MicroTech

MT Servo Controllers

Version 3.00

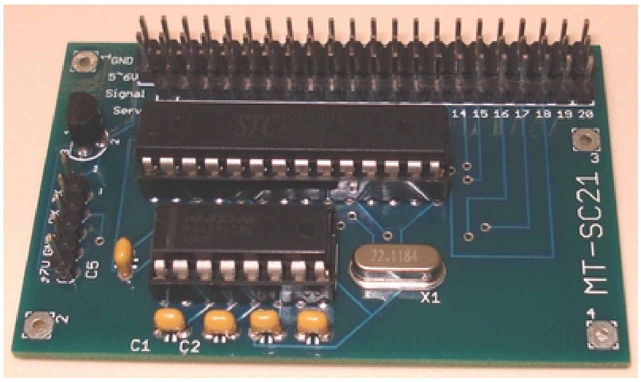
www.mcu.hk

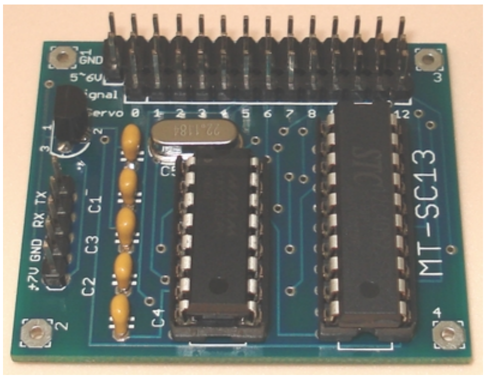
Warning:
Incorrect power connection to any electronic and electrical equipment may seriously damage them or even cause a fire hazard or explosion. Users must take care to identify the correct pins and supply an acceptable voltage to operate them safely.

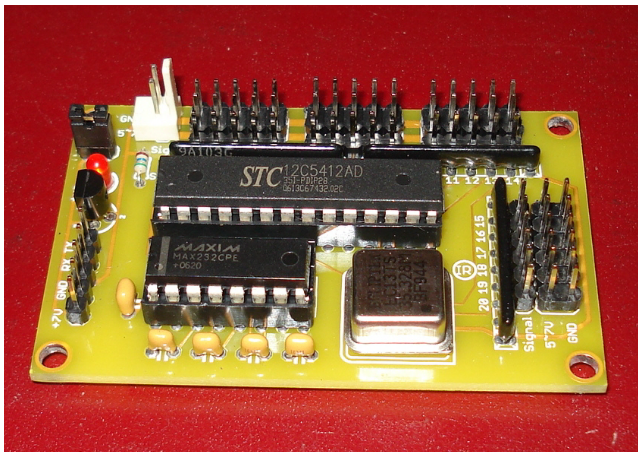
October, 2010

Important Safety Warning

This kit is not intended for young children! Assembly of this kit requires high-temperature soldering and the use of sharp cutting tools. Some included components may become hot, leak, or explode if used improperly. MicroTech strongly recommends that you wear safety glasses when building or working with any electronic equipment. Children should use this kit only under adult supervision. By using this product, you agree not to hold MicroTech liable for any injury or damage related to the use or to the performance of this product. This product is not designed for, and should not be used in, applications where the malfunction of the product could cause injury or damage.


MT-SC21


MT-SC13


MT-SC21-35M

Product Features

1. Parallel signal output mode, all the servo signals go High at the same time
2. Independent servo operation, each servo has it’s own set of speed and position settings
3. Individual servo speed control, i.e. different servos are capable of moving at different speeds simultaneously
4. Selectable baudrate from 600 to 19200
5. Command to retrieve servo position
6. Control up to 21/13 servos
7. 24 (MT-SC21, MT-SC21-35M), 19 (MT-SC13) storages for servo speeds and positions
8. Simple commands to operate
9. Small board size 65x41mm (MT-SC21), 46x38mm (MT-SC13)

Introduction

The MT Servo Controllers are 21/13 channel R/C servo drivermodule s designed using the parallel output mode which will drive up to 21/13 R/C servos. The servo controllers accept RS-232 serial data from a host PC or MCU and output PWM (pulse width modulation) signals to control the attached servos.

