - výpočetní úloha, 20
  - vstupně-výstupní ovladače, 20
- konstanta
  - Booleovská, 68
  - celočíslná, 70
  - logická, 68
  - reálná, 71
- konverze
  - reálného čísla na celé, 96
- krokový regulátor, 212, 215
- LC, 172, 558
- LEN, 287, 558
- LIN, 91, 558
- lineární
  - interpolace, 91
- LLC, 173, 558
- logické NEBO, 250
- LPBRK, 11, 27, 105, 558
- LPF, 121, 558
- maximum, 122
- MC\_AccelerationProfile, 388, 558
- MC\_AddAxisToGroup, 491, 496, 558
- MC\_CamIn, 456, 460, 463, 477, 480, 558
- MC\_CamOut, 456, 460, 558
- MC\_CombineAxes, 464, 558
- MC\_GearIn, 467, 470, 475, 477, 480, 558
- MC\_GearInPos, 470, 558
- MC\_GearOut, 467, 475, 558
- MC\_GroupContinue, 517, 518, 558
- MC\_GroupDisable, 499, 558
- MC\_GroupEnable, 491, 498, 558
- MC\_GroupHalt, 512, 558
- MC\_GroupInterrupt, 517, 518, 558
- MC\_GroupReadActualAcceleration, 491, 508, 558
- MC\_GroupReadActualPosition, 491, 506, 558
- MC\_GroupReadActualVelocity, 491, 507, 558
- MC\_GroupReadError, 521, 558
- MC\_GroupReadStatus, 519, 558
- MC\_GroupReset, 522, 558
- MC\_GroupSetOverride, 552, 558
- MC\_GroupSetPosition, 504, 558
- MC\_GroupStop, 509, 513, 558
- MC\_Halt, 392, 558
- MC\_HaltSuperimposed, 394, 558
- MC\_Home, 395, 426, 558
- MC\_MoveAbsolute, 397, 411, 434, 452, 471, 558
- MC\_MoveAdditive, 401, 558
- MC\_MoveCircularAbsolute, 486, 531, 558
- MC\_MoveCircularRelative, 486, 536, 558
- MC\_MoveContinuousAbsolute, 410, 558
- MC\_MoveContinuousRelative, 414, 558
- MC\_MoveDirectAbsolute, 486, 541, 559
- MC\_MoveDirectRelative, 486, 545, 559



