Albis motor controller IC

Setup manual

version 1.11

Adliswil, Switzerland
bmp72@hotmail.com

Some general remarks:

Before connecting the battery make sure options d and e in menu c are 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, even when current sensor calibration is not used (menu d, option l). 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.

When a sensored motor is used several times during the setup the chip will ask for the motor to be spun by hand. This can be done both in the forward or the reverse direction but be sure to always spin the motor in the same direction. Later when the setup data is written to EEPROM the direction can be changed (write to EEPROM or reverse and write to EEPROM). The state written to EEPROM must be for the forward direction.

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.

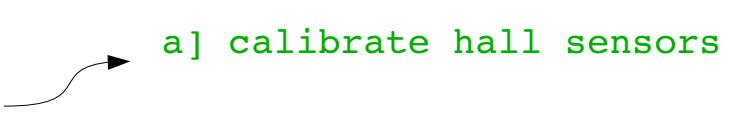
Some displayed values are calculated using more than one 16 bit variable, changing one variable can change the displayed decimal value of more than one menu option. Notable examples of far-reaching variables: sensor transimpedance (menu d, option b) and f_sample (menu e, option a)

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

```
########################################
#   (c)opyright 2013, B.M. Putter      #
#   Adliswil, Switzerland              #
#   bmp72@hotmail.com                  #
#                                      #
#  version 1.11                        #
#  experimental, use at your own risk  #
########################################


a] calibrate hall sensors
b] determine coil positions
c) PWM parameters
d) current settings
e) control loop parameters
f) throttle setup
g) running modes
h) CAN bus setup
i) Field Oriented Control
z) store parameters in EEPROM for motor use

------>
```

This is the main setup menu with the enter prompt. Make sure to selecting a menu option by using lowercase. Some options are only valid for sensored, some only for sensorless and some for both. This is indicated by:

x] a menu option only for sensored
x} a menu option only for sensorless
x) a menu option for both sensored and sensorless



Main menu option a] selects the hall sensor setup menu

```
a] number of e-rotations: 14
b] calibrate hall positions
c] table out hall signals
z] return to main menu

------>
```

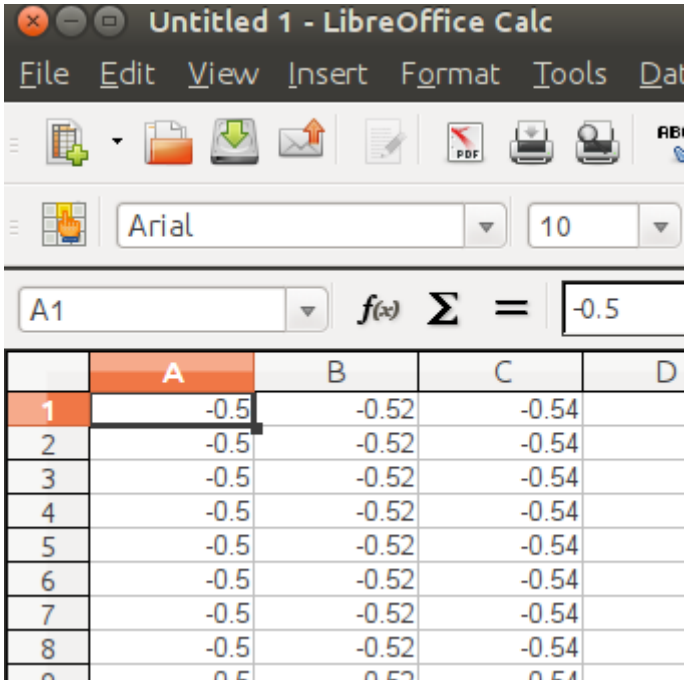
The controller IC can learn the relative position of the hall sensors with respect of each other. In order to do this it needs to see the hall signals toggle for a certain amount of e-rotations.

The amount of e-rotations is set using option a. It is best to choose a value such that the rotor makes an integer amount of revolutions. For a 7 pole motor 14 e-rotations equals 2 rotor rotations.

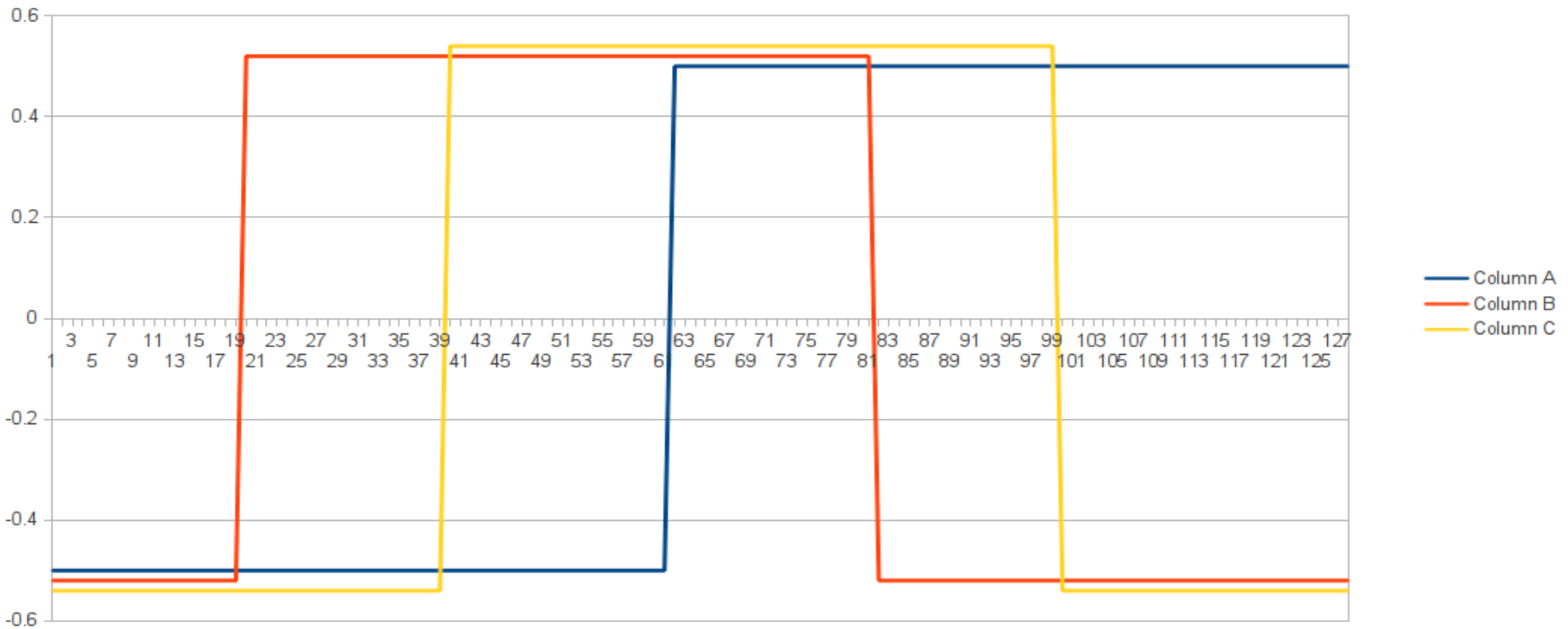
Option b will ask you to spin the motor (by hand) so that the IC can observe the toggling hall sensor signals. The IC will show a count of successful e-rotations. The motor is allowed to decelerate during this measurement but not at too fast a rate, add momentum to the rotor if necessary.

