Albis motor controller IC

Setup manual

version 2.A1

Some general remarks:

Before connecting the battery make sure the output stage polarity is set correctly. Use a fuse in the battery line. There should be no diode in the battery line. Make sure the current sensors are connected conform the schematic and that the default calibration values are written. An error here can cause severe damage to the output stage (again, make sure to have a fuse in the battery line).

The menu system was designed using 'gtkterm' under Ubuntu. The baudrate is 115200 baud, 8 data bits, 1 stop bit, no parity, no handshaking.

Under windows the freely available program 'termite' can be used, I tested version 2.8. The main difference between the two is that gtkterm transmits characters as they are typed while termite waits for an return before sending the characters. This means that whenever the chip waits for 'press any key', in termite the chip will only respond correctly when the return key is pressed. In the serial port settings of termite select 'append CR' and disable 'local echo'

The chip can be placed in the setup mode by closing the setup switch connecting pin 19 to ground while resetting the chip. Upon entering setup mode a keypress on the PC is necessary before the chip displays the main setup menu.

The chip keeps track of whether all sub-menu's have been accessed. Putting the chip in motor mode before doing so will make the drive LEDs flash 2 by 2. Correct the setup and try again.

Except for the output stage polarity all data during setup is recorded in RAM. It is only stored in EEPROM for motor use when the correct menu option for this has been selected. Do not turn off the power before writing the new setup to EEPROM !

Some menu options have a yes/no or high/low setting. Selecting this type of menu option toggles the setting, no further input is required.

Every time the chip displays a menu all internal 16 bit variables are translated into decimal numbers. When new numbers are entered the controller IC immediately translates these into a 16 bit number (which it uses for running the motor). The consequence of this is that in between entering a new number and the displaying of the updated menu 2 rounding operations have occurred, first after translating to 16 bits and then after translating back to decimal. This gives an insignificant discrepancy between the entered data and displayed data.

The chip does not perform any checks to see whether data is valid, this is up to the users common sense. Entering negative numbers or letters where positive numbers are required will result in the refreshed menu displaying data not correlated to the entered characters.

Some displayed values are calculated using more than one 16 bit variable, changing one variable can change the displayed decimal value of more than one menu option. Notable examples of far-reaching variables: sensor transimpedance and f_sample

Pressing the setup key while in motor mode will write the current sensor gain and offset calibration values to EEPROM. This is indicated by all four drive LEDs lighting up.

```
########################################
#   (c)opyright 2017, B.M. Putter      #
#   Adliswil, Switzerland              #
#   bmp72@hotmail.com                  #
#                                      #
#  version 2.A1                        #
#  experimental, use at your own risk  #
########################################


0) mode: Hall sensored
a) PWM parameters
b) current settings
c) throttle setup
d) erpm limits
e) battery
f) current sensor calibration
g) control loop coefficients
h) filter bandwidths
i) FOC motor impedance
j) CAN setup
k) recovery
l) hall sensored only
m) temperature sensors
n) miscellaneous
o) online Kv, L and R
y) chip status at last drive_1
z) store parameters in ROM for motor use
```

This is the main setup menu as displayed when the chip is in the setup mode. From this menu all the sub menus can be selected.

Option 0 toggles between the two motor start options of this version: Sensorless
or Hall sensored .

From the main menu the chip can be updated to a newer or older version by selecting '!'. The chip will respond with:

```
------> !

 Chip update has been triggered.
 If you do not want to update chip, power down or reset the controller NOW.

 For update, switch PC to 4800 baud and upload update file.
```

You can still abort the update by resetting, then you will go back to the main menu. All settings will however have been erased !

To update, the baudrate of the PC must be lowered to 4800 baud. Use your terminal program to send the toggle file to the controller IC. If everything is OK the chip will respond with running dots as the chip is updated, when the '*' appears the update has been completed. A reset will bring the chip back to the main menu (make sure to switch your terminal program back to 115200 baud first).

If the chip responds with an 'X' then something went wrong, check the baudrate and make sure the version number of your current chip appears in the name of the toggle file. With '%' in the main menu the checksum of the chip will be calculated. It can be compared to the one provided with the toggle file, to make sure the program memory is correct for that version.

A single toggle file will toggle the controller IC between two version. 'toggle2A0_2A1_.txt' for instance toggles between versions 2.A0 and 2.A1 . If this file were to be uploaded to a v2.93 chip no changes will be made and a 'X' will appear.

The information in the toggle file will be XOR-ed with the memory of the controller IC. This makes it possible to toggle between 2 versions using a single file. It also means that once the chip is corrupted (by for instance a power failure during the update) there is no way to 'repair' the chip.

```
a) PWM frequency: 19kHz
b) deadtime: 499ns
c) dutycycle testsignal: 47%
d) toggle high side polarity, now active HIGH
e) toggle low side polarity, now active HIGH
f) test PWM signals

g) autocomplete

h) loop sample frequency: 25.39 kHz

z) return to main menu
```

This is the setup menu for the output stage. Option a sets the PWM frequency that is used for operating the output stage. The deadtime used between the switching of the high/low side transistors is set using option b. With option c the dutycycle for the testsignal (option f) is set, this option has no effect when the controller is not in setup mode.

Options d & e MUST BE SET FIRST AND BEFORE THE BATTERY IS CONNECTED !!! These options determine whether a high (active HIGH) or low (active LOW) signal is used to turn on the high/low side FET. Unlike all other options (which are only saved to EEPROM when the controller IC is instructed to do so) these options are directly written to EEPROM.

With option f a test-signal having the properties of options a-c is generated. After selecting this option the controller will ask you to press any key before the test signal is turned on. After again pressing a key the test signal is turned off and the controller returns to the menu.

Option g is the autocomplete. In every menu where there is an autocomplete option, all the options below it are autocompleted. In this case option h, which is the frequency at which the control loops operate (the frequency at which the chip makes measurement, calculates the new output signals and writes new values to the PWM output stages). It is recommended to have h close to 25 kHz (by trying different values for a (close to desired PWM frequency) and using autocomplete)

```
      a) current sensor transimpedance: 40.00 mV/A
      b) maximum motor phase current: 49.9 A
      c) maximum battery current, motor use: 19.9 A
      d) maximum battery current, regen: 0.0 A

      e) autocomplete

      f) dr2, dr23: current to check: total current
      g) dr2, dr23: fixed part: 14.9 A
      h) dr2, dr23: proportional to throttle current, factor: 150 %
      i) dr3: maximum shutdown error current, fixed: 12.4 A
      j) dr3: maximum shutdown error current, proportional: 6.2 A
      k) dr3: shutdown based on : I_error
      l) applied braking current (phase) on direction change: 0.0 A
      m) offset filtering (phase) current limit: 0.0 A
      n) maximum field weakening current: 0.0 A

      z) return to main menu
```

This is the setup menu for anything current related. Options a-d are the minimum settings that must be entered by the user, the rest can be autocompleted. Options a sets the current sensor transconductance, option b the maximum motor phase current amplitude, option c the maximum battery current in motor mode and option d the maximum battery (charge) current during regen. Option a can be positive or negative, positive when current flowing from controller to motor results in increasing sensor output voltage, negative for when current flow in this direction causes decreasing sensor output)

Options f, g and h set the error detection during startup (drive_2) and the transition to FOC (drive_23). Total or error current can be checked (option f) with a fixed level (option g) and proportional to throttle (option h).

Options i, j and k set the error detection for drive_3. Option i set the maximum allowed error current at 0 signals to the motor, the sum of options i and j set the allowed error current at max signal amplitude to the motor. With option k the check can be based on I_error or the more strict abs(I_error).

The error current serves as a fault detection mechanism, when the level is violated the controller will default to drive_1 (recovery mode) and the conditions under which this happened are stored in memory.

```
    a) current sensor transimpedance: 40.00 mV/A
    b) maximum motor phase current: 49.9 A
    c) maximum battery current, motor use: 19.9 A
    d) maximum battery current, regen: 0.0 A

    e) autocomplete

    f) dr2, dr23: current to check: total current
    g) dr2, dr23: fixed part: 14.9 A
    h) dr2, dr23: proportional to throttle current, factor: 150 %
    i) dr3: maximum shutdown error current, fixed: 12.4 A
    j) dr3: maximum shutdown error current, proportional: 6.2 A
    k) dr3: shutdown based on : I_error
    l) applied braking current (phase) on direction change: 0.0 A
    m) offset filtering (phase) current limit: 0.0 A
    n) maximum field weakening current: 0.0 A

    z) return to main menu
```

When reverse is used the motor cannot instantly change direction. The mechanical energy of the spinning motor / moving vehicle must be reduced by either letting it spool down freely (option l = 0 A) or by specifying a regen braking current to actively slow the motor down (option l > 0 A). Note that when a braking current is specified, option d must be set to allow for regen current.

In drive_3 the controller can perform auto-offset calibration of the current sensors. This is not always a good idea, especially when currents are high and the sensors are non-linear. With option m the chip only calibrates the current sensors when the phase current is below the set value. By specifying 0 the auto offset calibration is effectively turned off. If auto calibration is used dependent on the motor and current sensors the motor can start bucking after about 30 seconds in drive_3, this is an indication that option m must be reduced.

b) current settings

```
    a) current sensor transimpedance: 40.00 mV/A
    b) maximum motor phase current: 49.9 A
    c) maximum battery current, motor use: 19.9 A
    d) maximum battery current, regen: 0.0 A

    e) autocomplete

    f) dr2, dr23: current to check: total current
    g) dr2, dr23: fixed part: 14.9 A
    h) dr2, dr23: proportional to throttle current, factor: 150 %
    i) dr3: maximum shutdown error current, fixed: 12.4 A
    j) dr3: maximum shutdown error current, proportional: 6.2 A
    k) dr3: shutdown based on : I_error
    l) applied braking current (phase) on direction change: 0.0 A
    m) offset filtering (phase) current limit: 0.0 A
    n) maximum field weakening current: 0.0 A

    z) return to main menu
```

Option n sets the maximum field weakening current. Field weakening is applied automatically upto the set current level (keep n at 0.0 A for no field weakening). Field weakening will increase dissipation, but gives a (motor and speed dependent) 'virtual' battery voltage increase of

4 * 3.14 * (e_rpm / 60) * L * I_fieldweak   Volt.

with L the inductance indicated by the FOC motor impedance menu. If you want to use fieldweakening a recommended value for k is 70% of the max phase current (option b)

```
c) throttle setup

a) calibrate throttle 1
b) calibrate throttle 2
c) polynomial coefficients throttle 1 (x, x^2, x^3): 1.0000, 0.0000, 0.0000
d) polynomial coefficients throttle 2 (x, x^2, x^3): -0.0002, -0.0002, -0.0002
e) use analog throttle 1: YES
f) use analog throttle 2: NO
   receive throttle over CAN: NO
g) TX throttle over CAN: NO
h) test throttle

z) return to main menu

------>
```

Upto 2 analog throttles can be connected to the controller IC. Options a and b are for calibration. The IC will measure the throttle voltage for throttle closed and throttle open. The throttle closed voltage must be lower than the throttle open voltage.

