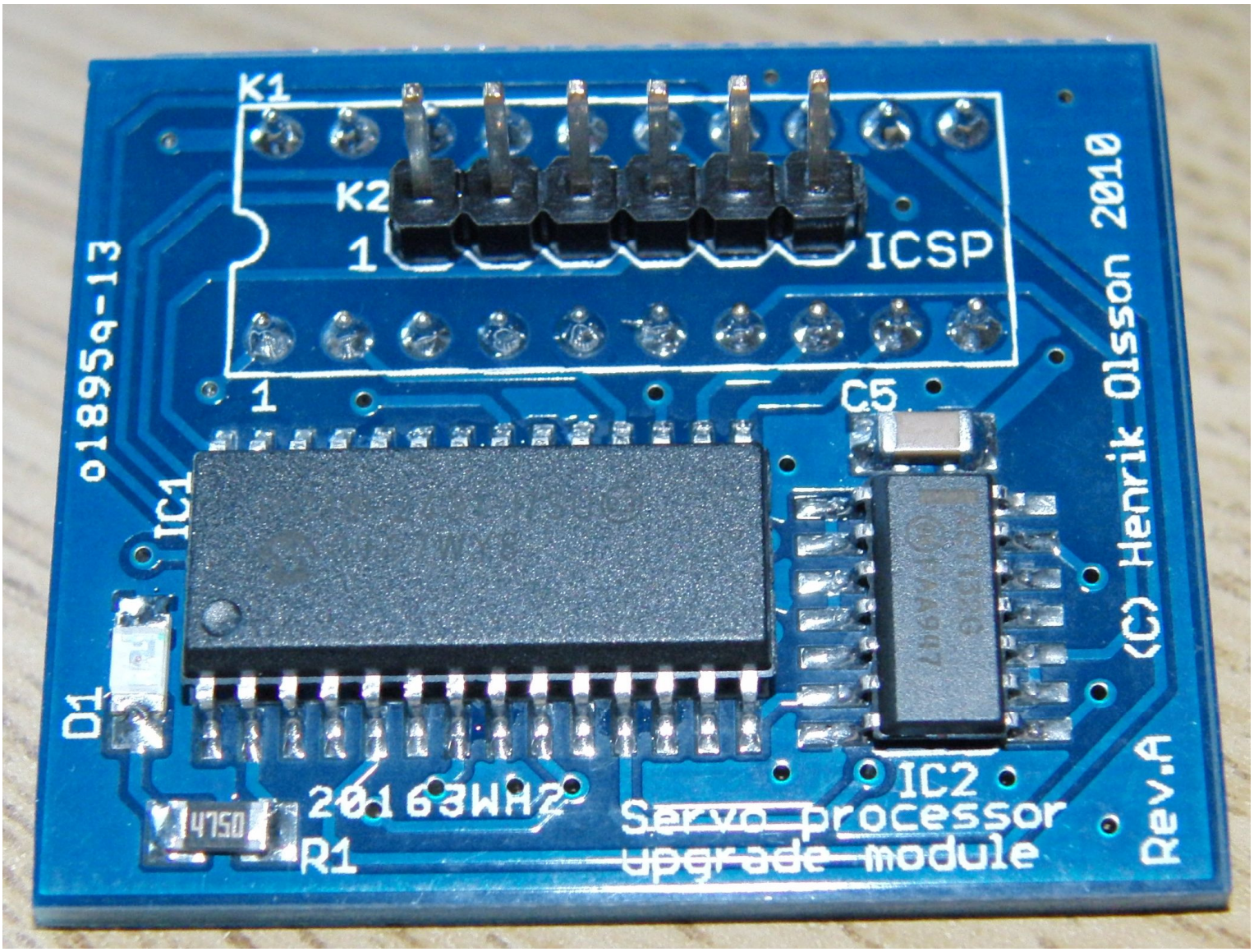


User Guide Rev. B

Servo processor upgrade module for UHU based drives.

Preliminary manual – work in progress 2011-06-03

Table of Contents

1 About this guide

This document is divided into the following chapters:

- Chapter 1, About this document
- Chapter 2, Introduction.
- Chapter 3, Installing the module and verify that it is working.
- Chapter 4, “Command reference”, describes the modules commands.
- Appendix A
- Appendix B, The following error scale feature explained.
- Appendix C

1.1 Who Should Use It

This guide is intended for users of the servo module. The user is expected to have a general understanding of how a closed loop servo system operates as well as general electronics and computer knowlege.

1.2 Typographical Conventions

This document uses the following typographical conventions:

- Angle brackets are used to illustrate a button press such as *<ENTER>*
- Commands, as entered by the user, appears in italic, like: *D1234<ENTER>*
- Response from the module, as shown in the terminal software window appears in monospace font, like: `(D) - Kd: 1234`

2 Introduction

2.1 Purpose

The purpose of the servo processor module is to be a near drop in replacement for the popular UHU servo controller chip. Simply replacing the UHU chip with servo processor module allows current hardware to attain much higher step-rates and/or use higher encoder resolutions than what is possible with the original UHU chip. On top of that several features not available on the UHU-chip have been added to the module.

2.2 Differences between the module and the UHU chip

The module was designed to be pin-compatible with v3.0 of the UHU chip. With that said there are differences between the two which, depending on the design of the particular drive on which the module is to be mounted, may or may not need to be considered. The following section of this document will try to cover these differences.

2.2.1 Module pinout

The following shows the actual pinout of the module:

Pin 1: Reset/MCLR (active low)
Pin 2: RXD (serial data to module)
Pin 3: TXD (serial data from module)
Pin 4: Not used, not connected to anything
Pin 5: Not used, not connected to anything
Pin 6: Step input
Pin 7: Dir input
Pin 8: Not used (connected to high Z pin)
Pin 9: Not used (connected to high Z pin)
Pin 10: GND / Vss
Pin 11: Not used, not connected to anything
Pin 12: Encoder input A
Pin 13: Encoder input B
Pin 14: Fault input (active low)
Pin 15: PWM output (locked antiphase)
Pin 16: Enable output (active high)
Pin 17: Fault output (active low)
Pin 18: Firmware status output (see 2.2.1)
Pin 19: Not used, not connected to anything
Pin 20: +5V / Vcc

2.2.2 Pins 4 & 5

On the UHU chip pins 4 and 5 are used for the crystal oscillator. The module features its own on board oscillator so these pins are not used nor are they connected to anything on the module. You can safely leave the 24MHz crystal for the UHU chip on your servo drive PCB.

