Hugin motor software user's guide

Document name: Hugin motor software user guide_revG.pdf.
April 2003 – Revised January 2009

SmartMotor
Energy Efficient Technology

Smart Motor AS

Dep.: R&D
Address: N-7041 TRONDHEIM
Phone: +47 73 98 25 00
Telefax: +47 73 98 25 01

REVISION HISTORY

REVISION	DATE	HIGHLIGHTS
A	2003.04.22	First version of this document created (WG)
B	2003.08.16	-State machine and power up sequence updated -Virtual function table updated -Control variables table updated. -Control variables description updated (KJ)
C	2003.10.27	-Updated error- and warning tables. -Control variables table updated. -Description of the rudder variables updated. (KJ)
D	2003.11.13	-Variable ranges -Motor state sequence table -State machine figure. (KJ)
E	2004.01.12	-State machine figure updated -Description of state machine -Description of speed reference ramp function. -Range of speed ref and applied speed ref. (KJ)
F	2004.03.11	-New variable, Lock_Propeller_Flag, in control variables table -Description of Lock_Propeller_Flag (KJ)
G	2006.08.01	-New variable, Enable2RudderControl in control variables table. -New variable, DisableRudderOutput, in control variables table. -New function, ResetSW in control functions table. -Description of the above. -Rearraranged the descriptions of variables to coincide with the order in the table. (KJ)
H (applies to sw version 1.13)	2009.01.29	-New variable, VoltageSystem30 in control variables table -Description of the above -Change of error and warning level DC low voltage -Swapped sign of rudder offset -Changed the duty cycle control for rudder outputs.*(see version history for elaboration) -Added a version history to the document (KJ)

Contents

Hardware summary

DSP settings

CPU clock	**150 MHz**
Internal flash	256 kB
Internal RAM	36 kB
External Flash	1 MB
External RAM	1 MB
Serial port A	Enabled, 115200 baud
Serial port B	Disabled
CAN	Disabled
Event manager A	Enabled, 75 MHz
Event manager B	Enabled, 75 MHz
AD converter	Enabled, dual sampling mode

I/O and AD

ADCINA0: Phase current A, range [-80 A, 80 A]
ADCINB0: Phase current B, range [-80 A, 80 A]
ADCINA1: DC BUS current, range [-80 A, 80 A]
ADCINB1: DC BUS voltage, range [0, 80 V>
ADCINA2: Rudder 1 current, range [-1.9 A, 1.9 A]
ADCINB2: Rudder 2 current, range [-1.9 A, 1.9 A]
ADCINA3: Rudder 1 position
ADCINB3: Rudder 2 position
ADCINA4: Rudder 3 current, range [-1.9 A, 1.9 A]
ADCINB4: Rudder 4 current, range [-1.9 A, 1.9 A]
ADCINA5: Rudder 3 position
ADCINB5: Rudder 4 position
ADCINA6: Temperature sensor 2, heat sink, range [2.5 °C, 100 °C>
ADCINB6: Temperature sensor 1, electronics [2.5 °C, 100 °C>
ADCINA7: 15 V DC voltage, range [0,33 V>

GPIOF0: Rudder 3 enable, active high
GPIOF1: Rudder 4 enable, active high
GPIOF10: Rudder 1 enable, active high
GPIOF13: Rudder 2 enable, active high
GPIOF11: Gate drive enable, active low
GPIOB0: Status LED 0
GPIOB1: Status LED 1
GPIOB2: Status LED 2
GPIOB3: Status LED 3

GPIOE0: Jumper JP1
GPIOE1: Jumper JP2

Motor control software

The motor control software is based on a sensorless control algorithm that uses the information that is available in the back emf from the motor. Since the back emf is only present when the motor is rotating, and its amplitude is proportional to the rotating speed, the algorithm will not work at very low frequencies. The motor is started using a standard u/f control, where the frequency is slowly ramped up to the desired value, while the applied voltage amplitude is increased proportionally with the frequency. After the motor has reached a certain speed, the flux estimators are enabled.

