Sensor Control Interface

Operation Manual

Jörg Conradt
Institute of Neuroinformatics
University and ETH Zürich
e-mail: conradt@ini.phys.ethz.ch
http://www.ini.ethz.ch/~conradt/projects/SensorControlInterface

Preliminary

Rev 2a. March 12, 2005

Funded by
The Institute of Neuroinformatics, ETH-Zurich, Switzerland
The Institute of Neuromorphic Engineering, University of Maryland, USA
# Table of Contents

1. Introduction ........................................................................................................................ 4  
2. Features ............................................................................................................................ 5  
3. Hardware Documentation .................................................................................................. 6  
   3.1. Power Supply to the SCI board .................................................................................. 6  
   3.2. Connecting to Host PC .............................................................................................. 7  
   3.3. Description of Analog IO ports, P0 – P3 .................................................................... 7  
   3.4. Description of Digital IO port, P4 ............................................................................. 9  
   3.5. Description of Special Ports ...................................................................................... 10  
   3.6. Description of On-Board Sensors ............................................................................. 12  
   3.7. Description of I/O Port Test Adapters ...................................................................... 12  
   3.8. Description of Generic I/O-PCB Adapter ................................................................. 13  
4. System Software Documentation .................................................................................... 14  
   4.1. Operating Principles .................................................................................................. 14  
   4.2. Command Line Interface on the Main Microcontroller Atmel M128 ......................... 14  
   4.3. Customized Software Routines on the Atmel M128 .................................................. 14  
   4.4. System Software Updates ......................................................................................... 15  
5. Communication with the Board ..................................................................................... 16  
   5.1. Direct Communication .............................................................................................. 16  
   5.2. On-Board Scripts ...................................................................................................... 16  
   5.3. C Routines / Libraries on the Host PC ..................................................................... 16  
   5.4. MatLab Scripts ........................................................................................................ 17  
   5.5. XML / Chip Database ............................................................................................... 17  
   5.6. Tradeoff: Convenience vs. Speed ............................................................................. 17  
6. Final Remarks .................................................................................................................. 18  
Appendix A: PCB Layout .................................................................................................... 19  
Appendix B: Data Sheets and Resources ........................................................................... 20  
Appendix C: C Function Calls provided by the System Software .................................... 21  
Appendix D: Modifications since Revision 0 .................................................................... 22  
Appendix E: Overview of available Command-Line Commands .................................... 23

Photograph on cover page shows the SencorControlInterface board with two side-ways attached neuromorphic motion sensors, controlling a small autonomous robot.

**SENSOR CONTROL INTERFACE, OPERATION MANUAL**

Jörg Conradt, conradt@ini.phys.ethz.ch
List of Figures

Figure 1: Sketch of the SCI board. Left: Top view of board and components. Right: Top view with super-imposed red marking of the “digital section” (DS) .........................................................4
Figure 2: Sketch of PCB, top view (left) and bottom view (right). Refer to appendix A for a detailed description of components on the board .................................................................6
Figure 3: Main power connector for the SCI board ....................................................................................6
Figure 4: Location and pin-out of the communication interface connector (top-view).......................7
Figure 5: Pin-out diagram of the “Analog Ports” (P0-P3), top view (frontal view). The non labeled pin is left empty to secure proper alignment of connectors. ........................................8
Figure 6: Analog Ports P0+P1 and P2+P3: one connector for each pair of ports.................................9
Figure 7: Pin-out diagram of the “Digital Port” (P4), top view (frontal view). The non labeled pin is left empty to secure proper alignment of connectors ..............................................9
Figure 8: Motor Connectors and special power supply .......................................................................11
Figure 9: Servo connectors and special power supply .........................................................................11
Figure 10: TWI connector for system extensions ................................................................................12
Figure 11: Two available generic I/O-PCB adapters that only differ in the arrangement of the DA-pins (left side of images) .................................................................13
1. Introduction

The SensorControlInterface (SCI) board is a multipurpose Input/Output board that provides interfaces to external electronic devices such as sensors and actuators. It either operates as standalone system or connects to a standard Personal Computer.