- MC\_MoveLinearAbsolute, 486, 510, 513, 515, 523, 552, 559
- MC\_MoveLinearRelative, 486, 527, 559
- MC\_MovePath, 486, 549, 559
- MC\_MovePath\_PH, 559
- MC\_MoveRelative, 404, 407, 415, 559
- MC\_MoveSuperimposed, 394, 407, 477, 480, 559
- MC\_MoveVelocity, 381, 418, 559
- MC\_PhasingAbsolute, 477, 559
- MC\_PhasingRelative, 480, 559
- MC\_PositionProfile, 421, 452, 462, 559
- MC\_Power, 425, 559
- MC\_ReadActualPosition, 426, 559
- MC\_ReadAxisError, 427, 559
- MC\_ReadBoolParameter, 428, 559
- MC\_ReadCartesianTransform, 503, 559
- MC\_ReadParameter, 429, 559
- MC\_ReadStatus, 431, 559
- MC\_Reset, 433, 522, 559
- MC\_SetCartesianTransform, 485, 500, 559
- MC\_SetCartesianTransforms, 491
- MC\_SetOverride, 434, 559
- MC\_SetPosition, 395
- MC\_Stop, 436, 559
- MC\_TorqueControl, 384, 438, 559
- MC\_UngroupAllAxes, 497, 559
- MC\_UngroupAllAxis, 491
- MC\_VelocityProfile, 441, 559
- MC\_WriteBoolParameter, 445, 559
- MC\_WriteParameter, 446, 559
- MCP\_AccelerationProfile, 388, 559
- MCP\_CamIn, 456, 462, 559
- MCP\_CamTableSelect, 456, 457, 462, 559
- MCP\_CombineAxes, 464, 559
- MCP\_GearIn, 467, 559
- MCP\_GearInPos, 470, 559
- MCP\_GroupHalt, 512, 559
- MCP\_GroupInterrupt, 517, 559
- MCP\_GroupSetOverride, 552, 559
- MCP\_GroupSetPosition, 504, 559
- MCP\_GroupStop, 509, 559
- MCP\_Halt, 392, 560
- MCP\_HaltSuperimposed, 394, 560
- MCP\_Home, 395, 560
- MCP\_MoveAbsolute, 397, 560
- MCP\_MoveAdditive, 401, 560
- MCP\_MoveCircularAbsolute, 531, 560
- MCP\_MoveCircularRelative, 536, 560
- MCP\_MoveContinuousAbsolute, 410, 560
- MCP\_MoveContinuousRelative, 414, 560
- MCP\_MoveDirectAbsolute, 541, 560
- MCP\_MoveDirectRelative, 545, 560
- MCP\_MoveLinearAbsolute, 523, 560
- MCP\_MoveLinearRelative, 527, 560
- MCP\_MovePath, 549, 560
- MCP\_MovePath\_PH, 560
- MCP\_MoveRelative, 404, 560
- MCP\_MoveSuperimposed, 407, 560
- MCP\_MoveVelocity, 418, 560
- MCP\_PhasingAbsolute, 477, 560
- MCP\_PhasingRelative, 480, 560
- MCP\_PositionProfile, 421, 560
- MCP\_SetCartesianTransform, 500, 560
- MCP\_SetKinTransform\_Arm, 560
- MCP\_SetOverride, 434, 560
- MCP\_Stop, 436, 560
- MCP\_TorqueControl, 438, 560
- MCP\_VelocityProfile, 441, 560
- MCU, 174, 228, 560
- MDL, 337, 338, 560
- MDLI, 338, 560
- metoda nejmenších čtverců, 115
- MID, 288, 560
- minimum, 122
- MINMAX, 122, 560
- mocnina
  - druhá, 97
- model
  - druhého řádu s dopravním zpožděním, 340
  - FOPDT, 173, 336
  - procesu, 337
  - procesu s proměnnými parametry, 338
  - prvního řádu s dopravním zpožděním, 336
  - SOPDT, 340
  - stavový

- diskrétní, 330
- diskrétní s dopravním zpožděním, 328
- spojitý, 325
- spojitý s dopravním zpožděním, 322
- modul, 28
  - rozšiřující, 24
  - rozšiřující řídicího systému REX, 28
- modulace
  - šířková, 201
- MODULE, 21, 24, 28, 560
- motion control, 13
- MP, 157, 560
- MUL, 92, 560
- multiplexer
  - bitový, 239
- MVD, 339, 560
- násobení
  - celočíslné, 87
  - dvou signálů, 92
  - konstantou, 80
  - rozšířené, 75
- negace
  - logická, 249
- nelineární transformace
  - jednoduchá, 123
- NOT\_, 249, 560
- NSCL, 123, 560
- obvod
  - klopný Reset-Set, 252
  - klopný Set-Reset, 253
- odčítání
  - celočíslné, 85
  - dvou signálů, 99
  - rozšířené, 74
- odchylka
  - směrodatná, 116
- odmocnina
  - druhá, 98
- omezovač strmosti, 126
- OPC server, 361
- operační systém, 34
- operace
  - binární, 95
  - bitová, 238
  - relace, 95
- optimalizace
  - gradientní, 81
- OR\_, 250, 251, 560
- ORHEXD, 251, 561
- OROCT, 250, 251, 561
- ORQUAD, 251, 561
- OSCALL, 34, 352, 561
- OUTHEXD, 53, 55, 561
- OUTOCT, 53, 55, 561
- Outport, 48, 50, 561
- OUTQUAD, 53, 55, 561
- OUTRHEXD, 55, 561
- OUTROCT, 55, 561
- OUTRQUAD, 55, 561
- OUTRSTD, 57, 561
- OUTSTD, 45, 51, 53, 561
- ovladač
  - konfigurační data, 24
  - pořadí inicializace, 24
  - soubor s příponou .rio, 24
  - systém REX, 11, 24
  - uživatelská dokumentace, 26
  - vstupně-výstupní, 11, 24
  - vstupně-výstupní s úlohami, 37
- pásmo propustnosti, 109
- překlad
  - program RexComp, 27
- překladač RexComp, 20
- přepínač
  - celočíslných signálů, 246
  - jednoduchý, 143
  - s automatickou volbou vstupu, 105
  - s rampovou funkcí, 144
  - vstupu pro vysledování, 228
- převrácená hodnota, 94
- PARA, 305, 561
- parametr
  - tick, 20
  - nastavitelný ze vstupu, 306
  - vzdáleně nastavovaný, 309, 311

