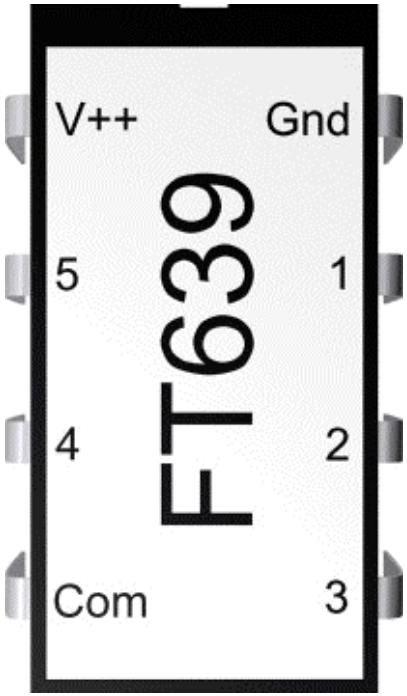
FerretTronics FT639 Servo Controller Chip

Data Sheet



Voltage on V++:	3.0V - 5.5V
Voltage on 2400 Baud In:	< = V++
Serial Line Setup:	2400 Baud, 8 Bit, No parity, and 1 Stop Bit

General Description:

The FT639 is an RC servo controller chip. The FT639 will control five radio-controlled servos through one 2400 baud serial line. It has a footprint of only eight pins. The only external components required are two resistors and a diode for an normal RS232 line such as the one found on a personal computer. No components are needed for a 0-5 volt serial line such as those found on the Parallax Basic Stamp ®. Just connect the servo control lines directly to the chip and connect the serial in line from a 2400 baud, No parity, 1 stop bit serial source, and five RC servos can be controlled, (see circuit setup).

Applications:

Radio control servo motors are used in remote control model airplanes, cars, and boats. They are widely available and can be used in robotics, automation, animation, and many other tasks. The problem with using RC servo motors in the past was the ability to control them. With the FT639 this is no longer a problem. It is possible now to control five RC servo motors with just one 2400 baud serial line. Each of the five RC servos is independently controlled.

Operations:

FT639 has two operating modes: Setup mode and Active mode. The chip starts in Setup mode. Setup mode is used to set the pulse length, header length and starting values for the 5 servos. Active mode sends the control pulses to the servos and controls the servos through the 2400 baud serial line.

Commands are sent to the FT639 through a 2400 baud, 8 bit, no parity, 1 stop bit serial line. The commands are all one byte. Each command is one character sent over the 2400 baud serial line.

Each RC servo has 256 positions. To send the position of a servo to the FT639 requires two commands. The first command contains the servo number and the lower nibble (lower 4 bits) of the positional number. The second command contains the servo number and the upper nibble (upper 4 bits) of the positional number.

The FT639 can set a typical servo in 256 different positions from 0 to 90 degrees with the short pulse length, or can control a typical servo in 256 different positions from 0 to 180 degrees with the long pulse length. The starting position of the servo can also be adjusted by using a different header length. The header length can be adjusted in the setup mode.

Setup Mode:

The servo controller starts in Setup mode. The default settings are the header is approximately 1ms with a short pulse length. This will control a typical servo in 256 steps from 0 to 90 degrees.

In setup mode the following settings can be adjusted:

1. Header length--this will allow adjustment of the starting position of the servo. The default setting is 12.
2. Servo pulse length--this allows positioning control of the servo between 0 to 90 degrees with the shorter pulse length or positioning control of the servo between 0 to 180 degrees with the longer pulse length. The default setting is short pulse length.
3. Initial setup of the servo positions--the FT639 will not send positioning pulses to the servo in Setup mode. However, positioning commands can be sent to the FT639 while in setup mode to allow the servos to energize in a known position. The default setting is position 0.