Either one of the throttle channels can be used for variable strength regen. This can be achieved by using negative polynomal coefficients (see the next slide)

Toggle options e and f indicate to the IC which analog throttle to use. When both are 'NO' the IC will automatically set the 'receive throttle over CAN' option. When the analog throttles are used there is an option to transmit the throttle information over CAN bus to other motor controller IC's.

When analog throttles are used also the 'reverse' switch can be used to select reverse. The state of this switch will also be transmitted over CAN. An IC receiving throttle information over CAN will also receive the 'reverse' information.

Based on the calibration information the throttle voltage will be transformed into variable x (x_1 for throttle 1, x_2 for throttle 2) in the range of 0 to 1. Out of range voltages are rounded to 0 or 1. Variables x_1, x_2 and the state of the reverse switch are transmitted or received over CAN bus.

Variables $x_{1,2}$ are transformed into variables $y_{1,2}$ by means of a polynomal function (the purpose of which is to be able to make different types of throttle response curves):

$$y_1 = a_1 x_1 + b_1 x_1^2 + c_1 x_1^3$$
$$y_2 = a_2 x_2 + b_2 x_2^2 + c_2 x_2^3$$

Options c and d are used to input the coefficients $a_{1,2}$, $b_{1,2}$ and $c_{1,2}$. Valid values are between -7.999 and +7.999.

Based on the throttle information the motor's phase current is given by:

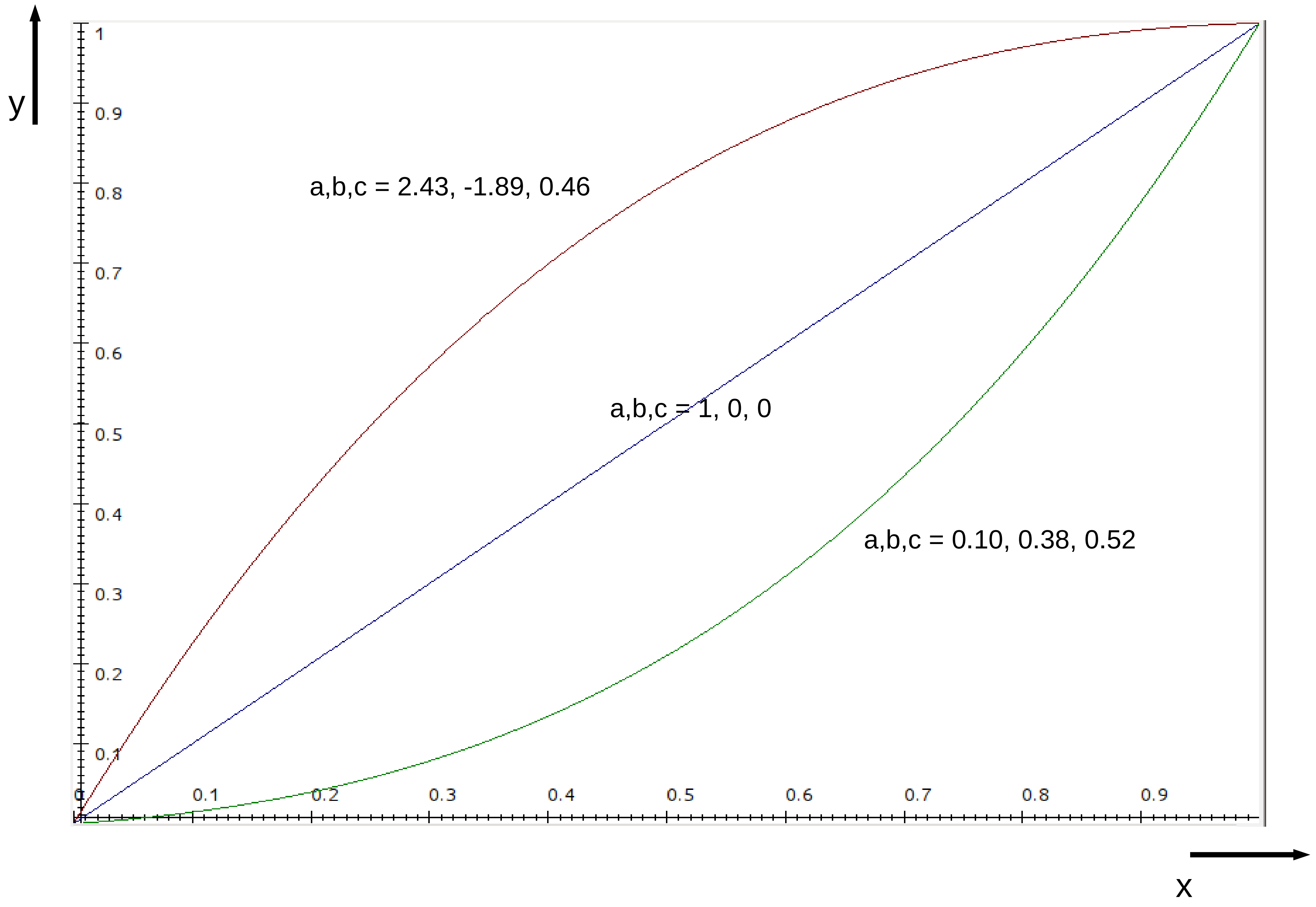
phase current = maximum phase current * ($y_1 + y_2$)

The maximum phase current is entered under menu d, option c.

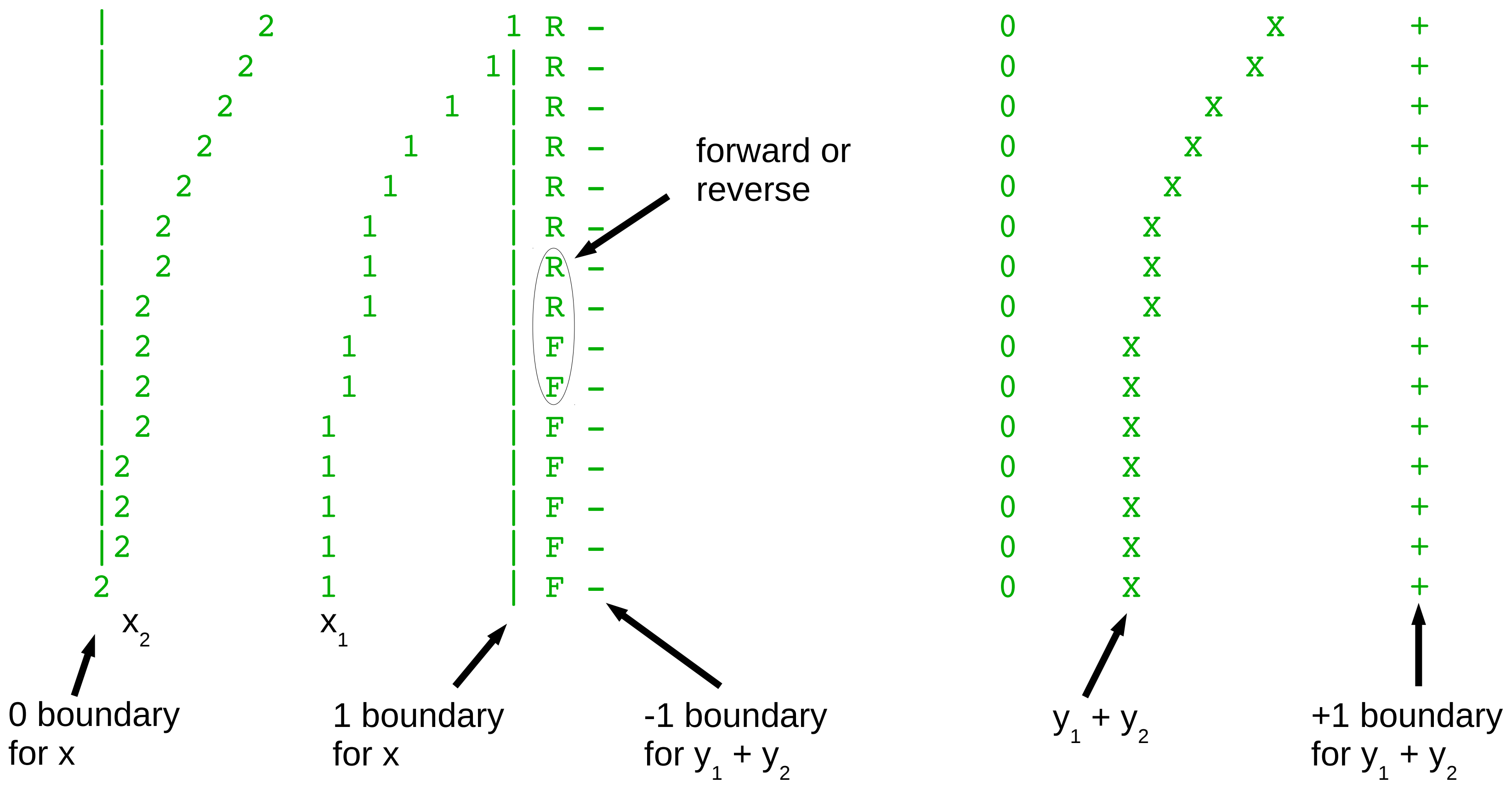
Variables x_1, x_2 are shared over CAN bus. Every motor controller IC however has its own set of a, b and c coefficients and its own maximum phase current setting. This allows the combining of motors with different ratings to operate of a shared throttle.