A spreadsheet program can be use to obtain graphical output of the hall signals. Option c will print a 128 row 3 column table. The 128 rows are conform the 360 degrees of an e-rotation, the 3 columns represent the output signals of the 3 hall sensors. Slightly different amplitudes are used (.5, .52, .54) for clarity, to make sure the graphs don’t overlap.

```
a] number of e-rotations: 14
b] calibrate hall positions
c] table out hall signals
z] return to main menu

------> c

hall 1, hall 2, hall 3
-.5000   -.5200   -.5400
-.5000   -.5200   -.5400
-.5000   -.5200   -.5400
```



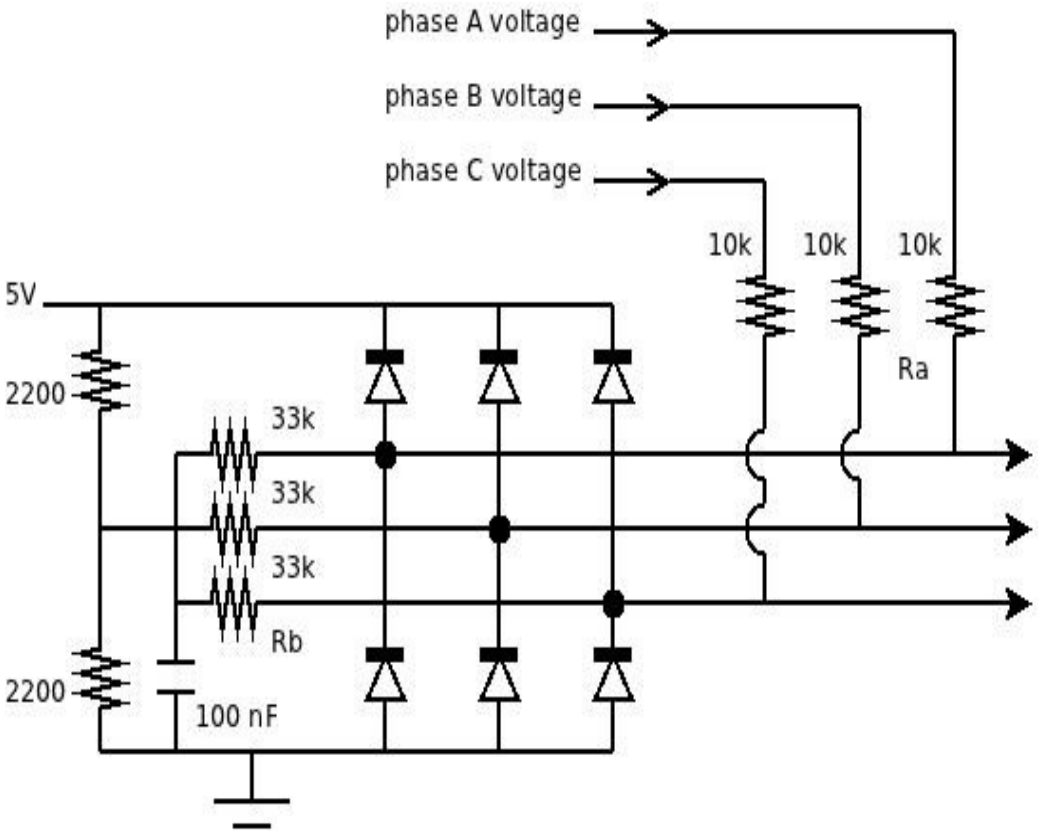
	A	B	C	D
1	-0.5	-0.52	-0.54	
2	-0.5	-0.52	-0.54	
3	-0.5	-0.52	-0.54	
4	-0.5	-0.52	-0.54	
5	-0.5	-0.52	-0.54	
6	-0.5	-0.52	-0.54	
7	-0.5	-0.52	-0.54	
8	-0.5	-0.52	-0.54	

