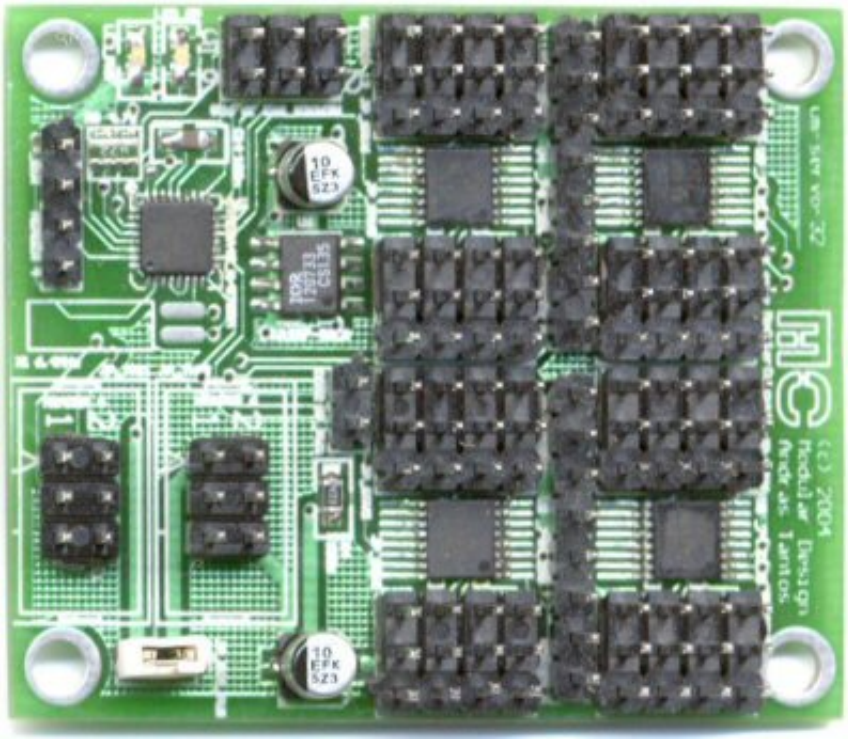
µM-Servo-32

Author : Andras Tantos



Introduction

This µModule implements a 32-channel R/C servo controller. The module interfaces to the external world using the TWI interface that's common among all µModules. It also has an optional CMOS-level RS-232 interface option. The module is ideal for biped (humanoid) or hexapod robot control applications where a large number of servos have to be controlled. For even higher number of servos multiple servo controllers can be connected to the same TWI channel. With this feature hundreds of servos can be controlled by a single central controller.

Features

- Standard µModule TWI with optional RS-232 interface
- Integrated 3.3V power supply, 5V operation is optional usnig an external power source
- Up-to 32 servos can be controlled in parallel
- High precision: around 7 bits of resolution per servo

License

H-Storm Non-Commercial License (HSNCL)

Redistribution and use in source or binary forms, or incorporated into a physical (hardware) product, with or without modification, are permitted **for non-commercial use only**, provided that the following conditions are met:

- **The redistribution doesn't result in financial gain.**
- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in any other form must contain in printed or electronical format the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- All advertising materials mentioning features or use of this technology must display the following acknowledgment:
  This product includes H-Storm technology developed by Andras Tantos and Modular Circuits.
- Neither the name of Andras Tantos or Modular Circuits may be used to endorse or promote products derived from or using this technology without specific prior written permission.

ALL THE INFORMATION, TECHNOLOGY, AND SOFTWARE IS PROVIDED BY THE AUTHORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ANDRAS TANTOS, MODULAR CIRCUITS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE OR TECHNOLOGY, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Design description

The bus interface

The TWI bus interface follows the µModule standards. There are six or four-pin connectors on the board, both with identical functionality. Two wires are used for the TWI signal transmission (clock and data) while the rest provide power and ground signals. The interface can be operated at a rate up to 400kHz. This module never initiates any transactions on the bus, it operates in slave-only mode. Status information can be acquired by polling and commands can be sent to the module at any time. The module implements the standard 8-bit register bank µModule

communication protocol with 32 8-bit registers, one for each channel.

Powering options

The module can source and sink power on this interface. The power to and from these sockets can be interrupted by an on-board jumper. With this there are three possible powering configurations with regards to the logic-level functions:

- The module is powered form the same power as the first four servos. The module provides power to other devices on the TWI bus
- The module is powered from the power supply of powering the first four servos, but it does not power other devices on the TWI bus
- The module is powered from the TWI bus

Servos can consume quite a large amount of power when under heavy load (we've measured around 0.6A on a standard servo). Since the module can control 32 servos that can add up to a huge amount of power. In order for this power to be delivered to the servos, the power lines for the servo headers are cut into 8 regions, each powering 4 servos. The power pins for the first set of four servos is connected to the power regulator for the digital circuitry on the module. It is therefore important that power to be connected to the connector for the first four servos even if no actual servos are connected to those headers. It is possible to connect one or more set of servo power pins together and provide power through a single wire but care must be taken to the maximum current that can flow through that wire.

Servo control

Each servo output is a PWM signal, with an approximately 50Hz repetition rate and a pulse width between 0 and 2.5ms. The pulse width is controlled by an 8-bit register so the step-increment in the pulse is around 10us. Most R/C servos require a 1..2ms pulse with, with 1.5ms corresponding to the center position. These values lead to 100 different values that servos can interpret or about 7 bits of usable resolution. The control signal for the servos is a 3.3V-compatible CMOS signal (unless an external 5V power supply is used) which is compatible with 5V TTL input levels that most R/C servos require.