The motor control software contains the following main parts:

Current regulators

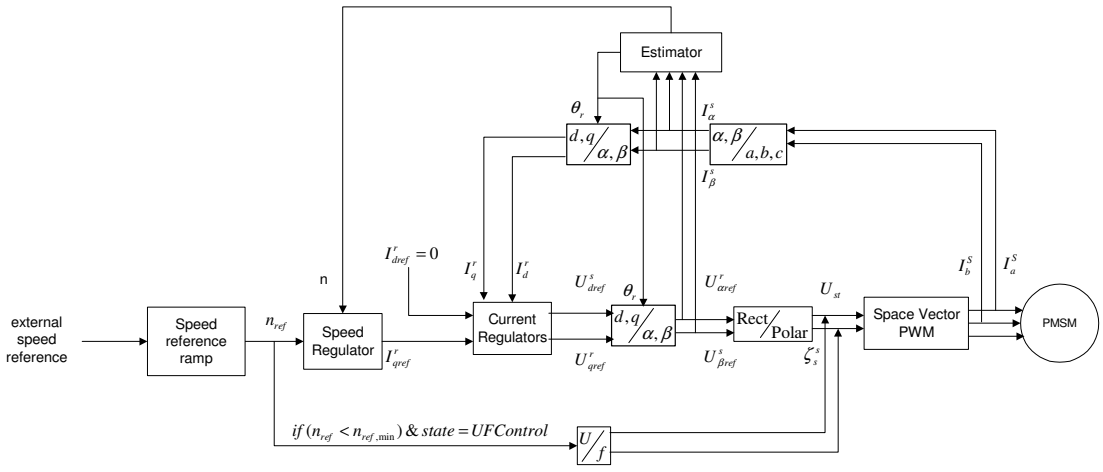
Speed regulator

Speed ramp function

Position estimator

Speed estimator

Space vector PWM modulator



Code generation tools

To compile and build the software, Code composer studio v2.20 is needed. This tool is available from Texas Instruments.

Flash programming

1. The startup, commros, reset vector and sine table has to be programmed in to the DSP's internal flash memory, this has to be done using a JTag and Spectrum digitals SDFlash program. This part of the flash programming is done only once (if the flash algorithms and software loader functions are unchanged).

2. The motor software, system software and rudder control software is placed in the external flash memory. This part of the program is programmed in to the external flash memory using the External flash-programming wizard in DSPComm. When new motor software is released, only this part of the software has to be reprogrammed.

Interrupts

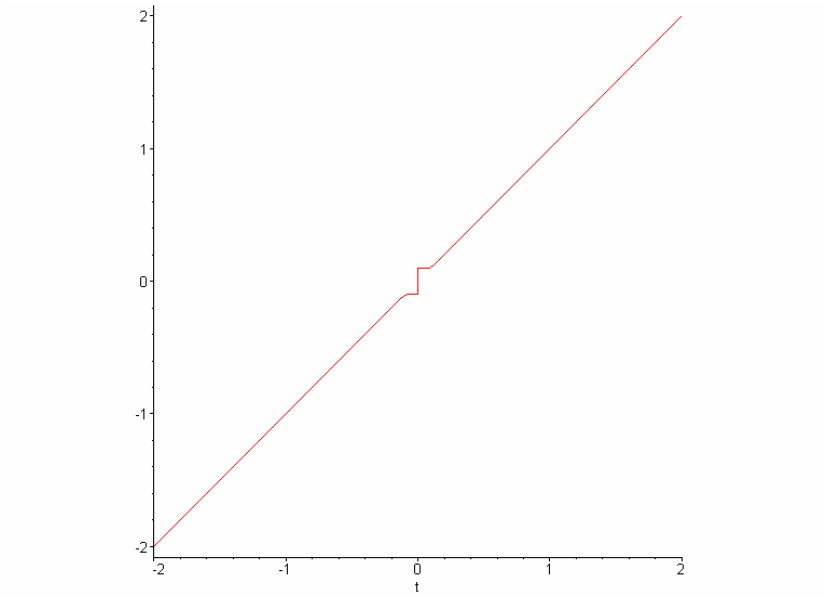
Interrupts are used to time many events on the DSP, the following interrupts are being used:

- AD interrupt
  - Interrupt frequency: Same as the switching frequency
  - Function:
    - Motor state machine
    - Current regulators
    - Current transformations
    - U/f control
    - Space vector PWM
- Timer 1 underflow interrupt
  - Interrupt frequency: Same as the switching frequency
  - Function:
    - Error control
    - Speed regulator
    - Speed reference ramp function
- CPU Timer 1 interrupt:
  - Interrupt frequency: Once every hour
  - Function:
    - Increase the system clocks hour count by 1.
- CPU Timer 2 interrupt:
  - Interrupt frequency: Same as rudder controls sampling frequency
  - Function:
    - Rudder control
- Serial port A interrupt:
  - Interrupt buffers incoming serial port data.

Functions used in SW

Speed reference ramp function

The external speed reference is passed through a ramp function before it is sent to the speed regulator. This ramp function has the following speed/time characteristic



The slope of the ramp function can be adjusted. Note that this function also limits the applied speed ref to its upper values given in the description of this variable, i.e. the characteristic is flat on the top and bottom. If a reference larger than 0, but less than the lower limit is given, the applied reference will be the lower limit.

Current regulators

Digital PI regulator with anti-windup.

Speed regulator

Standard PID regulator.

Position estimator

The rotor position is estimated using a sensorless algorithm that uses the information given from the back emf, and knowledge of motor parameters, currents and DC voltage.

Speed estimator

The rate of change in the estimated position is used to estimate the speed.

Space vector PWM modulator

The SVPWM modulator calculates the PWM outputs that corresponds to a given voltage vector. This modulator is based on standard theory. If the amplitude voltage reference vector from the current regulators exceeds the maximum limit, it is saturated at the limit value.

Flying start

The flying start algorithm makes it possible to start the motor if it is rotating, without risking over current trip.

System clock

The software is using one of the DSP's 32 bit timers as a system clock; this clock has a resolution of 1 μs. The system clock gives the time from the start command was given, if the program is stopped and started again, the system clock will be reset to zero. The system clock is available through virtual variables.
The system clock is used to time stamp events in the system log.

System log

The system log keeps a record of the following events:
- The state machine enters a new state
- Warnings
- Errors

All logged events are time stamped using the system clock.
The system log is placed in external RAM, and can be copied to the external flash using the virtual function "Save system log", this can only be done after the motor software has been stopped, if the motor software is running, a "Access denied" error will be returned.

State machine and power up sequence.

After a reset or power up of the DSP, certain functions are called. These functions initialize the DSP, verifies the program that has been programmed in to flash, and moves time critical code from flash in to faster internal or external RAM. If the flash memory has been erased, and not reprogrammed (contains only 0xFF), a “program not loaded” error will be given.
After a start command is sent to the DSP, the motor control software is started (if program exists). Before it is possible to run the motor, parameters are initialized from the external EEPROM, and sensors are being calibrated. After the calibration is finished (~0.14 s), the motor software is in the “Ready” state. To start the motor, it is necessary to run the virtual function “Start motor”, after this function has been run, the motor software will try to perform a flystart. If the flystart fails, the motor state will be set to “UFControl”(unless the speed reference is 0, then it will be set to “Stopped”), and the applied speed will follow the reference. When the applied motor frequency is higher than 12 Hz, the motor state is set to “Run”. The motor is now in normal speed regulated operation using flux estimators. The motor will stay in this state until the speed reference is set to zero, or the virtual function “Stop motor” is run. If any errors occur, the motor software will be put in the “Error state”.
If the error disappears, the respective error flag will be cleared after one second. To get the state machine out of “Error state”, one must run the virtual function 3, “Clear error”. If the error endures, one should always try to find the cause, and correct the error before bringing the state machine out of “Error state”. However, it is possible to run virtual function 4, “Start with error”, to mask out the respective error flags, and forcing the state machine back to ready. *SmartMotor cannot take responsibility for the electronics if this function is used, and the error hereby causes damage to equipment.*

In figure 1, the state machine is described graphically. Note that the check for applied UF-speed ref is just and internal variable, and will not be reflected in the applied speed ref read from the controller. Also note that the state machine will not go by Ufcontrol from RUN to STOPPED. This is because the Ufcontrol have no knowledge of the rotor position.