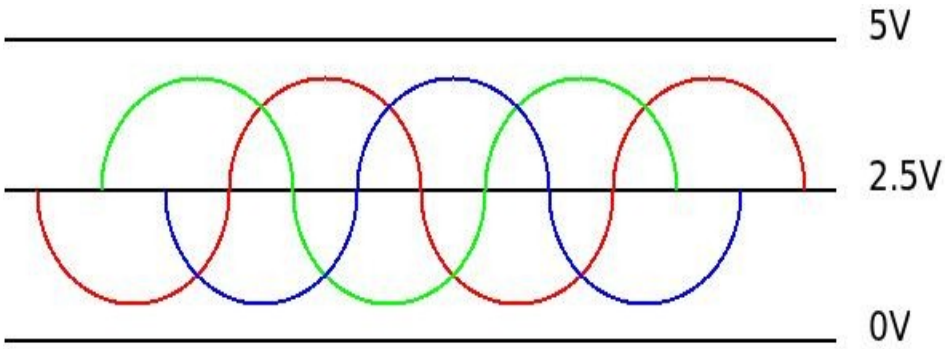


```
                                                    b] determine coil positions
```

```
a] number of back-emf samples: 850
b] calibrate coil positions
c] use only fundamental sine waves
d] reconstruct waveforms based on extracted parameters
e] table out data arrays
z] return to main menu

------>
```

This menu is only for a sensored setup, the purpose of it's functions are for the controller to learn the positions and polarity of stator windings and to relate this to the hall positions as measured earlier. This measurement will only be possible and yield valid results if the hall signals have been measured previously. Again, the controller will ask you to spin the motor after which it will measure the back emf signals. Essential for this is a relatively constant motor rpm, if the controller reports a failure to perform the measurements try to give the rotor more inertia.





back-emf to IC must fit in 5 V, adjust Ra, Rb accordingly.

For the measurements to be accepted by the controller the back-emf voltages must be within a certain range. If the rotor spins too fast (back emf voltages too high) it will wait for the rotor to slow down. The measurement will fail immediately if the back-emf voltage are too low. In this case the rotor must be spun faster or the resistors Ra / Rb in the schematic must be changed to yield a higher signal level at the controller pins.

It is best to perform this measurement with the battery connected. If the battery is not connected the diodes in the FET's can have an influence on the back-emf measurement. Dependent upon the type of output stage driver the output stage driver supply must be disconnected (for instance by removing the jumper as drawn in the schematic, see slide 28).

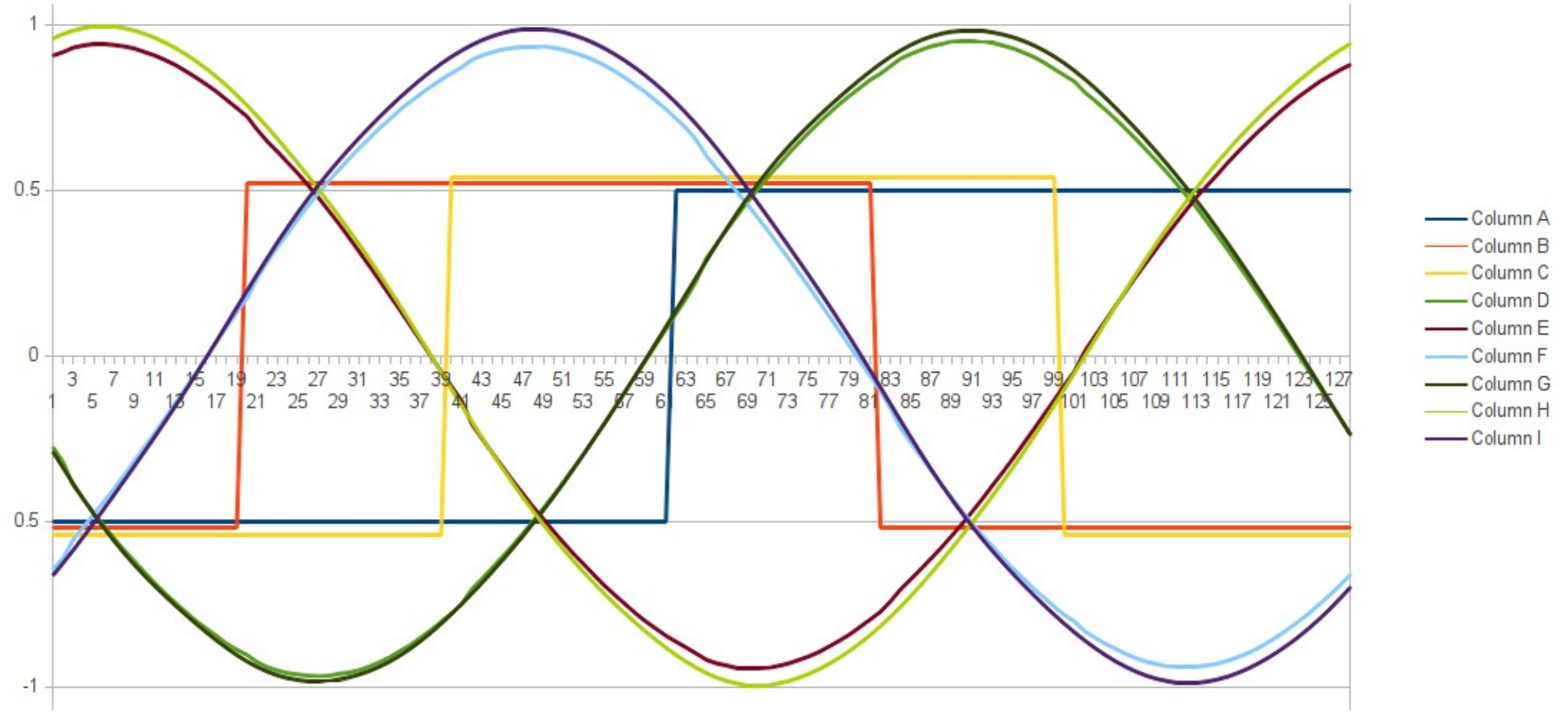
```
                                                    b] determine coil positions

 a] number of back-emf samples: 65535
 b] calibrate coil positions
 c] use only fundamental sine waves
 d] reconstruct waveforms based on extracted parameters
 e] table out data arrays
 z] return to main menu

 ------>
```

The back-emf voltages are measured over multiple e-rotations and averaged. Each e-rotation the controller will try to make 128 measurements. It is best to use an amount of measurements such that the rotor makes an integer amount of rotations. The amount of measurements will be round up such that an integer amount of e-rotations is measured. The number 850 under option a yields 850/128 = 6.6 e-rotations which will be rounded up to 7 (which for my motor equals 1 rotor revolution).