In the slowest speed setting (speed control set at 1) a servo moves from 0 to 180 degrees in 5s (MT-SC21) and 8s (MT-SC13). With speed control in action, different makes of servos move in unison, despite the fact that they might have different moving speeds!

The servo controller performs CPU intensive task which concurrently generating independent servo control signals of 0.5ms to 2.5ms width logic high pulses. The width of a pulse is greater or smaller than that required range by most servos and this allow it to operate it's full moving range of travel. However, **care must be taken not to overdrive servos** to their extreme limits at both ends. Start at the middle (90, 1.5ms) position first, then gradually work out the upper and lower limits the servo can takes. Otherwise, the gears inside the servo might easily get damaged due to overdriving.

The lower and upper travel bounds of servos can be set by the "servo-limits.ini" file, but this file has to be read in (by a click on the "Range" button) to be effective. User has to manually modify the values in this file by a text editor.

Servo Working Mechanism

Radio Controlled (R/C) servos are small electromechanical devices designed for wireless remotely controlled models, such as cars and airplanes. Inside a servo, a small motor drives a chain of gears which finally drives an output shaft which is linked to a potentiometer to measure it's position. An electronic feedback circuit reads the voltage value of this potentiometer and then controls the spinning direction of the motor to make the output shaft move to the desired position. A servo has three leads, two for power supply and one for a control signal. The control signal is a continuous stream of logic HIGH pulses that are 0.5ms to 2.5ms long. The width of the pulses determines the position to which the servo moves. Normally, a 1.5ms signal pulse width will set the servoto move to its middle neutral position. A servo moves back and forth as the width of the pulse changes. There are many different types of servos on the market that may differ from one manufacturer to another. However, the basic mechanism remains the same.

PCB Assembly & Soldering Tips

A soldering iron and a wire cutter is needed to assemble the servo controller. The general technique for soldering components onto PCB is that always start with the lowest in height components first, then work towards the tallest components. Here are the steps to assemble:

1. Insert all the capacitors (yellow beans) + crystal, flip PCB over, then solder.
2. Insert the 2 IC sockets, note the notch on one end that matches the PCB marking, flip PCB, then solder.
3. Insert all the pin headers, solder just 1 leg first, make sure they are sitting upright, then solder the rest.
4. Gently bend the middle lead of 78L05 voltage regulator outward, insert it into PCB, then solder.
5. Finally, gently bend the leads of the 2 ICs on a solid hard surface into 90 degrees angle, insert them into the IC sockets. Notice the notch on one end of the IC should match the IC socket, either facing the left side (MT-SC21) or the top side of PCB.

Power Supply

For a reliable operation of the servo controller, the recommended method to power the controller is to use two separate power sources, one to power the controller's circuit and the other to power the servos. Due to a voltage drop of about 1.6V by the 78L05 voltage regulator, a minimum of 6.6V is required to power the controller. Whereas a high current and stable power supply from a 4-cells Ni-Cad, Ni-MH or Li-ion battery pack of 5-6V is recommended to power the servos. Some servos might take voltage supply more than 7.2V, but this is risky and not recommended. Do a proper check with servo specification before operating.

Note: At a higher motor supplying voltage, servos might "vibrate" due to the "overshooting" phenomenon.

Connecting the Servo Controller

Extreme care must be taken when connecting servos, because it is easy to plug in a servo backwards. Make sure to connect your servos correctly or they may be destroyed. The signal wire (usually white, yellow or orange in colour) should be closest to the microcontroller and above the servo number markings. The power supply ground (black) wire should be closest to the top edge of the board.

Connect the supplied RS232 cable between a computer's serial port and the top 3 pins of the 5-pin header. Do notice the ground (black) wire which connects to the pin with GND marking.

Now, connect a 7V power supply to power the servo controller circuit. Beware of the polarity of power wires, wrong connection may seriously damage the controller or even cause explosion, the circuit was not protected from wrongly power connection!

