ADS Matlab a Simulink toolbox

# Kompilácia kódu

Pre každú odlišnú verziu Matlabu a Simulinku je nutné funkcie toolboxu individuálne prekompilovať.

Na tento účel slúži zvláštny skript kompilujúci ako implementované MEX funkcie tak aj S funkcie. Kompiláciu všetkých funkcii teda realizujeme spustením *build\_script.m*. Prirodzene je nutné mať všetky zdrojové súbory , hlavičkové súbory (.cpp .h) a knižnice (.lib,.dll) v pracovnom priečinku. Taktiež treba mať v Matlabe nainštalovaný a nastavený kompilátor (najlepšie MinGW). Kód je tak úspešne skompilovaný po vypísaní:

Building with 'MinGW64 Compiler (C++)'.

MEX completed successfully.

Kompilačný script vyzerá nasledovne.

mex ('ADS\_write\_mex.cpp','ADS\_lib.cpp',strcat('-L',pwd()),'-lTcAdsDll')

mex ('ADS\_read\_mex.cpp','ADS\_lib.cpp',strcat('-L',pwd()),'-lTcAdsDll')

mex ('ADS\_open\_mex.cpp','ADS\_lib.cpp',strcat('-L',pwd()),'-lTcAdsDll')

mex ('ADS\_close\_mex.cpp','ADS\_lib.cpp',strcat('-L',pwd()),'-lTcAdsDll')

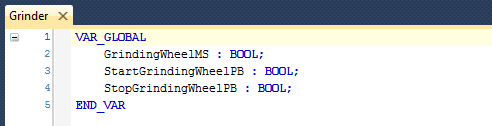
mex ('ADS\_setup\_SFUN.cpp','ADS\_lib.cpp',strcat('-L',pwd()),'-lTcAdsDll')

mex ('ADS\_write\_SFUN.cpp','ADS\_lib.cpp',strcat('-L',pwd()),'-lTcAdsDll')

mex ('ADS\_read\_SFUN.cpp','ADS\_lib.cpp',strcat('-L',pwd()),'-lTcAdsDll')

# Dátové typy

Každá premenná v systéme TwinCat má svoj dátový typ. Ten sa dá zistiť v zozname globálnych premenných (global variable list - GVL) na strane servera.



V rámci tohto ADS toolboxu sú týmto dátovým typom priradené identifikačné čísla. Preto pri práci s väčšinou funkcii je nutné poznať dátový typ premennej s ktorou chceme pracovať a teda poznať aj číslo tohto dátového typu. Identifikačné číslo dátového typu vstupuje ako parameter do MEX funkcii a rovnako ako parameter var\_type v prípade masiek S-funkcii.

Zoznam dátových typov vyzerá nasledovne , je rovnako obsiahnutý v súbore *TC\_data\_types.m*

Je vhodné si tento súbor pred prácou spustiť a zadefinovať si tak prehľadnejšie mená pre dátové typy.

%ciselne oznacenie datovych typov v TwinCAT

%identifikacne cisla datovych typov su nutne pri volani MEX funkcii a

%rovnako aj pri nastavovani S funkcii

%pre spravne fungovanie komunikacie (citanie a zapis) premennych je nutne

%zistit si v TwinCAT akeho datoveho typu premenna je

TC\_BOOL\_type=0

TC\_BYTE\_type=1

TC\_WORD\_type=2

TC\_DWORD\_type=3

TC\_INT\_type=4

TC\_DINT\_type=5

TC\_LINT\_type=6

TC\_USINT\_type=7

TC\_UINT\_type=8

TC\_UDINT\_type=9

TC\_ULINT\_type=10

TC\_REAL\_type=11

TC\_LREAL\_type=12

# Matlab MEX funkcie

MEX funkcie sú funkcie, ktoré je možné volať štandardným rozhraním Matlabu, teda priamo v príkazovom riadku. Rovnako je možné ich používať v skriptoch a Matlabovských funkciách. Tieto funkcie sú však kompilovaným kódom a nie je možné do nich pri debugovaní vstupovať a ani ich vnútornú činnosť modifikovať.

Pre účely tohoto toolboxu sú implementované nasledovné:

ADS\_open\_mex();

Funkcia slúži na otvorenie ADS komunikačného portu. Nevstupujú do nej žiadne argumenty a žiadny výsledok nevracia. Túto funkciu je nutné zavolať na začiatku programu.

ADS\_close\_mex();

Funkcia slúži na zatvorenie ADS komunikačného portu. Nevstupujú do nej žiadne argumenty a žiadny výsledok nevracia. Túto funkciu je nutné zavolať na konci programu/skriptu

ADS\_write\_mex([net\_id],’nazov\_premennej’,cislo\_datoveho\_typu ,vstupna\_hodnota);

Funkcia slúži na zápis hodnoty premennej.