The SCI board was specifically designed to support exploration of analog-VLSI (aVLSI) chips, providing a large number of adjustable voltages for generating chip biases, several analog input ports and several digital control lines. Most important, the SCI board is separated into a “digital” and an “analog” section (DS, AS), containing completely separated circuits for the digital control logic and the analog interface to the aVLSI chips (figure 1). All “connections” between DS and AS are routed through optic-couplers, guaranteeing two completely separated systems which do not even share their GND level. This complete separation provides an absolutely low noise environment on the aVLSI side.

Additional standard features on the SCI board include motor drivers, servo control outputs, and a magnetic compass. The on-board TWI-interface connector allows chaining multiple identical or different (customized) boards in a final system. Users can modify the on-board software and program upgrades during operation.

The SCI board is compact and lightweight, offering various potential application domains:

- Test setup for aVLSI chips in lab environments
- Remote chip tuning via the internet, e.g. for collaborative projects
- Data collection for aVLSI chips in controlled environments
- Data collection and sensor evaluation in “real world” environments
- Robot / System control

![Figure 1: Sketch of the SCI board. Left: Top view of board and components. Right: Top view with super-imposed red marking of the “digital section” (DS).](image-url)
2. Features

Main processor: Atmel M128-8AI
- 128 KB in-system reprogrammable Flash EEPROM
- 4 KB internal SRAM
- 4 KB internal EEPROM for storing setup information
- 7.372 MHz (7.372 MIPS), 16MHz version available
- 3.3 – 5.5 V operation
- Core program occupies less than 15% of program memory

Secondary processor: Atmel M88-10AI (for generating PWM signals)

Communication: Serial port (921.600 Baud)
- USB (virtual serial port at 921.600 Baud)
- Wireless Bluetooth (virtual serial port at 921.600 Baud)
  - Class 1 (100m range) or Class 2 (10m range) available

Power requirements: Ideally 2 separate power sources (e.g. batteries):
- 3.3 – 5.5 V digital section (average consumption 70 mA)
- 5.5 – 8 V analog section (average consumption 60 mA)
- Alternatively a common source at 5.5 V (not recommended)

Size and Weight: 91 x 74 mm (3.6 x 2.9 inches), 40g (1.4 ounces)

IO functionality:
- 4 x Analog Ports, each:
  - 16 x 16 bit D/A converter, 0 – 3.3/5V range, steps of <=0.1mV
  - 8 x 12 bit A/D converter, 0 – 3.3/5V range, steps of 0.8/1.2mV
  - 8 x digital output lines (0 – 3.3/5 V level)
  - 8 x digital input lines (0 – 3.3/5 V level)
  - 2 x fast digital clock line (0 – 3.3/5V level)
  - Various AGND and AVCC pins for power supply
- 1 x Digital Port
  - 2 x 8 bit general digital I/O port (pin individually selectable)
  - 1 x 5 bit analog converter (directly on the digital Atmel chip)
  - 2 x bi-directional PWM signal (low power, 0 / 5 V logic level)
  - 4 x PWM signal for servo control
  - 1 x serial port (0 / 5 V level)
  - Various GND, VCC, and a D-AVCC supply
- 4 x Bidirectional PWM Motor Drivers (up to 1A each)
- 5 x Servo Control Ports (for standard model servos)
- 1 x Two-Wire-Interface (TWI), I²C compatible
- 1 x on-board magnet field sensor (heading direction)
- 1 x Battery level sensor on board

Non-volatile EEPROM to store settings for power-up initialization
Software updates: http://www.ini.ethz.ch/~conradt/projects/SensorControlInterface
3. Hardware Documentation

3.1. Power Supply to the SCI board

The SCI board (Figure 2) has completely separate power for the digital and analog sections (DS / AS). Therefore, the power connector shown in Figure 3 provides four contacts for:

- Analog VCC input (AVCC, +5.5 – +8 V relative to AGND)
- Analog Ground input (AGND)
- Digital VCC input (DVCC, +3.3 - +5.5 V relative to DGND)
- Digital Ground input (DGND)

Ideally, these power supplies are completely separated, e.g. by two different batteries. In the case where separate power supplies are not available, both VCC signals can get powered from a single 5.5V source. A common power supply, however, is not recommended, because the shared supply introduces significant electric noise in the AS originating from the DS.
The small red power supply connector on the PCB flips the right and left side of the pins. All pins shown on the left side in Figure 3 are actually on the right side of the red connector in top view on the PCB. Therefore, AVCC and DVCC show contacts on the right whereas AGND and DGND show contacts on the left side. When connecting a 4-wire cable, the order within the cable remains unchanged: from top to bottom AVCC, AGND, DVCC, and DGND.

