Hitec Digital Servos Operation and Interface

Revision 0.4

1st June 2006

Introduction

This is based on the data gathered from the HFP-10 and the following servos:

HS-5475HB (Firmware Version 1.03)

HS-5245MG (Firmware Version 1.04)

HSR-5995TG (No Firmware Version given)

HS-5645MG (Firmware Version 1.04)

This document does not cover the HSR 8498HB. This robot servo is quite different. It is not compatible with HFP-10. It uses a different electrical interface to support multi-drop bus operation.

Inside the Hitec Digital Servo

This is mainly based on the HS-5475HB. The HS-5245MG looks similar, but being smaller the circuit is on two sandwiched boards, so I can't be sure.

The HSR-5995TG and the HS-5645MG have an ATmega8 processor instead of the AT90LS4433. The processors are pin compatible.

Processor

The HS-5475HB servo is based on the Atmel AT90LS4433 processor, which has:

2.7V to 6V
4K Flash not reprogrammable in servo
128 bytes RAM
256 bytes EEPROM
UART
10 bit ADC
4 MHz Clock

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

http://www.atmel.com/dyn/products/product_card.asp?part_id=1998

I didn't measure the clock; I think it is probably 4MHz

The HSR-5995TG and HS-5645MG are based on the 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

There does not appear to be any reprogramming of Flash by the HFP10 at least in version 1.02 software

Mechanics

The mechanics are conventional, motor, gear-train, and feedback pot (4K7Ohm). I didn't count the gear ratio.

Obviously the gears are of different material

Surprisingly the same pot seems to be used on all servos, and the quality while maybe acceptable on a $40 servo, is low for a $150 servo

Physical end stops are on output shaft at just over 180 degrees (198 degrees I think)

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

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

Motor Driver

The motor is driven by a pair of Vishay Siliconix Si9958 Complimentary MOSFETS in Bridge configuration.

http://www.vishay.com/doc?70141

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 is performed on motor.

The MOSFETS heat up pretty quick under stall conditions, and are close to their maximum ratings on the higher power servos.

Control Interface

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 input is protected by a 1K resistor and what appears to be a 4V7 zener on the input. The control then goes to what appears to be the emitter of an NPN transistor. The collector goes to the RX input of the processor, and the base is driven by the processor TX through a 10K resistor.

When driven in PWM mode the swing on control is 0 to 4.8 Volts

For serial mode a light pulldown (I used 47K), is needed. The control input is pulled up by the transistor. Too high a value for pulldown prevents a good low, too low reduces the high from the servo.