Vstupné parameter funkcie:

* net\_id – adresa servera v sieti, 6 miestny riadkový vektor v štandarnnej syntaxi Matlabu napríklad [10, 3, 1, 138, 3, 1]
* názov premennej – reťazec názvu premennej v TwinCAT, štandardnný tvar: ‘GVL.premenna’
* dátovy typ – klasický celočíselný skalár, identifikačné číslo dátového typu, napríklad 4 pre typ INT
* vstupná hodnota – hodnota, ktorú chceme zapisovať, reálnočíselný skalár. V prípade, že premenná je celočíselného typu, dôjde k zaokrúhleniu.

Funkcia nevracia žiadnu hodnotu

y=ADS\_read\_mex([net\_id],’nazov\_premennej’,cislo\_datoveho\_typu);

Funkcia slúži na čítanie hodnoty premennej.

Vstupné parameter funkcie:

* net\_id – adresa servera v sieti, 6 miestny riadkový vektor v štandarnnej syntaxi Matlabu napríklad [10, 3, 1, 138, 3, 1]
* názov premennej – reťazec názvu premennej v TwinCAT, štandardnný tvar: ‘GVL.premenna’
* dátovy typ – klasický celočíselný skalár, identifikačné číslo dátového typu, napríklad 4 pre INT

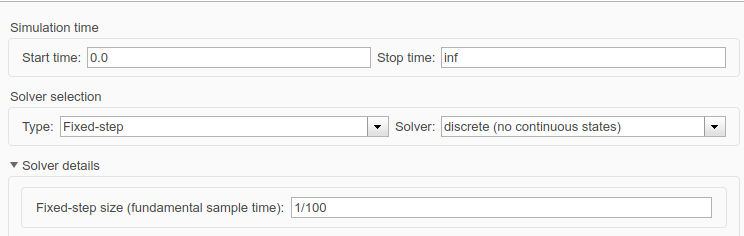
Funkcia vracia reálnočíselnú hodnotu premennej.

# Simulink S-funkcie

S funkcie sú rozšírením štandardných knižníc Simulinku o vlastný kompilovaný kód so špecifickou funkcionalitou. S funkcie teda umožňujú používať štandardné rozhranie Simulinku (vstupy, výstupy) vo vlastnom kóde , napríkald v tomto prípade v  jazyku C++. S-funkcie sú programované úplne nezávisle od MEX funkcii.

Pre správnu funkčnosť ADS toolboxu je nutné v simulačnej schéme spraviť viaceré úpravy.

Volanie užitočného kódu – komunikácie so serverom sa uskutočňuje v diskrétnych krokoch. Preto je nutné nastaviť solver simulácie na diskrétny (bez spojitých stavov) s konštantnou periódou.



Aby simulácia mohla bežať v reálnom čase je nutné do schémy vložiť jeden blok Real-Time sync z knižnice toolboxu Simulink desktop real-time



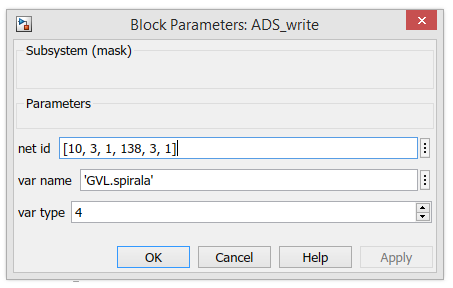
## ADS setup

Tento blok sa stará o automatické otvorenie komunikačného portu pri štarte simulácie a jeho uzatvorenie na konci. S-funkcia nemá žiadne vstupy, výstupy ani parametre a je nutné mať iba jeden takýto blok v schéme.



## ADS write

Blok na zápis do premennej. Vstupom do funkcie je signál s hodnotou, ktorú chceme zapísať. Týchto blokov môže byť v schéme ľubovoľný počet. Parametre S – funkcie sú reprezentované maskou a sú v princípe rovnaké ako tomu bolo v prípade ekvivalentnej MEX funkcie.





## ADS read

Blok na čítanie premennej. Platí to isté ako v prípade bloku ADS write.



# Demo skript a schéma

Pre účely demonštrovania funkcionality som napísal demo skript využívajúci všetky spomínané MEX- funkcie.

ADS\_open\_mex();

ventilator\_name='GVL.ventilator';

spirala\_name='GVL.spirala';

snimac\_name='GVL.snimac';

for i=1:N

t(i)=i\*Tvz;

u(i)=30000\*spirala\_in;

ADS\_write\_mex([10, 3, 1, 138, 3, 1],ventilator\_name,TC\_INT\_type,ventilator\_in);

ADS\_write\_mex([10, 3, 1, 138, 3, 1],spirala\_name,TC\_INT\_type,u(i));

y(i)=ADS\_read\_mex([10, 3, 1, 138, 3, 1],snimac\_name,TC\_INT\_type);

pause(Tvz);

end

ADS\_write\_mex([10, 3, 1, 138, 3, 1],ventilator\_name,TC\_INT\_type,0);

ADS\_write\_mex([10, 3, 1, 138, 3, 1],spirala\_name,TC\_INT\_type,0);

ADS\_close\_mex();

Podobne aj v prípade Simulinku a S-funkcii som vytvoril testovaciu schému obsahujúce spomínané bloky, ich využitie a nastavenie.

