VYUŽITÍ SÉRIOVÉ LINKY POD MATLABEM VERZE 6

František Dušek, Daniel Honc
Katedra řízení procesů a výpočetní techniky, FCHT, Univerzita Pardubice

Abstract

Jednou z nových možností MATLABu od verze 6 je podpora sériové linky. Je možné přijímat i vysílat data v různém datovém formátu prostřednictvím sériové linky RS232 a synchronizovat vykonávání příkazů MATLABu s příjmem dat. V příspěvku je na příkladě řízení elektrické soustavy v reálném čase ukázáno použití těchto standardních příkazů MATLABu a synchronizace pomocí semaforu. Dále byl zjišťován vliv konfigurace počítače na dosažitelnou rychlost vzorkování.

Úvod

Existuje značné množství různých zařízení podporujících komunikaci prostřednictvím sériové linky. S těmito přístroji lze nyní komunikovat na úrovni základních příkazů MATLABu. Ukažme jednoduchý příklad na řízení soustavy III. řádu (elektrického modelu) v uzavřené smyčce v reálném čase prostřednictvím externího A/Č a Č/A převodníku připojeného na standardní sériovou linku RS232. Převodník CTRL51 se vyráběl na UTIA ČAV Praha v sérii několika set kusů. Na tomto příkladě je ukázána práce s objektem typu serial reprezentujícím sériovou linkou, spouštění funkce MATLABu od události, použití časovače i synchronizace běhu programu se sběrem dat z paralelně běžícího reálného zařízení pomocí semaforu.

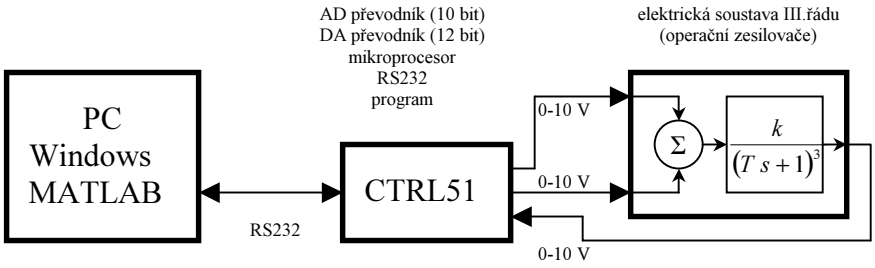
Soustava byla řízena diskrétním LQ regulátorem s pozorovatelem stavu a průběžně identifikovaným modelem řízené soustavy ve formě diferenční rovnice III. řádu. Bližší informace o použitých algoritmech jsou v článku [Honc 2002], který je též uveden ve sborníku.

Výsledné řízení bylo zkoušeno na třech různých konfiguracích počítačů

- MATLAB verze 6.1, Windows 98, Pentium 150 MHz, 128 MB EDO RAM
- MATLAB verze 6.1, Windows 98, Athlon XP 1600+ , 256 MB DDR (PC2700) RAM
- MATLAB verze 6.1, Windows 2000, Athlon XP 1900+ , 512 MB DDR (PC2700) RAM.

Řízená soustava a spojení s PC

Uspořádání zařízení je na obrázku 1. Ovládanou soustavou je elektrická soustava (operační zesilovače) s dvěma vstupy a jedním výstupem. Tato soustava je třetího řádu s ustálením cca 10 sec a zesílením cca 1. Pro připojení soustavy jsou použity dva výstupy (kanál 1 a 2) a jeden vstup (kanál 1) jednotky CTRL51. Tato jednotka obsahuje 10 bitový analogočíslicový (ADC Analog to Digital Convertor) s multiplexorem (12 kanálů) a čtyři 12 bitové číslico-analogové (DAC Digital to Analog Convertor) převodníky, oba s vstupním (výstupním) rozsahem 0-10 V. Činnost jednotky je řízena mikroprocesorem Philips 80C552, který kromě obsluhy AD převodníku s multiplexorem a DA převodníku také zajišťuje obsluhu komunikace po sériové lince RS232 tj. příjem příkazů a odeslání výsledků. Přenosová rychlost je 9600 Bd.



obrázek 1 Připojení soustavy k PC

Komunikace s CTRL v MATLABu

Komunikace s CTRL51 je řízena z PC. Komunikace se zahájí vysláním řídicí byte obsahující kromě typu požadavku i číslo kanálu. Při požadavku na nastavení výstupu následují další 2 byte hodnoty, která se má nastavit. Při požadavku na čtení hodnoty CTRL51 odpoví dvěma byte měřené hodnoty[1]. Bližší informace o komunikaci s CTRL51 viz [Dušek 93]. V MATLABu pak pro spolupráci s jednotkou CTRL stačí čtyři pomocné uživatelské funkce využívající příkazy pro práci se sériovou linkou.