2.2.3 Pins 8 & 9

On the module pins 8 and 9 are connected to pins on the modules CPU which, as of this writing, are not being used and are set to high impedence mode (inputs). In future firmware versions these pins MAY be used for other functions which would then require rework of the servo drive PCB.

Several incarnations of the UHU based drives have pin 8 and pin 9 in parallell with pins 12 and 13 of the chip (the encoder inputs). This seems to come from earlier versions of the original chip while the documentation for v3.0 states that the encoder inputs are on pin 12 and 13 only which is where they are located on the module.

> As long as your PCB is layed out to have the encoder signals on, at least, pins 12 and 13 it is OK to leave everything as it is. If the encoder signals happen to be available on pins 8 and 9 as well that won't be a problem with this version of the module.

2.2.4 Pins 11 & 18

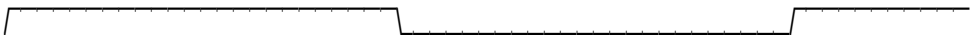
These pins are not used nor are they connected to anything on the module.

2.2.5 Firmware status output.

The operation of the firmware status output differs from that on the original UHU chip. On the original UHU chip this output was simply held at a constant low level (LED ON) when the chip was operating correctly. On the servo processor module the output toggles high and low in different patterns corresponding to the current state of the module. Internal states such as fault due to following error and external fault are two states that can easily be recognised by looking at the LED that normally is connected to the firmware status output pin.

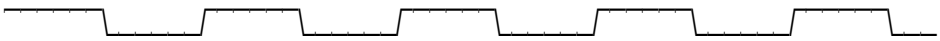
Currently four different states can be identified by the firmware status output, these are *Normal state*, *Fault due to following error*, *Fault due to external request* and *Servo loop aborted on user request (over serial command).*

Normal state:



Following error:

External fault request:



Servo abort (Q-command)

2.2.6 PWM Dutycycle and output swing

The UHU chip limits the output duty cycle, or output swing, to between 13 and 87% in order to allow the bootstrap capacitors in the MOSFET bridge drive circuitry enough time to refresh their charge. By default the servo processor modules C-parameter is setup to provide the same amount of output swing as the UHU-chip.

However, if the hardware on which the module is being used can accept a larger output swing it is possible to increase this up to 8-92%.

See the command reference section of this manual for details on the C-parameter.

2.2.7 Baud rate

The original UHU chip uses 34800 baud 8N1 for serial communication with the host PC. The servo processor module uses 57600 baud 8N1. This means that it's not possible to use any terminal software specifically developed for the UHU chip to communicate with this module. Any "general purpose" terminal software, such as Hyperterminal and Realterm will work if setup for 57600 8N1, no flow control or handshaking. See section 3.1 for further details on how to set up the terminal software.

2.2.8 Parameter banks

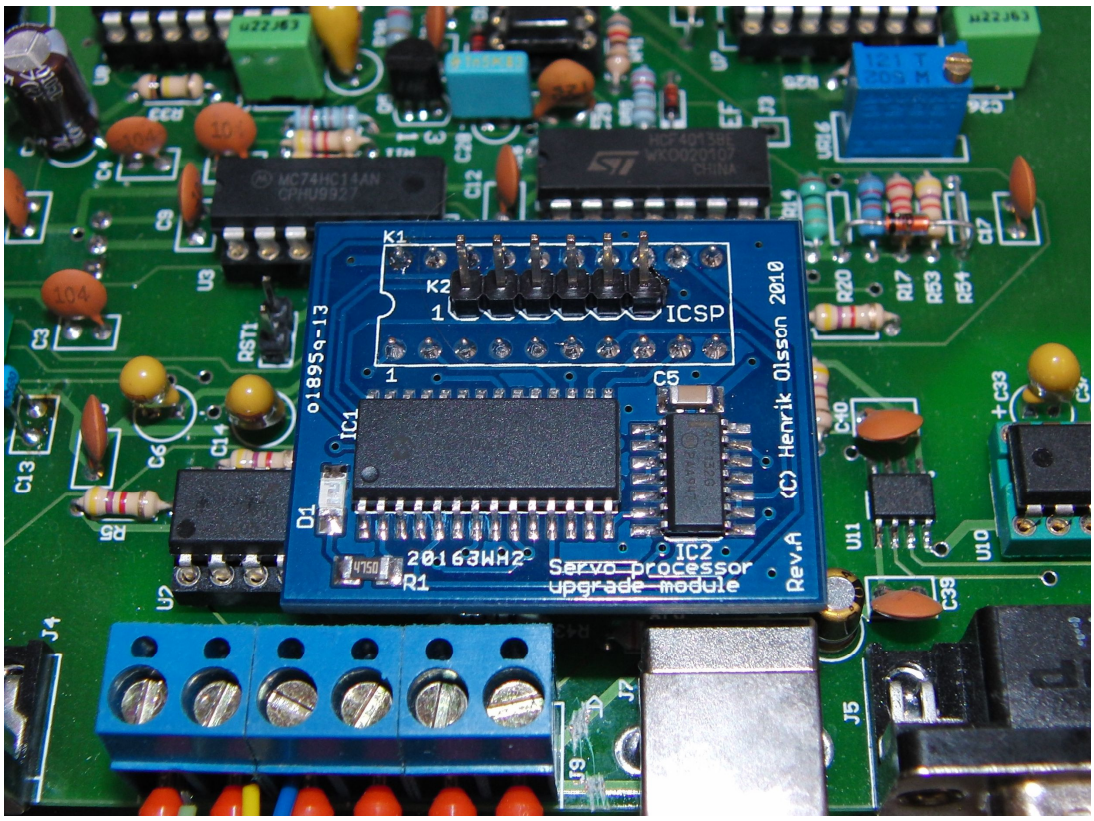
The original UHU features several banks to which complete sets of parameters can be saved and recalled. The servo processor module only has one bank.

3 Installing the module

The servo processor module consists of a printed circuit board with male headers matching the 20-pin dual-in-line footprint of the original UHU chip. Installing the module is a simple matter of removing the UHU chip and mounting the module in its place, making sure the pins lines up properly.