Note: These servo controllers were designed and based on a standard Futaba S3001 servo, some other makes of servos might need 5K~10K pull-up (to 5V) resistors (or resistors connected in serial to reduce current draw by the servos) at the signal output pins in order to work properly.

Position Storage

The MT Servo Controllers distinguished from other servo controllers on the market with this nifty feature. The current settings of all the servo positions, speeds and baudrate values can be saved to the controller's EEPROM memory and later be retrieved by issuing a single command. The total number of available storage for MT-SC21 is 24 and 19 for MT-SC13. Imagine that a humanoid robot is performing the act of Chinese TaiChi, this feature is very useful in setting different gestures of the robot and then save these gestures into memory locations. The robot can later be set to the required gesture by simply sending a 2-bytes Retrieve Command to it, instead of sending a series of bytes to move a large number of servos individually, which will cause the microcontroller to handle the incoming stream of data and might interrupt the current motion. This feature is vital for moving those servos at the robot's legs simultaneously, otherwise the motion might get out of pace if data are being received and processed over the RS232 serial line. The values stored at memory location 0 are retrieved after power applies to the controller to set the initial default preset positions.

About Feedback

Some servo controllers on the market claimed that they have a controlling command to return the position of a servo. This is misleading and meaningless, because servos are 'one-way traffic' devices that are manufactured to receive data only. Therefore, the servos will NOT send back any data to the servo controllers, which in turn send back to the controlling host. What the servo controllers sending back to the host are the data previously sent to them by the host, which do not reflect the **actual** positions the servos had moved to.

One way to find out whether or not a servo has moved to it's designated position is to solder a wire to it's potentiometer and use an ADC to convert this voltage value into a number that the controlling host can understand. By knowing this number the host knows the servo's current actual position and acts accordingly.

However, by providing such a command to retrieve these data might be useful for those microcontrollers with very limited run-time memory, i.e. they do not have enough memory to store these data.

New digital servos with actual feedback will be available in the market very soon.

Controlling Commands & Protocol

The baudrate for servo controller to receive commands is set at default 9600,8N1. It reacts on the received data as soon as the number of bytes for a particular command is satisfied. Since the controller was designed with running speed efficiency in mind (by eliminating error-checking codes), so, it does very little in checking the correctness of the received data. Therefore, user must not send any data to the controller outside the permitted values. A command is simply a set of 1 to 3 bytes data sent in pure ASCII format with no headers or delimiters. The following table summarizes the formats and ranges of allowed values for the controlling commands.

Format:
1st Byte 2nd Byte 3rd Byte
<command / servo number> <servo number / memory location number / position / baudrate> <speed>

Command Description	**1st Byte (ASCII value)**	**2nd Byte (ASCII value)**	**3rd Byte**
Initialize all memory storage	240	240	240
Get Version Number	244		
Retrieve Servo Position	245	0..20(SC21), 0..12(SC13)	
Set Baudrate	246	6,12,24,48,96,192	
Set All Servos Speeds Same	239	0..5	
Set Servo Speed	250	0..20(SC21), 0..12(SC13)	0..5
Initialize Memory Location	251	0..23(SC21), 0..18(SC13)	
Save to Memory Location	252	0..23(SC21), 0..18(SC13)	
Retrieve Memory Location	253	0..23(SC21), 0..18(SC13)	
Set Servo Position	0..20(SC21), 0..12(SC13)	0..180	

It is recommended to issue 3 bytes of 240 to initialize all the memory storage before use. The default speed for a servo is set to 0, which means no applied speed control setting and the servo is moving with it's maximum speed. Speed of 1 means slowest, 5 is the fastest, analogous to the number of gears in cars!

The position value (0..180) is just an arbitrary range of values that applies to servo, it does not necessary corresponding to the actual angular position the servo will finally be moved to. Since the characteristics of servos by different manufacturers (or even from the same manufacturer) are not always the same. Care must be taken not to overdrive servos to their extreme limits at both ends. Start with the value 90 first, then gradually work out the upper and lower limits the servo can take. Otherwise, the gears inside the servo might easily get damaged due to overdriving.