Základní uživatelskou funkcí je *o_CTRL*, která vytvoří objekt typu serial (**serial**), nastaví požadované parametry, napojí objekt na zvolený port (**fopen**), nastaví signály na rozhranní RS232 (zápis do vlastností otevřeného objektu), vyčte případné byte ve vstupním bufru (**fread**) a vrátí vytvořený objekt navázaný na zvolený port. Ve vlastnostech objektu je nastaveno používání časovače s periodickým spouštěním lokální uživatelské funkce *CtrlTimer*. Tato funkce při svém provedení pouze povýší číslo v položce .UserData určené pro volné použití. Tato položka bude sloužit pro synchronizaci hlavního programu. Zda se napojení na vybraný port "povedlo" je možné zjistit ve stavu položky s.Status. V případě úspěchu má hodnotu *'open'* v případě neúspěchu *'closed'*. Neúspěch je možný v podstatě pouze v případě zadání neexistujícího portu nebo v případě, že port je používán jiným programem.

```
function s=o_ctrl(port,period)
% zahájení práce s CTRL
%
%   s=o_ctrl(port,period)
%
% s      ... vytvořený objekt typu serial
% port   ... port pro komunikaci např. 'COM1'
% period .. perioda časovače

% vytvoření objektu
s=serial(port,'BaudRate',9600,'Parity','none',...
    'DataBits',8,'StopBits',1);
s.DataTerminalReady='off';  % DTR do Low
s.InputBufferSize=512;      % velikost vstupního bufru
s.OutputBufferSize=512;     % velikost výstupního bufru
s.FlowControl='none';       % bez řízení komunikace
s.TimeOut=5;                % max.čekání 5 s
s.TimerPeriod=period;       % perioda časovače
s.TimerFcn=@CtrlTimer;      % funkce spouštěná časovačem
s.UserData=0;               % příznak synchronizace

fopen(s);       % propojení objektu s fyzickým zařízením

pause(0.1)
% HW inicializace CTRL
s.DataTerminalReady='on';   % signal DSR do High
pause(0.5)                  % počkej na CTRL
x=s.BytesAvailable;         % počet byte ve vstup.bufru
if x>0,fread(s,x); end      % vše vyčti

% lokální funkce (subfuction) -dostupná pouze z o_ctrl
function CtrlTimer(obj,event)
% callback od Timeru
obj.UserData=obj.UserData+1;  % povyš hodnotu semaforu
```

Druhou uživatelskou funkcí je *c_CTRL* – funkce doplňující funkci *o_CTRL*. Funkce *c_CTRL* zajistí korektní ukončení práce se sériovou linkou tj. ukončí vazbu na hardware (**fclose**) a uvolní používaný objekt typu serial (**delete**). Je potřeba ji použít vždy po ukončení práce se sériovou linkou nejen proto, aby bylo možné znova použít funkci *o_CTRL* např. s jinou periodou, ale hlavně proto, aby bylo možné používat příslušný port ve Windows i v jiných programech po ukončení práce s MATLABem.

```
function c_ctrl(s)
% ukončení práce s objektem serial

fclose(s)           % odpojení od fyzického zařízení
delete(s)           % zrušení objektu
```

Zbývající funkce zajistí vlastní komunikaci s CTRL. Uživatelská funkce *w_CTRL* vyšle tři byte pomocí dvou volání **fwrite**. Požadavek na zápis hodnoty na zvolený kanál *k* je v prvním byte. To, že se hodnota výrazu MATLABu (standardně 8 byte v pohyblivé řádové čárce) konvertuje do čísla v