Option b performs the actual measurement. After selecting it the controller asks you to first spin the motor and then press any key. Measurements start immediately after the key press. If the rotor turns too fast (voltage too high) the controller will wait for the rotor to slow down. Measurement will fail if the rotation is not 'uniform' enough. After a successful measurement the data arrays are filled with the raw measured data, use option d to obtain the table (which can be used to generate a plot). Don't worry if the raw measured data looks 'spiky' and with missing samples, a cross-correlation algorithm has been implemented to deal with this eventuality.

For motor operation use the controller does not store the full back-emf waveforms. The controller dissects the measured waveforms of option b into it's fundamental sine wave and a 3^{rd}, 5^{th}, 7^{th} and 9^{th} overtone. The amplitude and phase of the fundamental and its overtones are stored and then (for motor operation) used to reconstruct the back-emf waveforms. Option d can be used to fill the data arrays with the reconstructed waveforms, they can be outputted using option e and compared with the raw measurements.

Option c removes all overtones and forces the controller to use sine waves (with moving midpoint).



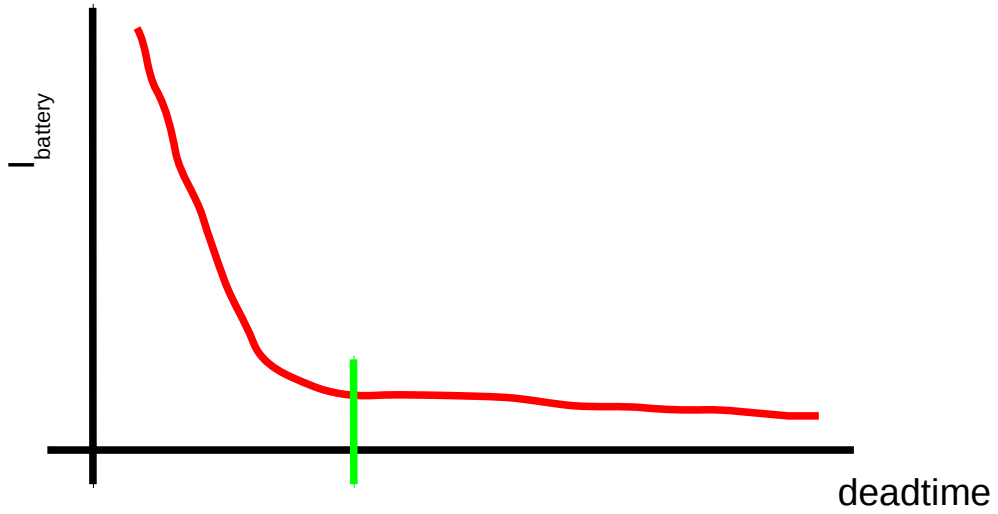
These are the waveforms collected from my motor. You can see the hall signals (A,B,C), the sampled waveforms (D,E,F) and the reconstructed waveforms (G,H,I). Note the very, very bad hall sensor placement for a MC33033 type controller. For this type of controller one of the back-emf waveform must cross 0V right in between two hall transitions. Here however the hall transitions practically coincide with the 0V crossings...

```
a) PWM frequency: 21kHz
b) deadtime: 599ns
c) dutycycle testsignal: 50%
d) toggle high side polarity, now active HIGH
e) toggle low side polarity, now active HIGH
f) test PWM signals
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.



Start by correctly setting options d & e. The first thing to do is to find the deadtime for which shoot-through (where both FETs are on) occurs. Start with a low PWM frequency (10 kHz), a large dead time (2000 nsec) and 50% dutycycle. It is better but not strictly necessary to have the motor connected to the output stage (all 3 motor phases will be driven in phase, there will be no effective motor voltage). Use option f to generate a test signal.

With the test signal active, measure the supply current of the output stage (battery current). Repeat this measurement several times with decreasing deadtimes. A plot of the measured supply currents should be relatively constant until a certain deadtime below which there is a sharp rise in current. The deadtime to use is longer than the point where the sharp rise in current occurs.

The PWM frequency can now set by inspection (using an oscilloscope) or by relating it to the deadtime. With the PWM period time around 50 times the dead time PWM_freq is 1 / (50 * deadtime). Make sure though that the driver stage does not overheat, else reduce the PWM frequency. It is normal that for a higher PWM frequency the supply current increases.

After setting up the output stage correctly a 10% dutycycle test signal should look nice and crisp on an oscilloscope. When performing this measurement, look at the gate drive signals.