Depending on how your particular servo drive is designed it is possible that the servo processor module won't fit directly in place of the UHU chip as surrounding components may interfere with the larger footprint of the module. If this is the case it's advisable to use one or more 20-pin dual-in-line IC sockets as 'raisers' to get the module "above" the surrounding components. Make sure to secure the module in case the 'stack' is high – using some hot melt glue, RTV silicone or similar.

Here's a photo of the module mounted on the HP-UHU drive:



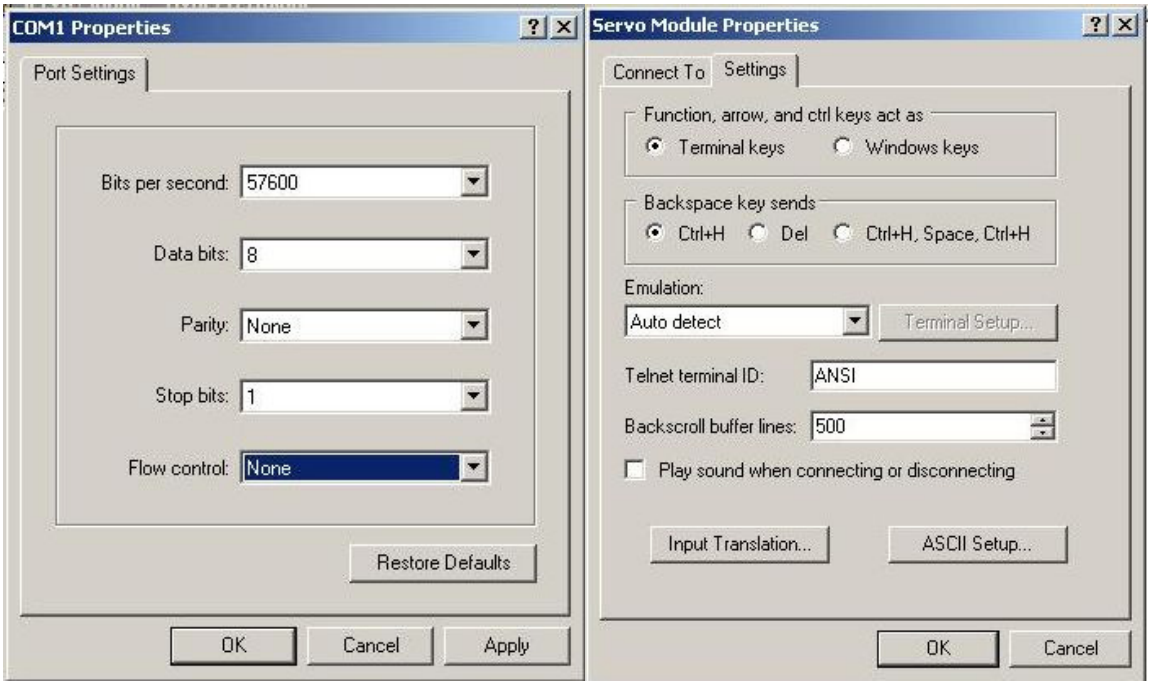
Note: It's highly recommended to use high quality IC sockets with machined pins.

3.1 Setting up the terminal software

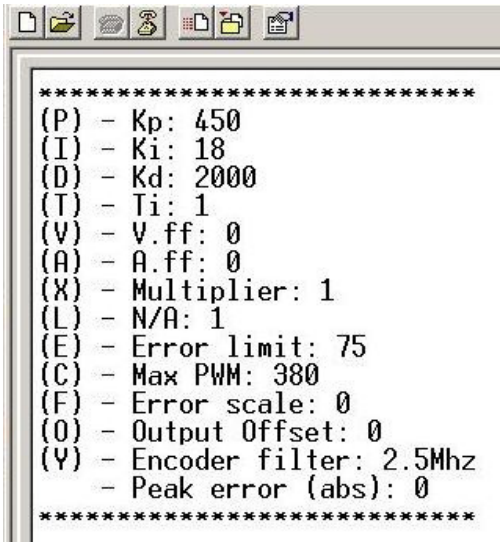
Here we will go thru how to set up Hyperterminal, which comes with most versions of Windows, to communicate with the module. However, any terminal emulating software capable of talking to a serial port should work.

To set up Hyperterminal, start a new connection, give it a name and click OK. Select the appropriate COM port to which you connect your drive and click OK.

Select 57600 baud, 8 databits, 1 stopbit, no parity and no flow control and click OK. Finally go to the File menu, select Properties, click the ASCII Setup button and tick the checkbox for echoing typed characters locally.



Power up the drive and you should see a message on screen displaying the serial number and firmware revision of the module. Now type *?<ENTER>* and the module should respond with the setup menu (actual content of menu may not match this screenshot).



4 Command reference

This section explains the various commands and settings that are available. To change a setting simply type the letter corresponding to the setting you want to change followed by the value to which you want to change it and press <ENTER>. The module will then respond with a confirmation. For example, to set the proportional gain you enter *P1234<ENTER>* and the module responds with: `(P) - Kp: 1234`

When all settings are to your liking the 'S' command is used to save them to the on board EEPROM memory. Simply send *S<ENTER>* and the module will respond with `Parameters saved.`

All settings accepts integer values only, some uses signed numbers, ie. they can be set to negative values by preceeding them with a "-". It is described in this manual when and where negative values are applicable.

No floating or fixed point numerical values are allowed (simply – no decimal points).

4.1 A – Acceleration feed forward

Sets the acceleration feed forward gain.
The acceleration feed forward works like the derivative term of the PID filter but instead of working with the difference of the error between PID updates (ie. the derivative) the acceleration feed forward works on the difference in commanded velocity (ie. the acceleration) giving the output some extra "push" when accelerating and "holding it back" when decelerating.

A100<ENTER>

`(A)-A.ff: 100`

Default value: 0. Minimum value: 0. Maximum value: 32000. Typical value: N/A

4.2 B – Integrator limit

Sets the integrator limt.

The B parameter can be used to limit the amount of “effort” that the integral term is allowed to contribute with to the total output of the PID filter.

B125<ENTER>

```
(B) - I Limit: 125
```

Whenever there’s an error between the setpoint and the target position the integral term of the PID filter will keep adding to (or subtracting from) the total output of the filter in an effort to drive that error away. This behaviour ensures that the motor will end up at its target but it also introduces some drawbacks.