The following commands can be sent in Setup mode:

Command	Binary Value	Decimal Value
Active Mode	01110101	117
Short Pulse	01010101	85
Long Pulse	01011010	90

The header length command is 0110xxxx, where xxxx is the setting for the header length. The actual length of the header will be different for the different pulse length as shown below:

Header Value	Short Pulse Length	Long Pulse Length	Control Byte	Control Decimal
0	.147 ms	.237 ms	01100000	96
1	.219 ms	.357 ms	01100001	97
2	.291 ms	.477 ms	01100010	98
3	.363 ms	.597 ms	01100011	99
4	.435 ms	.717 ms	01100100	100
5	.507 ms	.837 ms	01100101	101
6	.579 ms	.957 ms	01100110	102
7	.651 ms	1.077 ms	01100111	103
8	.723 ms	1.197 ms	01101000	104
9	.795 ms	1.317 ms	01101001	105
10	.867 ms	1.437 ms	01101010	106
11	.939 ms	1.557 ms	01101011	107
12	1.011 ms	1.677 ms	01101100	108
13	1.083 ms	1.797 ms	01101101	109
14	1.155 ms	1.917 ms	01101110	110
15	1.227 ms	2.037 ms	01101111	111

Active Mode:

In Active mode the servo control pulses are sent to the servos. The servos will be energized in this mode. There are only two commands that are allowed in this mode. Positional commands and the setup command. The setup command puts the FT639 back into Setup mode. The position of a servo can be changed by sending a positional command. The positional commands are sent in Active mode exactly the same as they were in Setup mode (see instructions above). Sending a positional command will make the servo move to the new position as soon as the upper byte command is sent.

The following commands are available in the active mode:

Command	Binary Value	Decimal Value
Setup Mode	01111010	122

Positional Commands:

To send a positional command to the individual servos, two bytes must be sent. The first byte sent contains the lower nibble of the position byte and the second byte sent contains the upper nibble of the position byte. The lower byte command must be sent before the upper byte command. The format for the bytes are:

```
Lower Byte = 0sssxxxx

Upper Byte = 1sssyyyy

sss = Servo number:
```

000 = servo 1

001 = servo 2

010 = servo 3

011 = servo 4

100 = servo 5

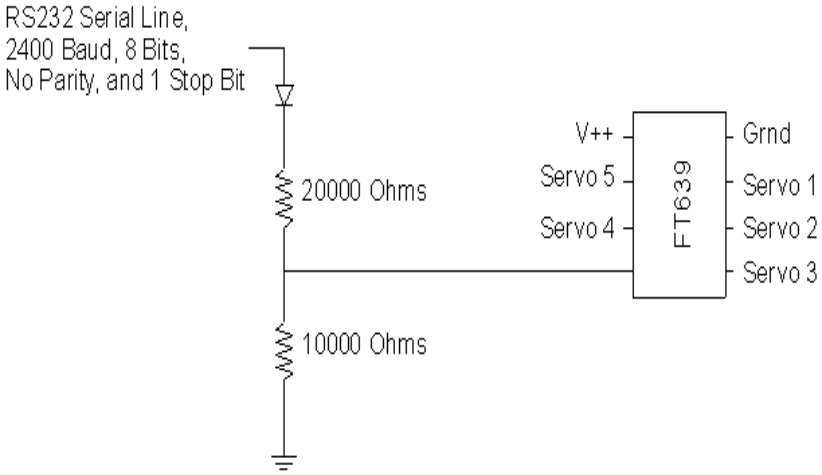
xxxx = the lower nibble of the position byte

yyyy = the upper nibble of the position byte

A table is shown below with the Lower and Upper Byte for various positional commands:

			Binary Value			
	Position Value				Decimal Value	
Servo	Decimal Value	Binary Value	Lower Byte (0sssxxxx)	Upper Byte (1sssyyyy)	Lower Byte	Upper Byte
1	0	00000000	00000000	10000000	0	128
1	49	00110001	00000001	10000011	1	131
1	185	10111001	00001001	10001011	9	139
1	255	11111111	00001111	10001111	15	143
2	0	00000000	00010000	10010000	16	144
2	49	00110001	00010001	10010011	17	147
2	185	10111001	00011001	10011011	25	155
2	255	11111111	00011111	10011111	31	159
3	0	00000000	00100000	10100000	32	160
3	49	00110001	00100001	10100011	33	163
3	185	10111001	00101001	10101011	41	171
3	255	11111111	00101111	10101111	47	175
4	0	00000000	00110000	10110000	48	176
4	49	00110001	00110001	10110011	49	179
4	185	10111001	00111001	10111011	57	187
4	255	11111111	00111111	10111111	63	191
5	0	00000000	01000000	11000000	64	192
5	49	00110001	01000001	11000011	65	195
5	185	10111001	01001001	11001011	73	203
5	255	11111111	01001111	11001111	79	207

Circuit Diagram:



Note: For a serial line that has voltage from 0 to V++ requires no diode resistor network. The line can be connected directly to the serial-in pin on the FT639.