```
a) number of current sensors: 3                                    d) current settings
b) current sensor transimpedance: 255.99 mV/A
c) maximum motor phase current: 52.0 A
d) maximum battery current, motor use: 52.0 A
e) maximum battery current, regen: 52.0 A
f) maximum shutdown error current, fixed: 52.0 A
g) maximum shutdown error current, proportional: 52.0 A
h) IIR filter coefficient, throttle current: 65535
i) IIR filter coefficient, error current: 65535
j) use additional comb filter: YES
k) use offset calibration: YES
l) restore default calibration
z) return to main menu
```

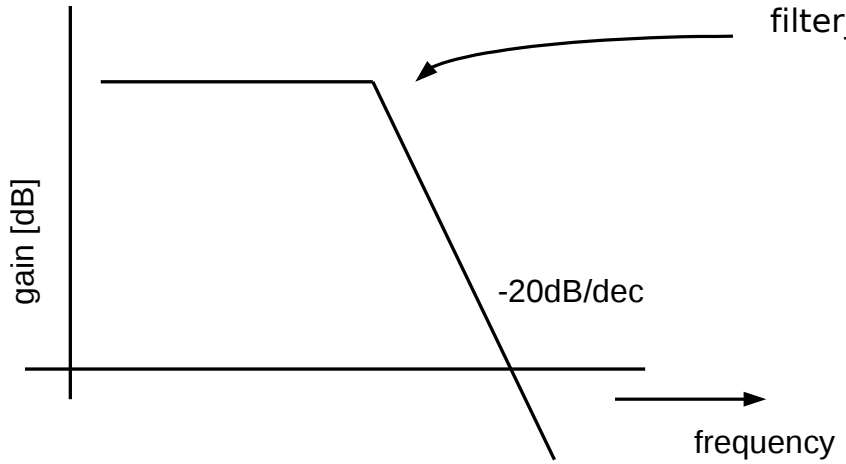
Option a tells the controller how many current sensors are present, valid values are 2 or 3. The main characteristic of a current sensor is its mV/A rating, this is specified under option b. Changing option b has an immediate effect on the current values displayed by the subsequent options and should therefore be set first !

Option c tells the controller what the maximum phase current of the motor is. The current specified here is multiplied by the effective throttle value (which is in the range of -1 .. 1) to produce the amount of motor phase current the controller will try to achieve. This under the condition that it does not violate the maximum ratings under options d and e. Note that there is no explicit over-current protection for the battery current, only indirect by limiting the throttle ! An error between the wanted and actual current can cause the protection to trip and shut down the motor. The maximum tolerated error has a fixed part (option f) and a proportional to amplitude part (option g), recommended for both is around 5% to 10% of option c.

Options b and c must be such that the maximum phase current times the trans impedance is less than 2 V, in equation form: (option b) * (option c) < 2 . The controller does not check for this, it is the responsibility of the user. The current sensor must be matched to the motor, the closer to 2 the better.

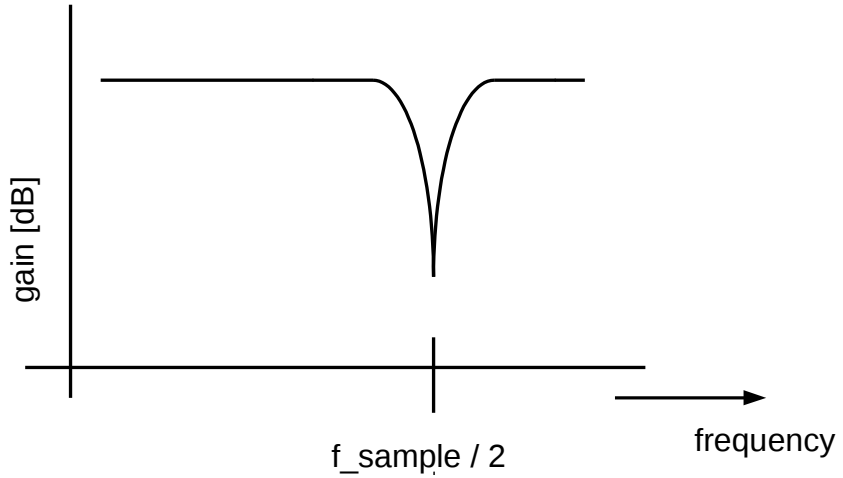
The measured currents are filtered by a first order IIR filter with selectable (-3dB) filter frequency (options h, i) and an optional comb filter (to reduce the PWM induced current). Using the comb filter is NOT recommended for sensorless - self start.

The transfer curve of the IIR filter:



If you're not sure what value to choose for option h: useful values are in the 2 to 8 range with 4 or 5 recommended. F_sample is set in menu e, option a. Option i accepts values in the 0-3 range with 3 recommended.

The transfer curve of the comb filter:



The comb filter is meant to reduce the PWM induced triangular current as seen by the controller (it does not reduce the triangular current in the motor phase wires !). Since it is a very sharp narrow filter it is only effective when the PWM frequency is close to half the sample frequency.

The trade-off w.r.t. the filters is to filter strong enough not to have a spike trip the shutdown error current but not to filter so strong that valid error conditions are no longer detected.

```
a) number of current sensors: 3                                  d) current settings
b) current sensor transimpedance: 255.99 mV/A
c) maximum motor phase current: 52.0 A
d) maximum battery current, motor use: 52.0 A
e) maximum battery current, regen: 52.0 A
f) maximum shutdown error current, fixed: 52.0 A
g) maximum shutdown error current, proportional: 52.0 A
h) IIR filter coefficient, throttle current: 65535
i) IIR filter coefficient, error current: 65535
j) use additional comb filter: YES
k) use offset calibration: YES
l) restore default calibration
z) return to main menu
```

Typically the current sensors have an offset, when option k is set the controller will try to remove the offset by calibration. Calibration occurs only when the controller is in drive 3 (see status LED) and can take upto 30 seconds. When option k is set the controller will always calibrate when in drive 3, it is an on-going process.

Option l must be used to restore the default calibration values which will later (using option z of the main menu) be written to EEPROM, failure to do so will cause very high currents in the motor and output stage. Option l must also be used when offset calibration is not selected !

Upon startup the controller will read the calibration values from EEPROM. It is highly unlikely that the default values as written using option l are correct. It is therefore possible to write the calibration values during motor use by pressing the setup switch. The controller will then write the calibration values as known at that moment to the EEPROM (run the motor in drive 3 for at least 30 seconds at medium rpm and low mechanical load to make sure the calibration algorithm has time to settle). The controller will flash all four drive LEDs to indictate the values are written, before jumping to drive 0. The next time the controller is powered up the new initial values for the calibration will be read. Once stored the calibration is turned off automatically.

It is recommended to leave calibration off, it's better not to use it when driving around.