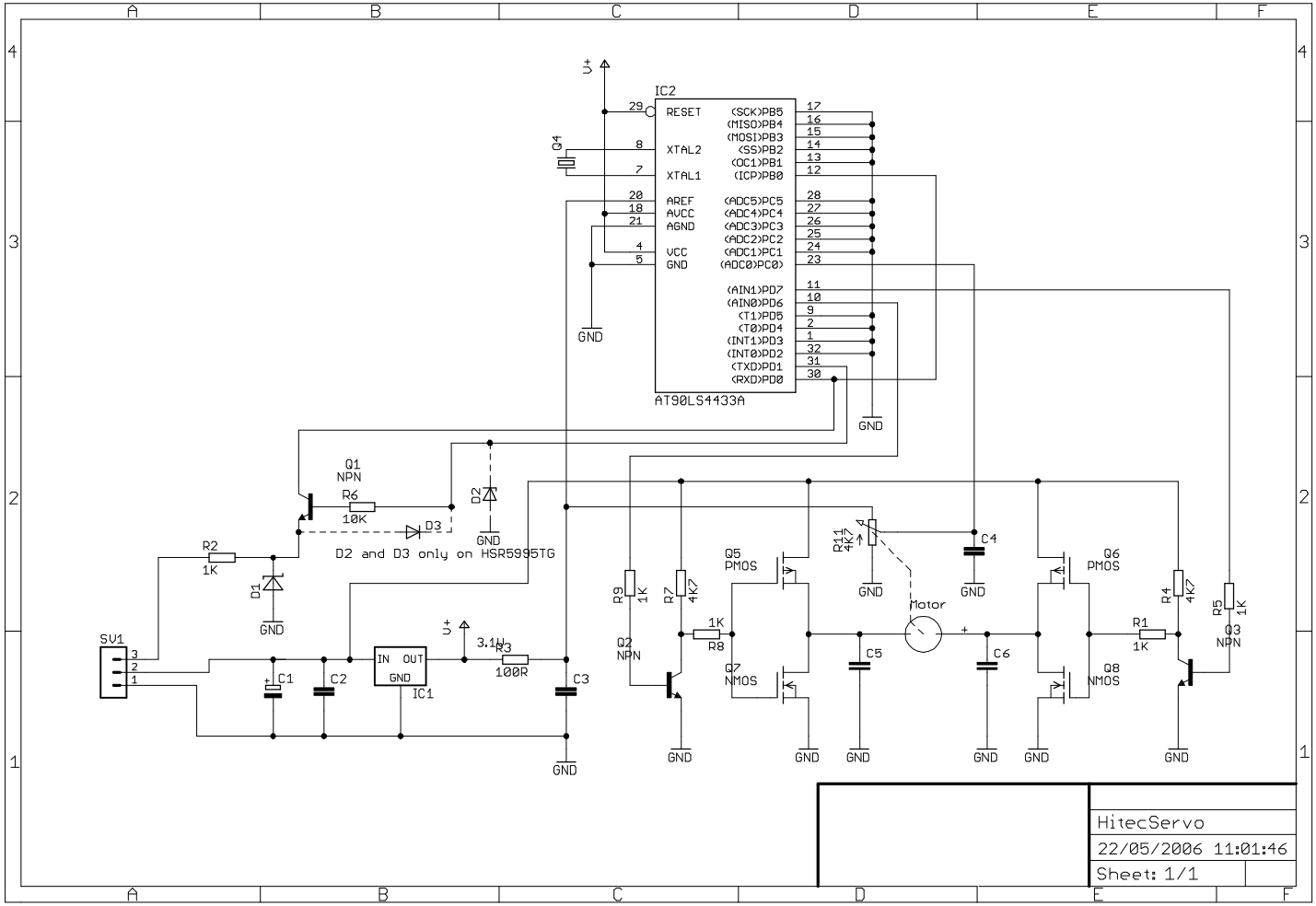
The HSR-5995TG has a couple of extra diodes on the TX pin; this may be for higher speed operation in HMI.

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.

Schematic


IC2
AT90LS4433A
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
(SCK)PB5
(MISO)PB4
(MOSI)PB3
(SS)PB2
(OC1)PB1
(ICP)PB0
(ADC5)PC5
(ADC4)PC4
(ADC3)PC3
(ADC2)PC2
(ADC1)PC1
(ADC0)PC0
(AIN1)PD7
(AIN0)PD6
(T1)PD5
(T0)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
Q4
U+
GND
Q1
NPN
R6
10K
D3
D2
D2 and D3 only on HSR5995TG
R2
1K
D1
SV1
C1
C2
IC1
IN OUT
GND
3.1V
R3
100R
C3
R9
1K
R7
4K7
Q2
NPN
1K
R8
Q5
PMOS
Q7
NMOS
R11
4K7
C4
C5
C6
Motor
Q6
PMOS
Q8
NMOS
R4
4K7
R5
1K
R1
1K
Q3
NPN
HitecServo
22/05/2006 11:01:46
Sheet: 1/1

Servo programmer HFP-10

I didn't spend too much time looking at the programmer. Mine has version 1.02 software.

I made a simple board with two 3 pin headers, and a link on the control pin, to enable isolation.

I used a Tek 318 logic analyser in serial mode on the control Pin. To determine if it was the servo or the HFP sending data, I put fine wires direct to the processor RX and TX pins on the servo.

Servo in PWM Mode

I didn't check out the servo in PWM mode.

I would like to know how the servo reacts to pulses outside the 900 – 2100 uSec range. Do they respond to 800, as if 900, or not at all?