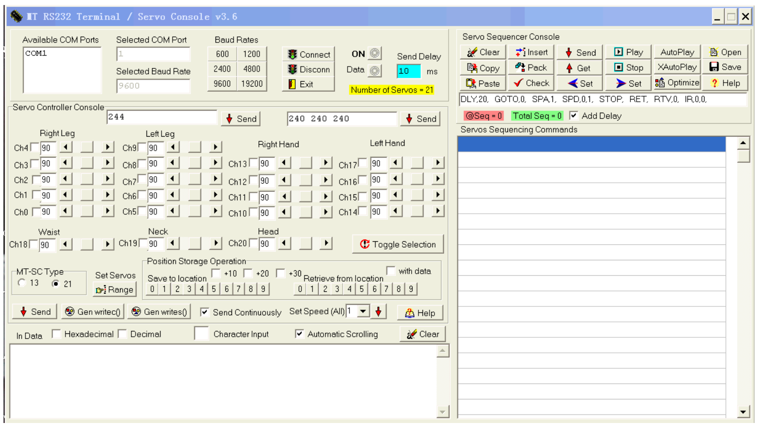
When setting baudrate, because the maximum value a byte can hold is 255, therefore, the baudrate value is divide by 100 (i.e. 192 means 19200, 6 means 600) to be kept within the range. The servo controller starts immediately operating in this new baudrate setting and this baudrate value is stored in the controller's memory, until it is reset with different baudrate value. Exception to this command is that the baudrate for MT-SC13AT is fixed at 9600, user cannot change it.

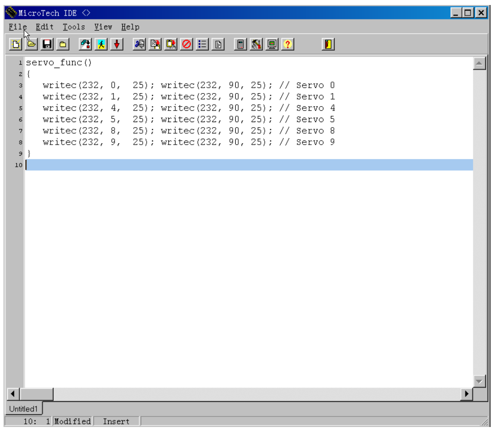
In case you forgot the previously set baudrate, you can use the MicroTech IDE's RS232 Test Terminal to find it out. At power up, the controller will send a "sign-on" message ("MT-SC21" or "MT-SC13") to the Terminal. By using this characteristics, you can select different baudrate settings on the Terminal, then switch Off/On the controller, until you are able to see the "sign-on" message. It's simple and easy!

Testing and Usage

By using the MT Servo Controller/Sequencer Console (SQ.exe or SQ2.exe), user can send test data to the servo sequencer to move the servos. The numbers within the text input boxes are the ASCII codes that will send to the controller. Note that, spaces between the numbers are skipped and ignored.

User can also simply use the commands within the Console to move servos interactively to determine the positions of all the required servos contributing to a particular robotic gesture. By pressing the **Gen writeX()** button, a TinyC function is generated which contains the codes for all the involved servos. The codes are temporarily stored in the Windows system buffer, which can later be pasted into Editors.





```
servo_func()
{
   writec(232, 0,  25); writec(232, 90, 25); // Servo 0
   writec(232, 1,  25); writec(232, 90, 25); // Servo 1
   writec(232, 4,  25); writec(232, 90, 25); // Servo 4
   writec(232, 5,  25); writec(232, 90, 25); // Servo 5
   writec(232, 8,  25); writec(232, 90, 25); // Servo 8
   writec(232, 9,  25); writec(232, 90, 25); // Servo 9
}
```

The codes on the right are pasted from Windows system buffer after pressing the **Gen writec()** button on the Servo Controller Console. This feature speeds up robotic development tremendously. User can just call this function from the main program control loop to perform the required robotic action. A TinyC program calls several sets of these generated functions to perform a series of motions. Don't forget to rename the function names to meaningful ones for the ease of maintenance.