A negative phase current means current will flow to the battery, this is how variable strength regen can be obtained. When the conditions are such that the throttle requested phase current means that the maximum battery current or maximum battery regen current will be violated the phase current is automatically reduced.

some example throttle curves :



Because of the complexity of the throttle setup and it's importance for safety a test function has been implemented (option h).



```
|       2           1 R -                   0            X      +
|      2           1| R -                   0           X       +
|     2          1  | R -                   0         X         +
|    2        1    | R -                   0        X          +
|   2        1     | R -                   0       X           +
|  2        1      | R -                   0      X            +
|  2        1      | R -                   0      X            +
| 2          1      | R -                   0      X            +
| 2         1       | F -                   0     X             +
| 2         1       | F -                   0     X             +
| 2        1        | F -                   0     X             +
|2         1        | F -                   0     X             +
|2         1        | F -                   0     X             +
|2         1        | F -                   0     X             +
2          1        | F -                   0     X             +
```

The throttle information as shown above is updated around 30 times a second. The receive / transmit over CAN functionality is active so it's possible to test a multi-controller setup.

```
                                                       d) erpm limits

     a) erpm limiter (forward) rampdown start, end: 49.98, 52.99 k-erpm
     b) erpm limiter (reverse) rampdown start, end: 49.98, 52.99 k-erpm
     c) regen rampup start, end: 490, 784 erpm
     d) accept direction change below: 98 erpm
     e) erpm dr3 back to dr2: 980 erpm
     f) erpm dr2 jump to dr3: 1494 erpm

     g) use acceleration limiter: yes
     h) acceleration limiter start, end: 10.10, 12.12 kerpm/sec

     z) return to main menu
```

The chip has 4 different running mode, called drive_0 to drive_3. When not using recovery, drive_0 is the startup mode where the chip ends up at reset or after it failed to synchronise with a running motor in drive_1. It transitions to drive_1 when the throttle is closed and the motor is at standstill (if selected later in the recovery menu). Drive_1 is the recovery mode. The controller goes through drive_1 at startup or when an error has been detected (by means of the error current settings). In drive_1 the controller will try to synchronise with the (running) motor. Finally drive_2 is the mode in which the motor is started and drive_3 is the main FOC running mode in which 99% of the driving around is done.

In the erpms menu, options a and b set the maximum motor speed (electrical rpm !) for forward and and reverse direction. Erpm limiting is implemented by automatically ramping down the throttle to 0, the ramping down starts at the first value (49.98) and ends (throttle at 0) at the second value (52.99)

Using option c the parameters for regen ramping are set. Below the first value (490) there is no regen, above the second value (784) full regen is available, inbetween the two values ramping is implemented.

The reverse pin is treated as a request for reverse. When the request comes the motor is spooled down with the braking current as set in the currents menu, and the request is granted once the motor speed is below the speed set in option d.

The transition erpms between starting the motor and the sensorless FOC running mode are set with options e and f. Note that some hysteresis should be build in by specifying option f larger than option e.

```
                                                                    d) erpm limits

    a) erpm limiter (forward) rampdown start, end: 49.98, 52.99 k-erpm
    b) erpm limiter (reverse) rampdown start, end: 49.98, 52.99 k-erpm
    c) regen rampup start, end: 490, 784 erpm
    d) accept direction change below: 98 erpm
    e) erpm dr3 back to dr2: 980 erpm
    f) erpm dr2 jump to dr3: 1494 erpm

    g) use acceleration limiter: yes
    h) acceleration limiter start, end: 10.10, 12.12 kerpm/sec

    z) return to main menu
```

Option g can be used to activate the accelleration limiter. This can be useful to limit mechanical shock when a mechanical 'clutch' is used. In a similar fashion as the erpm limiter, ramping down of the throttle is used for accelleration limiting. The two values entered under h represent the start and end of the ramp.

```
                                                                e) battery

     a) use HVC, LVC battery current limiting: yes
     b) battery voltage: 60.1 V
     c) HVC cutoff: 65.9 V
     d) HVC start : 63.9 V
     e) LVC start : 57.9 V
     f) LVC cutoff: 55.9 V

     z) return to main menu
```

With option a battery current limiting can be turned on or off. When off it is still necessary to enter a voltage under b, this value is used to calculate the correct motor inductance value in a later menu.

The High Voltage Control and Low Voltage Control act on the allowed battery current. When powering the motor, the controller will start reducing the allowed battery current when the battery voltage drops below the value at 'LVC start' and ramp down to 0 by the time battery voltage drops as low as 'LVC cutoff' . Same for regen and HVC. The battery current is affected, meaning the motor will still do lots of torque (high phase curent) at low erpm, but will quickly run out of steam as erpms increase.

The value under b) must be set to the current battery voltage. The way it works is that every time this menu is refreshed the controller IC measures the battery voltage (takes 0.5 sec so not really noticable) and stores the 16 bit digital code from the ADC. Then when you select b) and enter a value it calculates the ratio with regards to the 16 bit from the ADC and stores this.

The interesting consequence of this is that when you change the battery voltage and refresh the menu (press enter without selecting anything) the value under b) will be updated with to reflect the voltage change.

f) current sensor calibration

```
a) restore calibration, autocomplete
b) perform offset measurement
   sensor a: 0.0 mV
   sensor b: 0.0 mV
   sensor c: 0.0 mV

z) return to main menu

------>
```

The current sensors can be calibrated for offset. Option a resets all calibration values to default.
In practise I found offset calibration is not necessary and all values can stay at default.

Option b performs the offset measurement. Good offset calibration is necessary for smooth (very very) low erpm sensorless operation. When you select option b the motor will make a noise as the controller performs the measurement.

g) control loop coefficients

```
a) autocomplete

  phase control loop, drive 3
b) 1st order: 480
c) 2nd order: 24.0000
d) 3rd order: 0.3000
  phase control loop, drive 2
e) 1st order: 480
f) 2nd order: 24.0000
g) 3rd order: 0.0299
  amplitude control loop
h) 1st order: 100
i) 2nd order: 5.0000
j) invoke fieldweakening at amplitude: 95 %
k) reduce throttle from amplitude: 98 %
l) to closed throttle at amplitude: 100 %
  field weakening control loop
m) 2nd order: 5
  miscellaneous
n) immediate motor phase step: 48

z) return to main menu
```

The control loop coefficients dermine the behavior of the control loops. Normally the autocompleted values are correct and no changes are necessary.

```
                                                                                    h) filter bandwidths

      a) autocomplete

      b) dr2: speed filter 50% step response time: 50.5 msec
      c) dr2, dr23: error current 50% step response time: 20.0 msec
      d) dr3: error current 50% step response time: 5.00 msec
      e) battery voltage filtering time constant: 0.10 sec
      f) acceleration filter time constant: 200.85 msec

      z) return to main menu
```

This menu sets the filter bandwidths for various filter in the controller IC.

The internal motor speed variable is too noisy to reliably make the changes between drive modes 2 and 3. Therefore a very slow filter is added (option b) and the drive mode transitions are based on the output of this filter. This prevents incorrect transitions coming from spikes on the internal speed variable.

The error currents are filtered so that the controller doesn't conk out to drive_1 for spikes which are higher than the error current settings. The response time for these filter can be set using options c and d.

Option e determines the filtering of the noisy battery voltage, before using it for LVC etc.

Similary, option f sets the filtering for the acceleration filter, before using it for the accelertion limiter.

i) FOC motor impedance

```
    a) autocomplete

    b) FOC measurement current: 59.9 A
    c) FOC measurement erpm: 11.98 k-erpm
    d) perform impedance measurement

    L) inductance: 63.71 uH
    R) resistance: 25.66 mOhm

    z) return to main menu
```

For the Field Oriented Control the motor impedance must be measured.

Option a autocompletes the measurement currents and erpms. Option d must be manually selected and will perform the actual measurement (the motor will make a noise). In order to display the correct impedance value, the battery voltage must be set correctly (battery voltage menu).

If an 'out of range' error message is produced the measurement current (b) should be reduced.

j) CAN setup

```
a) CAN 'address': 16383
b) CAN CFG1 as per Microchip 30F manual: 65535
c) CAN CFG2 as per Microchip 30F manual: 65535
   RS232 output rate: 3636 Hz
z) return to main menu

------>
```

This menus sets the properties of the CAN bus. Option a sets the 'address' (acceptance filter in CAN speak), the addresses of the transmitting and receiving controller must match for communication to occur. Multiple master/slaves with different addresses can use a single CAN bus. Valid values are in the 0 to 2046 range.

Options b and c configure the CAN bus data rate, see the 30F manual from Microchip. For a typical robust 100 kHz bitrate setup, use '14' for option b and '664' for option c. Throttle and reverse information (60 to 70 bits) is sent at a rate of 100 Hz.