[1] ačkoliv jde o 10 bitový převodník měřená hodnota je v rozsahu 0-4095 tj. odpovídající hodnotám převodníku 12 bitového. Skutečná měřená hodnota je vynásobena 4.

pevné řádové čárce bez znaménka o délce 1 byte zajistí parametr *'uchar'* funkce fwrite. Zbývající dva byte představující výstupní hodnotu napětí jsou vyslány najednou. Příslušnou konverzi výsledku (přepočet hodnoty 0-10 tj. požadované výstupní napětí ve Voltech na hodnotu 0-4095 tj. 12 bitový DAC s rozsahem 0-10 V) zajistí parametr *'uint16'* druhého volání funkce fwrite.

```
function w_ctrl(s,k,x)
% zápis hodnoty x na kanál k CTRL
%
%   w_ctrl(s,k,x)
%
% s ... objekt typu serial
% k ... číslo výstupního kanálu CTRL (1-4)
% x ... hodnota (0-10)

% pošli 1 unsigned byte (řídicí byte)
fwrite(s,64+k,'uchar');
% pošli 1 unsigned short int (hodnota)
fwrite(s,x/10*4095,'uint16');
```

Funkce *r_CTRL* je trochu složitější. Nejprve je opět vyslán řídicí byte s požadavkem na čtení a číslem kanálu pomocí funkce **fwrite**. Na základě tohoto byte jednotka CTRL51 změří napětí na požadovaném kanále a vrátí hodnotu 0-4095 na dvou byte. Na tuto odpověď se čeká ve funkci **fread**. Formát čísla, na které se čeká tj. jedno 16 bitové číslo v pevné řádové čárce bez znaménka, je určen parametry funkce fread. Poté následuje přepočet hodnoty ADC na hodnotu 0-10. Správně by nejprve mělo následovat vyhodnocení, zda nedošlo k překročení nastaveného času při čekání na požadovaný počet byte (timeout). V případě timeoutu je proměnná *d* prázdná matice, počet čísel *c* je nulový a v proměnné *m* je text '*The specified amount of data was not returned within the Timeout period*'. V této souvislosti je potřeba si uvědomit i to, že fread čte byte z vyrovnávacího vstupního bufru o volitelné velikosti (položka *.InputBufferSize*). Plnění tohoto bufru je záležitostí Windows respektive ovladače portu. Plnění tedy probíhá po otevření portu (fopen) vždy při příchodu znaku - tedy i když není funkce fread používána. Kdyby před použitím funkce *r_CTRL* byly nějaké byte v bufru, pak vyčtená hodnota nebude odpovídat vyslanému požadavku.

```
function d=r_ctrl(s,k)
% vyčtení kanálu k CTRL
%
%   d=r_ctrl(s,k)
%
% s ... objekt typu serial
% k ... číslo vstupního kanálu CTRL (1-12)
% d ... měřená hodnota

% pošli 1 unsigned byte
fwrite(s,128+k,'uchar');
% vyčti 1 unsigned short int
[d,c,m]=fread(s,1,'uint16');
% převod (12 bit ADC) na V (0-10)
d=d*10/4095;
```

Synchronizace programu

Použití těchto funkcí včetně synchronizace běhu programu na reálný čas využívající časovač objektu serial ukažme na příkladě řízení elektrické soustavy na požadovaný průběh žádané hodnoty. Řešení ve formě skriptu je ukázáno v posledním rámečku. Příkazy jsou v textu komentovány a tak vysvětlení snad vyžaduje pouze synchronizace. Je použit mechanismus nazývaný "synchronizace pomocí semaforu[2]". Ve funkci *o_CTRL* je nastaveno použití časovače. To znamená, že od okamžiku spojení objektu s hardware (fopen) je, nezávisle na tom co dál v MATLABu děláme, periodicky[3] spouštěna zadaná lokální funkce. Tato funkce povyšuje počítadlo průchodů (s.UserData) – "semafor" s numerickou hodnotou. V hlavním programu testujeme hodnotu semaforu – nulová hodnota signalizuje, že interval ještě nevypršel a je třeba počkat[4]. Nenulová hodnota signalizuje, že interval již vypršel – je nutné zde hodnotu semaforu snížit. Pak provedeme potřebné příkazy a vrátíme se na test semaforu. Je-li hodnota semaforu nenulová uplynula již další perioda (provedení příkazů trvalo déle) a příkazy se provedou ihned znovu. Je-li hodnota semaforu nenulová trvale – trvá provádění

---

[2] semafor může být buď logický nebo numerický. V příkladu je použit numerický semafor, protože dovoluje rozpoznat situaci, kdy čas vykonávání jedné sady příkazů překročí nastavenou periodu opakování. Také automaticky zajistí dodržení celkového času v situaci, kdy k překročení periody dojde jen občas, ale v průměru je doba sady příkazů menší než perioda časovače.