```
a) loop sample frequency: 40.00 kHz
b) 1st order phase loop integrator coefficient: 12.7999
c) 2nd order phase loop integrator coefficient: 0.0709
d) amplitude loop integrator coefficient: 3.0000
e) maximum amplitude: 200 %
z) return to main menu

------>
```

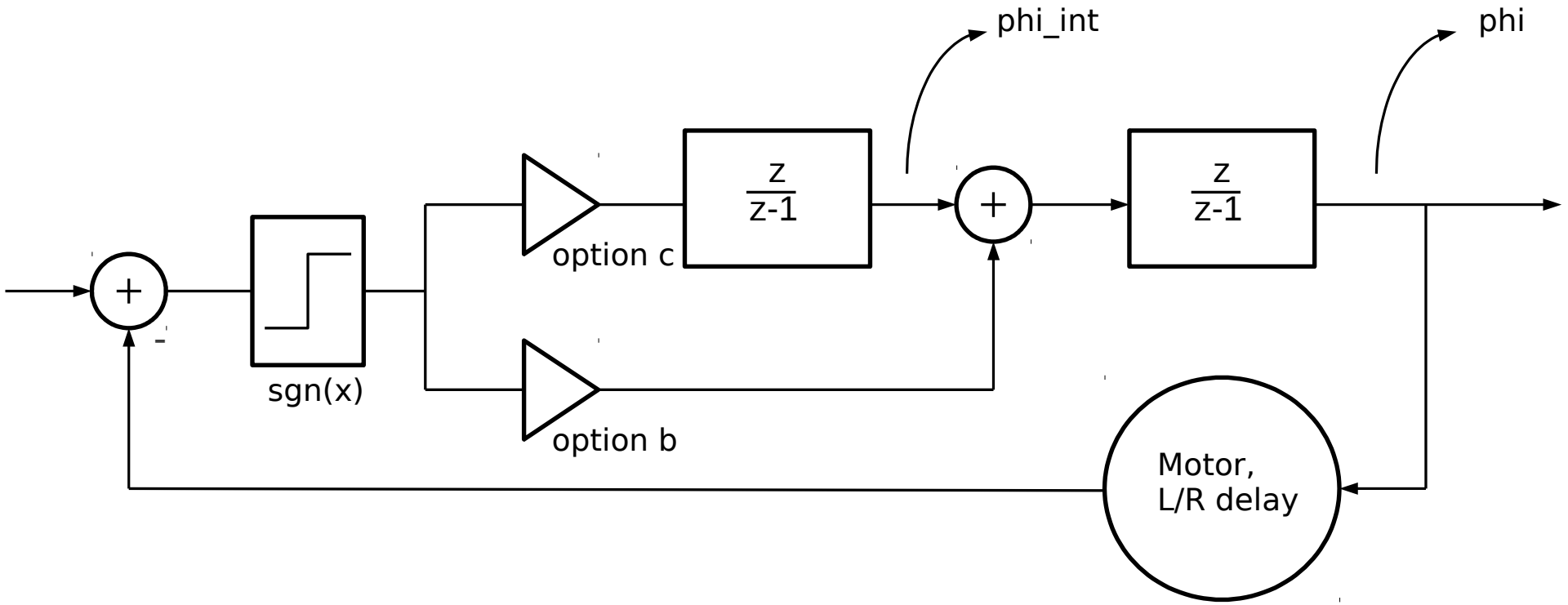
This is the setup menu for the phase and amplitude control loops.

The control loops perform measurements using the ADC's, update all parameters and write new values to the PWM outputs at a rate of f_sample times per second. F_sample can be set using option a and can be up to 100 kHz. Most sensible is a value a few kHz away from double the PWM frequency. This ensures a close to maximum update rate of the PWM dutycycle, an effective comb filter and reduced susceptibility to (PWM induced) interference.

Options b,c and d are explained on the next slides.

Option e sets the maximum amplitude of the PWM-approximated (moving mid-point) sine waves. A value of 100 % indicates a sine wave amplitude of half the battery voltage (so a full sine wave fits between ground and $V_{battery}$), 111% makes full use of the moving mid-point to generate 11% larger sine waves. Values larger than 111% will introduce clipping and allow the approximation of trapezoidal waves. Valid values are from 0% to 200% .

The phase control loop can be modeled by:



The phase control loop is a second order bang-bang loop (the sgn(x) block outputs +1 or -1). Options b and c are the gain factors for the first and second order. The input/output phase values are 16 bit integers which will be rounded to 7 bit for the PWM outputs. The motor introduces a delay in the system of L/R.

The following slide shows some easy to follow guidelines for calculating the values for options b and c.

The recommended method for calculating b and c is to define 2 intermediate values:

y = 256
x = 3 * LR_delay * f_sample

With these:

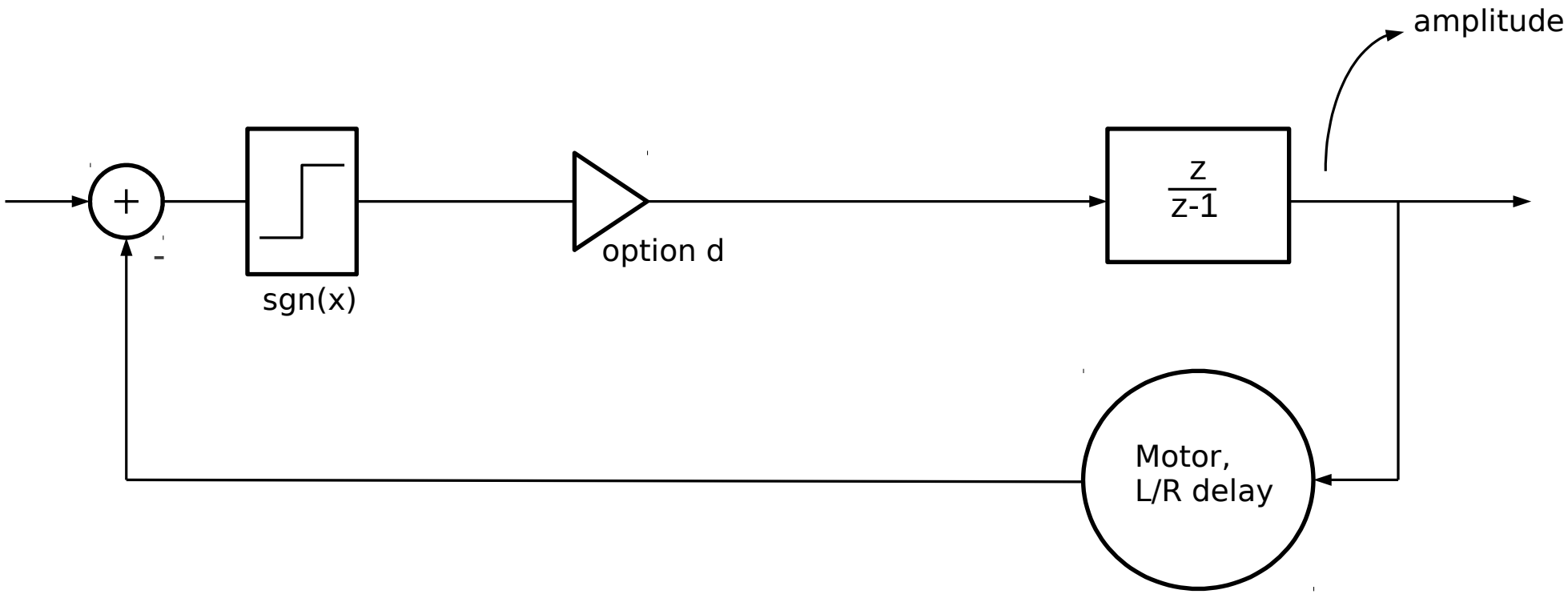
Option c = y / (x^2)
Option b = 3y / x

y represents a gain factor common to both coefficients, the 256 given here is a good initial guess for a low e-rpm hub motor (1024 for high e-rpm RC), but experiment with different values.

The LR_delay is typically not known but luckily also not very critical. Most motors hover around the 1 to 2 msec level, choose this as an initial guess (and experiment).

For a high f_sample the value for option c can become very small, around 0.0003 or lower it’s recommended to increase y and recalculate b and c.

The amplitude control loop can be modeled by:



The amplitude control loop is a first order bang-bang loop (the sgn(x) block outputs +1 or -1).
Options d is the gain factor. Recommended value (but again, not very critical so experiment):

Option d = PWM_frequency / (458 * LR_delay * f_sample)