Also do the pulses in the HMI document have any effect?
http://www.hitecrobotics.com/Tony%20information/HMI%20Protocol.pdf

The HSR-5995TG says position feedback on the box

Servo Electrical Interface in Serial Mode

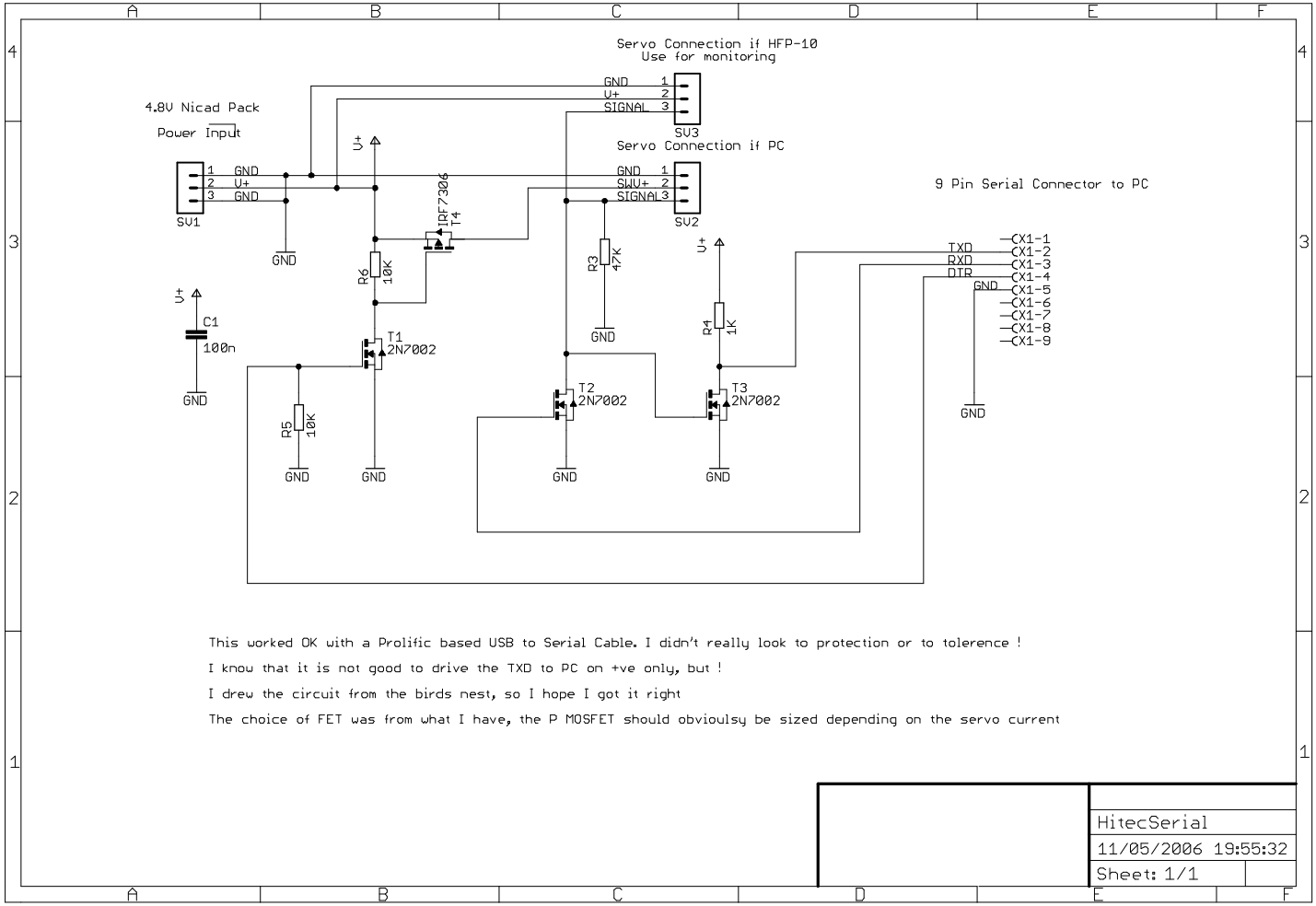
Electrical

I used the following interface from the PC.

This pulls down the control input, allows the PC to send, receives data (including what is sent), and controls the power up to the servo.