[3] ve Windows je slovo periodicky potřeba brát s rezervou. Dodržení periody je přibližné – pro delší periody je v průměru dodrženo. Menší periody (v uvedeném příkladu na Pentiu 150 MHz a Win98 je minimální v průměru dodržená perioda cca 0.4 sec) se nedodrží.

[4] např. pomocí konstrukce `while s.UserData == 0, end;`

potřebných příkazů déle než je nastavená perioda. Konkrétně v našem případě, když bude hodnota semaforu po skončení cyklu větší než 1 znamená to, že byla nastavena příliš krátká perioda.

```
% regulace elektrické soustavy pomocí CTRL
% LQG regulátor, Kalmánův filtr, průběžná identifikace

% volitelné parametry experimentu
N =180;                 % počet měření
T =0.5;                 % interval vzorkování
fi=0.99;                % faktor exponenciálního zapomínání
h =15;                  % horizont řízení
om=0.01;                % penalizace akční hodnoty LQ regulátoru
u0=5;                   % počáteční hodnota akční
% *volba modelu řízené soustavy Ay=Bu+Ce (řád + počáteční odhady parametrů)
B=[0,1,0,0]; A=[1,0,0,0]; C=1;
% *průběh žádané hodnoty
% **obdélníkový průběh Np period na délku experimentu
Np=4; yL=2; yH=8;       % počet period, dolní a horní úroveň

% příprava na experiment
s=o_ctrl('COM2',T);     % vytvoření a otevření linky
u0=5;                   % vstupní napětí soustavy
w_ctrl(s,1,u0);         % kanál 1 nastav na u0 V
w_ctrl(s,2,0);          % kanál 2 nastav na 0 V
hy=zeros(N,1);          % vektor výstupů
hu=zeros(N,1);          % vektor vstupů
ht=zeros(N,1);          % vektor času měření
disp('... čekám na ustálení')
pause(5)                % počkej 5 sec na ustálení
y0=r_ctrl(s,1);         % výstupní napětí soustavy

% inicializace dat pro průběžnou identifikaci (diferenční rovnice)
IP=[];
[B,A,C,IP,Idat]=RMNC(B,A,C,IP,[],y0,u0,1000);

% inicializace dat pro pozorovatele stavu (stavový model)
na=length(A);
AA=eye(na-2); AA=[AA;zeros(1,na-2)]; AA=[-A(2:na)',AA];
BB=B(2:na)';  CC=[1,zeros(1,na-2)];
XX=(eye(na-1)-AA)\BB*u0;
KP=0;
% inicializace dat pro LGQ regulátor
% *vytvoření průběhu žádané
nk=fix(N/2/Np+1);       % počet vzorků na 1/2 periody žádané
hw=[];
for k=1:2*Np,
    if mod(k,2)==0, t=ones(1,nk)*yH;
    else            t=ones(1,nk)*yL;
    end
    hw=[hw,t];
end
% *rozšíření poslední hodnoty na horizont řízení
hw=[hw,ones(1,h)*hw(end)];

% synchronizace na následující událost časovače
s.UserData=0;           % zrušení příznaku časovače
while s.UserData==0, end;   % čekej na příznak
s.UserData=0;           % zrušení příznaku časovače

% začátek experimentu
disp('... start experimentu')
tic                     % začátek kontrolního čas.intervalu
ht(1)=toc;              % čas prvního měření
hy(1)=r_ctrl(s,1);      % první měřený výstup soustavy
hu(1)=u0;               % ustálený vstup soustavy
u=u0;
```