During motor use the RS232 is dormant. Normally no information is transmitted but when the controller receives a single lower-case letter according to the table it will start transmitting 16 bit data at a rate indicated here under 'RS232 output rate'. Data is outputted as 2's complement with the high byte first. Transmission will stop once a character not in the table is received.

a	phi (slide 16)
b	phi_int (slide 16)
c	phase current, filtered
d	phase current, requested from throttle
e	amplitude (slide 18)
f	throttle 1 (x1)
g	throttle 2 (x2)
h	combined throttle after polynomals (y1+y2)

i	calibration value current sensor A
j	calibration value current sensor B
k	calibration value current sensor C

```
                                                                   j) CAN setup
a) CAN 'address': 16383
b) CAN CFG1 as per Microchip 30F manual: 65535
c) CAN CFG2 as per Microchip 30F manual: 65535
   RS232 output rate: 3636 Hz
z) return to main menu

------>
```

l	temperature (*2) reading from sensor 0
m	sensor 1
n	sensor 2
o	sensor 3
p	sensor 4
q	sensor 5
r	sensor 6
s	sensor 7
t	Max phase current based on temperature sensors
u	Field weakening current

```
                                                                      k) recovery
  a) autocomplete

    phase control loop, recovery
  b) 1st order: 0
  c) 2nd order: 120.0000
  d) 3rd order: 3.0000
    amplitude control loop, recovery
  e) 1st order: 240
  f) 2nd order: 12.0000
  g) pulse when current drops below: 6.4 A
  h) pulse width: 32 usec
  i) pulse % for exit: 95
  j) pulse % filter 50% step response time: 30.2 msec
  k) speed filter 50% step response time: 7.0 msec
  l) try restart for: 499 msec
  m) check for spinning motor, drive_0: enabled
  n) check for throttle closed, drive_0: enabled
  o) post recovery throttle ramp : 0.100 sec

  z) return to main menu
```

Recovery is done by pulsing the motor (shorting it out) and then observing the currents and trying to get phase and amplitude information from this. As phase and amplitude information come closer and closer to the correct value, a pulse generates less and less current. To maintain the current level the pulse rate will increase. When the pulse rate has reached a certain level the motor is 'recovered' and normal running will commence.

When recovery is used it is indicated by the drive_1 LED .

A pulse has the length of time as given by option h. During a pulse the motors backemf voltage will induce a current in the windings. Because of the inductance this current will keep flowing after the pulse has ended. When the current drops below the level of option g a new pulse is given. As the controller gets closer and closer to being in sync, the current induced from every pulse will be less and less. So after every pulse the current level of option g is reached sooner, to maintain this current level the pulse rate will go up as controller gets more and more in sync. The pulse rate is passed through a filter (option j) and when it reaches the level of option i the motor is declared 'recovered' and normal running will commence.

Options b to f are the phase and amplitude control loop coefficients used during recovery. They are set rather high to increase speed of recovery. Too high however and the phase/amplitude information will jump around a lot, making it difficult to accurately capture the motor. In practise this means the controller will not reach a high enough pulse rate to exit drive_1.

To start normal running the controller needs to obtain speed information during recovery, this information is filtered using option k. The number here must be less than the pulse filter option j, as the speed filter must be settled before pulse filter settles.

Whenever an error or glitch occurs the controller will exit to drive_1 and try to recover the motor. In case of a serious problem like a blown FET the controller will not be able to recover. Recovery is tried for the amount of time specified under option l, if unsuccesfull the controller will go to drive_0. Here the controller will yes/no wait for motor stop (option m) and/or throttle closed (option n). Having option m set to enabled will also detect a shorted out FET. Option n is there to prevent the motor from starting with high torque when the controller is powered on for the first time. Both options can be disabled (this enables for on the fly controller reset or turn-on).

To prevent mechanical shocks in the drivetrain, after recovery the throttle is slowly ramped up to the full user set (by means of throttle) value over the time specified under o.

```
      code: 0, angle: 358 deg, confidence: 0, used: no                          l) hall sensored only
      code: 1, angle: 241 deg, confidence: 6, used: yes
      code: 2, angle: 118 deg, confidence: 6, used: yes
      code: 3, angle: 177 deg, confidence: 6, used: yes
      code: 4, angle: 7 deg, confidence: 6, used: yes
      code: 5, angle: 305 deg, confidence: 6, used: yes
      code: 6, angle: 66 deg, confidence: 7, used: yes
      code: 7, angle: 358 deg, confidence: 0, used: no

     a) autocomplete

     b) toggle hall usage
     c) calibrate hall mode: no
     d) erpm for hall calibration: 4.99 k-erpm
     e) PLL bandwidth: 100 Hz
     f) PLL damping factor: 0.71
     g) hall offset, forward: 0.00 deg
     h) hall offset, reverse: 0.00 deg
     i) hall assisted sensorless: yes

     0) reset hall statistics
     #) display hall statistics
     z) return to main menu
```

The hall sensor menu gives information about the current hall calibration, allows you to use or disregard different 'codes' and to enable the automatic hall calibration.

The hall calibration works as follows. The chip must be set in hall mode (option 0 of the main menu), the calibrate hall positions in the above menu must be turned on (c) and the 'online parameter save' in the store in ROM menu must be enabled. The chip must be setup completely and the parameters must be saved.

Turn off the controller, remove the setup signal and power on the controller. The controller will enter motor mode, and because the calibrate hall is selected it will not go into hall mode but into sensorless mode. Make sure the motor is unloaded and give a bit of throttle. The motor will start spinning as it would in sensorless mode. Once spinning you can give full throttle, the motor erpm is automatically controlled to the setting of option e. To indicate the special measurement mode both drive_3 and drive_0 LEDs will light. After about 4 seconds all LEDs will light up indicating the measurement is finished and the data is saved. The controller will reset and is ready for normal use.

```
 code: 0, angle: 358 deg, confidence: 0, used: no                    l) hall sensored only
 code: 1, angle: 241 deg, confidence: 6, used: yes
 code: 2, angle: 118 deg, confidence: 6, used: yes
 code: 3, angle: 177 deg, confidence: 6, used: yes
 code: 4, angle: 7 deg, confidence: 6, used: yes
 code: 5, angle: 305 deg, confidence: 6, used: yes
 code: 6, angle: 66 deg, confidence: 7, used: yes
 code: 7, angle: 358 deg, confidence: 0, used: no

a) autocomplete

b) toggle hall usage
c) calibrate hall mode: no
d) erpm for hall calibration: 4.99 k-erpm
e) PLL bandwidth: 100 Hz
f) PLL damping factor: 0.71
g) hall offset, forward: 0.00 deg
h) hall offset, reverse: 0.00 deg
i) hall assisted sensorless: yes

0) reset hall statistics
#) display hall statistics
z) return to main menu
```

Even through it is not necessary, after calibration you can go into the hall menu and have a look at the calibration results, and turn on or off certain 'codes' using option b (see also next few pages).

The raw hall sensor signals are filtered using a 'PLL' with bandwidth and damping as under e and f. Dependent on direction a correction (g and h) can be added.

To find the correction hall statistics can be run. Hall statistics will measure the real-life phase difference between sensored and sensorless operation as you are running the controller in motor mode. Before using statistics it must be reset with option 0. When running in motor mode, at the moment of transition between sensored and sensorless the phase difference is stored. To store this data, before turning off the controller the data must be stored by pressing the setup button (online save must be turned on, see the 'store in ROM' menu).

```
l) hall sensored only
```

The result of the statistics can be viewed with '#'

```
 forward
count: 76
average: -3.33 deg
std-dev: 17.81 deg

 reverse
count: 0
```

In this case 76 sensored-to-sensorless transitions occurred. The average phase error was -3.33 degrees, with a standard deviation of 17.81 degrees. No transitions were seen for reverse.

To correct the phase used during sensored running the reported average must be ADDED to the corresponding hall offset value:

```
g) hall offset, forward: -3.32 deg
h) hall offset, reverse: 0.00 deg
```

When changing the hall offset values the statistics must be reset for meaningful results. If no changes were made data collection can continue without resetting. Note: after running in motor mode setup must be pressed else transition data is lost when the chip is powered down.

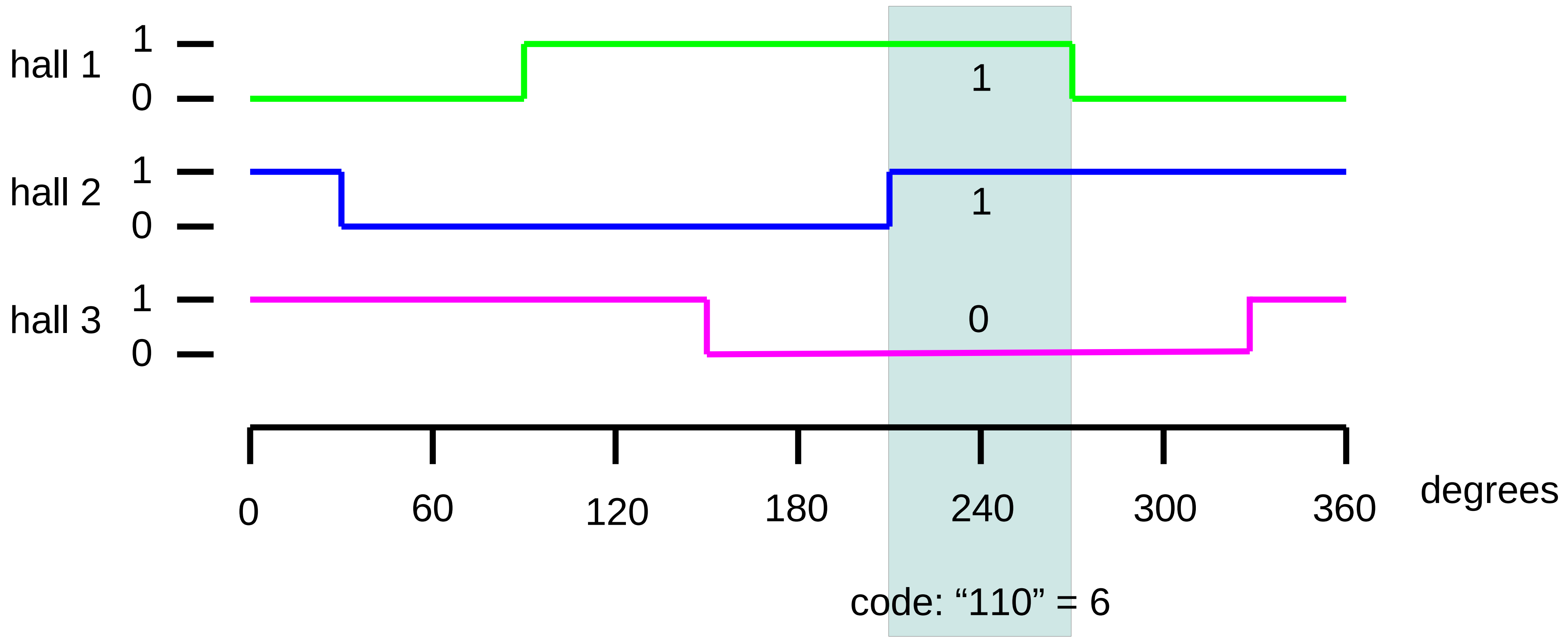
```
     code: 0, angle: 358 deg, confidence: 0, used: no                      l) hall sensored only
     code: 1, angle: 241 deg, confidence: 6, used: yes
     code: 2, angle: 118 deg, confidence: 6, used: yes
     code: 3, angle: 177 deg, confidence: 6, used: yes
     code: 4, angle: 7 deg, confidence: 6, used: yes
     code: 5, angle: 305 deg, confidence: 6, used: yes
     code: 6, angle: 66 deg, confidence: 7, used: yes
     code: 7, angle: 358 deg, confidence: 0, used: no

    a) autocomplete

    b) toggle hall usage
    c) calibrate hall mode: no
    d) erpm for hall calibration: 4.99 k-erpm
    e) PLL bandwidth: 100 Hz
    f) PLL damping factor: 0.71
    g) hall offset, forward: 0.00 deg
    h) hall offset, reverse: 0.00 deg
    i) hall assisted sensorless: yes

    0) reset hall statistics
    #) display hall statistics
    z) return to main menu
```