- vzdáleně získávaný, 300, 302
- PARB, 306, 561
- PARI, 306, 561
- PARR, 306, 561
- PARS, 308, 561
- PID
  - PID regulátor, 189
  - s autotunerem, 176
  - s momentovým autotunerem, 183
  - s přepínáním parametrů, 181
  - s parametry na vstupech, 192
  - se statikou, 179
- PIDAT, 12, 176, 561
- PIDE, 179, 561
- PIDGS, 12, 181, 561
- PIDMA, 12, 183, 212, 360, 561
- PIDU, 176, 179, 181, 189, 192, 212, 215, 228, 561
- PIDUI, 192, 561
- PJROCT, 289, 561
- PJSOCT, 291, 561
- pořadí
  - inicializace úloh, 35
  - inicializace modulů, 28
  - spouštění úloh, 35
  - zavádění modulů, 28
- podíl, 73
  - celočíselný, 89
- POL, 93, 561
- poloha
  - absolutní snímač, 103
- polynom
  - vyhodnocení, 93
- posloupnost
  - binární pseudonáhodná, 158
- potlačení
  - vibrací, 146
- POUT, 194, 561
- průměr
  - vlečný, 107
- PRBS, 158, 561
- predikce, 115
- prediktivní řízení, 197
- PRGM, 195, 561
- priorita
  - úloh, 35
  - logická, 20, 24, 30
  - závislost na operačním systému, 21
- program
  - externí, 350
  - RexComp, 27
  - RexDraw, 24
  - RexView, 18, 25, 35
  - RexView příznak Enable, 32
  - RexView tlačítko Halt/Run, 32
  - RexView tlačítko RESET, 32
  - týdenní, 263
- PROJECT, 29, 561
- projekt
  - hlavní soubor, 20, 24
- protokol
  - UDP/IP, 357
- prvek
  - třístavový, 229
- PSMPC, 197, 561
- pulz, 194
  - ručně generovaný, 157
- pulzní výstup, 194
- PWM, 182, 187, 191, 193, 201, 221, 561
- QFC, 58, 59, 367, 561
- QFD, 55, 57–59, 367, 561
- QTASK, 20, 21, 26, 30, 33, 35, 323, 326, 561
- Rate monotonic scheduling, 21
- RDC, 13, 357, 361, 561
- RDFT, 124
- reálný čas
  - exekutiva, 17
- režie
  - jádra řídicího systému, 20
- REC, 94, 561
- REGEXP, 293, 561
- regulátor
  - fuzzy, 166
  - krokový s polohovou zpětnou vazbou, 212
  - krokový s rychlostním vstupem, 215