... pokračování na další straně

```
      ... pokračování z předchozí strany
for k=2:N,
  % synchronizace
    while s.UserData==0, end;    % čekej na příznak
    s.UserData=s.UserData-1;     % snížení příznaku časovače
    ht(k)=toc;                   % skutečný čas měření (od startu)
    y=r_ctrl(s,1);               % aktuální hodnota výstupu soustavy
  % identifikace
    [B,A,C,IP,Idat]=RMNC(B,A,C,IP,Idat,y,u,fi);
    AA(1:end,1)=-A(2:end)'; BB=B(2:na)';
  % regulace
    [K,l]=synLQa(AA,BB,CC,0,hw(k:k+h-1),om);
    uu=K*XX+l;                   % vypočtený akční zásah
    u=uu;                        % omezení
    if uu<0, u=0; end
    if uu>10, u=10; end
    w_ctrl(s,1,u);               % realizace akčního zásahu
  % estimace (odhad budoucího stavu soustavy)
    [XX,KP]=eKalm(AA,BB,CC,y,u,XX,KP);
    hy(k)=y; hu(k)=u;    % záznam průběhu experimentu
end
s.UserData               % stav semaforu po ukončení experimentu
c_ctrl(s);               % ukonči práci se sériovou linkou
disp('... konec experimentu')
```

Závěr

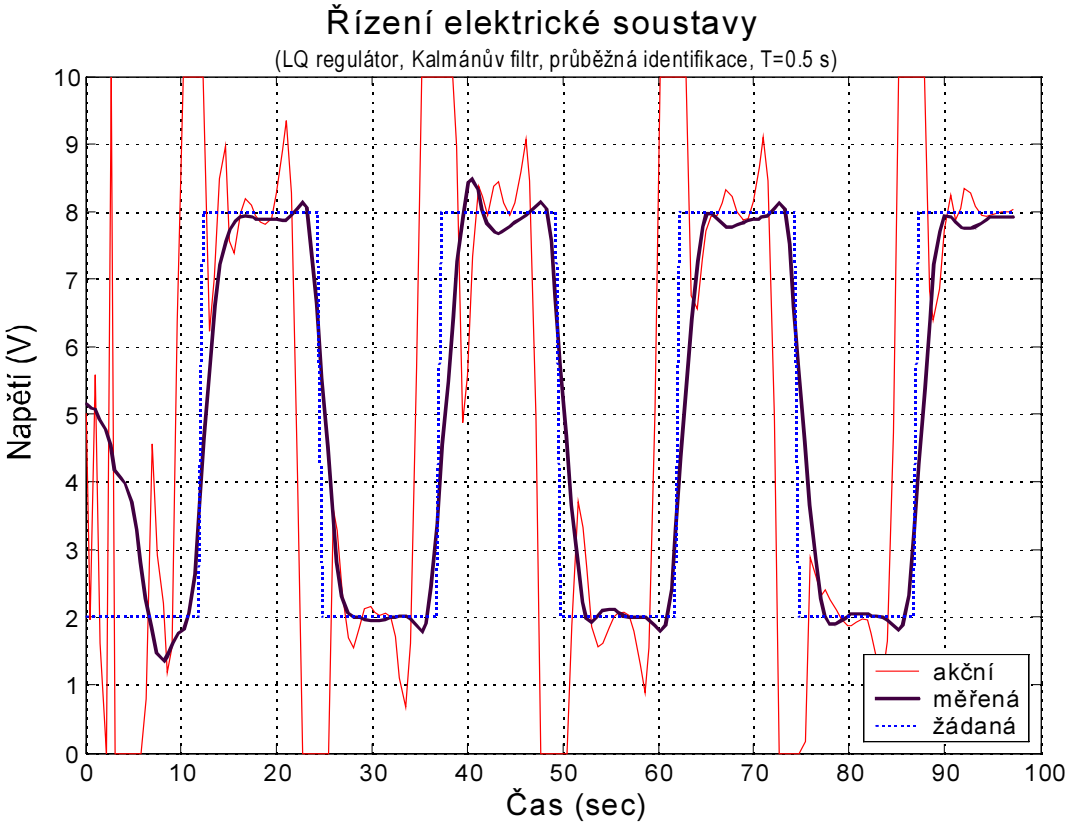
V souvislosti s používáním příkazů pro práci se sériovou linkou je potřeba upozornit na následující problém. V případě, že adresář s funkcemi používajícími příkazy pro práci se sériovou linkou je na jiném disku než je nainstalován MATLAB, příkazy pro práci se sériovou linkou nefungují. Tento problém lze jednoduše odstranit buď tak, že aktuální (pracovní) adresář bude na stejném disku co je MATLAB, nebo tak, že jednou vytvoříme, otevřeme a uzavřeme objekt serial v situaci, kdy je aktuální (pracovní) adresář na disku s instalací MATLABu a teprve pak změníme aktuální adresář.