3.2. Connecting to Host PC

The board allows stand-alone operations, e.g. controlling a mobile robot without user interaction. For most applications, however, a connection to a PC facilitates operation significantly. Therefore, the board offers a serial port connector as standard communication interface (CI), shown in Figure 4. The connector provides a full TTL-level serial interface, communicating at 921.600 baud. Three external adapter boards that attach directly to the communication interface provide a variety of further connecting options:

- CI to Standard PC serial port (+/- 12V operation)
- CI to USB converter
- CI to Wireless Bluetooth converter

All of the above options rely on a (possibly virtual) serial port on the PC that will be used for communication. The standard software on the board provides a simple command line language that can be accessed through a terminal program (e.g. Windows’ HyperTerm or Linux’ minicom). Please refer to chapter 4 for a detailed description.

3.3. Description of Analog IO ports, P0 – P3

Before we look in detail at the analog ports please remember that the board provides completely separated power for the analog and digital sections (AS / DS), as described in Chapter 3.1. Therefore, all pins in this chapter are part of the AS, even when they are called DIx or DOx. The notation DI / DO refers to the characteristics of the pin, being able to provide or sense a TTL compatible 0 – 5 V level rather than analog voltages. Nevertheless, these pins are part of the AS (and not of the DS), and thus they have the same low-noise characteristics as the other signals on the analog connectors.
Figure 5: Pin-out diagram of the “Analog Ports” (P0–P3), top view (frontal view).

The non labeled pin is left empty to secure proper alignment of connectors.

The pin-out of analog ports P0 – P3 is identical (Figure 5); therefore we will look at a single port only. All names of IO pins correspond to the SCI boards view: “output” generates a voltage for an external device, whereas “input” senses a voltage from an external device.

- Each analog port provides 16 analog output signals (DAC0 – DAC15), each of which has a resolution of 16 bits at 0 - 5V output voltages. Changing the output level in the least-significant bit (LSB) therefore results in a voltage change of 0.1mV. The DAC outputs are buffered (voltage follower Op-Amp) inside the DA-converter (Linear 2600), and can typically drive loads up to 20mA.
- Each analog port also provides 8 analog input signals (ADC0 – ADC7), with a sensing resolution of 12Bit at 0 – 5V level (yielding in a resolution of 1.25mV). The A/D-converter (Maxim 1226) internally performs 4 times over-sampling (software adjustable from no over-sampling to 32x over-sampling) at about 200 KHz sampling frequency.
- Each analog port additionally provides 8 digital output signals (DO0 – DO7) and 8 digital input signals (DI0 – DI7). These pins provide or read standard TTL level signals of 0 / 5V (logic low or logic high). They are not part of the SCI board’s digital section; the name DI / DO only refers to their characteristic of working with Boolean values rather than analog values.
- All analog ports share two high-speed digital output signals (CLK0 and CLK1), that can clock e.g. through scanners on aVLSI sensors at up to 1 MHz. If more than two fast clock lines are requires (e.g. for clocking 3 chips independently), the CLK signals need to get multiplexed by external logic (e.g. a logic AND with any pin of DO0-7).
- The remaining pins provide AGND and AVCC connections for external electronics. Ensure that the total power supply of your external hardware does not exceed 200mA (summed on all four ports). Also ensure that your external circuitry does not generate noise that feeds back into the AS. The pin D-AVCC shall be used for external circuitry that shows rather digital characteristics (e.g. clocks and scanners for readout), whereas the pins A-AVCC shall be used for pure analog electronics.
After power-up, the board initializes the analog and digital output ports to their default values. Users can store default power-up values in the microcontroller’s EEPROM by as explained in Chapter 5.1.

Each side of the board (upper and lower edges) contains two analog ports directly joined together: On the lower end P0 and P1; on the upper end P2 and P3. Instead of four single 50 pin connectors (for P0 – P3 independently) you will see two 100 pin connectors (jointly for P0+P1, and P2+P3 respectively). For clarification, the physical layout of these 100 pin connectors is provided in Figure 6.

