PERFORMANCE ANALYSIS OF SERIAL PORT INTERFACE IN MATLAB

Petr Blaha , Pavel Václavek

Centre for Applied Cybernetics, Brno University of Technology, 612 66 Brno, Božetěchova 2, Czech Republic,
Tel.: +420 541 141 160, Fax: +420 541 141 123, E-mail: blahap@feec.vutbr.cz, vaclavek@feec.vutbr.cz

Abstract: This article is interested in serial communication in the environment of Matlab. It compares Matlab's serial port interface with javax.comm package based realization of serial communication and the solution programmed as a C MEX-File. The stress is laid on the comparison of time requirements on data exchange. The testing is provided on two different baud rates and with two different data packet sizes.

Keywords: serial port, Java, C MEX-File

1. Introduction

The serial communication serves as a very simple tool for mutual data exchange between different devices, e.g. computers, scientific devices and peripherals. The serial port interface is implemented directly in Matlab. It enables to configure serial ports, to use control pins, to write and read data, to use events and actions and to record information to disk. It is supported identically on many platforms due to Java language based implementation. For more information about the Matlab's serial port interface see [1].

This article is motivated by bad experiences with Matlab's serial port interface. We needed to communicate with Motorola DSP56F805 evaluation module [2]. The data packets sent and obtained back from EVM were both short (about 6 bytes). The communication was required to run as fast as possible. We have tried to increase the baud rate in order to increase the communication speed but it did not lead to expected results. It forced us to write our own C MEX-File realizing the serial communication in C language using Windows API functions. The intention of this article is to compare these two realizations. The Java language solution using javax.comm is compared too.

The article is organized as follows. The second chapter contains the description of realized tests. The third chapter summarizes obtained results. The fourth chapter enumerates used hardware and software and the last chapter concludes the results.

2. Description of realizations

At first we have measured the time spent on sending and receiving one character using two different baud rates. Motorola DSP56F805 evaluation module was employed as a second communication device. It was working only as a responder since it immediately returned received character back to PC.

It is impossible to measure the time interval needed for exchange of one character using supported Matlab functions. Due to this we have measured the time interval consumed by the exchange of hundred characters. The latency caused by for cycle is negligible. This interval is already measurable with the Matlab functions tic and toc. Combination of these commands gives number of seconds required for the operations enclosed with them.

Secondly we have measured the time spend on sending and receiving buffer of characters. The buffer of hundred characters was sent out. When the DSP56F805 EVM obtained the last character of the buffer it immediately started to send the same buffer back to Matlab. The buffer was sent ten times to increase the precision of time interval measurement. The testing was realized again using two different baud rates.

The following paragraphs describe the different realizations in details.

2.1. Matlab's Serial Port Interface

This realization was implemented easily since it is in Matlab manner according to [1].

```
s = serial('com1','BaudRate',9600);
fopen(s);
data_out = 1;
tic;
for I = 1:100,
        fwrite(s,data_out,'schar');
        data_in = fread(s,1,'schar');
end;
t = toc;
fclose(s);
clear s;
```

Code 1: Matlab's SPI (100x one byte)

```
s = serial('com1','BaudRate',9600);
fopen(s);
data_out = 1:100;
tic;
for I = 1:10,
        fwrite(s,data_out,'schar');
        data_in = fread(s,100,'schar');
end;
t = toc;
fclose(s);
clear s;
```

Code 2: Matlab's SPI (10x hundred bytes)

2.2. C MEX-File mexserial

This C MEX-File opens, initializes, sends data, receives the same amount of data and closes the serial port each time it is called. It is implemented using API functions of Windows (used functions are CreateFile, GetCommState, SetCommState, SetupComm, WriteFile, ReadFile, CloseHandle).

```
data_out = 1;
tic;
for i=1:100,
        data_in = mexserial(data_out);
end;
t=toc
```

Code 3: mexserial (100x one byte)

```
data_out = 1:100;
tic;
for i=1:10,
        data_in = mexserial(data_out);
end;
t=toc
```

Code 4: mexserial (10x hundred bytes)

2.3. C MEX-File mexserialpersistent.

Its functionality changes according to the first input parameter. The number 0 means to open and to initialize serial port, 1 means to send data, 2 means to receive data and 3 means to close serial port. The serial port stays opened during the data transfers. Its handle is stored in persistent variable.

```
mexserialpersistent(0,0);
data_out = 1;
tic;
for i=1:100,
        mexserialpersistent(1,data_out);
        data_in  = mexserialpersistent(2,1);
end;
t=toc;
mexserialpersistent(3,0);
```