```
                                                        f) throttle setup

   a) calibrate throttle 1
   b) calibrate throttle 2
   c) polynomal coefficients throttle 1 (x, x^2, x^3): 0.5000, 0.0000, 0.0000
   d) polynomal 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:

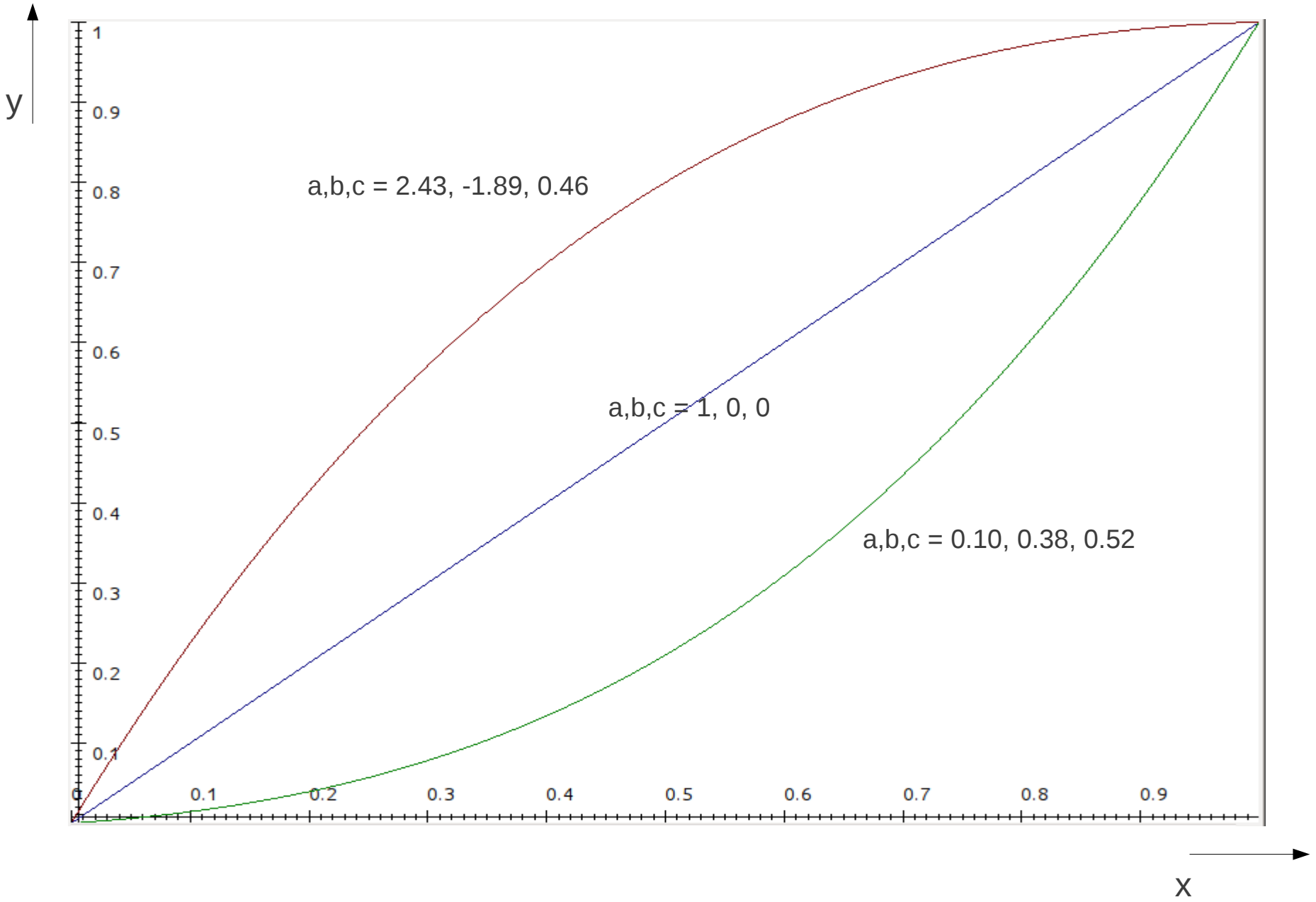
phase current = maximum phase current * (y_1 + y_2)

