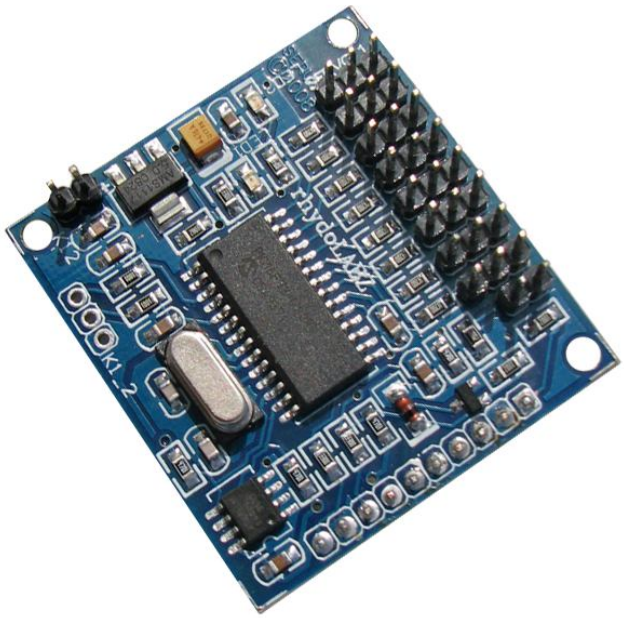
Document : Datasheet
Date : 16-Mar -07

Model # : ROB - 1185

Serial Servo Controller

- USART / I²C with ADC



Rhydo Technologies (P) Ltd.
(An ISO 9001:2008 Certified R&D Company)
Golden Plaza, Chitoor Road,
Cochin – 682018, Kerala State, India
Phone : 0091- 484-2370444, 2371666
Cell : 0091- 99466 70444
Fax : 0091 - 484-2370579
E-mail : info@rhydolabz.com, sales@rhydolabz.com
WebSite : http://www.rhydolabz.com

rhydolabz.com

The "Serial Servo Controller (USART & I^2C) with ADC" from Rhydolabz is a very compact solution for controlling up to eight RC servos from a PC or microcontroller. Each servo speed and range can be controlled independently. This servo controller supports USART and I^2C communications. The user could control the servomotors by sending the desired command from an I^2C or USART supporting device. Servo Controller supports two communication protocols - Mini SSC-II and RHYDO-SSC protocol. The built in three ADC Ports can be used for measuring any analog voltage where one channel is permanently connected to Servo Voltage and two other can be used to measure analog inputs like Sensor values. With appropriate command from host controller the module will do ADC conversion for the selected channel and send out the result via USART/ I^2C. This feature is extremely useful for microcontroller without On-chip ADC (like AT89S52), and also to detect the battery low condition. Up to 128 Servos can be controlled by connecting 16 Modules in Parallel.