- PID, 189
- PID s autotunerem, 176
- PID s momentovým autotunerem, 183
- PID s přepínáním parametrů, 181
- PID s parametry na vstupech, 192
- PID se statikou, 179
- prediktivní, 197
- s klouzavým režimem, 221
- stavový s frekvenčním autotunerem, 206
- REL, 95, 562
- relé
  - s hysterezí, 203
  - s předstihem, 165
- REPLACE, 294, 562
- REXLANG, 13, 362, 562
- RLIM, 126, 562
- RLY, 203, 562
- RM\_AxesGroup, 484, 488, 489, 496, 498, 499, 521, 562
- RM\_Axis, 347, 381, 382, 395, 426–429, 434, 445, 446, 448, 450, 488, 489, 491, 496, 499, 552, 562
- RM\_AxisOut, 448, 450, 488, 562
- RM\_AxisSpline, 381, 450, 488, 562
- RM\_Feed, 493, 562
- RM\_Gcode, 494, 562
- RM\_GroupTrack, 562
- RM\_Track, 452, 562
- rozdíl
  - celočíslný, 85
- rozptyl, 116
- rozvrh
  - týdenní, 263
- RS, 252, 562
- RTOI, 96, 562
- RTOS, 295, 562
- RTOV, 345, 351, 484, 562
- rychlá smyčka, 27
- S10F2, 127, 562
- sčítání
  - celočíslné, 83
  - dvou signálů, 66
  - rozšířené, 74
  - vícevstupové, 67
- SAI, 127, 129, 130, 562
- sample&hold, 139
- SAT, 204, 562
- saturace výstupu, 204
- SC2FA, 206, 562
- SCU, 182, 187, 191, 193, 212, 215, 562
- SCUV, 182–184, 187, 189–191, 193, 215, 562
- sekvenční řízení, 234
- SEL, 133, 562
- selektor
  - aktivního regulátoru, 219
  - analogového signálu, 133
  - signálu, 127
  - zabezpečený, 127
- SELHEXD, 133, 135, 562
- SELOCT, 133, 135, 562
- SELQUAD, 133, 135, 562
- SELSOCT, 296, 562
- SELU, 219, 347, 562
- sequential control, 241
- servoventil, 339
- SETPA, 309, 562
- SETPB, 311, 562
- SETPI, 311, 562
- SETPR, 311, 315, 562
- SETPS, 313, 562
- SG, 160, 562
- SGI, 160, 562
- SGSLP, 302, 312, 314, 318, 562
- SHIFTOCT, 137, 562
- SHLD, 139, 562
- SILO, 316, 318, 563
- SILOS, 319, 563
- simulace
  - běh v reálném čase, 31
  - parametry, 31
- Simulink, 27, 31, 357
- SINT, 117, 140, 563
- SLEEP, 11, 31, 563
- směrodatná odchylka, 116
- SMHCC, 221, 563
- SMHCCA, 225, 563
- SMTP, 355, 563

- snímač polohy
  - absolutní, 103
- SOPDT, 340, 563
- součet, 66
  - celočíselný, 83, 87
  - logický dvou signálů, 250
- součin
  - logický, 232, 233
- součinitel relativního tlumení, 109
- SPIKE, 130–132, 141, 563
- SQR, 97, 563
- SQRT\_, 98, 563
- SR, 253, 563
- SRTF, 32, 563
- SSW, 143, 347, 563
- střední hodnota, 116
- state machine, 241
- stavový model, 325, 330
  - s dopravním zpožděním, 322, 328
- STOR, 297, 563
- strmost
  - omezení, 126
- SUB, 67, 99, 563
- subsystém, 50
  - archivační, 265
- SubSystem, 50, 563
- SWR, 144, 347, 563
- SWU, 228, 563
- SWVMR, 347, 563
- systém
  - druhého řádu, 340
  - prvního řádu, 173, 336
- týdenní časovač, 263
- třístavový výstup, 229
- TASK, 20, 21, 26, 30, 33, 35, 37, 323, 326, 381, 563
- task
  - quick, 30
- TIME, 259, 262, 563
- TIMER\_, 254, 563
- TIODRV, 21, 26, 37, 563
- trajektorie
  - časově optimální, 108
- transformace
  - binárních čísel, 247
  - celých čísel, 247
- trend
  - záznam, 273, 276
- TRND, 273, 276, 563
- TRNDLF, 278, 563
- TRNDV, 276, 563
- TRNDVLF, 280, 563
- TSE, 212, 215, 229, 563
- tvarovač
  - pro potlačení vibrací, 146
- typ
  - parametr, 14
  - výstup, 14
  - vstup, 14
- typy
  - proměnných, 14
- výběr
  - analogového signálu, 127
- výstup
  - pulzní, 194
- VDEL, 145, 563
- ventil
  - s motorizovaným pohonem, 339
- vibrace
  - potlačení, 146
- VIN, 55, 57, 59, 60, 367, 563
- vlečný průměr, 107
- VOUT, 58, 62, 367, 563
- VTOR, 124, 348, 351, 484, 491, 563
- vzorkovač, 139
- WSCH, 263, 563
- WWW, 39, 563
- zásobník
  - velikost, 24
- záznam dat, 273, 276
- zabezpečený analogový vstup, 130
- zadávání
  - ruční, 174
- zesílení, 80
- zpětná vazba, 27

zpoždění

dopravní, [114](#), [336](#), [340](#)

ZV4IS, [146](#), [563](#)