![Figure 6: Analog Ports P0+P1 and P2+P3: one connector for each pair of ports](image)

### 3.4. Description of Digital IO port, P4

In contrast to the analog ports, the “Digital IO port” P4 is part of the DS. The direct connection to the Atmel microcontroller on the DS allows significantly faster digital IO pins and a large number of special functions like a serial port or motor control signals. As a drawback the signals on P4 suffer significantly from electric noise in the system. **Do not connect external circuitry on P0 – P3 (analog ports) with external circuitry on P4 (digital port), as this will require a common GND and thus introduce noise in the AS of the board!**

![Figure 7: Pin-out diagram of the “Digital Port” (P4), top view (frontal view). The non labeled pin is left empty to secure proper alignment of connectors](image)
The digital IO port P4 (Figure 7) offers a variety of special IO signals that are described in the following list:

- **TWI-SCL / TWI-SDA** offer a two-wire-interface (I²C compatible) to connect to external hardware or other microcontrollers.
- **MOTx-PWM+ / MOTx-PWM-** offer bi-directional pulse-width-modulated logic signals. These signals vary in duty cycle and direction (either PWM+ or PWM- active). They can e.g. drive an external actuator (motor). However, the signals provide logic levels only (no high power) – use the motor connectors described in Chapter 3.5) for powering small motors directly. In fact, the MOTx-PWM+/- signals are the identical signals used for driving motors 0-3; so use only either the MOTx-PWM+/+ on P4 or the direct motor connectors 0-3.
- **SERVOx** provides PWM signals that are tuned to control standard hobby-craft servos. They have a variable duty cycle of 0-5ms and a rest-cycle of roughly 50ms. They are the identical signals found on the direct servo connectors 0-4 (Chapter 3.5).
- **D-AVCC, D-AIN0-4** are connected directly to the on-board analog inputs (A-IN 3-7!) of the main Atmel M128 controller (Atmel A-IN 0-2 are reserved for battery voltage and compass readings). These A-IN ports can be used (possibly in conjunction with D-AVCC as a 5V reference) as 10-bit analog converters. However, do remember that P4 is part of the DS and therefore highly noise contaminated.
- **D-IO0-15** are general digital I/O pins, connected directly to the Atmel M128 ports E (0-7, default input) and B (8-15, default output). If you decide to modify software on the Atmel microcontroller, you can set each pin individually as input or output pin, and even change the functionality during operation. In input mode the Atmel IO pins provide software-selectable internal pull-up resistors between. In output mode the pins can typically provide currents up to 20mA.

After power-up, the board initializes the digital output ports to their default values. Users can store default power-up values in the microcontroller’s EEPROM by as explained in Chapter 5.1.

### 3.5. Description of Special Ports

In addition to general purpose analog and digital IO ports the SCI board provides a number of specialized connections:

- **Motor Control**: The SCI board offers bidirectional PWM signals to control up to four small motors (Figure 8). These PWM signals are amplified by Vishay SI9986 motor drivers to provide up to 1A each. For normal operation, provide a jumper between M-VCC and DVBATT. For large total motor currents (>= 1A) provide additional external power on M-GND and M-VCC. M-GND is directly connected to GND, so do not use a voltage that conflicts with respect to GND. M-VCC is directly connected to the power input of the Vishay motor driver chips. D-VBATT is directly connected to the power supply from the digital supply input (ref Chapter 3.1).
Servo Connectors: The SCI board offers five servo control outputs with PWM signals timed to fit standard model servos (Figure 9). The PWM timing for the active portion of the signal varies between 0 and 5ms, whereas the delay between consecutive control signals is about 50ms. The SCI board offers five servo-connectors each providing three pins: PWM-Signal, VCC, and GND. Ensure that your servo has the same order of pins as the PCB: Signal, VCC, and GND. Some model servos operate from as little as 3.5V; however, most servos are only rated to work between 4.5 and 5.5 V. For large total servo currents (>= 1A) additional power supply pads are placed close to the servo pads. Similarly as for the motors in the previous chapter, place a jumper between DVBATT and S-VCC for normal operation. For high current servos, however, provide external power on S-VCC and S-GND.
• **TWI Interface:** The Two-Wire-Interface (TWI) is a standard bus interface featuring master to many-slaves communication using only a unidirectional clock and a bidirectional data line. The TWI bus is compatible with industry standard \( \text{I}^2\text{C} \). Using the TWI interface facilitates adding customized hardware to the SCI board; e.g. further digital IO boards or further on-board processing power. Additional microcontrollers can use the standard interface to exchange information with the main controller, and thus also exchange data with the host PC. For “standard” operations you will not need the port. The SCL and SDA signals are identical to those found on P4 (digital port).