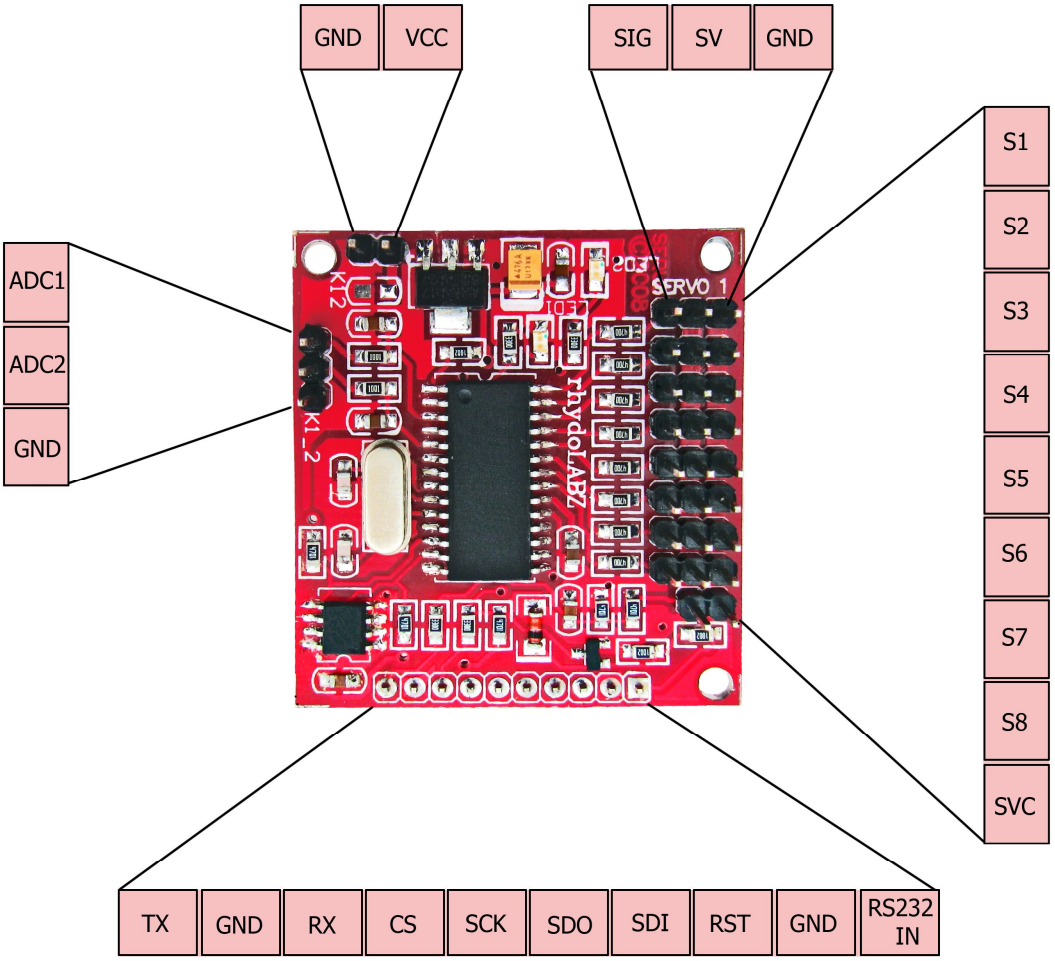
FEATURES

- Professional EMI/RFI Complaint PCB Layout Design for Noise Reduction
- Support Mini SSC-II and RHYDO-SSC Protocol
- USART and I^2C interface with host Controller
- 3 Channel ADC measurement using command
- Servo voltage measurement using command
- Onboard EEPROM to save Configuration Parameters
- RS232 to TTL conversion for PC based controlling
- On Board Status LED Indicator
- On Board TTL / RS232 / I^2C connector
- Can set Neutral Point for each servo
- Mini SSC-II Mode in default and Ready to use

SSC SPECIFICATIONS

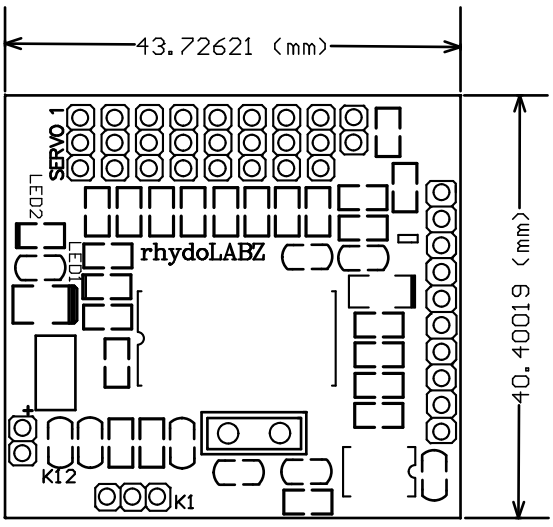
- Number of servo ports - 8 Nos
- Pulse Width Range - 0.25 to 2.5ms
- Resolution - (~0.05 degree)
- Supply voltage - 6.5 -12V
- I/O Voltage - 0 - 5V
- Baud Rate - 9600 - 115200
- Current consumption - 5 mA (average)

PIN DIAGRAM AND FUNCTIONS



PIN	PIN NAME	DETAILS
VCC	Power Supply	Power Supply Input (5V)
GND	Ground	Ground Level of Power supply
SIG	Signal	Signal pin for servo
SV	Servo Voltage	Power Supply Input(for servos)
S1-S8	Servo1-Servo8	Servo Connectors
SVC	Servo Volt Connector	Power Supply Connector (for servos)
RS232 IN	RS232 Input	RS232 input Receiver Pin
RST	Reset Input	A low level on this pin for longer than the minimum pulse length will generate a Reset.
SDI/SDA	Serial data in/out	Serial data input/output pin for I2C
SCK/SCL	Serial clock	Serial clock pin for I2C
ADC1	Analog1	Analog input Pin 1
ADC2	Analog2	Analog input Pin2
TX	Serial output	Data Output Pin for Serial Programming
RX	Serial Input	Data Input Pin for Serial Programming

DIMENSIONS



CONNECTIONS

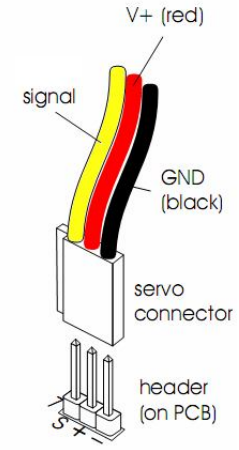
Power

Controller and Servos are powered separately. Servos are powered by their supply consisting of 4-6V. This can be connected at the bottom right of the board (SERVO+ and GND). Supply for servo should be capable of providing several amps of current. The controller is powered by a supply of 6.5-12V. This can be given at the top left of the board (+VE and GND). A regulator circuitry is provided for supply.

Note : *Power supply beyond the limit or reverse voltage can damage both the controller and servos.*

Control signals

A logical level serial input (0-5V) or an RS232 level serial input or an I2C input can used to provide control signals to servo controller. Do not use both RS232 and logical input at a time. Board can receive both Serial and I2C data simultaneously.



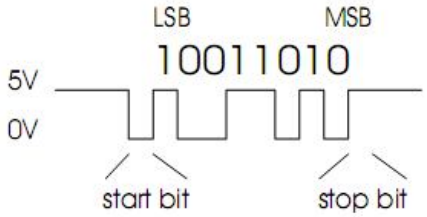
Servos

Improper connection of servo pins will cause damage to servos. The signal pin (yellow or white) should be toward the inside of the board, And the black wire (GND) should be closest to the edge of the board.