In this version of the controller IC a new running mode is introduced, 'hall assisted sensorless' (option I).

In this mode, in drive_3 (the sensorless mode) the motor phase is calculated using sensorless back-emf information while motor speed is taken from the hall sensors. It makes running the motor more robust for when large sudden speed changes occur, like in a sporty mid-drive bike used off-road (jumping roots) in the forest.

Best results are obtained in hall assisted sensorless when the control loop coefficient in menu g, option c is increased from 24 to 96.

1) hall sensored only



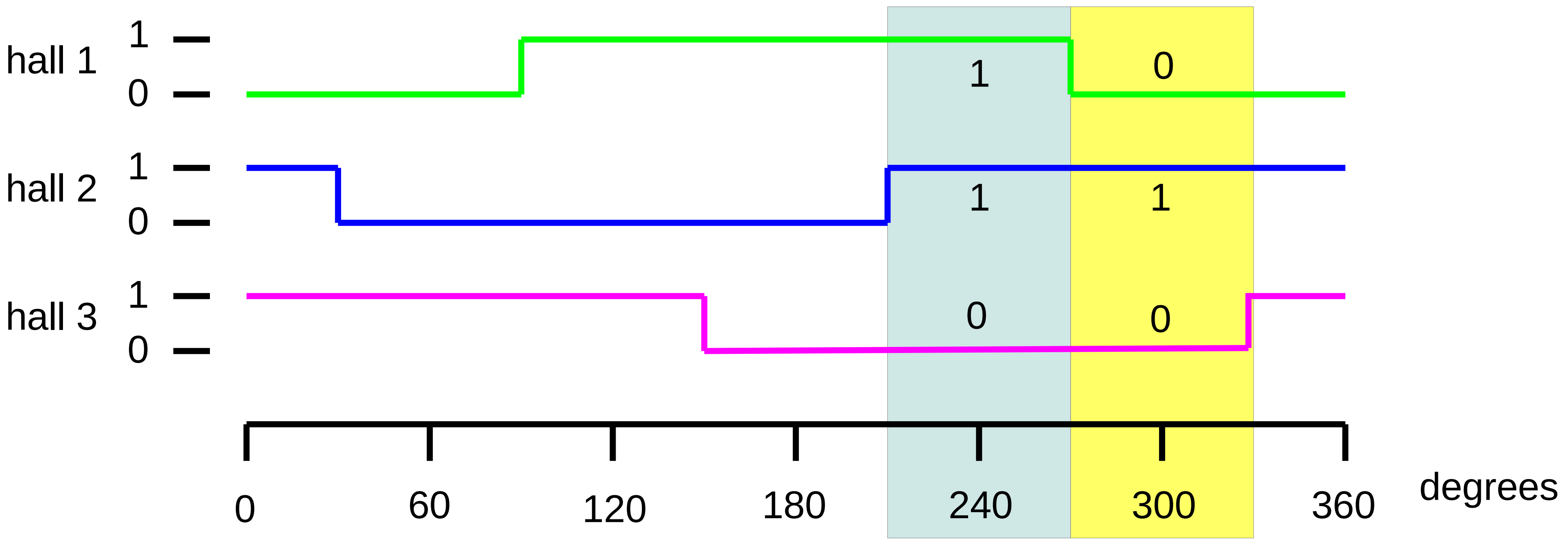
The figure above shows how the three hall signals toggle as the motor moves from 0 to 360 e-degrees.

The controller looks at the three hall signals as if they are part of a binary number. In the blue box for instance, hall 1 is digital '1', hall 2 is digital '1' and hall 3 is '0'. Together this makes the binary number “110”, which translates to the decimal number 6. This is called the 'code'.

When running the motor, the controller looks at the hall signals, transforms this into the code and then looks in an array to find the correct e-degrees at which to apply the signals to the motor.

```
1) hall sensored only
```

```
code: 0, angle: 111 deg, confidence: 0, used: no
code: 1, angle: 59 deg, confidence: 7, used: yes
code: 2, angle: 298 deg, confidence: 7, used: yes
code: 3, angle: 357 deg, confidence: 7, used: yes
code: 4, angle: 177 deg, confidence: 7, used: yes
code: 5, angle: 118 deg, confidence: 7, used: yes
code: 6, angle: 239 deg, confidence: 7, used: yes
code: 7, angle: 358 deg, confidence: 0, used: no
```

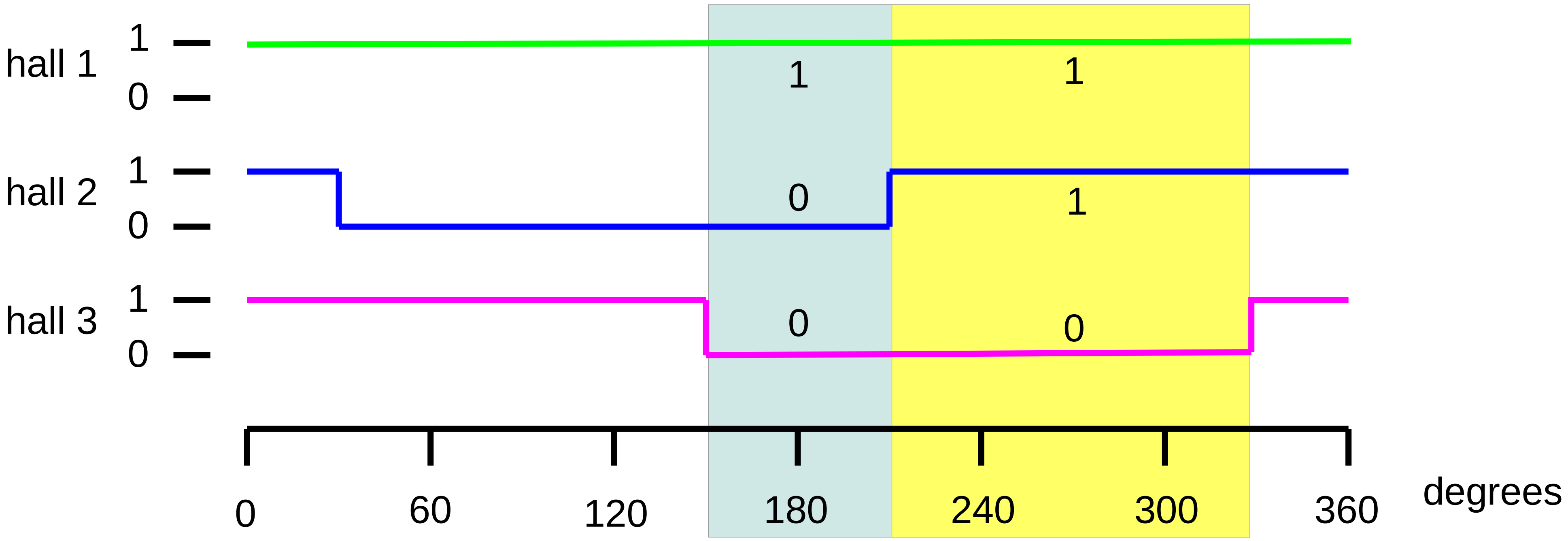


The table in the hall menu shows the calibration result. If you look at code 6, you can see the chip measured an average of 239 degrees, as indicated by the blue box in the graph. In a similar way, code 2 is indicated by the yellow box.

The confidence in the measurement is indicated by a number between 0 and 7. The confidence is computed based on how often a code occurred during the calibration run and how wide the range of degrees is for the code. Above for instance the confidence in codes 0 and 7 is 0, these codes were not seen by the chip during the calibration run. A low confidence (below 2.3) means the chip will not use the code when running the motor. You can override whether a code will be used of not with option a. If the chip encounter an unused code when running the motor, it will use the last valid code it saw.