![TWI connector for system extensions](image)

Figure 10: TWI connector for system extensions

• **Reprogramming Sockets:** The SCI board additionally offers two Atmel specific ISP (In-System-Programming) sockets. A standard Atmel STK500 reprogramming device can update both microcontrollers, in case the provided Bootloader software in the Atmel M128 gets corrupted (refer to Chapter 4.4). These ISP ports are only of interest in specific situations; usually software updates shall get applied through the communication serial port. Please refer to appendix A to locate the reprogramming sockets.

3.6. **Description of On-Board Sensors**

• **Magnet Sensor:** The SCI interface was originally designed for mobile flying robots. For these robots a simple initial sense of orientation is provided through the earth’s magnetic field. An ‘on-board’ compass chip provides this information in terms of magnetic field strength in two orthogonal directions. Alternatively, the SCI board provides a single angle (rotation relative to magnetic north) after a calibration process. Please refer to appendix A to locate the magnet sensor if required.

• **Power Supply Voltage Sensor:** The SCI board can measure the two power supply voltages (analog and digital supply). Users can thus verify that the power supplies still provide sufficient energy for their application.

3.7. **Description of I/O Port Test Adapters**

The SCI board provides a large number of individual IO pins on the analog ports (P0-P3). The supplied small I/O-port-test adapters offer a simple mechanism by which the analog ports can
get checked for electric and mechanical failures (shortcuts, broken wires, etc). Simply attach
the small test adapter to a single port (either P0, P1, P2, or P3) and run the supplied software
scripts for tests (see Chapter 5.4). The test will report problems with the analog ports.

3.8. Description of Generic I/O-PCB Adapter

To facilitate prototyping the SCI board offers simple I/O-PCB adapters that connect to the
analog (P0-P3) or digital (P4) ports and provide a regular 0.1” grid space for customized
circuits. The additional hardware allows immediate prototyping of circuits that attach to the
SCI board, rather than designing an adapter PCB. The two options of the generic I/O-PCB
adapters for the analog ports are shown in figure 11; the pin-layout follows directly from
Figure 5 and Figure 11. The two boards only differ in the arrangement of D/A output pins.

![Figure 11: Two available generic I/O-PCB adapters that only differ in the arrangement of the DA-pins (left side of images)](image-url)
4. System Software Documentation

4.1. Operating Principles

The SCIF board contains a microcontroller that provides basic functionality to set and read I/O ports, memorize setup information, and read on-board sensors. Users can decide whether to use the provided simple command interface or to extend the program in the microcontroller to support faster I/O. For example, a typical task of analyzing an aVLSI vision chip might consist of clocking through an array of pixels and reading brightness values. The “read and clock” cycle can get implemented by individual commands using the on-board scripting language. Alternatively, users can write functions directly in C that perform the requested actions and return an array of brightness values for increased read-out speed.

4.2. Command Line Interface on the Main Microcontroller Atmel M128

The standard software on the SCIF board provides a command line interpreter. Commands are send and received over the serial communication interface (refer to Chapter 3.2) in human-readable ASCII format. These commands denote simple (“atomic”) actions, such as “Set Analog Voltage on Port P2 Pin 4 to 3.4V”. To enhance communication throughput, the commands are abbreviated to a compact code instead of “full English sentences”. Refer to chapter 5.1 for a detailed description or enter “???” to see a list of available commands. The main advantage of using the command line interface lies in the simplicity of triggering arbitrary actions. The major drawback of the command line interface is the slow processing speed due to the communication bottleneck at the serial port.