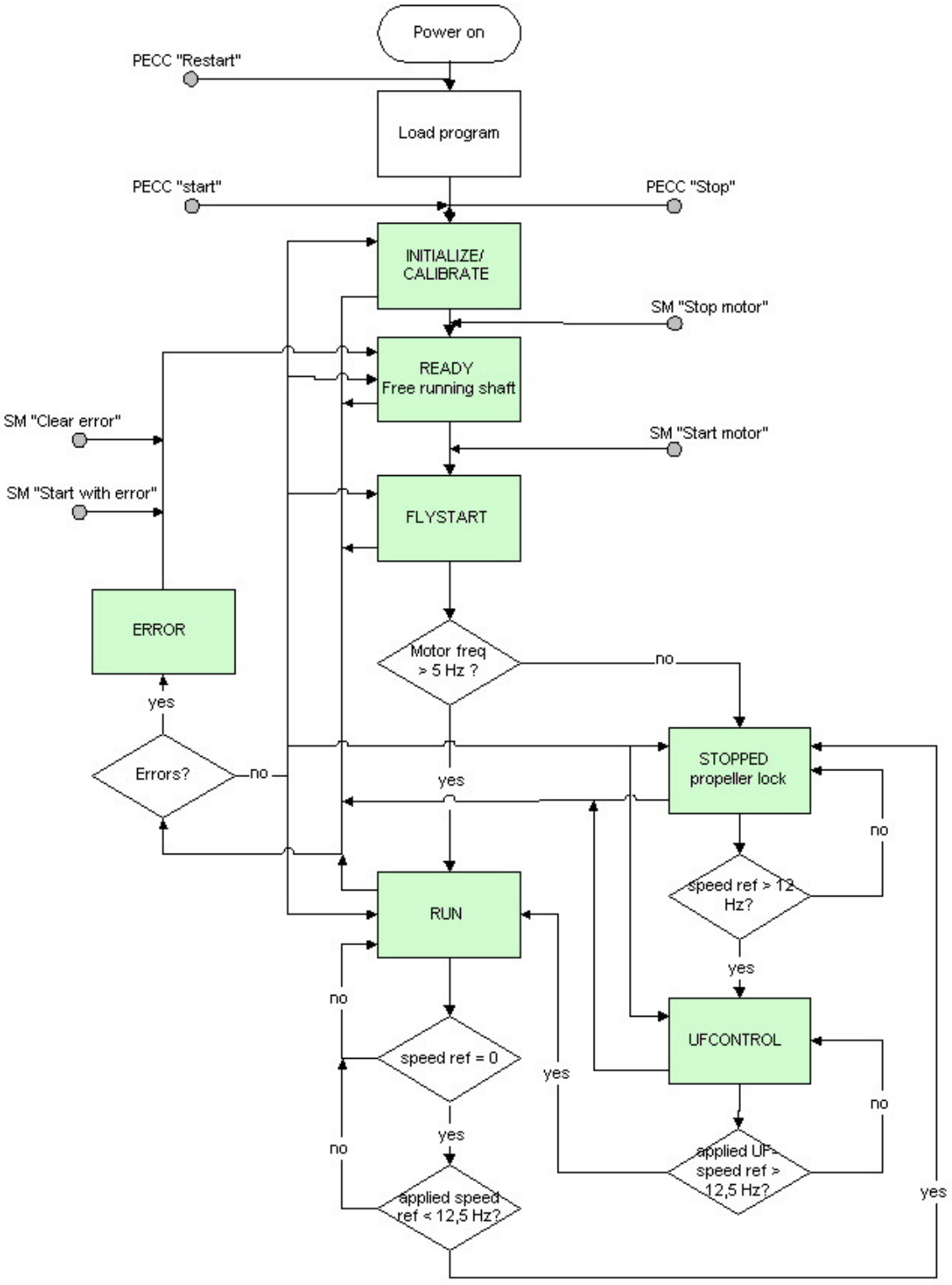


Figure 1. Hugin application state machine.

