Scicos Serial-Interface-Block Manual 12.05 (pre-Alpha)

Dipl.-Ing. Klaus Weichinger / snaky.1@gmx.at / bioe.sourceforge.net

May 4, 2012

This document was written in a short time and no review were performed. So this document will have failures. Please give positive and/or negative feedback of your experience with this toolbox.

Abstract

This document describes the installation and usage of the Serial-Interface Scicos *block and a demo with the Arduino-UNO board (see http://www.arduino.cc).*

1 Introduction

In [1] a low cost do-it-self automation system called BIOE was presented. This system uses the parallel printer port to interconnect a PC with a physical plant and to control this plant with a hard real-time system as discussed in [2, 3, 4, 5]).

I was often asked why i'm using the parallel port and why i'm not using the Arduino-UNO board [6] because many people do have this.

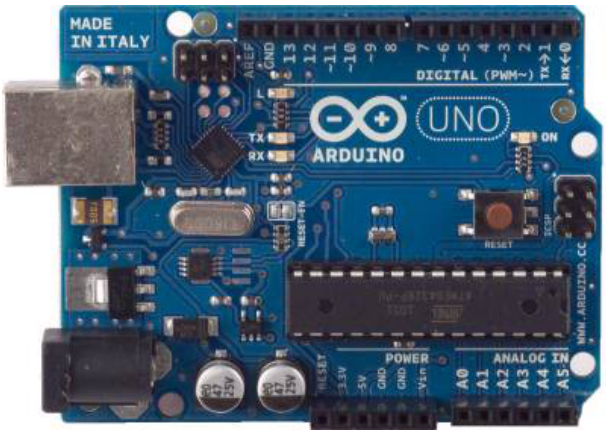

Figure 1: Arduino-UNO

So i decided to develop a ScicosLab block that can exchange signals via the serial port with other devices like the Arduino-UNO. The provided approach uses the real-time scaling capability of a ScicosLab simulation so that soft real-time applications can be realized and run directly within ScicosLab.

At the moment the toolbox supports only ScicosLab 4.4.1 on WIN32.

In the following sections the concept, installation and usage of the Scicos block called *serialinterface-block* are discussed.

2 Concept

The concept uses a single ScicosLab block, that sends a set of signals to the device and receives a set of signals from the device like illustrated in Figure 2.

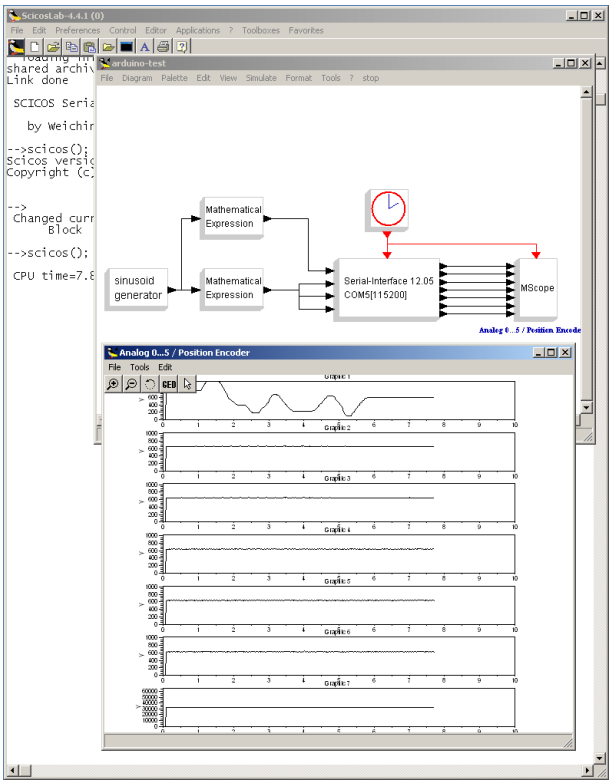

Figure 2: Example using Arduino-UNO and the *serialinterfaceblock*

For the realization the following techniques were used:

- Serial-Port: the data exchange between SciCosLab and the device uses serial ports like the USB virtual comport used at the Arduino-UNO
- Threading: the handling of the serial port is realized within a seperate thread so that the SciCosLab simulation does not influence the serial communication
- Base64: The signals are exchanged in base64 [7] formated binary blocks
- Arduino-UNO: This board is a well known piece of hardware that could be used as simple input/output interface

The different components are discussed in more detail in the following subsections.

