Hitec HSR-8498HB Digital Servo Operation and Interface

Revision 0.5

Updated 24th January 2007

Introduction

This is based on the data gathered from the HSR 8498HB, both physical and in operation

Also includes data provided by Hitec in their servo specification and their description of the operation of the HMI protocol.

Inside the HSR 8498HB Digital Servo

The HSR-8498HB has an ATmega8 processor.

Processor

The HS-5475HB servo is based on the Atmel ATmega8 processor which has:

2.7V to 5.5V
8K Flash self reprogrammable
1024 bytes RAM
512 bytes EEPROM
UART
10 bit ADC
4 MHz Clock

The processor data sheet is available on the Atmel web site:

http://www.atmel.com/dyn/resources/prod_documents/doc2486.pdf

Looking at the processor via the SPI programming interface show it has a boot loader section located at 0x0E000. Therefore we might assume the firmware in the flash is upgradeable.

- ATmega8 Configuration:
- 512 Boot Flash at 0x0E000, Boot Vector Enabled
- Brown Out at 2.7V, Brown Out enabled
- Mode 3 further program and verify disabled
- No lock on LPM/SPM in app, Lock on LPM in boot

Mechanics

The mechanics are conventional, motor, gear-train, and feedback pot (4K7Ohm).

The gear train is Karbolite, and has a ratio of about 300:1 between motor and output.

Physical end stops are on output shaft at just over 190 degrees

Mechanical specs (torque, speed, etc.) are on Hitecrcd web page:

http://www.hitecrcd.com
And elsewhere.

To remove the servo PCB it is necessary to de-solder the 3 connections to the base of the motor. This de-soldering would also be necessary to replace the case, or to move the cable from one side of the case to the other.

Motor

The motor has a DC resistance of 4.4 ohms, which gives a maximum current draw of 1.35A at 6 Volts, or 1.7A at 7.4 Volts supply

The motor is glued to the servo case, but may be removed is sufficient pressure is applied.

Motor Driver

The motor is driven by a pair of International Rectifier IRF7389 Complimentary MOSFETS in Bridge configuration.

Datasheet: http://www.irf.com/product-info/datasheets/data/irf7389.pdf

The Bridges are driven from transistors and linked together. This has the unfortunate situation that the bridge cannot be turned completely off. So the motor is either driven or braked while power is applied to servo. No current or temperature monitoring appears to be performed on motor.

Control Interface

The control interface differs from earlier Hitec digital servos, in that it provides a true open collector output. This bidirectional control allows the servos to be daisy chained.

The control interface is connected to the processor Port D0 and Port D1 pins which are also the Uart TX and RX pins. The Port B0/ICP pin is also connected to the UART RX pin, and likely measures the pulse width in PWM mode

The control interface to the processor is made using what appears to be a dual digital NPN transistor. The control signal is bi-directional and requires an external pull-up.