1) hall sensored only

```
code: 0, angle: 358 deg, confidence: 0, used: no
code: 1, angle: 358 deg, confidence: 0, used: no
code: 2, angle: 358 deg, confidence: 0, used: no
code: 3, angle: 358 deg, confidence: 0, used: no
code: 4, angle: 177 deg, confidence: 6, used: yes
code: 5, angle: 85 deg, confidence: 6, used: yes
code: 6, angle: 267 deg, confidence: 6, used: yes
code: 7, angle: 357 deg, confidence: 6, used: yes
```

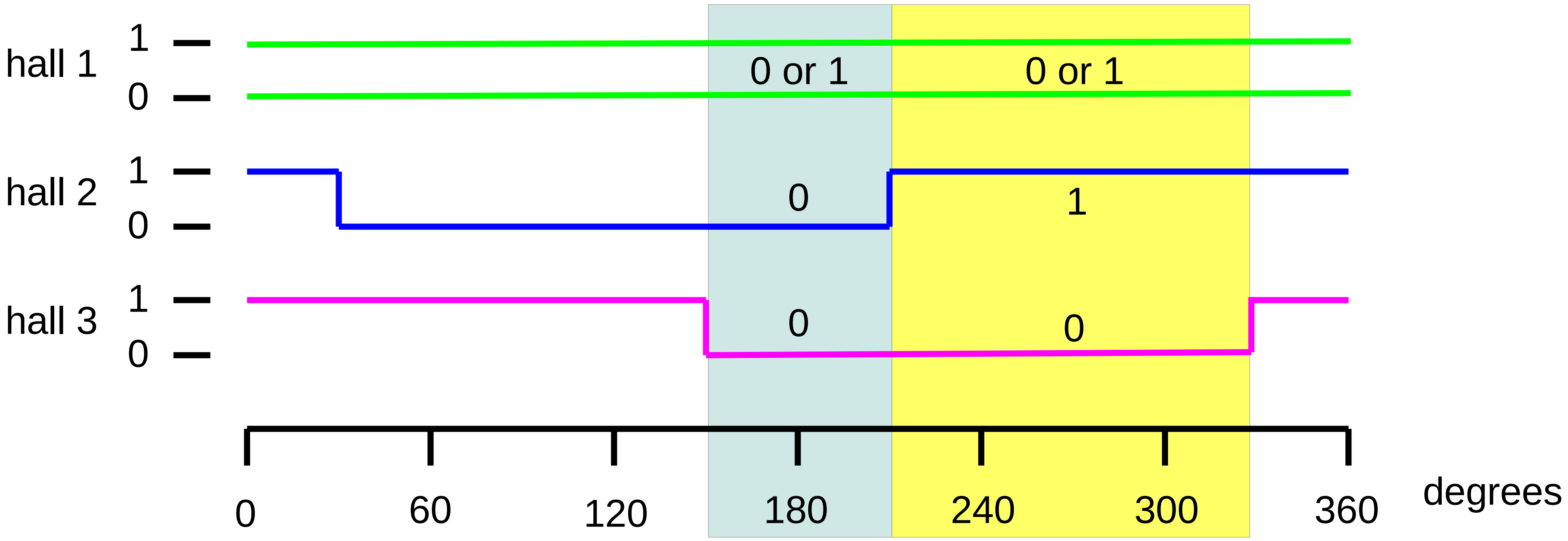


The system also works when a hall sensor is broken, above for instance hall 1 is for some reason always showing a high signal. The blue square shows code 4 which is indeed centered around 177 degrees. The yellow box shows code 6, which now is centered around 267 degrees. Confidence has dropped from 7 to 6, either because the codes don't occur so often (the narrow blue box) or because the degree range is wide (wide yellow box). Codes 0 to 3 don't occur, confidence is 0 and they are not used.

The controller will still be able to run the motor. Especially in the wide (yellow) ranges the motor control is not so accurate as the specific code covers a range of 120 e-degrees. But since most of the time the controller runs completely sensorless and the halls are only used for motor start, performance is not really affected.

```
l) hall sensored only
```

```
code: 0, angle: 177 deg, confidence: 6, used: yes
code: 1, angle: 85 deg, confidence: 6, used: yes
code: 2, angle: 265 deg, confidence: 6, used: yes
code: 3, angle: 355 deg, confidence: 6, used: yes
code: 4, angle: 175 deg, confidence: 6, used: yes
code: 5, angle: 85 deg, confidence: 6, used: yes
code: 6, angle: 267 deg, confidence: 6, used: yes
code: 7, angle: 357 deg, confidence: 6, used: yes
```

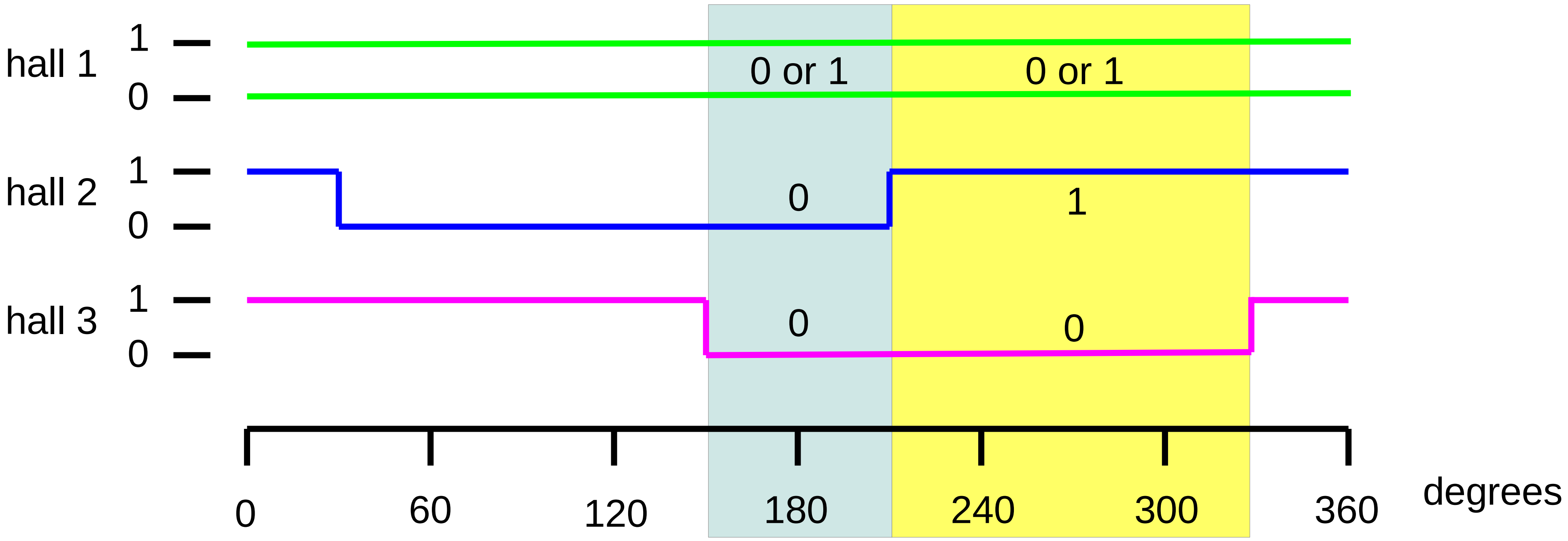


Above shows the case when hall 1 is broken and outputs random 1's and 0's. Now both codes 0 and 4 correspond to the same blue box, codes 2 and 6 map to the yellow box.

Since all codes occur at random and therefore both quite often, confidence is high at 6. All codes are used.

```
code: 0, angle: 177 deg, confidence: 6, used: yes
code: 1, angle: 85 deg, confidence: 6, used: yes
code: 2, angle: 265 deg, confidence: 6, used: yes
code: 3, angle: 355 deg, confidence: 6, used: yes
code: 4, angle: 175 deg, confidence: 6, used: yes
code: 5, angle: 85 deg, confidence: 6, used: yes
code: 6, angle: 267 deg, confidence: 6, used: yes
code: 7, angle: 357 deg, confidence: 6, used: yes
```

1) hall sensored only



Above shows the case when hall 1 is broken and outputs random 1's and 0's. Now both codes 0 and 4 correspond to the same blue box, codes 2 and 6 map to the yellow box.

Since all codes occur at random and therefore both quite often, confidence is high at 6. All codes are used.

```
a) use temp sensors: yes
b) identify temp sensors
c) temperature readings
z) return to main menu

 0: 5E00 0802 BBD6 1010
 1: 8F00 0802 BBF8 6E10
 2: 5400 0802 A8C4 E110

0) reduce max phase current above 50.0 degC by 0.9 A/degC
1) reduce max phase current above 60.0 degC by 1.9 A/degC
2) reduce max phase current above 74.0 degC by 3.4 A/degC

------>
```

Option a turns the use of temperature sensors on or off.

From the factory each produced temperature sensor gets its own unique 64 bit code. With this code the sensors can be addressed independently, even though they are all connected in parallel. Option b searches for all connected temperature sensors and retreives their 64 bit codes (upto 8 sensors max). These are then displayed after option z.

For each temperature sensor you can set above which temperature the motor phase current must be reduced and by how many Amperes per degree Celsius.