4.3. Customized Software Routines on the Atmel M128

As alternative to the command line interface, users can access and modify the complete source code of the existing system. The Atmel software written in C provides a framework of function calls similar to the command line interface described above (refer to Appendix E). Using these functions, customized routines running directly on the micro controller can achieve a similar flexibility with significantly increased execution speed. In this mode the communication interface is only used for triggering sequences of actions and retrieving data from the board.

Preferably, users do not modify the core program. Instead, upon request, we will add a call to a chip-specific function in the ParseRS232Data.c file. Once this call is established, all user specific commands for the new chip start with a particular prefix (e.g. “#ABM6-“) and call the chip specific function in a separate C file. This will ensure compatibility with future releases of the main software. An example for handling special chips is provided with the “ABM6-“-call that is handled in the “ABM6Functions.c” file.

The software originally was developed on Imagecraft’s AVR-ICC C-compiler (references in Appendix B), but should compile under GNU-C for Atmel microcontrollers without modifications. Please remember that the core software includes a WatchDogTimer that causes a device reset if not cleared within roughly 140ms. Simply call “WDT_ClearTimer()” to reset the WatchDog if your software requires more time for execution.
4.4. System Software Updates

New software (customized software or upgrades provided on the web page) typically gets transmitted to the Atmel M128 using the build-in boot-loader option. After power-up, the Atmel M128 runs the boot-loader program for about 5 seconds, during which software on the PC can contact the SCI board for sending software updates. The boot-loader operation is indicated by the on-board control LED showing fast blinking. Users do not need special programming hardware. The required software on the PC side is currently available as MatLab script “AtmelProgBootLoader.m”.

In case the Atmel boot-loader software gets corrupted, the SCI board offers two reprogramming sockets (refer to Appendix A) to connect an Atmel STK500 programming device and renew the software from scratch. The secondary controller Atmel M8 used for generating PWM signals typically does not need to get reprogrammed and thus does not offer a boot-loader. However, the SCI board does provide a reprogramming socket for emergency use.
5. Communication with the Board

5.1. Direct Communication

The simplest way to communicate with the board is using a terminal program at 921.600 Baud, 8N1, and to enter commands by hand. Start with “??<CR>” to display a list of available commands. An overview of currently available commands is provided in Appendix E. As an example, the following command will set pin 2 of analog output port P0 to 5V:

```
!AOUT0,2=65535 <CR>
```

The “!” (exclamation mark) denotes an output change, “AOUT” refers to the type of signal, “0” refers to the port P0, “2” refers to pin number 2 of that port and “65535” refers to the highest possible value of a 16-bit DA converter (0..65535). Another example:

```
?AIN3,7 <CR>
```

reads the analog voltage of P3, pin7. The “?” (question mark) denotes a request to read a voltage, “AIN” refers to the analog input pins, “3” refers to port P3, “7” refers to pin 7 of that port. You can also request the voltages of output pins, e.g.:

```
?AOUT3,2 <CR>
```

which will report the current analog-output voltage on port P3, pin 2. So no AD conversion happens in this example, the board only reports the memorized setting of the DA converter P3, pin2. The special command

```
!E <CR>
```

stores the current settings of all analog and digital output ports in EEPROM. These will be used as default setting for future power-up events. Ensure that the settings stored in EEPROM match your chips’ biases; otherwise your chip might get damaged immediately on power-up.

5.2. On-Board Scripts

Storing sequences of commands that get executed upon a trigger command is highly useful to enhance operation speed without programming customized microcontroller software. Currently the SCI board does not provide scripting support, but we are planning to add scripting functionality as soon as time permits. Please indicate urgent need. Until scripting is implemented you need to issue individual commands or program customized routines in Atmel-C (refer to Chapter 4.3).

5.3. C Routines / Libraries on the Host PC

Alternatively to using a terminal program to enter commands by hand, we supply a library of function calls in C, running on a host PC under Linux that handle the serial port communication. Do not confuse this library with customizing the program on the Atmel
microcontroller (as outlined in chapter 4.3)! The C library discussed here simple generates command strings and transmits them over the serial port – much like typing commands by hand. The same advantages (flexibility and simple use) apply but also the same drawback (slow due to communication delays). However, simple control solutions or automatic bias tuning is very quickly realized in C.