Virtual control functions

The following virtual functions are available in the SW:

Address	Name	Description
0	Start motor	The motor state machine will go from ready to flystart-stopped-ufcontrol-run
1	Stop motor	The motor state machine will go to Ready (except if it is in Error state)
2	Save system log	The system log is programmed in to flash; this function can only be run after the program has been stopped.
3	Clear error	The motor state machine will go from error to ready, provided all error flags are cleared.
4	Start with error	The active error flags will be masked out, and the state machine brought to ready. This is only to be used in an emergency. USE WITH CAUTION!
5	ResetSW	Running this function will cause the software to reset as if the power was reset. Requires the communication to function.

Virtual control variables

All control variables are accessed through the serial communication interface.

Virtual address	Access (**R**ead / **W**rite)	Variable
Motor Control		
0	R/W	Speed reference
1	R	Applied speed reference
2	R	Estimated speed
3	R	Temperature PCB
4	R	Temperature Heat sink
5	R	48 V DC voltage
6	R	15 V DC voltage
7	R	Motor torque
8	R	DC current
9	R	DC Power
Rudder control		
10	R/W	Rudder Top offset
11	R/W	Rudder Bottom offset
12	R/W	Rudder Starboard offset
13	R/W	Rudder Port offset
14	R/W	Rudder Top reference
15	R/W	Rudder Bottom reference
16	R/W	Rudder Starboard reference
17	R/W	Rudder Port reference
18	R	Rudder Top position

19	R	Rudder Bottom position
20	R	Rudder Starboard position
21	R	Rudder Port position
Software control		
22	R/W	Timeout
23	R	Software version
24	R	Software state
25	R	Error word
26	R	Warning word
27	R	Status word
System clock		
28	R	System clock hours
29	R	System clock minutes
30	R	System clock seconds
31	R	System clock milliseconds
32	R	System clock micro seconds
New additions after first release		
33	R	Remaining torque
34	R/W	Rudder deadband
35	R/W	Rudder max deflection
36	R/W	Lock propeller flag
37	R/W	Enable2RudderControl
38	R/W	DisableRudderOutput
39	R/W	VoltageSystem30

Speed reference (Read/Write)

Size: 32 bit, signed
Format: Q.20

This variable is the speed reference in pu Q.20 format.

Range: <-q3.0, q3.0>
Error range: <-∞, -3.5> and <3.5, ∞>

Table 1: Speed reference encoding/decoding

Q.12	**Decimal view**	**RPM**
-2.0	-2097152	-353
-1,5	-1572864	-264.75
-1.0	-1048576	-176.5
-0.5	-524288	-88.25
-0.1	-104858	-17.6
0.1	104858	17.6
0.5	524288	88.25
1.0	1048576	176.5
1.5	1572864	264.75
2.0	2097152	353

Applied speed reference (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-q1.45, -q0.24] and [q0.24, q1.45]

This variable is the output from the speed reference ramp function, and the input to the speed regulator. Value's are in the same format as the "Speed reference".

Estimated speed (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-2^12, (2^12-1)]

This variable is the speed estimate from the sensorless speed estimator. The format used is the same as for the Speed reference.

Temperature PCB (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-2^12, (2^12-1)]

This variable is the temperature on the electronics board in Q.20 format, i.e. q30.0=30.0 degree Celsius.

Temperature Heat sink (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-2^12, (2^12-1)]

This variable is the temperature on the heat sink for the transistor in Q.20 format, i.e. q30.0=30.0 degree Celsius.

48V DC voltage (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-2^12, (2^12-1)]

This variable is the voltage on the DC-link (battery voltage) in Q.20 format, i.e. q48.0=48.0 Volt.

15V DC voltage (Read)

Size: 32 bit, signed
Format: Q.20

Range: [-2^12, (2^12-1)]

This variable is the voltage on the internal 15V supply in Q.20 format, i.e. q15.0=15.0 Volt.

Motor torque (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-2^12, (2^12-1)]

This variable is the applied torque on the motor axis in Q.20 format, i.e. q15.0=15.0 Nm.

DC current (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-2^12, (2^12-1)]

This variable is the current that is drawn from the DC-link in Q.20 format, i.e. q5.0=5.0 A.

DC power (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-2^12, (2^12-1)]

This variable is the power used by the motor drive in Q.20 format, i.e. q200.0=200.0 Watt.