Option c will show multiple columns, one for each sensor diplaying the temperature, with the final column showing the temperature allowed max phase current (based on the user entered data under 0), 1) etc)

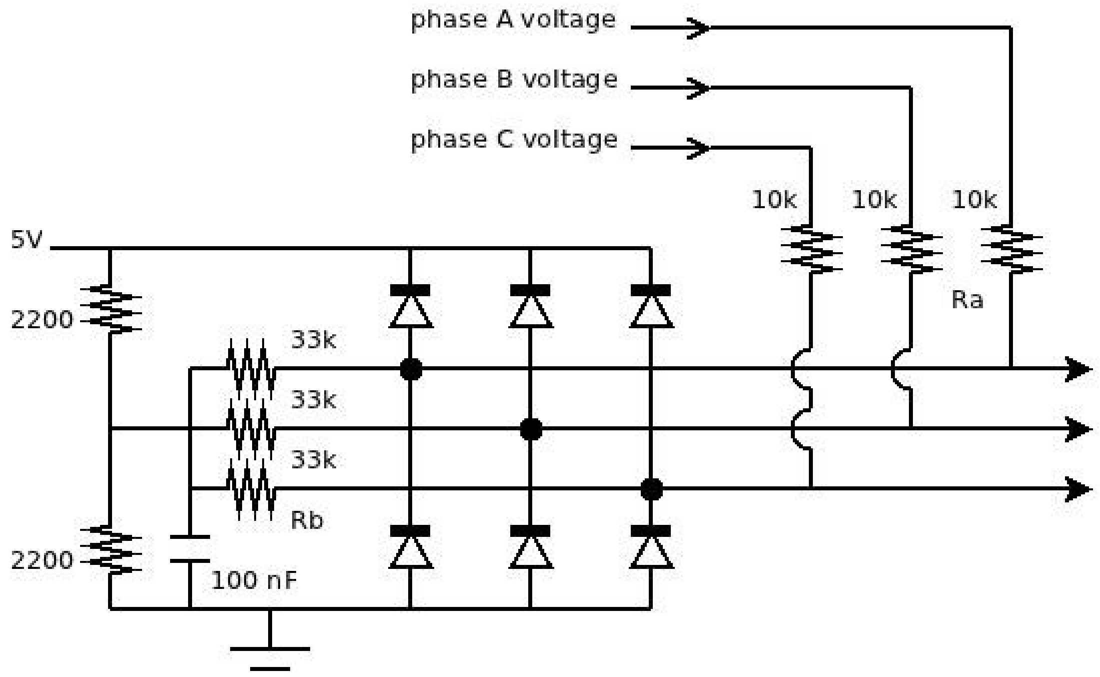
```
                                                                   n) miscellaneous
    a) autocomplete

    b) motor standstill voltage threshold: 0.48 V
    c) low side pulsing in drive 0: enabled
    d) low side pulsing rate: 20 Hz
    e) low side pulsing width: 20 usec
    f) wiggle range: 19 deg
    g) wiggle rate: 6 Hz
    h) minimum # of cycles going from drive 2 to 3: 1000

    z) return to main menu
```

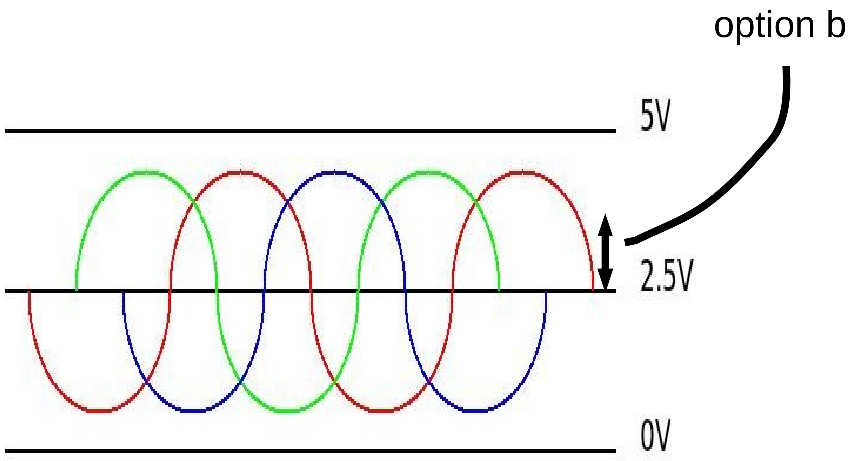
Options b-e are discussed on the following pages.

When sensorless start is selected, a wiggle on the phase (drive_2 only) is used to intise a response from the motor that the sensorless algorithm can work with. Option f sets the amount of e-phase the the motor should be wiggled over, option g sets the rate. The wiggle is especially usefull for losening up RC motors which have a relatively large amount of cogging. The wiggle can be disabled by making option f 0.

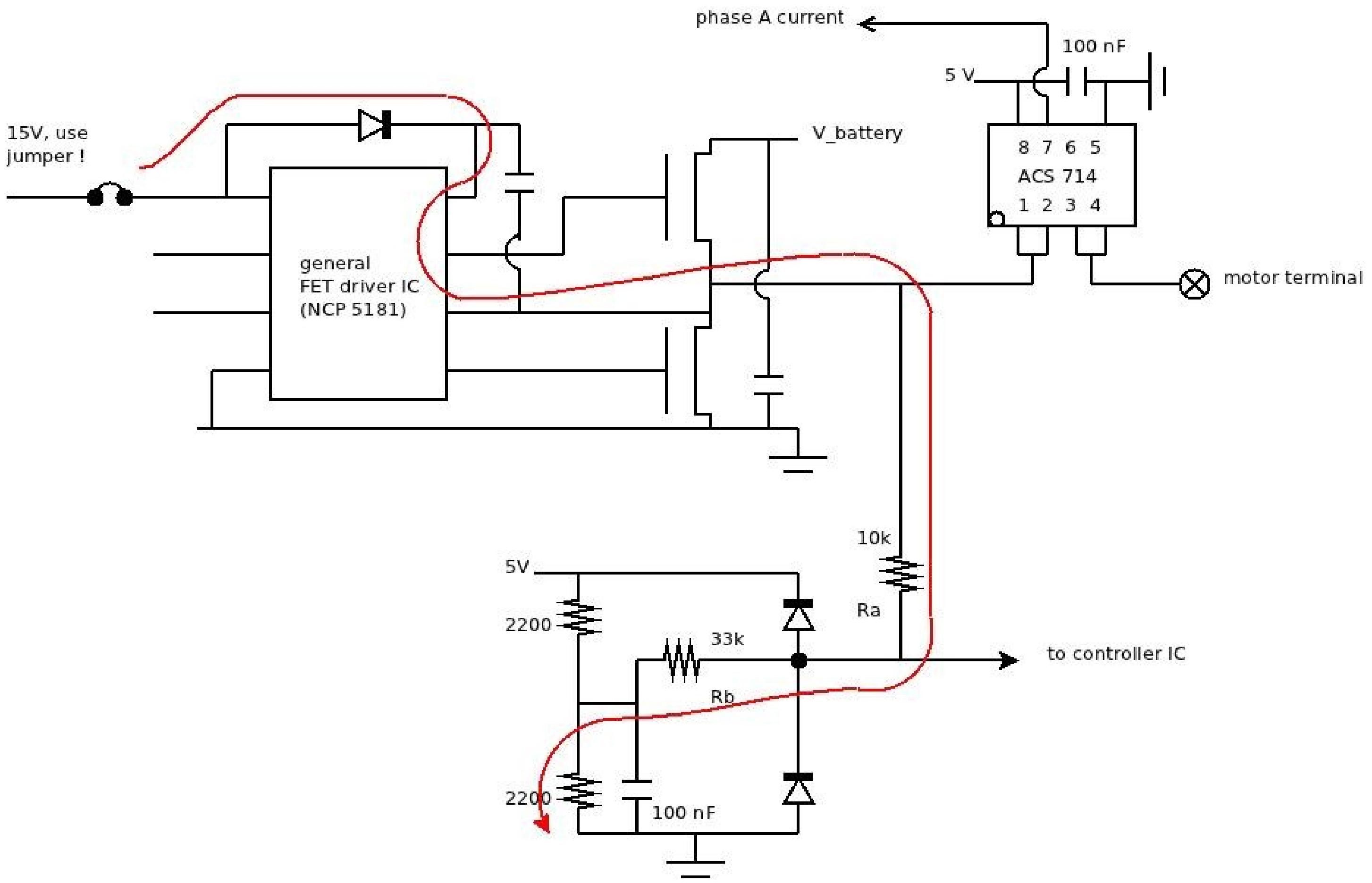
The actual backemf based sensorless will see the wiggle and try to follow it. When the chip transides to drive_3, the wiggle can be disruptive and cause an error current event. Therefore there is an inbetween mode (drive_2to3) during which error current detection is disabled. Option h sets the minimum amount of cycles to be spend in this transitional mode. After this amount of cycles, the chip checks for low error current before transiding to drive_3.

Since the controller powers all motor terminals all of the time a mechanism has to be in place to tell the controller when it's safe to take control of the motor. This is determined in drive_0, the only mode in which the motor terminals are 'released'. To transide out of this mode the throttle has to be closed (or on regen) and the motor speed has to be low. The motor voltage amplitudes as seen by the controller have to be below the voltage of option b.