HOW TO USE SERVO CONTROLLER

Servo controller can generate 8 independent servo control signals simultaneously. The pulses vary from 0.25 milli sec to 2.5 milli sec which can drive the servo motors at an operating range of 0-180 deg.

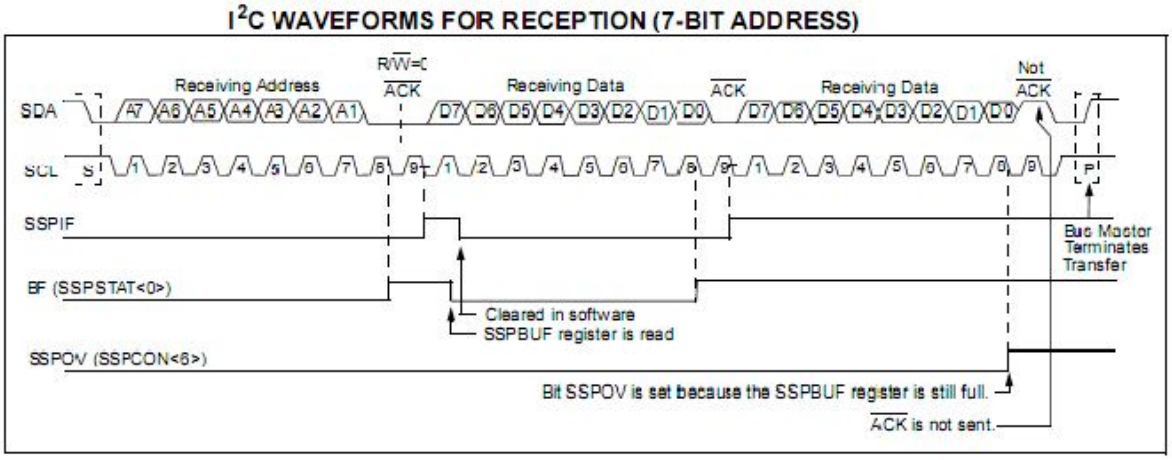
➢ Serial input



The commands given should be eight bits at a time, with no parity and one stop bit. Logic level serial input must be non-inverted, meaning that a zero is sent as a low voltage, and a one is sent as a high voltage, as shown in the diagram. Any inverted serial input, whether at logic levels or at RS-232 levels, may be connected to the RS-232 serial input.

➢ I^2C input

Here also the commands given should be eight bits at a time. Devices which can support only I^2C communication can take the advantage of this mode. The command patterns are same for both serial and I^2C communication. The only difference is that you have to follow the I^2C communication protocol. The servo controller is in I^2C slave mode with slave address 0xE0. The clock diagram for slave reception is shown below.



➢ LED indication

There are two LED's for the status indication. One LED is for process completion and another for the completion of signal reception (Serial/I2C). The LED 3 will remain ON if the process is incomplete. The LED 1 will turn ON when a valid start byte of command arrives and turn off when the entire bytes in one control signal pattern is received.

➢ Reset condition

The servo is factory set to Mini SSC-II mode. When the power turns on, the controller waits for a synchronization signal 0x55 through serial data-line @ 9600 baud rate or I2C data pin. The controller will then read the previous data stored in an external EEPROM. Baud Rate and mode of operation will be transmitted through serial output (Tx) pin at a Baud Rate of 9600. The servo controller automatically sets to that particular Baud Rate and Mode. It will configure parameters like neutral angle, speed, range, Device ID and servo numbering based on the data read from EEPROM location. The location that is read on reset will be the previously selected location by the user. The user can change the mode, configuration location and Baud Rate using commands. During the reset condition LED3 will be ON and is turned OFF only after EEPROM reading and configuring is over.

Serial output Value and their corresponding indication

Tx Value	Baud Rate
1	9600
2	19200
3	38400
4	57600
5	96000
6	115200

Eg: 60 => 115200 : Rhydo SSC
21 => 19200 : Mini SSC II

Tx Value	Mode
00	Rhydo
01	Mini SSC-II

➢ Interfacing the board

To communicate with the servo controller, there are two communication protocols. On reset, controller will be configured to the previously selected mode. The user can change the mode using commands. The user should use the command format of the selected mode.

Rhydo Mode: This mode allows the user to access all the features of servo controller such as speed, range, neutral settings. Several controllers can be connected on same serial line and can be controlled independently.
Mini SSC-II Mode: This mode allows the servo controller to respond to the protocol used in Mini SSC-II servo controller made by Scott Edwards Electronics. This is a simple protocol which can specify the desired servo positions in only one way. However, it's not compatible with other serial peripheral products.

MINI SSC-II MODE

Baud Rate: Mini SSC-II will work on the previously configured BD. For changing Baud Rate, the user has to switch it to Rhydo mode.