State of control input on power up seems to determine the mode (PWM or serial). I do not know if you can send PWM in serial mode. I do know you have to follow the right sequence to get into serial mode.

The data is inverted (Space = High), so can drive a PIC direct on RXD. The TXD must be open collector or open source driver, which inverts, so best to use a processor where TXD can be inverted.


Servo Connection if HFP-10
Use for monitoring
GND 1
U+ 2
SIGNAL 3
SV3
4.8V Nicad Pack
Power Input
Servo Connection if PC
1 GND
2 U+
3 GND
SV1
GND 1
SWU+ 2
SIGNAL3
SV2
9 Pin Serial Connector to PC
U+
IRF7306
T4
R6
10K
R3
47K
U+
TXD
RXD
DTR
GND
CX1-1
CX1-2
CX1-3
CX1-4
CX1-5
CX1-6
CX1-7
CX1-8
CX1-9
C1
100n
T1
2N7002
R4
1K
T2
2N7002
T3
2N7002
R5
10K
GND
GND
GND
GND
GND
GND
GND
GND
This worked OK with a Prolific based USB to Serial Cable. I didn't really look to protection or to tolerence !
I know that it is not good to drive the TXD to PC on +ve only, but !
I drew the circuit from the birds nest, so I hope I got it right
The choice of FET was from what I have, the P MOSFET should obvioulsy be sized depending on the servo current
HitecSerial
11/05/2006 19:55:32
Sheet: 1/1

Serial data configuration

The serial data is at 19.2K Async 8 bits 1 stop

Getting into Serial Mode

When the servo powers up with only a light pulldown (47K) on the control input, after about 12mS the servo sends a "+" character and pulls up the control input. You must respond to this "+" by sending back a "+". This has to be done within a specific time (needs to be measured).

A couple of seconds after you send the "+", the servo sends another "+". You are then in serial communication.

Serial commands are ignored unless this start-up is performed. This is one of many locks and checks in the servo software to prevent corruption of the configuration.

Command Format

Commands seem all to be of the 4 byte format

Command Type	Address	Data	Checksum

When address and data are not required, the servo just seems to ignore them.

Response

Response 1	Response 2

An acknowledgment response is a "++", a byte response is BYTE"+", and a word response is WORD only.

The checksum is

Command Type + Address + Data + Checksum = 0 (Mod 256)

There is no checksum on the servo response. The checksum is for protection of the servo configuration, not for serial errors.

A bad checksum is responded with a "-", and seems to drop out of serial mode, but I haven't really checked this.

Command Types

Command Type Character	Hex	Command Function	Addr	Data	Response 1	Response 2
a	61	Read EEPROM	Addr	n/u	Data Byte	+
b	62	Write EEPROM	Addr	Data	+	+
c	63	Read Data Memory	Addr	n/u	Data Byte	+
d	64	Write Data Memory	Addr	Data	+	+

e	65	Move Servo	Data High	Data Low	+	+
f	66	Read Position	n/u	n/u	Data High	Data Low

EPROM Commands

The “a” and “b” commands read or write to the 256byte EEPROM respectively. The HFP-10 only ever seems to access the locations 0 through 0x1C for writing, and the locations 0 through 0x1C, and 0x30 through 0x4C for reading. The second set of locations seems to be very much the same as the lower set. The second set may be the factory defaults, or maybe a second configuration.

The location 0x1C appears to be a checksum of the bytes from 0 through 0x1B, so the bytes add up to 0 (mod256). Therefore location 0x1C must also be modified for all changes. If the EEPROM Checksum is not correct the servo does not operate on power up (does not respond to PWM). When connected to the HFP-10 with a bad checksum the HFP-10 displays “Fail”, and resets the EEPROM to default values.