2.1 Serial-Port

The real RS232 serial port is more and more obsolete on standard PCs, Notebooks and Netbooks. The USB is normally on each available and there is also the possibility to connect mikrocontrollers with an UART to the PC by the use of USB to SERIAL bridges. This concept is also used on the Arduino-UNO board. On both sides (PC and mikrocontroller) the interface acts like a normal serial port.

But there are some differences: Due to the fact that USB uses a cycle based communication concept there are delays (1ms and more) within the communication. These delays influences the real-time capabilities dramatically compared to a real serial or parallel port.

Nevertheless control systems with cycle times of 20ms ore more can be realized for educational or prototyping usage.

2.2 Threading

ScicosLab will be used as platform to implement a control low. Normally cycle times of 20ms or more can be realized directly on WIN32 systems for soft real-time educational or prototype experiments. To provide a fast data exchange via the serial interface the *serialinterfaceblock* creates a new thread that handles incoming and outgoing data.

For the detailed implementation just have a look to `scicosthreading.h` and `serialinterfaceblock.c` of this toolbox.

2.3 Base64

To reduce the code and CPU effort to create serial frame and to encapsulate or to extract the signal information of different type the memory block of the idea to store the signals in a C structure and to transmit the memory block was used. This memory block is transformed to a base64 [7] because it reduces the transmitted data size compared to a ASCII-HEX transformation. The binary blockes is encapsulated between "<...>". Furthermore a CRC16 checksum is placed at the end of a C structure so that the integrity of a frame can be ensured.

The example frame sent from the Arduino-UNO **<nwKQAokCggJ6AnICAoCnhD==>** contains the binary information of the following C structure

```
struct txFrame
{
  unsigned short Analog0;
  unsigned short Analog1;
  unsigned short Analog2;
  unsigned short Analog3;
  unsigned short Analog4;
  unsigned short Analog5;
  unsigned short Position;
  unsigned char  crc1;
  unsigned char  crc2;
};
```

So at the end 26 bytes (including overhead) are sent to transmit a buffer with 16 bytes. The symbol "<" signalizes that a new base64 will follow. The implementation translates the base64 stream to the binary on demand. If the symbol ">" is received, the CRC16 if the buffer is checked and if it is valid the signal values are passed to the inputs/outputs of the ScicosLab block / Arduino-UNO.

2.4 Arduino-UNO

The Arduino-UNO contains a 8bit AVR mikrocontroller and provides a serial port via USB. It contains 10bit analog digital converter and 8bit PWM signals. The firmware `ArduinoScicos-UniversalIO` contained with the toolbox passes following signals from/to ScicosLab:

- The 10bit value of Analog0...5 are sent to ScicosLab
- Pin2&3 are the inputs for position encoder signals. The position is sent to ScicosLab
- Pin9, Pin10, Pin11 are 8bit PWM signals that can be set by ScicosLab
- Pin5, Pin6, Pin7, Pin8, Pin12 and Pin13 (LED L) can be set by ScicosLab

3 Installation

General follow the installation instructions provided with the `readme.txt` within the toolbox package. If the installation was successful and ScicosLab is started, the start window should show you the version information of the toolbox (see Figure 3).

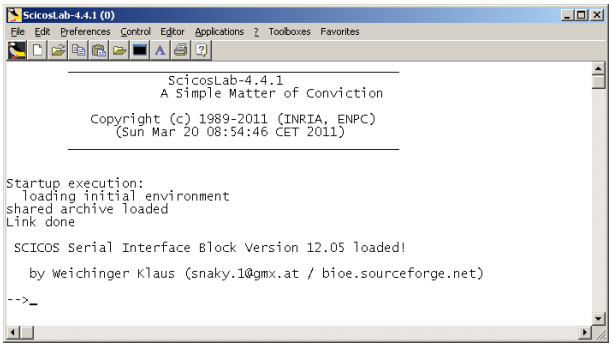

Figure 3: Startscreen of ScicosLab shows the version information of the toolbox

4 Usage of the ScicosLab Block

The ScicosLab block *serialinterfaceblock* can not be found within the palette. To insert the block select **menu-Edit-Ad new block**, type *serialinterfaceblock* (Figure 4), press *OK* ans place the block within the workspace (Figure 5).