5.4. MatLab Scripts
For processing data in MatLab we provide simple scripts that rely on the C library to support basic functionality, as e.g. setting and reading voltages. Using the provided scripts as templates allows adding further calls for specific purposes. The board is easy to integrate in a MatLab test environment for evaluating aVLSI chips, or even into MatLab’s Simulink to generate real-time control algorithms. However, remember that the same advantages and disadvantages apply as described above because MatLab uses the serial communication protocol.

MatLab provides a convenient way to verify the functionality of IO-ports: Applying the IO-port test adapter (refer to chapter 3.7) and running the script “SICheckPort.m”, MatLab will report problems detected on the selected port.

5.5. XML / Chip Database
The XML interface developed by Matthias Oster <mao@ini.phys.ethz.ch> provides an extremely convenient solution to tune aVLSI chips. Initially, basic information of a particular chip is stored in an XML table. Using this information, several scripts provide convenient graphical user interfaces to tune, sweep, and store bias settings. This tuning can even happen remotely through an internet connection. The only major drawback is the further significantly decreased operation speed compared to all above methods. Please contact Jörg or Matthias by email to receive complete documentation.

5.6. Tradeoff: Convenience vs. Speed
In general, adapting the Atmel software (Chapter 4.3) is the fastest and ultimately most powerful way to work with aVLSI chips. However, during the development period, the interfacing methods described in the current chapter might provide results significantly faster and more convenient. Users always face a tradeoff between convenience and operating speed, as shown in the following list:

- Customized Atmel Software:
  Most flexible and fastest, but inconvenient during development
- On-Board Scripting:
  Very flexible and fast, but limited in functionality (one script at a time)
- Direct Communication / C Libraries
  Reasonably fast, requires programming additional software on PC
- MatLab Scripts:
  Convenient but slow
- XML Interface:
  Extremely convenient, but slow
6. Final Remarks

We hope that the SCI board facilitates working with aVLSI chips and a variety of other sensors that need analog setup voltages and provide analog output values.

The SCI board is work permanently under development. Please do not hesitate to contact the developer to suggest improvement, additions, etc. Although the boards are already used at various places, it is not clear that the software is free of bugs. Please report errors or inconsistencies to the developer.
Appendix A: PCB Layout

PCB top view:
Appendix B: Data Sheets and Resources

Main Microcontroller: Atmel M128
Secondary Microcontroller: Atmel M88
In-System Programmer: ISP STK500
   http://www.atmel.com/products/avr/

AD converter: MAX1226
   http://www.maxim-ic.com

DA converter: Linear 2600
   http://www.linear.com

Motor Driver: Vishay SI9986
   http://www.vishay.com

Optical-Coupler: HCPL-0631
   http://www.agilent.com

Digital Output Buffer on AS: HCF 4094
Digital Input on AS: HCF 4021
Magnetic Field Sensor: KMZ 52
   http://www.semiconductors.philips.com

Voltage regulators: XC6203
   http://www.torex-usa.com/

C-Compiler ICC-AVR
   http://www.imagecraft.com/software/adevtools.html

GNU C-Compiler
   http://www.gnu.org/

XML software documentation
   Please request documentation by email from conradt@ini.phys.ethz.ch

Libraries for Eagle PCB-layout
PCB Layout for Generic Adapter Boards
PCB Layout for Port-Extension to 0.1” Spacing Connectors
   Please request files by email from conradt@ini.phys.ethz.ch

Online support (software updates, documentation updates, etc)
   http://www.ini.ethz.ch/~conradt/projects/SensorControlInterface
Appendix C: C Function Calls provided by the System Software

extern signed char SetDigitalOut(unsigned char BoardID,
                                  unsigned char PinID,
                                  unsigned char Value);

extern unsigned char GetDigitalIn(unsigned char BoardID,
                                  unsigned char PinID);

extern signed char SetAnalogOut(unsigned char BoardID,
                                 unsigned char PinID,
                                 unsigned short Value);

extern unsigned short GetAnalogIn(unsigned char BoardID,
                                   unsigned char PinID);

extern unsigned short GetAnalogBatteryVoltage(void);
extern unsigned short GetDigitalBatteryVoltage(void);

extern unsigned short GetDigitalCompassX(void);
extern unsigned short GetDigitalCompassY(void);

extern void SetMotorSpeed(void);
                    // motor speed stored in global variable
Appendix D: Modifications since Revision 0