Note: Do select the correct servo controller type before issuing any command. The last parameter for writec() function is the delay setting, NOT the COM port number. Use sys(0, x) function to set the default COM port for subsequent output by various serial communication functions. See sample programs, "2xCOM.prg" and "4xCOM.prg" for demonstration.

Sample commands

Initialize all the memory storage locations
240 240 240

This is the only single byte command that is used to retrieve the controller's firmware information.
244

Move servos 0, 1 and 2 to position 30
0 30 1 30 2 30

Move servos 0, 1 and 2 to position 150
0 150 1 150 2 150

Initialize memory storage locations 0 and 1 (all channels are set to position 90, speed control off)
251 0 251 1

Set all servos to moving speed 1 (slowest)
239 1

Set all servos to moving speed 0 (fastest, speed control off)
239 0

Set servos 0 and 1 to moving speed 1 (slowest)
250 0 1 250 1 1

Save current settings to memory storage location 4
252 4

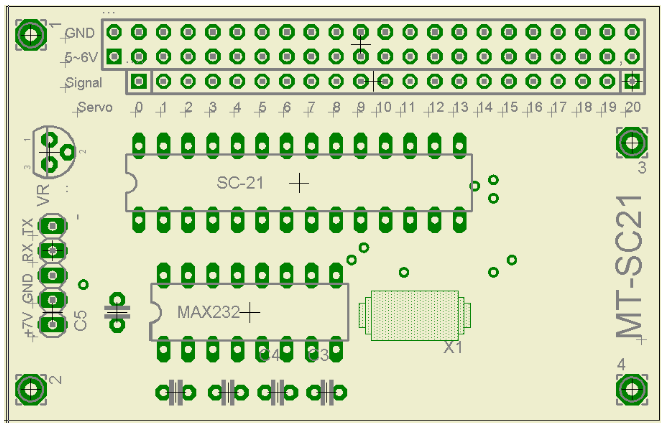
Retrieve settings from memory storage location 4
253 4

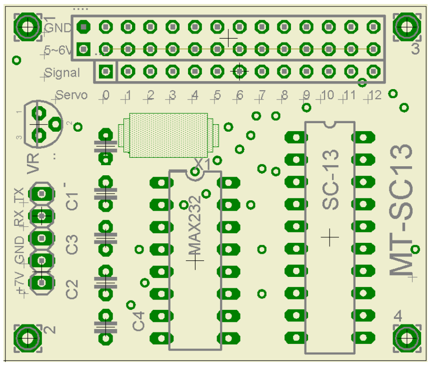
Retrieve the 1st servo position
245 0

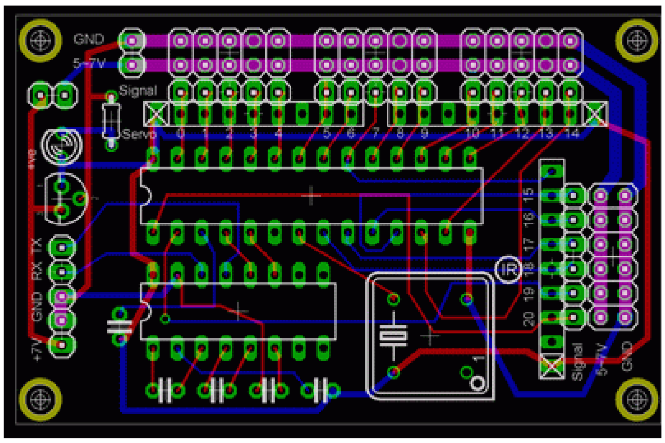
Set baudrate to 1200,8N1
246 12

After issuing this command, the controller starts operating in this new baudrate setting, you need to re-select the Terminal's baudrate to 1200 (don't forget to press the "Connect" button as well), in order be able to communicate with the controller again.