It is not apparent how the higher locations are addressed on the ATMega8

EEPROM Locations

EEPROM Address	Function	Reset Default HS-5475HB	Reset Default HS-5245MG	Reset Default HS-5645MG	Reset Default HSR-5995TG
00		78	50	50	2B
01		DC	BE	B4	B4
02	Dead Band	02	02	02	02
03		64	96	96	46
04		01	01	01	01
05		05	06	09	05
06	Speed	40	40	40	40
07	Neutral High	0F	0F	0F	0F
08	Neutral Low	A0	A0	A0	A0
09		0D	0D	0B	0D
0A		48	48	B8	48
0B		21	21	23	21
0C		98	98	28	98
0D		00	00	00	00
0E		38	39	0A	46
0F		03	03	03	03
10		C9	AF	E8	C1
11		17	17	17	17
12		CC	CC	D0	CC
13	Endpoint Left	70	75	74	F3

14	Endpoint Right	70	75	74	F3
15		03	03	03	03
16		E8	E8	E8	E8
17		03	03	03	03
18		E8	E8	E8	E8
19	Failsafe Point High	17	17	17	17
1A	Failsafe Point Low	CC	CC	70	CC
1B	Clockwise(bit 0)/Failsafe Enable(bit 1)	01	01	01	01
1C	EEPROM Checksum (00 to 0x1B)	BA	E4	3B	10
1D		08	06	04	0E
1E		FF	FF	FF	FF
1F		FF	00	00	00
20 to 2F		FF	FF	FF	FF
30		78	50	50	64
31		DC	BE	B4	B4
32		02	02	02	02
33		64	96	96	46
34		01	01	01	01
35		05	06	09	05
36		40	40	40	40
37		0F	0F	0F	0F
38		A0	A0	A0	A0
39		0B	0B	0B	11
3A		B8	B8	B8	40
3B		23	23	23	1E
3C		28	28	28	60
3D		00	00	00	00
3E		0A	0A	0A	0A
3F		03	03	03	03
40		E8	E8	E8	E8
41		17	17	17	17
42		D0	D0	D0	D0
43		70	75	74	F3
44		70	75	74	F3
45		03	03	03	03
46		E8	E8	E8	E8
47		03	03	03	03
48		E8	E8	E8	E8
49		17	17	17	17
4A		70	70	70	70
4B		01	01	01	01
4C	EEPROM Checksum (0x30 to 0x4)	BC	29	3B	32
4D		08	06	04	0E
4E		FF	FF	FF	FF

4F		FF	00	00	00
50 to 5F		FF	FF	FF	FF
60	Servo Type	48(H)	48(H)	48(H)	48(H)
61		53(S)	53(S)	53(S)	53(S)
62		35(5)	35(5)	35(5)	52(R)
63		34(4)	32(2)	36(6)	35(5)
64		37(7)	34(4)	34(4)	39(9)
65		35(5)	35(5)	35(5)	39(9)
66		48(H)	4D(M)	4D(M)	35(5)
67		47(B)	47(G)	47(G)	54(T)
68	Null	00	00	00	47(G)
69		FF	41(A)	41(A)	00
6A		FF	4C(L)	4C(L)	4C(L)
6B		FF	49(I)	49(I)	49(I)
6C		FF	5A(Z)	5A(Z)	5A(Z)
6D		FF	45(E)	45(E)	45(E)
6E		FF	44(D)	44(D)	44(D)
6F		FF	00	00	00
70		15	08	07	08
71		08	0C	0A	0C
72		07	07	07	07
73		D4	D5	D4	D5
74 to 7F		FF	FF	FF	FF
80	Serial Number?	30(0)	30(0)	30(0)	30(0)
81		30(0)	32(2)	30(0)	32(2)
82		37(7)	34(4)	39(9)	37(7)
83		32(2)	39(9)	33(3)	35(5)
84		35(5)	37(7)	38(8)	31(1)
85		33(3)	32(2)	32(2)	37(7)
86		34(4)	30(0)	35(5)	37(7)
87	Null	00	00	00	00
88 to FF		FF	FF	FF	FF

Data Memory Commands

The “c” and “d” commands read or write to the 256byte data memory area respectively. The HFP-10 only ever seems to use the d command.

The format of the data memory is in the Atmel data sheet. It appears possible to access contents of both the RAM and also the registers including I/O.

No idea how the Data memory locations higher than 256 are accessed on ATMega8. Also no idea how flash can be boot loaded

Some data memory contents are important and accessed by the HFP10 to control the operation of the servo.