Protocol: A sequence of three bytes is needed to set the servo position. The first byte must be always a start byte 255 (0xff). Byte 2 is the servo number, and it can be between 0 and 254. Byte 3 is the position to which you want servo to move, can be 1-254. The servo position should not be 0. Giving a 0 as the 3rd byte will change the protocol to Rhydo mode.

Start byte = 0XFF	**Servo number**	**Servo position - 0X01-0XFE**

There are two motion ranges available in this mode. Each controller responds to 16 servo numbers. Addressing lower 8 will move them within an approximately 90 degree range, while addressing upper 8 servo numbers will give twice the range. The user can set the group of servo numbers to which your servo controllers responds. If your servo controller is set to servo number 0, it responds to servo numbers 0-15 and sending the command sequence [255,12, 254] will move servo 4 to one extreme of its range in 180 degree mode. If you send servo numbers that are not recognized, the servo controller will ignore the command. Up to 16 servo controllers can be connected in one serial line to control up to 128 servos independently. By default the servo controller responds to servo numbers 0-15. You can change this numbering as 16-31, 32-47 … 240-254. To set the servo numbers, put the servo controller in Rhydo mode and send the command.

RHYDO MODE

This mode will support several features for servo controlling. To decide a position of a servo we use a range of values from 500 – 5000. One count corresponds to half of a micro second. There are several command formats to change the features. We can individually configure the speed, position, neutral angle, minimum range, maximum range etc… A command for changing the position will turn on a servo automatically. Configuration commands will not turn on a servo.

Protocol: This protocol contains a sequence of 6 bytes. The first byte is the start byte – 0x80. Byte2 is a Device ID which can be used for controlling serially connected servo controllers independently. Byte3 is a command for configuring or controlling the servo controller. Byte4 is the servo number to which the command should be applied. Byte5 and Byte6 are the data values of the given command. In every Byte except start byte, the seventh bit must be clear so the range is 0- 0x7f (0-127).

Start byte=0x80	Device ID	Command	Servo number (1-8)	Data1	Data2

COMMANDS

Value assigning commands

Command 0x01: Position setting (without considering range settings)

This command allows control of the servo position. The range settings will not affect this command. The range of value given should be in between 500 and 5000. The data2 contains lower 7 bits and data1 contains the upper bits. A servo position setting will automatically turns on the servo.

EX: 0X80 0X01(ID) 0X01(COM) SERVO_NUM DATA1 DATA2

Command 0x02: Rotate left (range settings will be considered)

This command allows movement of motor to the left. The data given decides the degree of rotation. If we are giving a degree which will exceeds the limit (minimum range) the motor will rotate up to that range.

EX: 0X80 0X01(ID) 0X02(COM) SERVO_NUM DATA1 DATA2

Command 0x03: Rotate right (range settings will be considered)

This command allows movement of motor to the right. The data given decides the degree of rotation. If we are giving a degree which will exceeds the limit (maximum range) it will rotate up to that range.

EX: 0X80 0X01(ID) 0X03(COM) SERVO_NUM DATA1 DATA2

Command 0x04: To stop a servo

This command will stop the signal to the specified servo.

EX: 0X80 0X01(ID) 0X04(COM) SERVO_NUM 0X00 0X00

Command 0x05: Set to neutral angle

This command will rotate all the enabled servos to its neutral angle. The disabled servos will not turn on while getting this command.

EX: 0X80 0X01(ID) 0X05(COM) 0X00 0X00 0X00

Command 0x06: To stop all servos

This command will stop the signal to all the servos.

EX: 0X80 0X01(ID) 0X06(COM) 0X00 0X00 0X00

Command 0x07: To pause all servos

This command will pause the movement of servo motors. This command will not stop the pulses to the servos but will stop their rotation. The same command should be given again to continue the movement from the same location.

EX: 0X80 0X01(ID) 0X07(COM) 0X00 0X00 0X00

Command 0x08: To ON a specified servo at its neutral angle

This command can turn on a servo at its neutral angle and also can rotate an enabled servo to its neutral angle.

EX: 0X80 0X01(ID) 0X08(COM) SERVO_NUM 0X00 0X00

Configuration Commands

Command 0x11: To configure Neutral angle of a servo

This command allows Neutral angle configuration of each servo independently. The default value of neutral angle is 3000.

EX: 0X80 0X01(ID) 0X11(COM) SERVO_NUM DATA1 DATA2

Command 0x12: To configure minimum angle of a servo

This command will configure the minimum range of each servo independently.

EX: 0X80 0X01(ID) 0X12(COM) SERVO_NUM DATA1 DATA2

Command 0x13: To configure maximum angle of a servo

This command will configure the maximum range of each servo independently.

EX: 0X80 0X01(ID) 0X13(COM) SERVO_NUM DATA1 DATA2

Command 0x14: To configure speed of a servo

This command will configure the speed of each servo independently. The range of value must be 1-127. Value 1 will give maximum speed that is 5 milli sec (pulse width) per second and 127 will give a minimum speed that is 40 microseconds per second.

EX: 0X80 0X01(ID) 0X14(COM) SERVO_NUM 0X00 SPEED

Command 0x15: To configure speed of all servos

This command will configure the speed of all servos simultaneously. The range of value must be 1-127. Value 1 will give maximum speed that is 5ms (pulse width) per second and 127 will give a minimum speed that is 40 microseconds per second.