Example Code:

Other programming examples can be found at:
http://www.ferrettronics.com/software.html

```
'########################################
'# This is a QBASIC programming example
'# For controlling the FT639
'########################################

DECLARE SUB servoMove (servoNum!, value!)
DECLARE SUB servo1 (value AS INTEGER)
DECLARE SUB servo2 (value AS INTEGER)
DECLARE SUB servo3 (value AS INTEGER)
DECLARE SUB servo4 (value AS INTEGER)
DECLARE SUB servo5 (value AS INTEGER)

CONST ACTIVE = 117
CONST LONGPULSE = 90
CONST SHORTPULSE = 85
CONST HEADER = 96
CONST SETUP = 122

' Opens COM Port 1 for sending out serial commands
OPEN
"COM1:2400,N,8,1,CD0,CS0,DS0,OP0,RS,TB2048,RB2048"
FOR RANDOM AS #1
' This command will put the FT639 in the setup
mode
PRINT #1, CHR$(SETUP);
' This command will put the FT639 in the long
pulse mode
PRINT #1, CHR$(LONGPULSE);
' This command will put the FT639 in the Short
pulse mode
'PRINT #1, CHR$(SHORTPULSE);
' This command will set the header at 3
PRINT #1, CHR$(HEADER + 3);
' This command will put the FT639 in the active
mode
PRINT #1, CHR$(ACTIVE);

'-----------------------------------
' Loop to cycle through all positions
'-----------------------------------
FOR i = 0 TO 255

' Cause a delay
FOR J = 1 TO 100000
NEXT J

' Moves the servos through all positions
```

```
servo1 (I)
servo2 (I)
servo3 (I)
servo4 (I)
servo5 (I)
NEXT i

'------------------
' Positions servo 1
'------------------
SUB servo1 (value AS INTEGER)
DIM uV AS INTEGER
DIM lV AS INTEGER
uV = INT(value / 16)
lV = value - (uV * 16)
uV = uV + 128
PRINT #1, CHR$(lV);
PRINT #1, CHR$(uV);
END SUB

'------------------
' Positions servo 2
'------------------
SUB servo2 (value AS INTEGER)
DIM uV AS INTEGER
DIM lV AS INTEGER
uV = INT(value / 16)
lV = value - (uV * 16)
uV = uV + 128 + 16
lV = lV + 16
PRINT #1, CHR$(lV);
PRINT #1, CHR$(uV);
END SUB

'------------------
' Positions servo 3
'------------------
SUB servo3 (value AS INTEGER)
DIM uV AS INTEGER
DIM lV AS INTEGER
uV = INT(value / 16)
lV = value - (uV * 16)
uV = uV + 128 + 32
lV = lV + 32
PRINT #1, CHR$(lV);
PRINT #1, CHR$(uV);
END SUB

'------------------
' Positions servo 4
'------------------
SUB servo4 (value AS INTEGER)
DIM uV AS INTEGER
DIM lV AS INTEGER
uV = INT(value / 16)
lV = value - (uV * 16)
uV = uV + 128 + 48
lV = lV + 48
PRINT #1, CHR$(lV);
PRINT #1, CHR$(uV);
END SUB

'------------------
' Positions servo 5
'------------------
SUB servo5 (value AS INTEGER)
DIM uV AS INTEGER
DIM lV AS INTEGER
uV = INT(value / 16)
lV = value - (uV * 16)
uV = uV + 128 + 64
lV = lV + 64
PRINT #1, CHR$(lV);
PRINT #1, CHR$(uV);
END SUB

'------------------------------------------------
-----------
' Positions any servo given servo number and
positional value
'------------------------------------------------
-----------
```

```
SUB servoMove (servoNum, value)
DIM uV AS INTEGER
DIM lV AS INTEGER
uV = INT(value / 16)
lV = value - (uV * 16)
uV = uV + 128 + (servoNum - 1) * 16
lV = lV + (servoNum - 1) * 16
PRINT #1, CHR$(lV);
PRINT #1, CHR$(uV);
END SUB
```