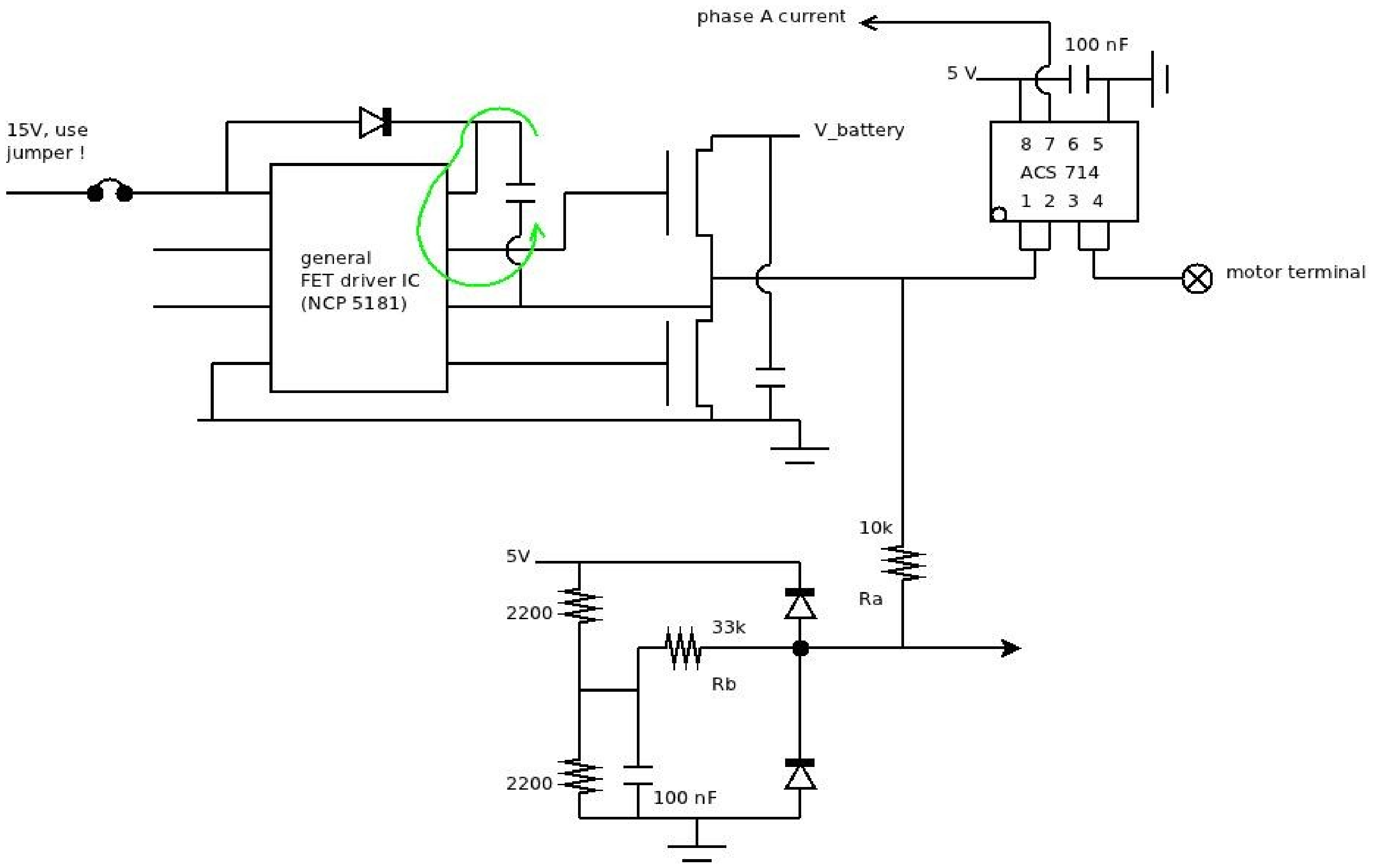




Dependent on the type of output driver a bias current can pass through resistors Rb in which case the detected voltages will never fall below 2.5V+option b.



The figure shows a typical output stage driver built using a NCP 5181. The high side driver is supplied from a capacitor which is charged using a diode. When the capacitor is not fully charged the diode will conduct, causing a current to flow according to the red path. The current passes through Rb, raising the voltage to the controller IC. When the voltage stays above 2.5V+option b the IC will stay in drive_0.
To prevent this from happening the high side driver's supply capacitor must be fully charged. The diode will then no longer conduct and no parasitic current will flow though Rb.



The figure shows the high side driver's supply current path when its supply capactor is properly charged.

To charge the supply capacitor the controller offers the option to pulse-wise turn on the low side FET. When the low side FET is on the supply capacitor will be charged, enabling the correct detection of 2.5V+option b.

The pulsing of the low side FET during drive 0 is turned on or off with option c. Options d and e set the pulse frequency and duration. The options should be set low enough not to drastically brake the motor and high enough to keep the high side driver supply capacitor charged to a sufficient level.

```
                                                                    o) online Kv, L and R

    a) use online Kv, L and R measurement: yes

     make sure f_sample is below 25kHz !

    b) autocomplete
    c) #data points for online impedance measurement: 16384
    d) reset data collection
    e) current data collection, R: 9.9 A
    f) current data collection, L: 2.4 A
    g) speed data collection: 5.97 k-erpm
    h) inject extra current : yes

    z) return to main menu
```

Online Kv, L and R is an experimental feature at the moment. Fine for playing around but not really suitable yet for driving around. Best is to use option a to turn it off.

It works based on running statistics and making a least square fit. Option c sets the amount of data points it uses for the statistics. More means it will be slower to update. D resets the data points.

E, f and g are the limits below which no data points are collected. Finally h injects an extra current that aids in the L measurement.

At the start after a data collection reset L and R are not updated immediately, it will need some time to collect an initial data set. Until then it uses the L and R from the FOC measurement menu.

```
                                                                                        y) chip status at last drive_1

        Chip status at button press:

       status bits:
       drive LEDS:                0.2.
       total on time:             408.836 sec
       time in drive mode:        11.864 sec
       throttle:                  0 %
       wanted_i_torque:           0.0 A
       wanted_i_fieldweak:        0.0 A
       measured_i_torque:         0.0 A
       measured_i_fieldweak:      0.0 A
       measured_i_error:          0.0 A
       Vout_real:                 0 %
       Vout_imag:                 0 %
       speed:                     0.00 k-erpm
       L:                         63.71 uH
       R:                         25.66 mOhm


       a) clear all

       z) return to main menu
```

Every time the chip enters drive_1 the status is written to RAM. When 'online parameter save' is enabled in the ROM menu, pressing setup will save the data to ROM so it can be upserved from the setup menus. The last three entries into drive_1 plus the status at button press are saved.

```
     Last entry into drive_1:

status bits:                  over_i_error
drive LEDS:                   ...3
total on time:                344.057 sec
time in drive mode:           0.758 sec
throttle:                     98 %
wanted_i_torque:              127.6 A
wanted_i_fieldweak:           6.8 A
measured_i_torque:            135.2 A
measured_i_fieldweak:         -22.6 A
measured_i_error:             -37.4 A
Vout_real:                    30 %
Vout_imag:                    37 %
speed:                        16.07 k-erpm
L:                            43.49 uH
R:                            28.36 mOhm
```

Above an example of a report for a conkout into drive_1 is shown. The error bits show the reason, an over error current was detected. While this happened the chip was in drive_3, where it spent only 0.758 seconds. The controller was on for 344 seconds

Throttle was at 98% of maximum (positive number for powering, negative for regen). This made the wanted torque current 127.6A (positive for powering, negative for regen). The wanted torque current is not always the throttle % times the max phase current, it can be lower to keep the battery current within the set limit, or 0 A when the motor speed is at or above the set maximum.

Wanted_i_fieldweak shows whether field weakening was used.

Measured_i_torque and measured_i_fieldweak are the torque and fieldweak currents at the moment of entering drive 1.

y) chip status at last drive_1

```
     Last entry into drive_1:

status bits:                over_i_error
drive LEDS:                 ...3
total on time:              344.057 sec
time in drive mode:         0.758 sec
throttle:                   98 %
wanted_i_torque:            127.6 A
wanted_i_fieldweak:         6.8 A
measured_i_torque:          135.2 A
measured_i_fieldweak:       -22.6 A
measured_i_error:           -37.4 A
Vout_real:                  30 %
Vout_imag:                  37 %
speed:                      16.07 k-erpm
L:                          43.49 uH
R:                          28.36 mOhm
```

Measured_i_error shows the error current (the large value is the cause of the conkout, as it shown by the over_i_error status bits.

Vout_real and imag show the amplitude of the signal to the motor.

Speed shows motor speed at moment of conkout. L and R show the momentary L and R. These should be the same as the FOC measured L and R, except for when online L and R measurement is turned on.

```
                                                            z) store parameters in ROM for motor use

      a) save data to ROM for motor use
      b) print data in HEX format
      c) enter data in HEX format
      d) online parameter save: disabled

      z) return to main menu

      ------>
```

This menu allows for the saving of data to the chips internal ROM memory, so that all the settings can be used for running the motor. For this option a must be used.

Option b print out all the variables in HEX format, which can then be saved in a text file on the computer.

```
------> b

 save the following HEX lines in a text file, including the '*' termination character

0x0085    0x0085    0x00D6    0x0015    0x0005    0x0002    0x7FBC    0x0623
0x0000    0x0F5B    0x0000    0x07AD    0x03D6    0x0E92    0x0400    0x0400
0x0400    0xAAAA    0xAAAA    0xAAAA    0x02CA    0x0248    0x000C    0x02DB
0x000B    0xF44C    0x0403    0x1000    0x0000    0x0000    0xFFFF    0xFFFF
0xFFFF    0x00D1    0x01D7    0xFFFF    0xFFFF    0x1EB6    0x0189    0x03D6
0x0000    0x0800    0x0258    0x0064    0xFFFF    0xFFFF    0xFFFF    0xFFFF
0x0000    0x4CCD    0x000C    0x0000    0x00F0    0xFFFF    0xB333    0xFFF4
0x0000    0xFF10    0x0000    0x07AE    0x0018    0x0000    0x01E0    0xFFFF
0xF852    0xFFE8    0x0000    0xFE20    0x0003    0x0000    0x0078    0x0000
0x0000    0xFFFD    0x0000    0xFF88    0x0000    0x0000    0x003C    0x0003
0x0000    0x0000    0x0000    0xFFC4    0xFFFD    0x0000    0x0000    0x0000
0x00F0    0x000C    0x0000    0x0000    0x0000    0xFF10    0xFFF4    0x0000
0x0000    0x0000    0x038A    0x6400    0x05DC    0x05DC    0x0F5B    0x0035
0x00E4    0x02C6    0x013F    0x0007    0x017A    0x0042    0x0010    0x0E10
0x0000    0x03E8    0x00C8    0x5027    0x03B6    0x6000    0x0623    0x013F
0x4700    0x2A3F    0xD441    0xFD3F    0x7D3F    0x5441    0xAA3F    0xFF00
0xFFFF    0xC519    0x764B    0x5482    0x41B3    0x35C3    0x2D7A    0x276B
0x22C9    0x1F1E    0x1C28    0x19B5    0x17A6    0x15E6    0x1463    0x1312
0x11EB    0x10E4    0x0FFB    0x0F28    0x0E6B    0x0DC0    0x0D23    0x0C94
0x0C10    0x0B97    0x0B27    0x0ABF    0x0A5F    0x0A05    0x09B1    0x0962
*
```

The data from the text file can later be read into the controller IC by using option c. Most terminal programs have the option to send a raw file over RS232, this can be used to send a previously saved HEX file to the chip.

After using option c, press enter to restore the menu and then use option a to save to ROM.

Finally, when option d is turned on, offset calibration data, hall sensor information (when hall calibration selected) or entry into drive_1 gathered while running the motor can be stored to ROM. While in motor mode, activate setup to store the data. This step is not necessary for the (automated) hall measurement. All drive mode indicating LEDs will light up and the chip will return to drive_0.