EX: 0X80 0X01(ID) 0X15(COM) 0X00 0X00 SPEED

EEPROM Read/Write commands

Command 0x20: To write ID to EEPROM

This command is used to write the ID of a servo controller to EEPROM.

EX: 0X80 0X01(ID) 0X20(COM) LOCATION NUMBER 0X00 ID

Command 0x21: To write speed to EEPROM

This command is used to write default speed of all servos to EEPROM

EX: 0X80 0X01(ID) 0X21(COM) LOCATION NUMBER 0X00 SPEED

Command 0x22: To write Range min to EEPROM

This command can be used to write default minimum range of all servos to EEPROM.

EX: 0X80 0X01(ID) 0X22(COM) LOCATION NUMBER DATA1 DATA2

Command 0x23: To write Range max to EEPROM

This command can be used to write default maximum range of all servos to EEPROM.

EX: 0X80 0X01(ID) 0X23(COM) LOCATION NUMBER DATA1 DATA2

Command 0x24: To write Neutral angle to EEPROM

This command can be used to write default Neutral angle of all servos to EEPROM. The 2nd location only can be used, location 1 is only readable.

EX: 0X80 0X01(ID) 0X24(COM) LOCATION NUMBER DATA1 DATA2

Command 0x30: To read default data from EEPROM

This command will read data stored in desired location in EEPROM and will transmit it back.
This command can be used to check the default data given. The locations can be 1 and 2.

EX: 0X80 0X01(ID) 0X30(COM) LOCATION NUMBER 0X00 0X00

Command 0x33: To read and configure default data from EEPROM

This command will read data stored in desired location in EEPROM and will configure the controller.
The locations can be 1 and 2.

EX: 0X80 0X01(ID) 0X33(COM) LOCATION NUMBER 0X00 0X00

ADC read commands

Command 0x40: To read ADC

This command will read the ADC channels. It can be 1, 2 or 3. Channel 1 will give a decimal value corresponds to the servo supply. Channel 2 and 3 will give decimal value corresponds to pins ADC1 and ADC2 respectively.

EX: 0X80 0X01(ID) 0X40(COM) CHANNEL_NUM 0X00 0X00

Note: The example command formats are uses 0x01 as device ID. data1 is the higher bits and data2 is the lower 7 bits of a data.
Ex: The data corresponds to 1500 micro second is 3000

Hex value of 3000 is 0x0bb8 ie 00001011 10111000

The 8th bit of each byte should be 0 so that we are shifting the 8th bit of lower byte to higher byte.

ie 00010111 00111000 or 0x1738

So for rotating 2nd servo of a servo controller having Device ID 0x03, to an angle 90 deg(count 3000), the command format should be as follows

0X80 0X03(ID) 0X01(COM) 0X02 0X17 0X38

ID Independent commands

0x80 0x00 0x00 0x01 0x00 id : To change Device ID

0x80 0x00 0x00 0x02 0x00 id : To transmit the Device id /servo number/ Configuration location selected

(id value should be 1 for Device ID , 2 for servo numbering and 3 for configuration location)

0x80 0x00 0x00 0x03 0x00 BD : To set the baud rate (it can be 1, 2,3,4,5 and 6)

0x80 0x00 0x00 0x04 0x00 0x00 : To switch from the Rhydo Mode to Mini SSC-II mode

0x80 0x00 0x00 0x05 0x00 num : To change Servo numbering

Mini SSC-II Commands

Format: 0xFF servo_number angle

In this mode, the servo number can be 0-254 and angle can be 1-254.

Ex1: 0xFF 3 254 will turn the 4th servo to 90 degree

Ex2: 0xFF 11 254 will turn the 4th servo to 180 degree

Ex3. 0xFF 0 0 will change the mode to Rhydo mode

➢ Position changing

There are two types of position changing commands. One will help you to rotate a servo to a particular angle. This command will not consider range settings of a servo. This command is useful for a user who knows about servo angles. The other one will rotate a servo to a particular degree right or left. It's a relative motion. This command is useful for a user who doesn't know the present angle of a servo. This command will consider range settings of a servo. We can rotate a servo with in the range limit.

➢ Neutral angle settings

Neutral angle of a servo means its default angle. On reset, all the angles will have same neutral angle. But we can configure neutral angle of each servo using commands. Neutral angle setting command helps you to make all the servos to a safe resting position using a single command. Also, we can rotate a servo to its neutral angle individually.

➢ Range settings

Range setting commands helps the user to set safe ranges to each servo. If you are using a left rotation command it will rotate only up to the minimum range of that particular servo and if you are using a right rotation command it will rotate up to the maximum range.

➢ ADC reading

The servo controller gives an option for reading **3** analog values. One is the voltage given to servo motors. This value (0-0xfe) helps the user to check the servo supply for preventing abnormal power supply. 0-5V supply will provide the corresponding digital values (0-0xfe), 5-5.8V will provide a value 0xfe, and any voltage above 5.8, which can cause damage to servos, will out an error value (0xff) to user.