Data Memory Address	Function	Reset Default HS-5475HB	Reset Default HS-5245MG	Reset Default HS-5645MG	Reset Default HSR-5995TG
18	May be direction	01	01	01	01

80		78	50	50	64
81		78	50	50	64
82		DC	BE	B4	B4
87		40	40	40	40
88		0F	0F	0F	0F
89		A0	A0	A0	A0
8E		00	00	00	00
8F		38	39	0A	46
90		03	03	03	03
91		C7	AF	E8	C2
92		17	17	17	17
93		CC	CC	D0	CC
96	End Point Left	70	75	75	F3
97	End Point Right	70	75	75	F3

Read Position Command

The read position command (e) appears to be a direct read of the ADC in the processor from the feedback pot. I get the same result as when I read the ADC registers from the data memory. The ADC is 10 bit, so the range of values is from 0 to 1024. The physical endpoint stops give an actual range of values of 68(0x44) to 956 (0x3BC) for my servo. So this is the best accuracy that could be achieved, though this may be compromised by the pot.

Move Servo Command

The move servo command (f) appears to move the servo position. The range of acceptable values on my servo seems to be 3200 (0xC80) to 8344 (0x8344). This is strange, and I do not really understand the positioning and the relationship between actual position, desired position, neutral and endpoints. Values outside the acceptable range do not move the servo.

During reset the servo is calibrated moving from end stop to end stop and reading position. Here the values of 3600 (900uSec * 4) and 8400 (2100uSec * 4) are used. These are of course consistent with an internal timer for the PWM clocked at 4MHz.

I just don't understand where the 3200 and 8344 come from, close but no cigar !

Key Operations

The following is how the HPF-10 performs the following actions:

HFP-10 Reset

When the reset command is first invoked the HFP-10 does the following:

Action	**Location**	**Content**	**Comment**
Write Data Memory	80	00	
Write Data Memory	82	00	After this the motor is off
Write Data Memory	81	00	
Read EEPROM	00 to 1C	….	
Read EEPROM	30 to 4C	….	

Write Data Memory	18	00	
Write Data Memory	87	01	
Write Data Memory	96	FF	
Write Data Memory	97	FF	
Write Data Memory	88	0F	
Write Data Memory	89	CC	

If the actual reset is invoked (L & R buttons together) the HFP-10 does the following:

Action	**Location**	**Content**	**Comment**
Write Data Memory	90	03	
Write Data Memory	91	FF	
Write Data Memory	8E	00	
Write Data Memory	8F	00	
Write Data Memory	92	17	
Write Data Memory	93	CC	
Write Data Memory	87	30	
Write Data Memory	80	00	
Write Data Memory	81	00	
Move Servo		0E10	Go to left end stop
Read Position		44	Get left position
Move Servo		20D0	Go to right end stop
Read Position		03BC	Get right position
Write Data Memory	80	00	
Write Data Memory	82	00	After this the motor is off
Write Data Memory	81	00	
Write EEPROM	0 to 0x1C	…	Initialises the EEPROM

HFP-10 Set Deadband

When set deadband is invoked on the HFP-10 does the following:

Action	**Location**	**Content**	**Comment**
Write Data Memory	80	00	
Write Data Memory	82	00	After this the motor is off
Write Data Memory	81	00	
Read EEPROM	00 to 1C	….	
Read EEPROM	30 to 4C	….	
Write Data Memory	18	00	
Write Data Memory	87	01	
Write Data Memory	96	FF	
Write Data Memory	97	FF	
Write Data Memory	88	0F	
Write Data Memory	89	CC	

When the deadband value is changed (m button) the HFP-10 sends:

Action	Location	Content	Comment
Write EEPROM	02	Deadband	Range 0x00 to 0x10
Write EEPROM	1C	Check	

HFP-10 Set Direction

When set direction is invoked on the HFP-10 does the following:

Action	Location	Content	Comment
Write Data Memory	80	00	
Write Data Memory	82	00	After this the motor is off
Write Data Memory	81	00	
Read EEPROM	00 to 1C	….	
Read EEPROM	30 to 4C	….	
Write Data Memory	18	00	
Write Data Memory	87	01	
Write Data Memory	96	FF	
Write Data Memory	97	FF	
Write Data Memory	88	0F	
Write Data Memory	89	CC	