One drawback is that if the error persists for an extended period of time (due to locked mechanics, stalled motor etc) the integral term can grow very big. If the mechanics then suddenly frees up the motor might accelerate very hard possibly damaging the mechanics etc. This behaviour is often called integrator wind-up.

Another is that when the motor is moving there’s almost always a small error present as the motor is “chasing” the moving target position. This can make the I-term grow quite large and cause a quite substantial overshoot and increased settling time at the end of the move.

Note: The B-parameter is not availbale in firmware version below 1.010

Default value: 450. Minimum value: 0. Maximum value: 32000. Typical value: N/A

4.3 C – Clamp

Sets the maximum PWM duty cycle.
This can be used to limit the effective voltage swing of the servo drive on which the module is mounted. By default this is set to 760 which results in an effective output swing of 13-87% which is the same as the original UHU chip uses.

C500<ENTER>

```
(C)-Max PWM: 500
```

Internally a dutycycle of 50% - resulting in an average motor current of 0A - is represented by the value 1024. 0% would be 0 and 100% would be 2048. The value of 'C' is simply the maximum amount that the duty cycle value is allowed to swing, up or down from 1024.

Example:
If you set C=400 the dutycyle will get clamped to 30.5 - 69.5% because
(100% / 2048 * (1024 - 400) = 30.5%) and (100% / 2048 * (1024 + 400) = 69.5%

Default value: 760. Minimum value: 0. Maximum value: 860. Typical value: 760

The maximum output swing of the module is 8-92% (C=860). Setting 'C' to a value higher than 860 is possible but will have no effect.

4.4 D – Differential gain

Sets the differential gain of the PID-loop.
The differential term of the PID loop works by looking at the difference in position error between two consecutive servo loops. If the position error gets larger the differential term adds "effort" to the total output, if the position error gets smaller it subtracts "effort" from the final output.
Some times the differential term is reffered to as damping.

D1100<ENTER>

```
(D)-Kd: 1100 (900)
```

The value shown within paranthesis is the gain that will be applied when the error is within the antidither region. See the J-parameter and the K-parameter.

Default value: 5000. Minimum value: 0. Maximum value: 32000. Typical value: 5-20 times P

4.5 E – Following error limit trip point

Sets the allowed following error in encoder counts. Please also see the F-parameter.

E75<ENTER>

```
(E) - Error limit: 75
```

Setting the E-parameter to anything above 32767 will effectively disable the trip limit all together.

Warning: Setting the E-parameter to very large values, or disabling it completely, can result in the internal calculations overflowing causing unexpected results like motor reversal and/or runaway etc. Be careful with this. Generally a value of E equal to a couple of degrees of motor shaft movement is considered a “proper” value.

Default value:75. Minimum value: 0. Maximum value: 32767. Typical value: 25-250

4.6 F – Following error scale

Sets the following error scale value.
This setting provides a way to scale the allowed following error relative to the currently commanded motor velocity. That is, the higher the commanded velocity is the larger following error is allowed without faulting.

F256<ENTER>

```
(F)-Error scale: 256
```

If you don’t want to use this feature it should be set to 0. The following error trip limit you specify with the E command will then be what you get – no matter what the commanded velocity is.

Default value: 0. Minimum value: 0. Maximum value: 32000. Typical value: 5-20 times P

For further details regarding this setting please see Apendix B.

4.7 G – Get parameter

The G command is used in conjunction with a second letter to get various internal, non changeable parameters and variables which can be useful during set-up and tuning:

4.7.1 GA – Get all PID filter terms

This gets and prints the value of all terms, except the offset, that makes up the total output. The P-term, I-term, D-term, acceleration and velocity feedforward.

GA<ENTER>

```
P10 I120 D0 A0 V25
```

4.7.2 GV – Get Velocity

This gets the current motor velocity in encoder counts per servo loop cycle. It is the actual motor velocity, not the commanded velocity and it's presented as a signed value.

GV<ENTER>

```
Velocity: -63
```

If the servo loop update rate was set to 1500Hz and the motor was equipped with a 500 line encoder the above would mean a motor velocity of 63*1500/500*4*60=2835rpm.

4.7.3 GP – Get Position

This gets the current motor position. Currently it's being presented as two 16bit numbers corresponding to the the high and low word of the internal 32bit position register.

GP<ENTER>

```
Pos: 3, 46259
```

In the above example the actual motor position is 3*65536 + 46259 = 242867 encoder counts. The 32bit position register is signed 2's compliment.

4.7.4 GT – Get Target (setpoint)

This gets the current target position or setpoint.

GT<ENTER>

```
Setpoint: 3, 46259
```

As with the GP-command the setpoint is preseneted as two 16-bit numbers representing the high and low word of the internal 32bit setpoint register.

4.7.5 GE – Get Error

This gets the current error. It's presented as a signed value.

GE<ENTER>

```
Error: -8
```

The error is the difference between the commanded and actual position as measured by the encoder on the motorshaft. It's being calculated every servo loop update.

4.7.6 GF – Get scaled limit

This gets the actual following error trip limit as calculated based on the currently commanded motor velocity, the following error trip limit (the E-parameter) and the following error scale value (the F-parameter).

GF<ENTER>

```
Scaled limit: 36
```

4.7.7 GO – Get Output

This gets the current output from the PID filter including velocity and acceleration feedforward but excluding any offset that has been applied with O-command. The value is presented in signed form.

GO<ENTER>

```
PID Output: 12
```

4.7.8 GD – Get Dutycycle

This gets the current dutycycle being output by the module. The value is presented like it is being written to the internal PWM generator where a value of 1024 means 50% dutycyle.

```
GD<ENTER>
PWM Dutycyle: 1214
```

1214 in this case would mean a duty cycle of 100/2048*1214=59% or 9% in the “forward direction” because we’re using locked antiphase PWM where 50% means 0 torque.

4.7.9 GS – Get Status

This gets the current status of the module. It’s presented as a 16bit binary number with the most significant bit to the left.