Other bits

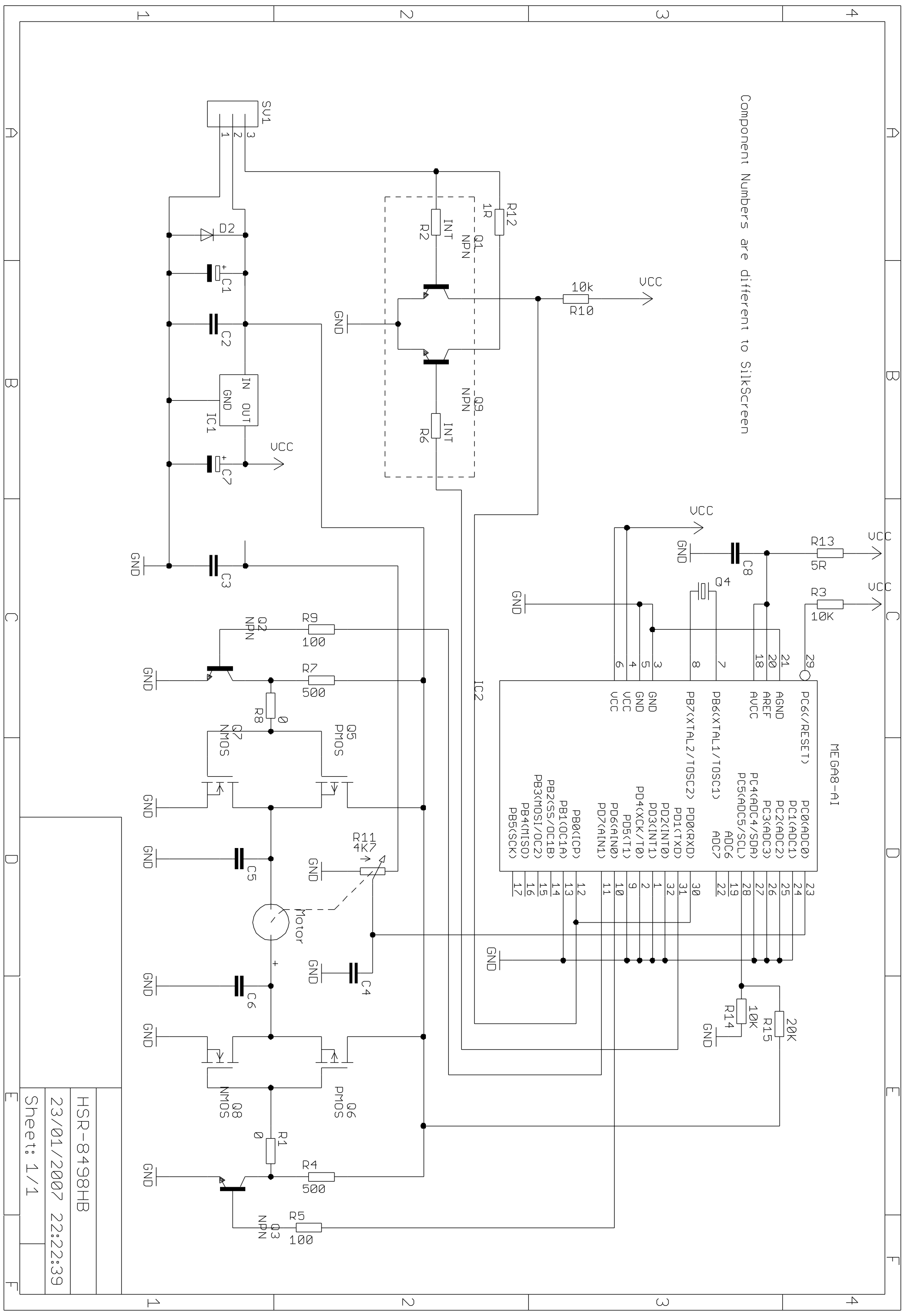
Processor power is regulated to 3.1 Volts by regulator IC.

4K7 Feedback pot feeds to ADC input on processor between Vref and ground.

The processor measures the servo supply voltage through a simple potential divider

There is no immediate evidence that the processor can actually measure current. This capability is described in some specification documents and the current may be read via the HMI protocol. It may be that the current is calculated from the PWM applied to the servo drive, maybe taking into account the position and the supply voltage.

Schematic


Component Numbers are different to SilkScreen
HSR-8498HB
23/01/2007 22:22:39
Sheet: 1/1
SU1
D2
C1
C2
C3
C4
C5
C6
C7
C8
IC1
IN
OUT
GND
IC2
MEGA8-AI
Q1 NPN
Q2 NPN
Q3 NPN
Q4
Q5 PMOS
Q6 PMOS
Q7 NMOS
Q8 NMOS
Q9 NPN
R1 0
R2 INT
R3 10K
R4 500
R5 100
R6 INT
R7 500
R8 0
R9 100
R10 10k
R11 4K7
R12 1R
R13 5R
R14 10K
R15 20K
Motor
VCC
GND

Hitec HMI Interface

The Hitec Multi-protocol Interface has been described in several documents from Hitec, but never in a complete or consistent way.