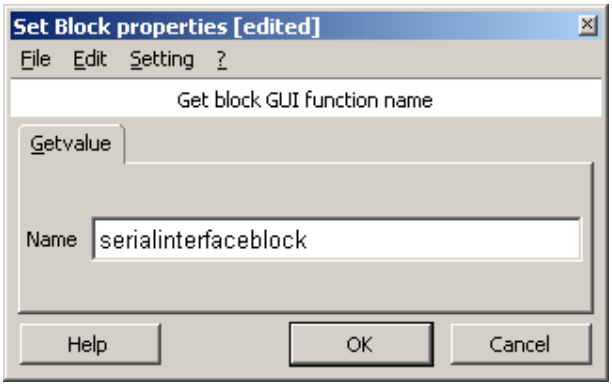

Figure 4: Type the block name

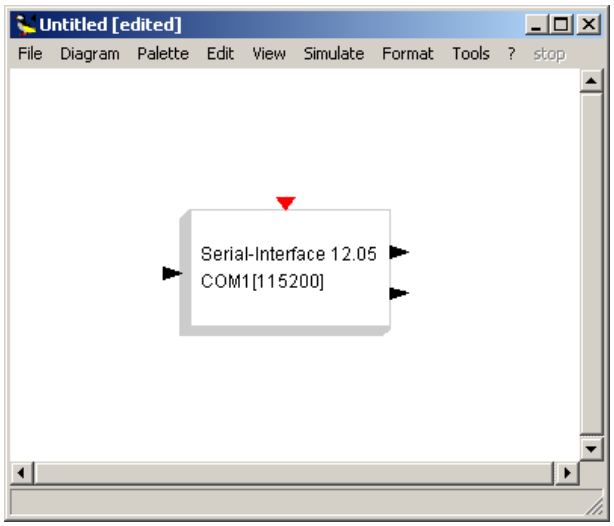

Figure 5: Place the ScicosLab block

Within the block options the serial port name and the data frame format can specified. The block uses the sequence of characters to extract the relating bytes of the c structur memory block and convert it to a signal or the other way round. Detaild information of the supported format is noted within the options dialog them self.

5 Revision History

The version number xx.yy represents the year xx and month yy when the version was released.

5.1 Version 12.05 (pre-Alpha)

The first release of this project to the community for testing and additional feature requests. This version still provides all necessary functionalities so that it can be used for educational purposes.

References

[1] Weichinger, K.: *A Nonlinear Model-Based Control realized with an Open Framework for Educational Purposes*. Proceedings of the 13th Real-Time Linux Workshop in Prague, Open Source Automation Development Lab (OSADL) eG, 2011.

[2] RTAI - Real Time Application Interface Official Website: *https://www.rtai.org*. Web. 20 Aug. 2011.

[3] Real-Time Linux Wiki: *http://rt.wiki.kernel.org*. Web. 20 Aug. 2011.

[4] Bucher, R.: *rt-preempt Code Generation Package for ScicosLab* *http://www.dti.supsi.ch/~bucher*. Web. 20 Aug. 2011.

[5] Basic Input Output Elements Project Website: *http://bioe.sourceforge.net*. Web. 27 Dec. 2011.

[6] Arduino Project Website: *http://www.arduino.cc*. Web. 03 May 2012.

[7] Josefsson, S.: *RFC4648: The Base16, Base32, and Base64 Data Encodings*, October 2006, http://www.ietf.org/rfc/rfc4648.txt, Web. 20 Aug. 2011.

[8] ScicosLab 4.4.1 - Project Website: *http://www.scicoslab.org*. Web. 20 Aug. 2011.

[9] Scilab / Scicos Website: *http://www.scilab.org*. Web. 20 Aug. 2011.