Rudder (x) offset (Read)

Size: 32 bit, signed
Format: Q.20
Default: q0.0
Range: [-2^12, (2^12-1)]

This variable is the offset value sent to the respective rudder to tune the potmeter feedback/rudder motor mounting. The default value of q0.0 will give a centered rudder provided correct mounting of the motor and potmeter. The value entered here is the desired offset in degrees; i.e. q2.5 = 2.5 degrees.

Note: from version 1.13, the sign of the offset is swapped.

Rudder (x) reference (Read/write)

Size: 32 bit, signed
Format: Q.20
Default: q0.0
Range: [-2^12, (2^12-1)]

This variable is the reference position for the respective rudder. A value of q0.0 will turn the

rudder to its center position, given a correct calibration of the rudder offset. The value entered her is the desired position in degrees; i.e. q2.5 = 2.5 degrees.

Rudder (x) position (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-0.9*RudderDeflection, 0.9*RudderDeflection]

This variable is the position of the rudder from the potmeter feedback in degrees; i.e. q12.5 = 12.5 degrees. The value is restricted to 0.9 times the max deflection of the rudder potmeter. A value out of range will disable the respective rudder drive, and give a warning in the warning word. The value is scaled by the max deflection parameter.

Timeout (R/W)
Size: 32 bit, unsigned
Format: Decimal
Default: 0
Range: [0, 2^32]

This variable is used as a watchdog in case of a communication failure. If a value different from zero is written to this variable, a count down from this value will start. The count down will have the same frequency as the switching frequency. When it reaches zero, an error flag will be set. Writing zero to this variable disables this function.

Software version (Read)
Size: 32 bit, unsigned
Format: Decimal
Range: [0, 2^16]

This variable contains the software version number. To get the correct number, divide the variables value by 100, e.g. Software version = 112 => Version 1.12.

Motor state (Read)

Size: 32 bit, unsigned
Format: Decimal

Value	Name	Description
0	Start_SW	Initialisation phase
1	Calibrate	Sensor calibration
2	UFControl	Low speed start up state.
3	Run	Normal speed regulated operation.
4	Ready	Ready, inverter and plane outputs disabled.
5	Stopped	Motor phases short-circuited
6	Flystart	
7	IDRun	Estimates machine parameters like phase resistance, and inductance.
8	DCInitialize	DC current rotor alignment state

99	Test state	
100	Error	Safe error state, inverter and plane outputs disabled.

Error word (R)

Size: 32 bit, unsigned
Format: Hex

The error word variable gives the reason of errors in the drive, according to the table underneath. The according bit is cleared after one second when the value re-enters the non-warning range.

Value (Hex)	**Error**	**Limit**
0x00000001	High current phase A	74.2 A
0x00000002	High current phase B	74.2 A
0x00000004	High current phase C	74.2 A
0x00000008	Low DC voltage	18.0 V
0x00000010	High DC voltage	55.0V
0x00000020	High temperature; heat sink	80.0 Deg
0x00000040	High temperature; PCB	75.0 Deg
0x00000080	Reserved	
0x00000100	Reserved	
0x00000200	Reserved	
0x00000400	Reserved	
0x00000800	Reserved	
0x00001000	Reserved	
0x00002000	Reserved	
0x00004000	Reserved	
0x00008000	Reserved	
0x00010000	Reserved	
0x00020000	Motor state machine; illegal state	
0x00040000	Reserved	
0x00080000	Communication timeout; Hugin	Timeout = 0

Warning word (R)

Size: 32 bit, unsigned
Format: Hex

The warning word variable notifies the user of triggered warning levels according to the table underneath. The according bit is cleared after one second when the value re-enters the non-warning range.

Value (Hex)	**Warning**	**Limit**
0x00000001	High current phase A	63.1 A
0x00000002	High current phase B	63.1 A
0x00000004	High current phase C	63.1 A
0x00000008	Low DC voltage	20.0 V

0x00000010	High DC voltage	50.0 V
0x00000020	High temperature; heat sink	60.0 Deg
0x00000040	High temperature; PCB	60.0 Deg
0x00000400	Rudder timeout	Not reached position after 1 minute.
0x00000800	Rudder deflection	Potmeter value > 0.9*range

Status word (R)

Size: 32 bit, unsigned
Format: Hex

The status word variable gives different status messages according to the table underneath.