- Revision 1
  - Added Digital Port P4
  - Added servo output
  - Changed PCB design (4-layer instead of 2-layer design)
    - Moved location of reset button
    - Moved location of TWI interface connector
    - Added auxiliary power pads for motors and servos
    - Increased Width of Power Route to Motors
    - Changed voltage stabilizers to SMD types
    - Changed all capacitors from Elko to Tantal

- Revision 2
  - Added fast digital clock output lines on analog port, added 7th optic-coupler for digital clock on bottom of PCB
  - Re-arranged IO lines on analog output adapter
    **sensor interface boards from Rev 1 are no longer compatible!**
  - All motor control signals on hardware PWM control in M88
  - Servo PWM signal generated in software on M88
  - Software update: enhanced speed for input and output on analog connectors
  - Changed over-sampling of AD input from 16x to 4x
Appendix E: Overview of available Command-Line Commands

?Ain#P,#p -> -Ain#P,#p=#AX  request analog in value
?Aout#P,#p -> -Aout#P,#p=#AX  retrieve analog out value
!Aout#P,#p=AX -> -Aout#P,#p=AX  set analog out value
!Aout#P,#p=-  -> -Aout#P,#p=-    set analog out to tri-state
!Aout#P,#p=+  -> -Aout#P,#p=+    restore analog out to active

?Din#P,#p -> -Din#P,#p=#DX  request digital in value
?Dout#P,#p -> -Dout#P,#p=#DX  retrieve digital out value
!Dout#P,#P=DX -> -Dout#P,#P=DX  set digital out value
!Dclk#p=DX  -> -Dclk#p=DX    set high-speed digital clock line
?Dclk#p  -> -Dclk#p=DX    retrieve high-speed digital clock line

with #P = port [0..4]
#p = pin   [0..15] for analog-out (P0..3 only)
#p = pin   [0..7] for analog-in, digital-in/-out
#p = pin   [0..1] for analog clock lines
#AX = analog voltage out value [0..65535]
#AX = analog voltage in value [0..4095]
#DX = digital (logic level) value [0..1]

?M   -> -M0=MX,1=MX,2=MX,3=MX
?Mn  -> -Mn=MX
!Mn=MX  -> -Mn=MX
!M0=MX,1=MX,2=MX,3=MX -> -M0=MX,1=MX,2=MX,3=MX
retrieve / set motor speed and direction
with #MX = motor speed [-127 .. +127], n=[0..3]
#MX = "B" to force stop (0=free running, B=break)

?S   -> -S0=SX,1=SX,2=SX,3=SX,4=SX
?Sn  -> -Sn=SX
!Sn=SX  -> -Sn=SX
!S0=SX,1=SX,2=SX,3=SX,4=SX -> -S0=SX,1=SX,2=SX,3=SX,4=SX
retrieve / set servo angle
with #SX  = servo angle [0 .. 4095], n=[0..4]

?B  -> -BA=DX,AX,S=SX  get battery voltages
with #DX  = battery voltage [0 .. 1023] for digital battery
#AX  = battery voltage [0 .. 4095] for analog battery
#SX  = battery voltage [0 .. 4095] for external analog supply

!C   calibrate compass
?C  -> -C=angle  get compass reading
with #CX  = compass angle [0 .. 359]

!L=LX  -> -L=LX    set LED status to #LX [0..2]
?L   -> -L=LX    get LED status [0..2]
with #LX = 0=off, 1=on, 2=blinking

?E -> -EEPROM-FetchedSettings fetch output values (D/Aout) from EEprom
!E -> -EEPROM-SavingSetup... save the current setup (D/Aout) in EEprom
-EEPROM-done these values get defaults on power-up
!Tx -> Test OK        Test analog Port Px (assuming test adapter on Px)

!Ix -> BoardID: x    Store Board ID String (<32 chars)
?I  -> BoardID: x    Read Board ID String

!R                    Reset device (complete)
!FR                   Reset to Factory Settings

??        Show Usage