Multi-Protocol presumably describes the ability to work in 3 modes:

- Standard Pulse servo mode compatible with radio control servos

- Extended Pulse mode to add position feedback and change settings
  Described:
  http://www.robonova.de/store/support/index.php?_m=downloads&_a=viewdownload&downloaditemid=33&nav=0,3

- Bidirectional Serial Interface at 19200 bps
  Described:
  http://www.robonova.de/store/support/index.php?_m=downloads&_a=viewdownload&downloaditemid=94

The width of pulses sent to the servo determines the mode:

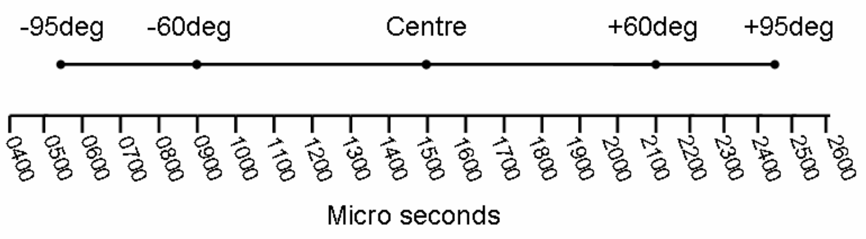
Pulse width 550 to 2450 Microseconds = Standard Pulse Mode
Pulse width 50 to 200 Microseconds = Extended Pulse Mode
Pulse width 416 Microseconds = Serial Mode

1. Servo in Standard Pulse Mode

Moving the Servo

The servo responds to PWM type signals in the range 550• S to 2450• S duration. Since the servo is digital a single pulse is sufficient to move the servo to the desired position. Pulses outside this range do not move the servo.

The servo midpoint is at 1500• S. In standard operation pulses from 900• S to 2100• S move the servo through an angle of about 120 degrees. Using the full range of 550• S to 2450• S an overall operating angle of 190 degrees can be obtained.



Servo Angle (degrees) = (Pulse Width (• S) – 1500) / 10

2. Servo in Extended Pulse Mode

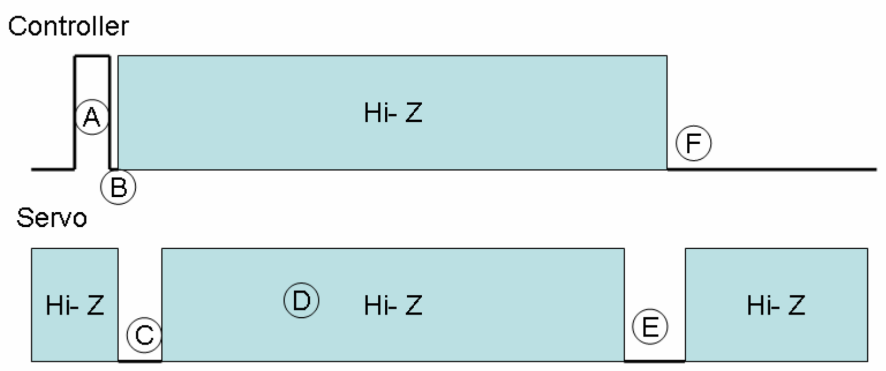
Position Feedback

Hitec have provided some information on the HMI position feedback protocol in their datasheet. This mode is used by the RoboNova for Position feedback

The first thing to note is that the 50• S pulse which initiates the position feedback also disables the motor drive. This means that reading the position interrupts the motion, and so reduces speed and torque that can be achieved by the servo. This also explains why Robonova jerks when the position is read, and joints under load sag as position is read continuously.

The advantage of having the servo motion turned off during position feedback is that the servo can be moved by hand during this time.

A more detailed description of the position feedback based on observed operation is:



- (A) Controller issues 50 microSecond pulse (+ve or Hi-Z)
- (B) Controller returns low and holds for 2 to 20 microseconds (see below)
- (C) Servo recognises pulse and sets low for 125 microseconds
- (D) Servo sends position pulse (550 to 2450 microseconds)
- (E) Servo sets low for 250 microseconds
- (F) Controller recognises end of sequence and holds low again

The timing B above needs to hold the control signal low for at least 2• S to be sure to be recognised. The servo will set low about 10• S after recognition. To avoid a glitch on the control line it is best to wait 10• S after the pulse before the controller goes Hi-Z.