Value (Hex)	**Warning**
0x00000001	Flystart OK
0x00000002	Flystart failed
0x00000004	Positive current saturation active
0x00000008	Negative current saturation active

System clock (R)

The system clock gives the elapsed time since the start command was given. If a stop command is sent, the system clock will stop, and it will be reset at the next start command.

Remaining torque (Read)

Size: 32 bit, signed
Format: Q.20
Range: [-2^12, (2^12-1)]

This variable gives an estimate of how much more torque the drive can provide at the current speed and voltage level. This is not an accurate value, merely an indication. The variable is in Q.20 format, i.e. q15.0=15.0 Nm.

Rudder deadband (Read/write)

Size: 32 bit, signed
Format: Q.20
Default: q0.4
Range: [-2^12, (2^12-1)]

This variable is the hysteresis deadband variable in degrees; i.e. q0.5 = 0.5 degrees deadband. The value is internally scaled by the rudder deflection paramter.

Rudder max deflection (Read/write)

Size: 32 bit, signed
Format: Q.20

Default: q25.0
Range: [-2^12, (2^12-1)]

This variable scales the potmeter reading to an angle value, and is the maximum deflection of the potmeter in degrees.

Lock propeller flag (Read/write)

Size: 32 bit, unsigned
Format: bool
Default: true
Range: [true, false]

This variable flags the wanted state of the propeller in the Stopped state. If true, the propeller is locked (short circuited) in the Stopped state, and if false the propeller is free running (open connections).

Enable2RudderControl (Read/write)

Size: 32 bit, unsigned
Format: bool
Default: false
Range: [true, false]

This flag will disable rudder 3 and 4 (bottom and port). It will also set the position values for these rudders equal to respectively rudder 1 and 2. The flag can only be set/reset in state Ready. When disabling the two rudders, the warning for rudder timeout is also disabled for these two rudders.

DisableRudderOutput (Read/write)

Size: 32 bit, unsigned
Format: hex
Default: 0x0000
Range: 0x0000-0x000f

When writing bits 1-4 to this variable, the corresponding rudder drive will be disabled, either it's a single bit or several. The bits are written in hex, i.e. 0x0001=rudder 1, 0x0002=rudder 2, 0x0004=rudder 3 and 0x0008=rudder 4. Or 0x000f disables all rudders. The position is still read from the position sensor. The variable can be used in state Ready, Run and Stopped. This will also disable the timeout warning for the disabled rudders.

VoltageSystem30 (Read/Write)

Size: 32 bit, unsigned
Format: bool
Default: false
Range: [true, false]

This flag can only be set in Ready state. The flag enables the 30 V system. When set, it also sets the Enable2RudderControl flag, hence rudder 3 and 4 will be disabled (see description).
Differences of voltage modes:
48 V
- nominal rudder voltage is 48 V.
- all rudders enabled (user may disable rudder at choice by using Enable2Rudder or DisableRudderOutput).

30 V
- nominal rudder voltage is 24 V.
- rudder 1 & 2 enabled.

Version history

Version history for Hugin 1000 software
2003 (c) SmartMotor AS

version 1.00

- First Release

version 1.01

- Fixed bug with no short circuit of motor phases in stopped state; Propeller lock.
- Changed motor state machine according to Hugin 3000 SW. New state machine:
  - -0- Start_SW
  - -1- Calibrate
  - -2- UFControl
  - -3- Run
  - -4- Ready
  - -5- Stopped
  - -6- Flystart
  - -7- IDRun
  - -8- DCInitialize
  - -99- Teststate
  - -100- Error

version 1.02

- implemented SW filtering on rudder position measurements

version 1.03 (20031029)

- Fixed bug that didn't disable rudder drives inside deadband. Bug was introduced with the new state machine in version 1.01

version 1.04 (20031030)