The other two analog (ADC1 and ADC2) pins can be used for user defined applications like sensor input etc.. Controllers having no inbuilt ADC can take advantage of this feature. The output range of these two channels will be between **0 and 0xff** depending on the ADC input voltage.

➢ EEPROM read/write and ADC read

The servo controller will acknowledge for every write by an "OK" only in USART communication. No such acknowledgment will be provided for I2C communication.

➢ I^2C communication (read/write)

To write/read a data to/from servo controller, we have to follow the I2C communication protocol. The slave ID for a servo controller is 0xE0 (7bit addressing).

*The writing format is **start, device_address write** for data write **(0xE0), data write** and **stop.***

*The reading format is **start, device_address write** for data read **(0xE1), data read** and **stop.***

User has to follow this format for each command. For sending a command pattern, we have to write 6 bytes ie we have to start, write and stop 6 times for a command pattern and have to provide sufficient delays (approx 5 millisec or more) between stop and starts. But for reading any data from servo controller, we have to write the servo commands first and then the I2C read command (R/W bit as 1). If there are more than one data to be read, servo controller will out the remaining data one by one for every dummy byte given to it by the master controller. For the last read operation, the user should instruct the master controller for "Not Acknowledge". Then give I2C stop command.

Ex: 0x80, 0x01, 0x30, 0x01, 0x00, 0x00 is an EEPROM read command for servo controller

First write these 6 bytes separately, then out the read command 0xE1 for the result.

The read command 0xE1 and the first dummy byte will give the first data Speed, the second set of dummy data will give range minimum and so on.

That is, for getting the data shown below,
we have to write **0xE1,0x00 0x00,0x00 0x00,0x00 and 0x00,0x00**

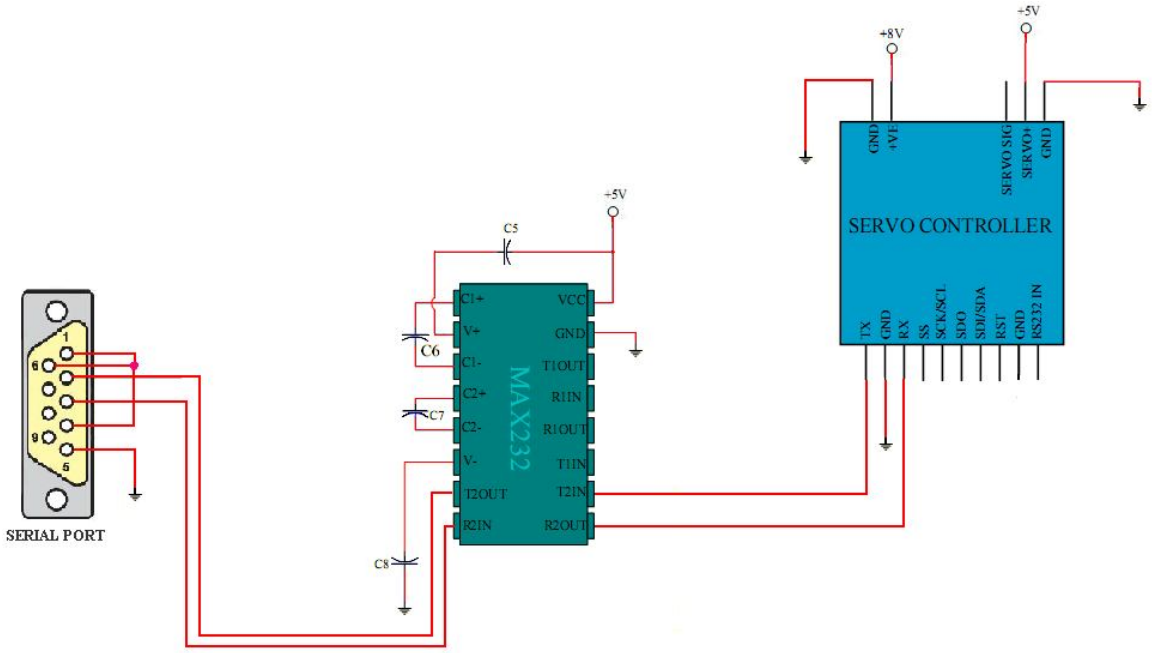
0xE1	0x00	← speed	(Ex: 0x0001)
0x00	0x00	← Range minimum	(Ex: 0x04b0)
0x00	0x00	← Range maximum	(Ex: 0x12c0)
0x00	0x00	← Neutral angle	(Ex: 0x0bb8)

For ADC reading also the user must give an I2C read command 0xE1 after giving the ADC read command for servo controller. The ADC out put is a single byte hex value.