After reading the position the servo motion must be reinstated using the PWM signal.

The impact of the removal of motor power during position feedback has a significant impact on suitability for use for closed loop feedback. The entire position feedback cycle is 2900• S then we have up to 2500• S to re-establish the motion. Therefore reading the position every 10mS would reduce the speed and torque by over 50%.

Parameter changes

The servo is decsribed to accept parameter changes by receipt of pulses of 100• S (default Parameter Set 1), 150• S (Parameter Set 2), and 200• S (Parameter Set 3).

No discernable difference was found during testing of these different settings**.**

The HMI protocol description describes the ability to change the P gain, D gain and dead band, so presumably they are used then.

Similarity of the protocol to that of the ICS servo suggests the types of parameter which may be changed.

http://www.kondo-robot.com/software/SVM2346R.EXE

http://www.kondo-robot.com/pdf/ServoManagerfor2346REDManual.pdf

3. Servo in Serial Mode

Electrical Interface

As described above and shown in the circuit diagram the interface is open collector. Pulses to the servos are positive going pulses using an external pull-up.

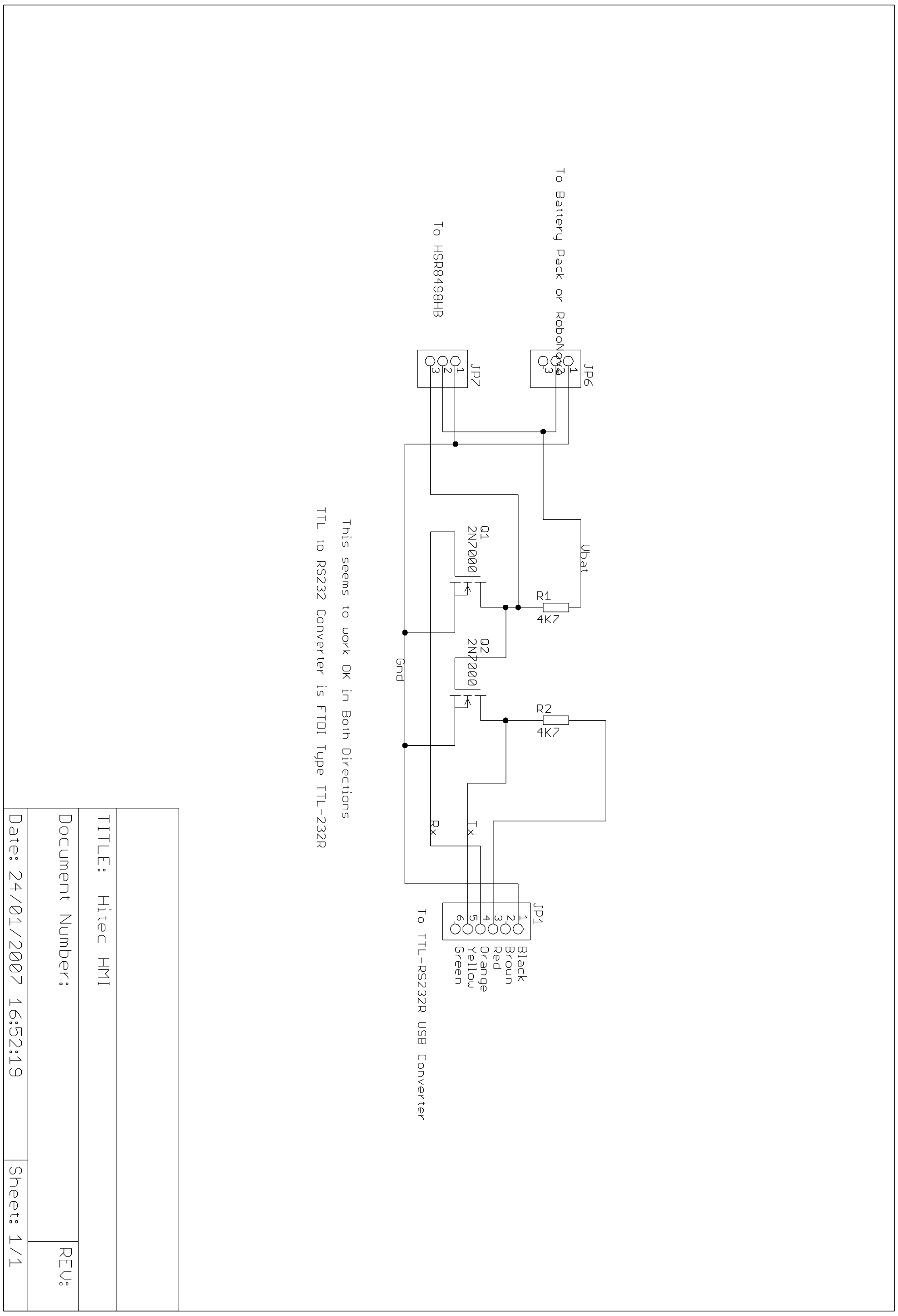
When operating in Serial Mode the serial data is not inverted:

Idle State = Stop Bit = “1” Bit = logic low = Ground

Start Bit = “0” Bit = Logic high = +5V via pull-up

To Interface to a PC a non inverting interface from RS232 levels to ttl levels is required.

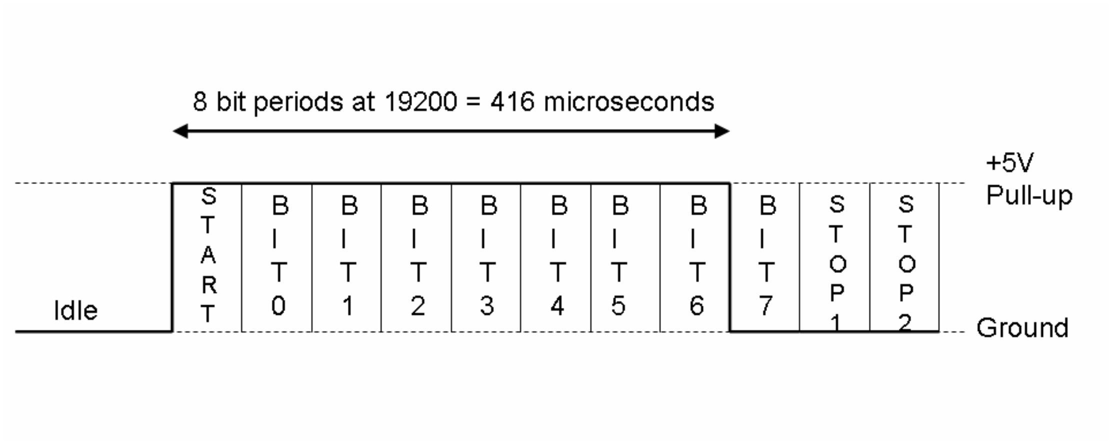
The following example uses an FTDI Model TTL-232R USB to TTL cable and a pair of MOSFET to invert and covert to one- wire.


To HSR8498HB
To Battery Pack or RoboNova
JP7
JP6
1
2
3
Q1
2N7000
Q2
2N7000
R1
4K7
R2
4K7
Vbat
Gnd
Rx
Tx
JP1
Black
Brown
Red
Orange
Yellow
Green
To TTL-RS232R USB Converter
This seems to work OK in Both Directions
TTL to RS232 Converter is FTDI Type TTL-232R
TITLE: Hitec HMI
Document Number:
REV:
Date: 24/01/2007 16:52:19
Sheet: 1/1

Frame Level Interface

Serial data is interchanged at 19200 bits per second, 8 bits per character, 2 stop bits, no parity.