Bit 0: Fault.
Bit 1: Following error.
Bit 2: External fault
Bit 3:
Bit 4:
Bit 5:
Bit 6:
Bit 7: Data recorder mode
Bit 8: Data recorder trigged
Bit 9:
Bit 10: PWM Output saturated.
Bit 11: Commanded motor direction
Bit 12:
Bit 13:
Bit 14: UART Receive error
Bit 15: UART Transmit error

GS<ENTER>

```
Status: 0000010000000101
```

The following describes each bit briefly:

Bit 0: Fault.

This bit will be set whenever the operation of the servo loop is inhibited – ie. power to the motor is turned off by making the Enable output low.

Bit 1: Following error

This bit will be set when a following error larger than what is specified with the E-parameter + any applied scaling (the F-paramater) stops the servo-loop.

Bit 2: External fault.
This bit will be set when an external fault issued thru the external fault input has stopped the operation of the servo-loop.

Bit 3-6: Currently not in use.

Bit 7: Data recorder mode.
This bit will be set when the module is in data-record mode.

Bit 8: Data recorder trigged.
This bit will be set when the module is in data-record mode and in process of recording data to the internal buffer.

Bit 10: PWM Output saturated.
This bit will be set if the output, at one point or another since power-up or reset, has been saturated. When this happens the servo-loop no longer has the motor “under control”. It (the servo-loop) would like to apply more power in order to reduce the error but it simply can’t because the duty cycle is already at its max which means that error is likely to grow bigger instead of smaller. This bit can be useful to monitor while tuning the system.

Bit 11: Commanded motor direction.
This bit will be set when the commanded motor direction is in the “negative” direction. It is primarily being used internally for the data recorder mode.

Bit 14: UART Receive error
This bit will be set if the internal hardware UART has detected an error when receiving data on the serial line, like buffer overruns etc. This SHOULD never happen.

Bit 15: UART Transmit error
This bit will be set if the internal hardware UART has detected an error when transmitting data on the serial line, this could happen if the firmware in the module tried to send data when the UART is already in process of sending. This SHOULD never happen.

4.8 I – Integral gain

Sets the integral gain of PID filter.
The integral term of the PID filter works by accumulating any errors and provides “effort” to the final output in relation to the sum of all errors over time. This ensures that even the smallest steady state error will be driven away.

I105<ENTER>

```
(I)-Ki: 105 (75)
```

Even though having velocity feedforward (see the V-parameter) helps keeping the following error low while the motor is moving there is almost always an error present during motion. The motor is more or less constantly “chasing” the target position, lagging slightly behind the moving target position. The integral term “acts” on this error and adds to the total output of the PID filter in an effort to reduce the error. This is good as it tries to reduce the following error but at the end of the move, when motion is stopped the intergral term may have grown large enough to cause a substantial overshoot or even oscillation with increased settling time as a result.

By specifying a negative integral gain you have the option to only have the integral term of the PID filter active when the commanded motor velocity is zero – in other words when the motor is supposed to be stationary and in position. This prevents the integral term to grow when the motor is moving. Yet, as soon as the commanded velocity is zero the integral term is re-enabled and starts integrating any remaining steady state error in an effort to drive it away.

I-75<ENTER>

```
(I)-Ki: -75 (-50)
```

The value shown within paranthesis is the gain that will be applied when the error is within the antidither region. See the J-parameter and the K-parameter.

Note: Specifying a negative integral gain is not possible in firmware versions below 1.100

Note: See the T-parameter and B-parameter as well.

Default value: 30. Minimum value: 0. Maximum value: 32000. Typical value: 1-10% of P

4.9 J – Antidither region

Sets the range, or region, within which the antidither function is active.
The antidither function works by scaling the P, I and D gains by an adjustable factor (see the K-parameter) when the error is within +/- what the J-parameter is set to. For example, if the J-parameter is set to 2 the antidither function will be active when the error is within +/-2 counts.

J4<ENTER>

```
(J)-AD Region: 4
```

Default value: 2. Minimum value: 1. Maximum value: 5000. Typical value: 0-20

See the K-parameter regarding what to do if you don't want to use the antidither function.

4.10 K – Antidither scale

This sets the scale factor, or gain, of the antidither function.
When the error is within the range specified by the J-parameter the normal P, I and D gains will get scaled by the value of the K-parameter.

This parameter is represented in 1/256 units meaning that if K=128 the three PID gains will be reduced to 50% (128/256=0.5) when the error is within the antidither region set by the J-parameter. With K=192 the gains will be reduced to 75% (192/256 = 0.75) and so on. It is perfectly valid to specify a value higher than 256 in which case the gains will get scaled UP when the error is within the antidither region if the user sees an application for that.

K128<ENTER>

```
(K)-AD Scale: 128
```

Setting the K-parameter to 256, resulting in a gain of 1, will effecitvely disable the antidither feature in case it's not needed or desiried. This is also the default setting.

Default value: 256. Minimum value: 0. Maximum value: 4000. Typical value: 32-256.

4.11 L – Servo loop update rate

Sets the frequency at which the PID-loop is ran. There are 10 settings ranging from 0 to 9:

0: 2785Hz	5: 1650Hz
1: 2450Hz	6: 1500Hz (Default)
2: 2200Hz	7: 1400Hz
3: 1950Hz	8: 1300Hz
4: 1800Hz	9: 1220Hz

L9<ENTER>

```
(L)-Servo rate: 1220Hz
```

Internally the L parameter sets a dividing ratio between the PWM frequency and the mechanism that triggers the execution of the servo-loop. The actual dividing ratio is simply the value of 'L' plus 7. The PWM frequency is fixed at 19.5kHz so with L set to 3 the resulting servo loop rate is 19500 / (7+3) = 1950Hz.

Setting L to any value above 9 will result in a servo-loop rate of 1220Hz.

Generally speaking a highly dynamic and/or low inertia system will benefit from a higher loop frequency while a large and heavy system might do better with a lower frequency because it can't respond fast enough to the "commands" generated by the higher update frequencies anyway.

Please note that changing the servo loop update rate effectively changes the effect of several other tuning parameters. This is because if you double the servo loop update rate you get half the number of encoder counts per servo loop update for any given motor velocity. This means that the effect of the D-parameter is reduced with increased servo loop update rate while the effect on the I-term is the oposite. This is because any error present gets accumulated and because the servo loop is run at a higher rate the I-term "grows" faster.