Note that these are approximate values and are not intended to be used as a source of any kind of timing. Our measurements show that for example a Hitec HS-425BB servo requires value 61 for its left-most position and the value 220 for its rightmost position. Exact values and limits differ from servo to servo and must be experimented with for each application.

Power consumption

The current drawn by the circuit is around 7mA when operated from the TWI interface. The optional external RS-232 level shifter consumes an additional 4mA.

Miscellaneous functions

The module on the top of the standard TWI interface, that is common among all µModules also contains a (logical level) RS-232 interface. This interface can be used to connect the module to other microcontroller modules or (after level-shifting) to a PC which doesn't have a TWI interface.

Serial interface

The serial interface is configured to 38400 BAUD, 8 data bits, no parity and 2 stop bits. The communication protocol is very simple.

The host uses commands to initiate communication with the servo controller. The command consists of a single character identifying the command followed by zero or more data transmitted by either the servo controller or the host. For each byte the servo controller receives it will send exactly one byte. This way the host can make sure that the communication channel is intact.

The connect command

The purpose of this command is to establish connection from the host to the servo controller or re-synchronize the communication in case it is lost. The host transmits a single 'c' (dec 99) character to which the servo controller responds with the same 'c' character. If any other response is received it can be assumed that there was a communication error and/or the syncronization between the host and the servo controller has been lost. The connect command can be repeated multiple times until a valid response is received from the servo controller at which point the synchronization is re-established. If no response is received within a reasonable time (5ms), it can be assumed that the communication channel is broken or there is no servo controller connected to the serial port.

The channel address command

The channel read and write commands use a channel address stored in the servo controller to index between the 32 available channels. This address can be set by this command. The command starts with the host transmitting an 'a' (dec 97) character to the servo controller. The controller acknowledges it with a '*' (dec 42) character. If any other character is received by the host, the synchronization is probably lost between the host and the servo controller and can be re-established using the connect command. After the reception of the acknowledge, the host transmits the channel address as a single byte (between 0 and 31). If the servo controller received a valid channel number, ir updates its internal channel address pointer, and acknowledges the command with a '*' character again. If the channel address was invalid, it responds with an error flag, the 'E' (dec 69) character.

The channel read command

The channel read command reads the current value of the currently addressed channel and after successful completion increments the channel address in the servo controller (the channel

address wraps around after the last channel, 31). The command start by the host transmitting the 'r' (dec 114) character. The servo controller responds to this command by sending the current value of the addressed channel as a single byte.

The channel read without increment command

This command is very similar to the previous one, with the exception that it does not change the channel address after execution. It is initiated by the host with the 'R' (dec 82) character to which the servo controller responds with the current value of the addressed channel.

The channel write command

This is the counterpart of the channel read command: it writes the value of the currently addressed channel and increments (in a wrap-around fashion) the channel-address afterwards. It is started by the host transmitting the 'w' (dec 119) character to which the servo controller responds with a '*' acknowledge character. After this the host transmits the new value for the addressed channel as a single byte. The value 0 turns the channel off. Note that values above 250 are not handled by the servo controller and corrected to 250. After the servo controller received the channel value, it responds with the acknowledge character '*'.

The channel write without increment command

This command behaves the same way as the channel write command with the exception that it does not increment the channel address after execution. It is initiated by the 'W' (dec 87) character, followed by the new channel value. Both characters are acknowledged by the host with a '*' response.

The TWI interface

The TWI interface adheres to the standard µModule communication protocol. It implements 8-bit register-bank addressing, and defines 32 registers. Each register corresponds to a single channel.

Register offset	Register name	Comment
0	Servo0	Servo channel 0 position control
1	Servo1	Servo channel 1 position control
2	Servo2	Servo channel 2 position control
3	Servo3	Servo channel 3 position control
4	Servo4	Servo channel 4 position control
5	Servo5	Servo channel 5 position control

6	Servo6	Servo channel 6 position control
7	Servo7	Servo channel 7 position control
8	Servo8	Servo channel 8 position control
9	Servo9	Servo channel 9 position control
10	Servo10	Servo channel 10 position control
11	Servo11	Servo channel 11 position control
12	Servo12	Servo channel 12 position control
13	Servo13	Servo channel 13 position control
14	Servo14	Servo channel 14 position control
15	Servo15	Servo channel 15 position control
16	Servo16	Servo channel 16 position control
17	Servo17	Servo channel 17 position control
18	Servo18	Servo channel 18 position control
19	Servo19	Servo channel 19 position control
20	Servo20	Servo channel 20 position control
21	Servo21	Servo channel 21 position control
22	Servo22	Servo channel 22 position control
23	Servo23	Servo channel 23 position control
24	Servo24	Servo channel 24 position control
25	Servo25	Servo channel 25 position control
26	Servo26	Servo channel 26 position control
27	Servo27	Servo channel 27 position control
28	Servo28	Servo channel 28 position control

29	Servo29	Servo channel 29 position control
30	Servo30	Servo channel 30 position control
31	Servo31	Servo channel 31 position control

Design files

µModule Users Manual (HSOL)
Schematic and PCB in PDF format (HSNCL)
Firmware source code (HSNCL)