Ukázka průběhu regulačního experimentu pro interval vzorkování 0.5 sec je obrázku 2. Popis použitých algoritmů ani způsob implementace není předmětem tohoto článku. Proto i obrázek je uveden bez dalšího komentáře pouze pro ilustraci dosaženého průběhu experimentu v reálném čase.

Pro představu o časových záležitostech je ukázána odchylka skutečné od teoretické doby provádění výše uvedeného skriptu pro zkracující se interval vzorkování na různých konfiguracích. Skript pro daný interval vzorkování byl proveden vždy dvakrát. Získané hodnoty jsou shrnuty v tabulce 1. Uvedena je hodnota semaforu po skončení skriptu a délka provádění vyjádřená jako procentuální odchylka od teoretické délky. Pro sledované časy není omezení na straně sériové komunikace ani jednotky CTRL51 – pod OS DOS lze bez problémů dosáhnout 100 čtení a zápisů za sekundu tj. interval 0.01 s. Samozřejmě limit je dán dobou výpočtů MATLABu v rámci jednoho intervalu, ale ta byla pro P150 cca 60 msec. Proto bylo skript proveden i bez výpočtů ve smyčce (posílala se konstntní hodnota) a dosažené hodnoty byly stejné. Z výsledků plyne, že se nedodrží požadovaná perioda ani při malém zatížení a že nelze dosáhnout kratších period. Tyto problémy neřeší ani použití Windows 2000.

Tabulka 1 - Srovnání provádění skriptu pro zkracující se interval vzorkování

Nastavená perioda opakování (interval vzrokování)	P150 MHz/ 128 MB / W98 doba výpočtu 0.055425 s				XP1600 / 256 MB / W98 doba výpočtu 0.001395				XP1900 / 512 MB / W2000 doba výpočtu 0.001172			
	Semafor		Odchylka %		Semafor		Odchylka %		Semafor		Odchylka %	
	I.	II.	I.	II.	I.	II.	I.	II.	I.	II.	I.	II.
0.7	0	0	2.7	2.7								
0.6	0	0	1.9	2.5								
0.5	0	0	8.1	7.8	0	0	7.6	7.9	0	0	-0.5	-0.5
0.4	1	1	10.3	11.2	0	0	9.2	9.5	0	0	1.6	1.2
0.3	53	64	52.1	54.0	1	1	9.4	9.3	0	1	8.8	9.3
0.2	209	208	152.0	153.8	84	84	59.8	60.2	4	5	3.6	3.7
0.1					353	354	223.1	224.4	159	156	104.1	102.3



obrázek 2 Průběh regulačního experimentu

Literatura

Dušek, F.; Klán, P.: *Laboratorní jednotka styku s prostředím pro počítače PC.* Automatizace (36)10, 1993, s.294-296
Dušek, F.: *MATLAB a SIMULINK úvod do používání.* Druhé rozšířené vydání. [skriptum] Pardubice 2002, ISBN 80-7194-475-0
Dušek, F.: *Adaptivní řízení.* [učební text], Pardubice 1999
Honc, D.; Dušek, F.: *Maticové operace v SIMULINKu verze 4.* [v tomto sborníku], 2002
MATLAB Help: *External Interfaces/API Reference, Seriál Port I/O Function.* 2001

doc. Ing. František Dušek, CSc. e-mail: frantisek.dusek@upce.cz tel.: 040-6037125
Ing. Daniel Honc, Ph.D. e-mail: daniel.honc@upce.cz tel.: 040-6037107

Katedra řízení procesů a výpočetní techniky
Fakulta chemickotechnologická
Univerzita Pardubice
nám. Čs. legií 565
53210 Pardubice