The first byte sent from the controller in a command is the header or packet start byte of 0x080. This looks like:



A complete serial Packet is 7 characters long = 7 * (8 +3) bits = 77 bits = 4.03 milliseconds.

Byte 1 = Header (0x080) Packet start

Byte 2 = Command

Byte 3 = Data 1

Byte 4 = Data 2

Byte 5 = Checksum

Byte 6 = Received 1

Byte 7 = Received 2

The checksum is described in the HMI description as Byte 1 + Byte 2 + Byte 3 + Byte 4. This in not correct!

The checksum is actually 256-(Byte 1 + Byte 2+ Byte 3 + Byte4)

To enable the received Bytes to work ‘0x00’ characters must be sent from the controller to disable drive. The servo will then drive the required bits to zero to make the response. Therefore the controller sends 7 characters with the last two as ‘0x00’. Since the bus is bidirectional the serial receiver will receive all 7 bytes.

Command Interface

The currently known commands from Hitec are given below. We expect more when the serial cable is released.

A) Read version and ID

Byte 1 = Header = 0x080

Byte 2 = Command = 0xE7

Byte 3 = Data 1 = 0x00

Byte 4 = Data 2 = 0x00

Byte 5 = Checksum = 256 – (Byte 1+ Byte2 +Byte3+Byte4)

Byte 6 = Received Byte 1 = Version

Byte 7 = Received Byte 2 = ID

B) Read Current and Voltage

Byte 1 = Header = 0x080

Byte 2 = Command = 0xE8

Byte 3 = Data 1 = 0x00

Byte 4 = Data 2 = 0x00

Byte 5 = Checksum = 256 – (Byte 1+ Byte2 +Byte3+Byte4)

Byte 6 = Received Byte 1 = Current (0 to 255)

Byte 7 = Received Byte 2 = Voltage (0 to 255)

C) Set Motor Speed and Read Position

Byte 1 = Header = 0x080

Byte 2 = Command = 0xE9

Byte 3 = Data 1 = Servo ID (0x00 to 0x7F) : use 0x00

Byte 4 = Data 2 = Speed (0x00 to 0xFF)

Byte 5 = Checksum = 256 – (Byte 1+ Byte2 +Byte3+Byte4)

Byte 6 = Received Byte 1 = Position High

Byte 7 = Received Byte 2 = Position Low

The position is given by ((Position High << 8) + Position Low). Value is in milliseconds of duration of pulse (1500 = centre). Error between measured and desired position may be due to dead-band.

D) Set Servo Position

Byte 1 = Header = 0x080

Byte 2 = Command (0x00 to 0x7F) = Servo ID

Byte 3 = Data 1 = Position High

Byte 4 = Data 2 = Position Low

Byte 5 = Checksum = 256 – (Byte 1+ Byte2 +Byte3+Byte4)

Byte 6 = Received Byte 1 = 0x00

Byte 7 = Received Byte 2 = 0x00

The position is given by ((Position High << 8) + Position Low). Value is in milliseconds of duration of pulse (1500 = centre).

E) Set Motor Go / Stop

Byte 1 = Header = 0x080

Byte 2 = Command = 0xEB

Byte 3 = Data 1 = 0x00

Byte 4 = Data 2 = Stop(0) / Go(1)

Byte 5 = Checksum = 256 – (Byte 1+ Byte2 +Byte3+Byte4)

Byte 6 = Received Byte 1 = 0x03

Byte 7 = Received Byte 2 = 0x03

I cannot make this command work !

That is all the command that Hitec have given to us. It cannot be all, since we cannot set the ID, and the commands related to setting the P gain, D gain, and dead-band are not there.

Hitec have given Visual Basic code for the HMI Interface:

http://www.robonova.de/store/support/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=5

Here is a complete VB.Net program to demonstrate the HMI interface. This can be compiled with the 2005 Express Edition :

http://robosavvy.com/Builders/i-Bot/HMITest.zip

Please send any corrections, new insights or comments, to me via Robosavvy forum or direct to richard.ibbotson@btinternet.com so I can incorporate them.