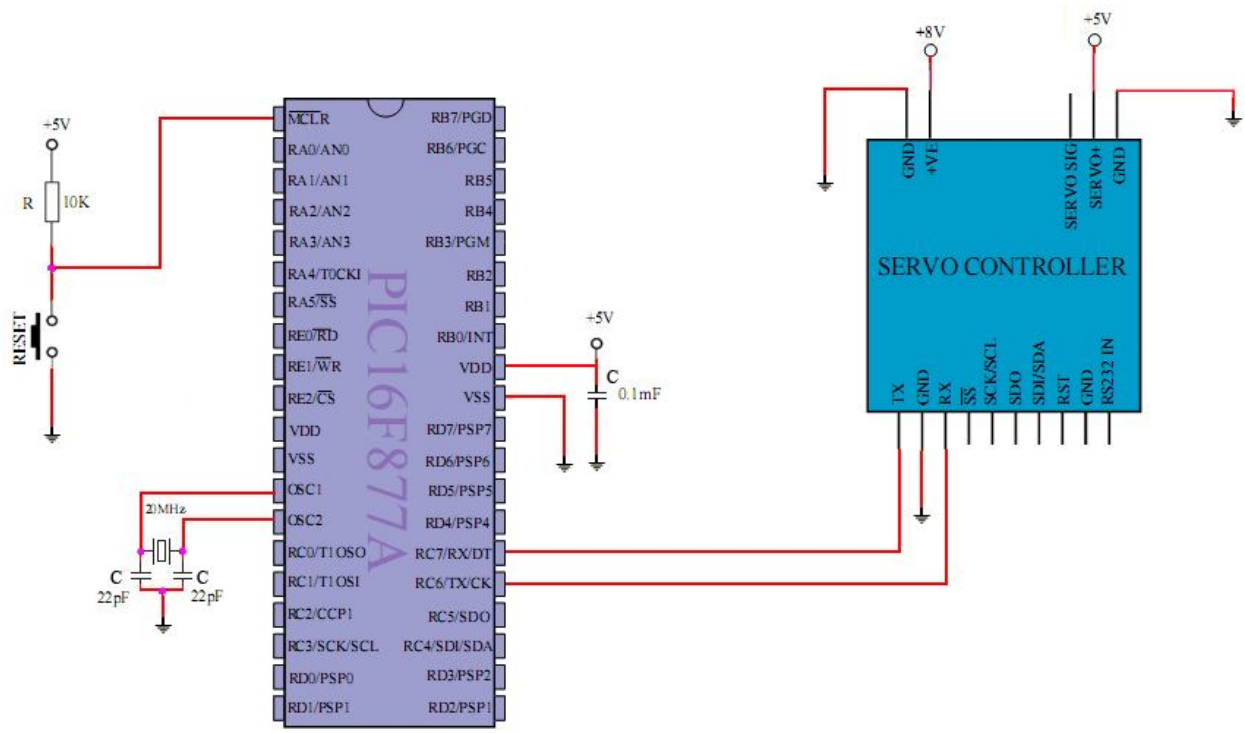
Ex: 0x80 0x01 0x40 0x02 0x00 0x00 and 0xE1 (I2C read command) -- The 6-bytes are command pattern for reading 2nd channel ADC and the last byte is for getting the ADC output.

INTERFACING THE SERVO CONTROLLER

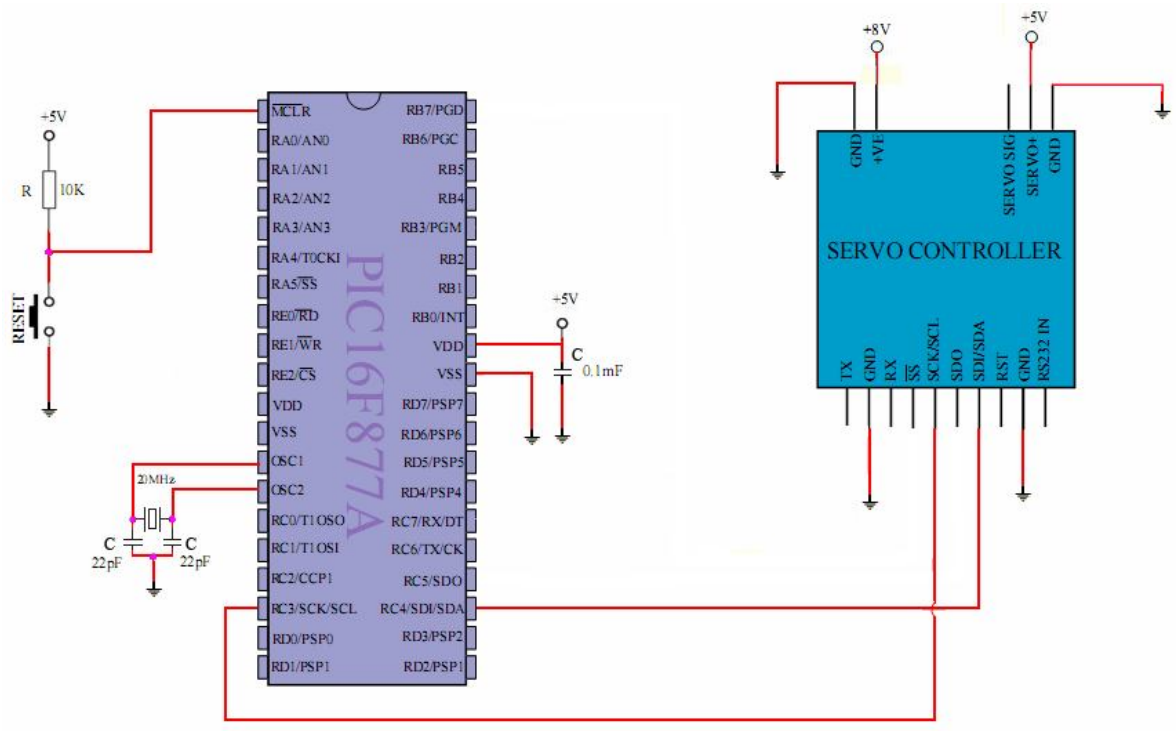
➢ *Interfacing the Servo Controller with PC Serial Port (using MAX 232)*