Default value: 6 (1500Hz). Minimum value: 0. Maximum value: 9. Typical value: 6

4.12 O – Offset

This sets the amount of offset applied to the output. Normally the internal dutycyle value is 1024, representing a dutycycle of 50%, when no torque is required from the motor. The O-parameter offsets this up or down by the amount specified.

It can be used to compensate for directional unbalance when driving hanging loads like the Z-axis on a typical three axis VMC or knee-mill. It can also be used to run the motor “open loop” by setting the main gains to 0 and then simply specifying an offset.
The value is entered in signed form, ie:

O-25<ENTER>

```
(O)-Output Offset: -25
```

Warning: Values above 32767 are internally treated as negative values (two’s compliment). This means that accidently entering very large values can have undesireable effects.

Default value: 0. Minimum value: -1024. Maximum value: 1024. Typical value: 0

4.13 P – Proportional gain

Sets the proportional gain of the PID filter.

P1250<ENTER>

```
(P)-Kp: 1250 (1000)
```

The value shown within paranthesis is the gain that will be applied when the error is within the antidither region. See the J-parameter and the K-parameter.

Default value: 0. Minimum value: 0. Maximum value: 32000. Typical value: 500-5000

4.14 Q – Quit servo

This stops the internal execution of the servo-loop, sets the PWM duty cycle to 50% + Offset and pulls the *Enable* output low which effectively disables the drive signals to the bridge drive circuitry on typical UHU based designs. (Removes power from the motor and freewheels it).

Q<ENTER>

```
Servo is OFF
```

Use the R-command to re-activate the module.

> Even though the execution of the servo loop is suspended and the enable signal removed from the bridge drive circuitry the encoder is still being read and the position register updated. However, once the servo loop is turned back on with the R-command the actual position at that time will be considered the new target position.

4.15 R – Reset / Restart

This performs a soft reset of the module and restart of the servo loop. Use this command to recover from a fault or to re-enable the drive after it has been disabled with the Q-command.

When the R-command is executed the following happens internally:

- All internal error flags in the status register are cleared.
- The PWM output saturated flag in the status register is cleared.
- The detected peak error value is reset to 0.
- The setpoint register is loaded with the current motor position (ie. the current motor position becomes the new setpoint and therefor the error becomes 0)
- The Fault output is returned to its normal state (high)
- The Enable output is turned 'on' (set high) enabling power to the motor on a typical UHU based design.

R<ENTER>

```
Servo RESET!
```

Note: If you've made, but not yet saved, changes to any of the parameters these will NOT be reverted when executing the R-command. A hardware reset will "undo" any unsaved changes.

4.16 S – Save parameters

This saves all parameters to the on board non volatile memory. While there are several banks available on the UHU chip there's only a single bank available on the servo processor module.

S<ENTER>

```
Parameters saved
```

Note: While saving the parameters to the onboard memory the interrupt that normally triggers the execution of the servo-loop is temporarily turned off as it otherwise can cause corruption of the data being written to the EEPROM memory. Because of this the S-command is only allowed to execute when the motor is not commanded to move.

4.17 T – Integral time constant (Ti)

The T-command is used to set the time constant for the PID-filter integrator.
A value of 1 (default) means that the I-term is updated every time the PID-loop is executed. A value of 2 means it's updated every second run of the PID-loop and so on. The frequency at which the PID loop is exectued is controlled by the L-parameter.

T3<ENTER>

```
(T)-Ti: 3
```

Default value: 1. Minimum value: 1. Maximum value: 128. Typical value: 1-5

4.18 V – Velocity feed forward gain

The V-command sets the gain of the velocity feed forward.

The velocity feed forward works by adding a scaled value of the commanded motor velocity to the final PWM output, providing some extra "push". Unlike the proportional term the velocity feedforward works outside of the actual control loop which means it can't cause instabillity

and/or oscillations. And its “input” is the commanded motor velocity instead of the current position error.

Normally, when the motor is moving, there’s almost always a small following error as the motor is “chasing” the ever changing target position. The velocity feedforward can help reduce this following error because instead of “draging” the motor along it “pushes” the motor forward. This, however, means that adding too much velocity feedforward may introduce a positive following error – ie the motor is ahead of the targer position.

V750<ENTER>

```
(V)-V.ff: 750
```

Default value: 0. Minimum value: 0. Maximum value: 32000. Typical value: N/A

4.19 W – Fault input digital filter

The W-parameter controls the digital filtering of the fault signal input.
Normally the fault input signal is polled (checked) before executing the servo-loop (1500 times per second by default). If the input is active (logic low level) the operation of the servo-loop is aborted and the module enters fault state. With the W-parameter it’s possible to specify for how many concecutive servo-loop cycles the fault input has to be active before the module actually enters fault state. Running the servo-loop at 1500Hz and setting the W-parameter to 15 means that fault input must be active for 10ms before the module aborts servo operation and enters fault state.

W5<ENTER>

```
(W)-Fault input filter: 5
```

The W-parameter also provides the option to completely disable the fault input. This is done by setting the value to 0. This however is not recommended for normal applications.

Default value: 1. Minimum value: 0. Maximum value: 255. Typical value: 1-10

Note: This feature was added in firmware version 1.1

4.20 X – Step multiplier

The step multiplier can be used to “gear up” the motor velocity based on the frequency of the incomming step-pulses. This can be useful when the step-generating source has a limited top frequency while, at the same time, the motor is equipped with a high resolution encoder.

For each incoming step pulse the internal setpoint register, or target position, is incremented (or decremented depending on the direction signal) by 1 * *Step multiplier setting* counts.

The benefit of using a high resolution encoder even with a “slow” step signal source is that the high resolution encoder provides more “information” per shaft revolution for the PID-filter to “work with”. A very small motor shaft displacement will get noticed and corrected while, with a lower resolution encoder, that same displacement may not result in even a single encoder count.

X2<ENTER>

```
(X)-Step multiplier: 2
```

Default value: 1. Minimum value: 1. Maximum value: See note. Typical value: 1-5

Warning: Although it is possible to set the step multiplier to any integer value it’s recommended to keep it below 10. In all cases should *Step frequency * Stepmultiplier / servo loop update rate* be below 1000. You must also make sure that the resulting encoder frequency is kept below 2.5Mhz or the value to which the encoder filter setting is set – which ever is lowest. (See the Y-parameter).