- Implemented timeout for rudder drives; if not in position after 1 minute => disable and warning.
- Implemented max deflection limit for rudders at 90% of potmeter deflection. => disable and warning.
- Scaled the rudder postion similar to Hugin3000 SW. That is; Position=ScaleFactor*PotmeterDeflection [Deg].
- Also rescaled rudder position offset variables to be given in degrees.
- Included virtual variable "RudderScale" - factor to scale from potmeter deflection to rudder angle.
- Included virtual variable "RudderDeadband" - Gives the user the option to adjust the deadband.

version 1.05 (20031201)

- Scaled the rudder regulator parameters according to variable PotmeterDeflection.
- Fixed rudder deflection warning to be set when either one of the rudders are out of bounds. (Not all four of them at once).
- Changed rudder deflection check to apply to position reference, not position.
- Changed scaling of potmeter reading to comprise the whole digital range; i.e. changed gain scaling.

version 1.06 (20040210)

- Implemented SW filtering on heatsink temperature, to avoid noise peaks. This was already done for PCB temperature.

- Fixed bug in error check for heatsink temperature. This check tested the PCB temperature for error flagging, instead of heat sink.

version 1.07 (20040311)

- Implemented a SW flag for switching between locked/unlocked propeller in stopped state. This was done to conform with Hugin 3000 SW.

version 1.08 (20040428)

- Bug fix; Rudder deadband was only updated after a change in the Rudder Scale variable. This is now updated on the fly.
- Rudder deadband set to 0 degrees as default.
- Rudder regulator integral effect set to 0.
- Rudder regulator proportional effect increased to increase bandwith. This was erroneous initialized. Tested with a 2 degree/1 Hz sine as reference.

version 1.09 (20040715)

- Doubled frequency of rudder switching. This was done due to problems with high ripple current in rudder motors. The doubled frequency gives half the ripple current.
- Implemented ramp function for rudder modulation such that the time from zero to full modulation is approximately 35 ms. This decreases the starting current to app. 0.5 A. As a result, the acceleration of course is decreased.
- The rudder control interrupt time (updating of rudder references) is decreased from 20 ms to 1 ms. This was done to be able to control the starting current.

version 1.10 (20040920)

- Changed warning limit for over voltage from 50.0 V to 50.5 V

version 1.11 (20051130)

- Implemented some changes in the calibration of the temperature sensors to get better accuracy over the full measuring range.

version 1.12 (20060801)

- New virtual variable: Enable2RudderControl. Makes it possible to disable rudder 3&4 for use with Hugin3000 rudder unit.
- New virtual variable: DiasbleRudderOutput. Makes it possible to disable rudder outputs of choice.
- New virtual function: resetSW. Resets the SW for use in case of SW hangups.

version 1.13 (20090129)

- The sign of the rudder_offset is swapped. This to ensure similar operation to the Hugin 3000 system.
- Voltage limits are lowered to respectively 18 V and 20 V for error and warning. This to protect the driver electronics, rather than the motor. Thus also allowing the same electronics to be used for the 30 V system.
- The duty cycle of the rudder output voltage are changed to regard the input DC voltage, in such a way that it outputs the nominal motor voltage for the system until supply voltage drops below this threshold. That is, it holds 24 V/48 V output for the 30 V/48 V system

Appendix I - The Q. format

The Q.x format is a standard method for representing floating point numbers in fixed point DSPs and micro controllers.

16 bit numbers:

Q.x means that the lower x bits are used to represent the decimal part of the number, and the remaining (16-x) bits is used to represent the integer part, and the sign bit.

Example:

$$1.45 \text{ in Q.12} = 1.45 \cdot 2^{12} = \underline{\underline{5939}}$$

$$1.45 \text{ in Q.8} = 1.45 \cdot 2^{8} = \underline{\underline{371}}$$

32 bit numbers:

Q.x means that the lower x bits are used to represent the decimal part of the number, and the remaining (32-x) bits is used to represent the integer part, and the sign bit.

Example:

$1.45 \text{ in Q.20} = 1.45 \cdot 2^{20} = \underline{\underline{1520435}}$

$1.45 \text{ in Q.24} = 1.45 \cdot 2^{24} = \underline{\underline{24326963}}$

Notation:
The notation q1.45 in the respective Q.x format means 1.45*2^x. That is: q1.45 in Q.20 format means 1.45*2^20.