When the direction is changed (L /R button) the HFP-10 sends:

Action	Location	Content	Comment
Write EEPROM	13	Endpoint	Move Right Endpoint
Write EEPROM	14	Endpoint	Move Left Endpoint
Write EEPROM	19	Failsafe High	Move Failsafe
Write EEPROM	1A	Failsafe Low	Move Failsafe
Write EEPROM	1B	Direction	Bit 0 is set for CW note this is a bit change and must preserve the other bits
Write EEPROM	1C	Check	

HFP-10 Set Speed

When set speed is invoked on the HFP-10 does the following:

Action	Location	Content	Comment
Write Data Memory	80	00	
Write Data Memory	82	00	After this the motor is off
Write Data Memory	81	00	
Read EEPROM	00 to 1C	….	
Read EEPROM	30 to 4C	….	

Write Data Memory	18	00	
Write Data Memory	87	01	
Write Data Memory	96	FF	
Write Data Memory	97	FF	
Write Data Memory	88	0F	
Write Data Memory	89	CC	

When the speed value is changed (m button) the HFP-10 sends:

Action	**Location**	**Content**	**Comment**
Write EEPROM	06	Speed	Range 0x00 to 0x40
Write EEPROM	1C	Check	

HFP-10 Enable Failsafe

When set enable failsafe is invoked on the HFP-10 does the following:

Action	**Location**	**Content**	**Comment**
Write Data Memory	80	00	
Write Data Memory	82	00	After this the motor is off
Write Data Memory	81	00	
Read EEPROM	00 to 1C	….	
Read EEPROM	30 to 4C	….	
Write Data Memory	18	00	
Write Data Memory	87	01	
Write Data Memory	96	FF	
Write Data Memory	97	FF	
Write Data Memory	88	0F	
Write Data Memory	89	CC	

When the failsafe is changed (L /R button) the HFP-10 sends:

Action	**Location**	**Content**	**Comment**
Write EEPROM	1B	Failsafe	Bit 1 is set for enable, note this is a bit change and must preserve the other bits
Write EEPROM	1C	Check	

HFP-10 Change Failsafe/Neutral/Endpoint Position

When change failsafe/neutral/endpoint position is invoked on the HFP-10 (after centre set on pot) does the following:

Action	**Location**	**Content**	**Comment**
Write Data Memory	80	00	
Write Data Memory	82	00	After this the motor is

			off
Write Data Memory	81	00	
Read EEPROM	00 to 1C	….	
Read EEPROM	30 to 4C	….	
Write Data Memory	18	00	
Write Data Memory	87	01	
Write Data Memory	96	FF	
Write Data Memory	97	FF	
Write Data Memory	88	0F	
Write Data Memory	89	CC	
Move Servo		Values determined by the HFP-10 pot	Does these two in a loop until a function is requested
Read Position		Values determined by the servo position	

When the failsafe is set the HFP-10 sends:

Action	**Location**	**Content**	**Comment**
Write EEPROM	19	Failsafe High	
Write EEPROM	1A	Failsafe Low	
Write EEPROM	1C	Check	

When the neutral is set the HFP-10 sends:

Action	**Location**	**Content**	**Comment**
Write EEPROM	07	Neutral High	
Write EEPROM	08	Neutral Low	
Write EEPROM	13	Endpoint Left	Value set to 0x16
Write EEPROM	14	Endpoint Right	Value set to 0x16
Write EEPROM	1C	Check	

When the left endpoint is set the HFP-10 sends:

Action	**Location**	**Content**	**Comment**
Write EEPROM	13	Endpoint Left	
Write EEPROM	1C	Check	

When the right endpoint is set the HFP-10 sends:

Action	Location	Content	Comment
Write EEPROM	14	Endpoint Right	
Write EEPROM	1C	Check	

On the HSR-5995TG the endpoints on reset are set outside the end stops of the servo; hence it is possible to overheat the servo by driving past the physical end points.

PC Code

The Visual Basic 2005 code I used is supplied as separate file