Code 5: mexserialpersistent (100x one byte)

```
mexserialpersistent(0,0);
data_out = 1:100;
tic;
for i=1:10,
        mexserialpersistent(1,data_out);
        data_in  = mexserialpersistent(2,100);
end;
t=toc;
mexserialpersistent(3,0);
```

Code 6: mexserialpersistent (10x hundred bytes)

2.4. javax.comm library

New versions of Matlab support operations with Java classes and objects in the environment of Matlab. The way of operating with Java language is detailed in [1]. In fact, the Serial Port Interface is based on this library.

```
commPort = javax.comm.CommPortIdentifier.getPortIdentifier('COM1');
serialPort = open(commPort,'serial',1000);
setSerialPortParams(serialPort, 9600, 8, 1, 0);
out = java.io.OutputStreamWriter(getOutputStream(serialPort));
in = getInputStream(serialPort);
data_out = 'a';
tic;
for i = 1:100
        out.write(data_out);
        flush(out);
        numAvail = available(in);
        while numAvail < 1
          numAvail = available(in);
        end
        data_in = read(in);
end
t=toc;
close(in);
close(out);
close(serialPort);
```

```
b = char(1:100);
data_out = java.lang.String(b);
tic;
for i = 1:10
        out.write(data_out,0,100);
        flush(out);
        obtained = 0;
        numAvail = available(in);
        while obtained  <  100
                numAvail = available(in);
                if(numAvail >0)
                        data_in = read(in);
                        obtained = obtained + 1;
                end
        end
end
```

Code 7: javax.comm library (100x one byte) (10x hundred bytes)

3. Results summary

All the measurements are summarized in Table 1. As one can see, the fastest data transfer, i.e. the lowest communication delay is accomplished with the C MEX-File mexserialpersistent. Little bit more time consuming was realization using javax.comm library. Even more time consuming was realization using C MEX-File mexserial where the serial port was opened and closed repeatedly for each data transfer. The realization using Matlab's SPI under W2000 as well as under Linux was very slow. It can be seen in Table 1 that the increase of baud rate in this case almost did not bring together the increase of the data transfer speed (when transferring 100 times one character). Note also that the mexserialpersistent realization is more that 150 times faster than realization using Matlab's serial port interface (baud rate - 115200, style of the transfer – 100times one character).

First we thought that the problem of big latency in communication lies in the implementation of Matlab's serial port interface using java language. The results above show that this is not the truth and that the Matlab is the one who causes the latency. This is high tax for user-friendly manipulation with the serial port under Matlab.

Baud rate	Style of the transfer	SPI under W2000	SPI under Linux	C MEX - File mexserial under W2000	javax.comm library under W2000	C MEX - File mexserial-persistent under W2000	Theoretical time needed only for the data transter
9600	100 times 1 character	10.22	12.11	2.02	0.75	0.65	0.23
9600	10 times 100 characters	3.09	3.34	2.41	2.46	2.22	2.29
115200	100 times 1 character	10.08	12.01	1.01	0.18	0.07	0.019
115200	10 times 100 characters	1.312	1.30	0.31	0.43	0.23	0.19

Table 1: Summary of the results

4. Hardware and software description

The following hardware was used to realize the tests:

- standard PC with Duron 650 processor
- Motorola DSP56F805 evaluation module

The PC was equipped with following software:

- Windows 2000
- Matlab v. 6.1 for Windows (the results obtained using new version v. 6.5 were the same)
- Linux RedHat 7.3
- Matlab v. 6.1 for Linux
- Metrowerks CodeWarrior for DSP56800 v. 5.0

5. Conclusion

The serial communication using Matlab's serial port interface is very ineffective. It is apparent from result obtained in this paper. Its usage is not suitable for exchange of large amount of data divided into small packets and in the cases where the time spent in communication is critical. The C MEX-File realization is needed for such a cases.

This article could be useful for those who plan to use serial link communication in their project, especially for those who need to communicate through the serial link as fast as possible.

6. References

[1] MATLAB External Interfaces. The MathWorks Inc., Version 6 , November 2000.

[2] DSP56F80x User's Manual. The Motorola Inc., Rev. 3.0. 2001.

Acknowledgement

This work was supported by the Ministry of Education of the Czech Republic under Project LN00B096.