4.21 Y – Encoder filter

This sets the internal divider ratio of the encoder filter.
The servo modules microcontroler have a digital filter incorporated on the encoder signal inputs which can be setup to provide different amount of filtering. There are 7 different settings available:

0: 2.5Mhz (default)
1: 1.25Mhz
2: 625kHz
3: 156kHz
4: 78kHz
5: 39kHz
6: 19.5kHz

Y2<ENTER>

```
(Y)-Enocder filter: 1.25MHz
```

Entering a value higher than 6 will result in the filter getting set to 19.5kHz.

If your motor is rated 3500rpm and has a 2000counts/rev encoder (500 lines) the resulting encoder count frequency will be 3500 * 2000 / 60 = 117kHz. If filtering is needed, set the Y-parameter to the nearest higher value which, in this case, would be 3 (156kHz).

Default value: 0. Minimum value: 0. Maximum value: 6. Typical value: 0

On a properly designed system there should be no need for any filtering. If you don't experience any problems that may be due to noise on the encoder signals you can safely leave the Y-parameter at 0 (default).

Differential signals transmitted over twisted pair cables with line drivers and receivers at their respective ends are highly recommended, especially when using any combination of high resolution, high speed and long cables.

4.22 Z – Data recorder mode

The Z-command is used to enter & setup as well as to exit or abort the data recorder mode of the module. To activate the data recorder mode send Z followed by a numerical value that is built up of the following three parts: The sample rate, the “trigger source” or when to start recording and the source or “data” to be recorded.

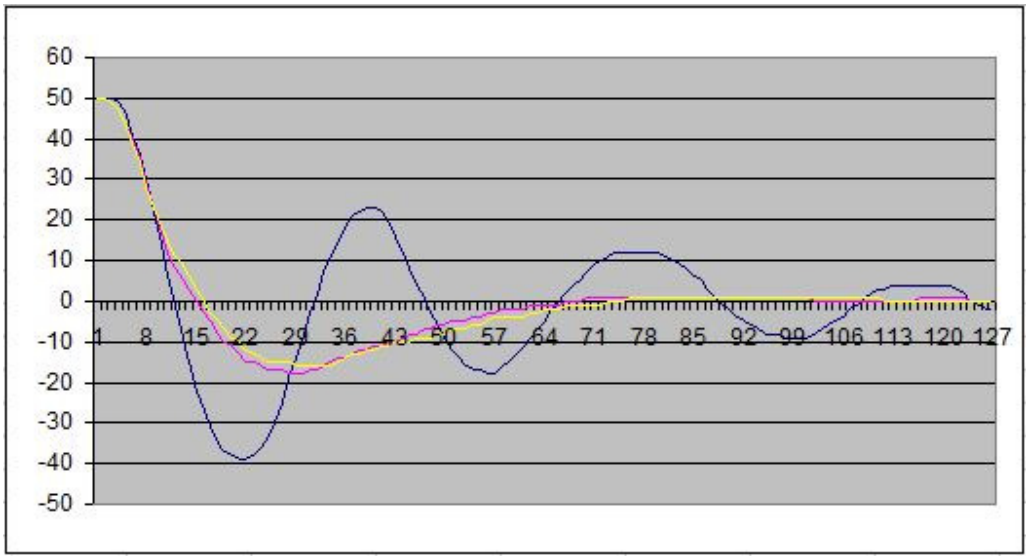
Sample rate	Trig source	Data source	Value
1:1	-	-	0
1:2	-	-	1
1:4	-	-	2
1:8	-	-	3
1:16	-	-	4
1:32	-	-	5
1:64	-	-	6
1:128	-	-	7
-	Vel cmd > 0	-	0
-	Dir change (cmd)	-	8
-	N/U	-	16
-	No Trig *	-	24
-	-	Following error	0
-	-	Motor velocity	32
-	-	Velocity cmd	64
-	-	PID Output	96
-	-	PWM Output	128
-	-	Scaled limit	160
-	-	N/U	192
-	-	N/U	224

Sample rate is simply how often you want the module to record. Maximum sample rate is whatever the servo loop frequency is set to (see the L-parameter). If the L-parameter is set to 4 for a servo loop frequency of 1800Hz, selecting the 1:16 sample rate will sample at 1800/16=112.5Hz. The data buffer is 128 entries “deep” so sampling at 112Hz means ~1.1seconds will fit in the buffer.

Choose one item in each column and add the values for each of them together.
Example: To sample the PWM output at a rate of 1:32 starting on the next servo loop send Z157<ENTER> (1:32 = 5, No Trig = 24, PWM Output = 128, 5+24+128=157)
Once the trig condition is met the module will start to record the value you've selected into the buffer. Once the buffer is full (128 entries) the recording will automatically stop and the data will get sent to the host. You can then copy-paste them into EXCEL etc.

To abort the data record mode simply send Z<ENTER>

The data recorder mode can be used to analyze the step response of the motor and evaluate different tuning settings. Activate the data recorder mode with the trig source set to 0 and use the + or – commands to issue 'instant' relative moves. Here's an example EXCEL plot of three such moves with different PID parameters:



Note: If an error occurs while the data recorder is active the recording will be inhibited (because the internal servo loop is not running). If the module is issued a soft reset (the R-command) the recording will continue and the data will be sent once the buffer is full. If you want to abort the recording simply send Z<ENTER> before you issue the soft reset.

4.23 ? – Menu

This prints the menu showing all changeable settings and their current values.

4.24 +/- - Relative move

Enter +100 to make a move corresponding to 100 encoder counts in the positive direction. Enter -75 to make a move corresponding to 75 encoder counts in the negative direction. This is an “instant” or “step” move, no trajectory planning (acceleration / deceleration) are being performed.

+125<ENTER>

```
Move +125
```

The + and – commands can be useful in conjunction with the data recorder mode to anaylze the step response of the system, see the Z-command.