Pinouts and Board Layout







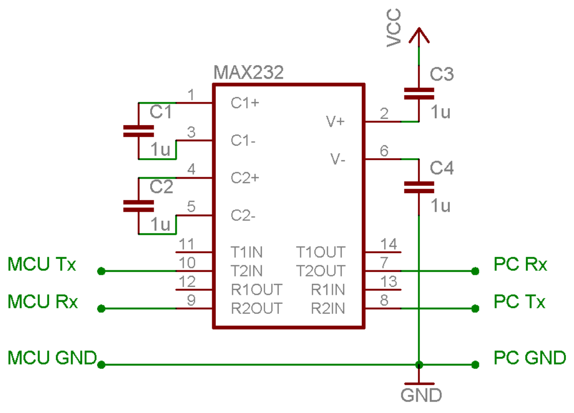
Optionally connect the jumper to power the board ICs from the power supply to servos.

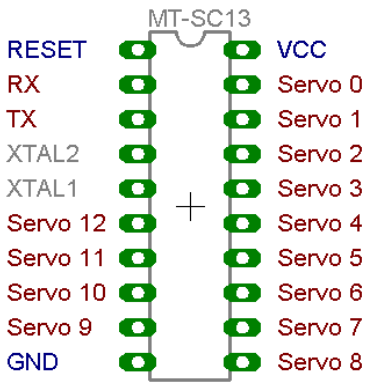
Others

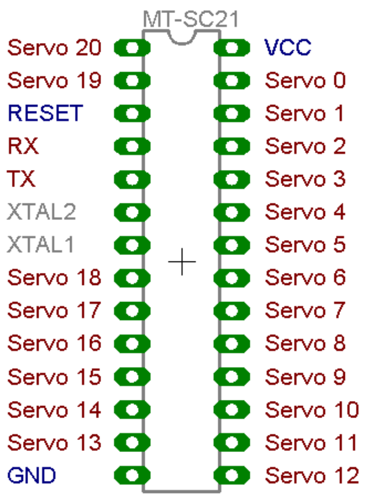
Apart from the aforementioned 3 servo controllers offered in kits or assembled forms, MicroTech also offers servo controllers in chips form only, together with the required crystals for OEM application and development. The types and specification for these chips are listed below for reference. All of these chips are 5V operating ICs, all you need to do is just hook them up with a RS232 serial communication with a 5V operating MAX232 IC, plug in the crystals, off they go!

Note: About the 'resolution', since the MT Servo Controller Console is able to send position values from 0 to 180 (total of 181 values) only and the servo controller firmware is programmed to accept this range of values, so, the number of steps within this range will be scaled up or down accordingly to represent this range. Therefore, the higher the steps values offer better resolution.

Model Number	Resolution (Steps)	Servos	Storage	BaudRate	XTAL
MT-SC13	160:181	13	19	600-19200	22.1184MHz
MT-SC21	115:181	21	24	600-19200	22.1184MHz
MT-SC21-35M	160:181	21	24	600-19200	35.3284MHz


VCC
MAX232
C3
1u
C4
1u
C1
1u
C2
1u
C1+
C1-
C2+
C2-
V+
V-
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
MCU Tx
MCU Rx
MCU GND
PC Rx
PC Tx
PC GND
GND


MT-SC13
RESET
RX
TX
XTAL2
XTAL1
Servo 12
Servo 11
Servo 10
Servo 9
GND
VCC
Servo 0
Servo 1
Servo 2
Servo 3
Servo 4
Servo 5
Servo 6
Servo 7
Servo 8


MT-SC21
Servo 20
Servo 19
RESET
RX
TX
XTAL2
XTAL1
Servo 18
Servo 17
Servo 16
Servo 15
Servo 14
Servo 13
GND
VCC
Servo 0
Servo 1
Servo 2
Servo 3
Servo 4
Servo 5
Servo 6
Servo 7
Servo 8
Servo 9
Servo 10
Servo 11
Servo 12