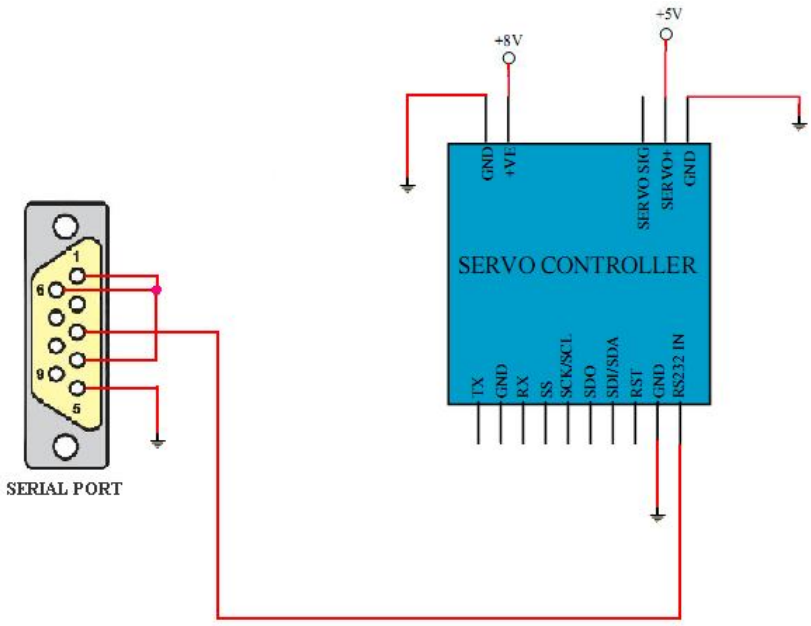
➢ *Interfacing the Servo Controller with PIC 16F877A microcontroller USART module*



➢ *Interfacing the Servo Controller with PIC 16F877 microcontroller I^2C Module.*



➢ *Interfacing the Servo Controller with PC Serial Port (Direct Connection)*



PRACTICAL CONSIDERATIONS FOR USE

i. EEPROM write/read cannot be performed while servos are working.
You should stop all signals to servo or should configure them at the initial stage.

ii. Changing Baud Rate and Mode will stop signals to servo.

iii. The Baud Rate change will be effective only after the board is reset.

iv. In Mini SSC-II mode, zero as the 3rd byte will change the mode

v. Value assigning commands will affect the pulse given to servo

vi. Configuration commands will not affect the pulses. These commands will only configure the motor values. A configuration command will not turn ON a servo.

vii. ID independent commands will turn off pulses to servo.

viii. The user cannot edit the default location 1 in EEPROM. Only location 2 can be edited.

ix. On reset the controller will load the default conditions from the previously selected location by the user.

x. Do not use both RS232 and logical input at a time.
Board can receive both Serial and I2C data simultaneously.

xi. In Rhydo Mode, the servo numbers used should be between 1 and 8 (both inclusive)

xii. A synchronization command 0x55 must be send @ 9600 baud rate or I2C pin to initialize a servo controller.

xiii. In ADC reading, a servo voltage of 5-5.8V will out 0xfe and above 5.8V will out 0xff as an error report

Note: *This product has been tested and certified by the company before shipping. Removing or replacing the components from the PCB could damage the product. In this case, the company won't be liable for the damages caused and no replacement/ refunding are entertained.*

TECHNICAL SUPPORT

If you are experiencing a problem that is not described in this manual, please contact us. Our phone lines are open from 9:00 AM – 5.00 PM (*Indian Standard Time*) Monday through Saturday excluding holidays. Email can be sent to *support@rhydolabz.com*

LIMITATIONS AND WARRANTEES

This product is intended for personal or lab experimental purpose and in no case should be used where it harmfully effect human and nature. No liability will be accepted by the publisher for any consequence of its use. Use of the product software and or hardware is with the understanding that any outcome whatsoever is at the users own risk. All products are tested for their best performance before shipping, still rhydoLABZ is offering One year Free service warranty (Components cost + Shipping cost will be charged from Customer).

DISCLAIMER

Rhydo Technologies (P) Ltd.
(An ISO 9001:2008 Certified R&D Company)
Golden Plaza, Chitoor Road,
Cochin – 682018, Kerala State, India
Phone : 0091- 484-2370444, 2371666
Cell : 0091- 99466 70444
Fax : 0091 - 484-2370579
E-mail : info@rhydolabz.com, sales@rhydolabz.com
WebSite : http://www.rhydolabz.com

rhydolabz.com