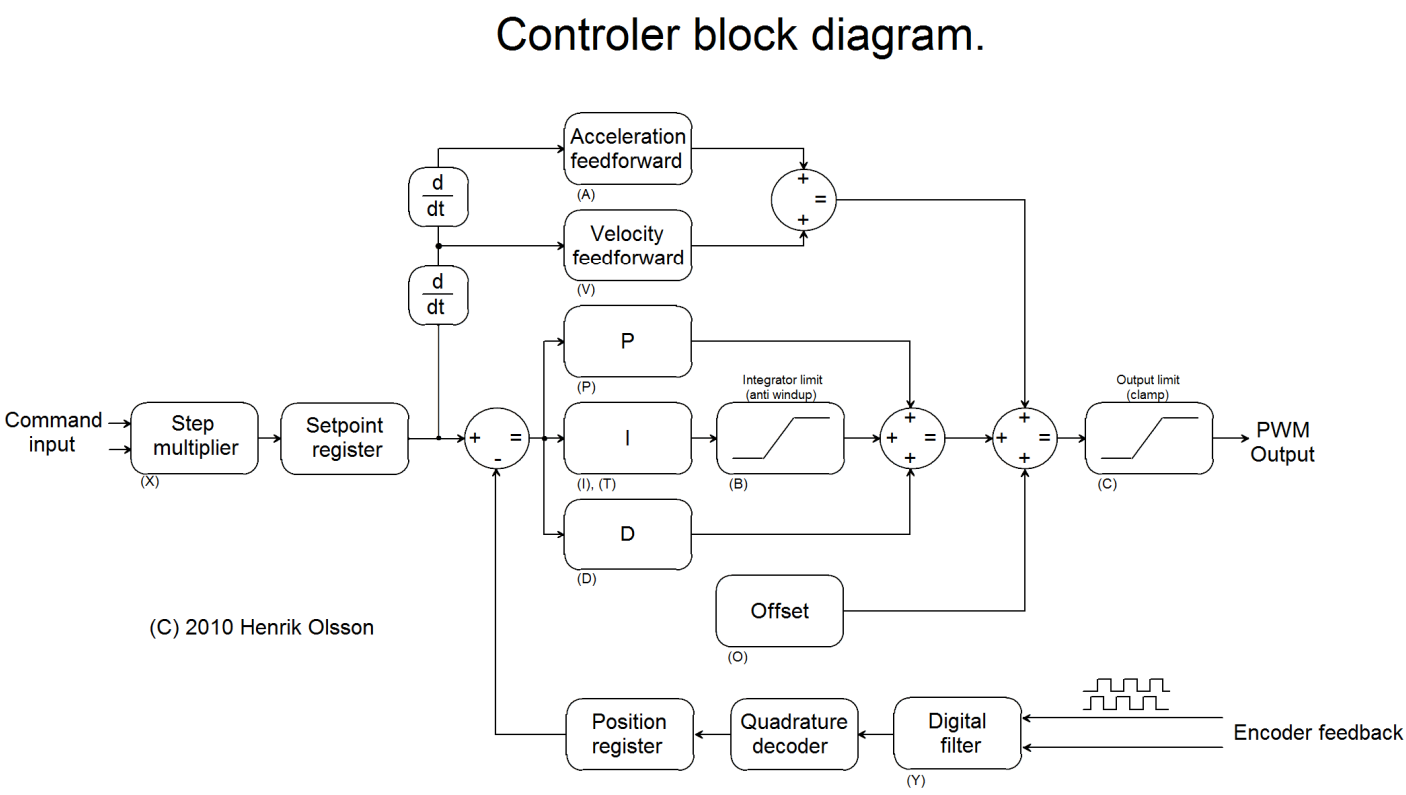
Note: If the commanded move is larger than the currently set following error trip limit (the E-parameter) the module will enter fault state immediately. The step-multiplier setting has no effect on moves issued with the + and – commands, all moves are in “unscaled” encoder counts.

4.25 - Peak Error

The peak error shown on the last line in the menu is not a setting or parameter. It’s simply the largest following error detected since power-up or module reset. It’s just another diagnostics tool which can be useful during setup

Appendix A – Controler block diagram

Below is a graphical representation of the controler.



Appendix B – Following error scale

The following error scale setting allows the following error trip limit to be scaled proportionaly to the commanded motor velocity.

Each servo update cycle the commanded velocity is calculated in encoder counts. This value is then multiplied by the following error scale value (the F-parameter) and divided by 256. Finally it is added to the value of the E-parameter and compared to the current difference between target position and actual motor position (ie. the following error).

Let's say your motors rated velocity is 2500rpm and that you have a 4000 counts/revolution (1000 lines or cycles) encoder mounted to it. Assuming the L-parameter is being set for 1220Hz servo loop update rate this means that at 2500rpm you'll get 2500 * 4000 / 60 / 1220 = 137 counts per servo update.

At stand still you may want to have a trip limit of 10 counts (0.9°) so you set the E-parameter to 10. But at full speed you may want to allow up to 50 counts (4.5°) following error before faulting. Calculating the value of the F-parameter in this case is simply a matter of taking (50 - 10) / 137 * 256 = 75

How does it work?

At motor velocity = 0 the trip limit is 10, at motor velocity 137 the trip limit is 10+(137*75/256)=50 counts.

At 1250rpm (50% of the motors rated speed) the velocity in encoder counts/servo update loop is 68 so the trip limit becomes 10+(68*75/256)=29 or roughly "half way" between 10 and 50.

If you set the F-parameter to 0 the value of the E-parameter is your following error trip limit at all motor velocities – no scaling will be applied.

Appendix C – Motor / encoder settings

In this appendix I plan to provide some baseline settings for various motor/encoder combinations. Obviously every application and motor/encoder/drive/powersupply combination is unique so these are shown for reference only and may (or may not) serve as a starting point.

Motor model: MAE 642 0860
Framesize: ~NEMA34
Power supply voltage: 55VDC
Armature resistance: 0.25ohm
Armature inductance: 0.42mH
Continous current: 8A
Peak current: 40A
Torque constant: 0.0816Nm/A

Encoder	AMT 102 (changable resolution)			
Line count:	500	1000	2048	3600
P				
I				
D				
L				
J				
K				
A				
V				

Motor model: Indramat MDC 10.30D
Framesize: ~NEMA56
Power supply voltage: 130VDC
Armature resistance: 0.24ohm
Armature inductance: 1.2mH
Continous current: 19A
Peak current: 150A
Torque constant: 0.47Nm/A

Document revision history.

2011-06-04. Added W-parameter. Added min, max & typical values. Minor cleanup.

2010-09-01. Added controller diagram, B-parameter and minor changes.

2010-08-20. Minor fixes, no additions.

2010-06-10. Minor fixes and additions.

2010-06-06. First release of manual.