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

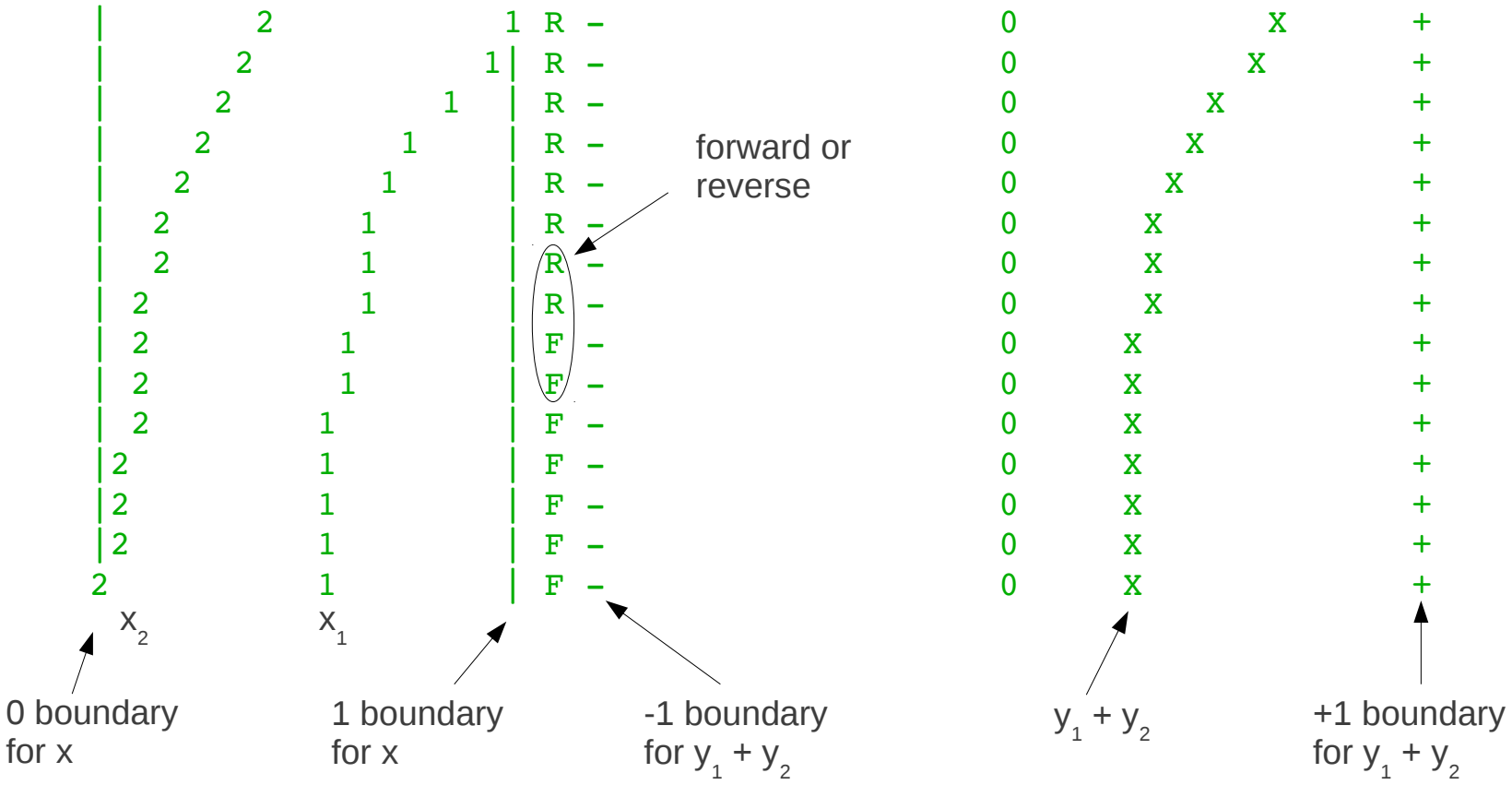
Variables x_1, x_2 are shared over CAN bus. Every motor controller IC however has its own set of a, b and c coefficients and its own maximum phase current setting. This allows the combining of motors with different ratings to operate of a shared throttle.

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).



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.

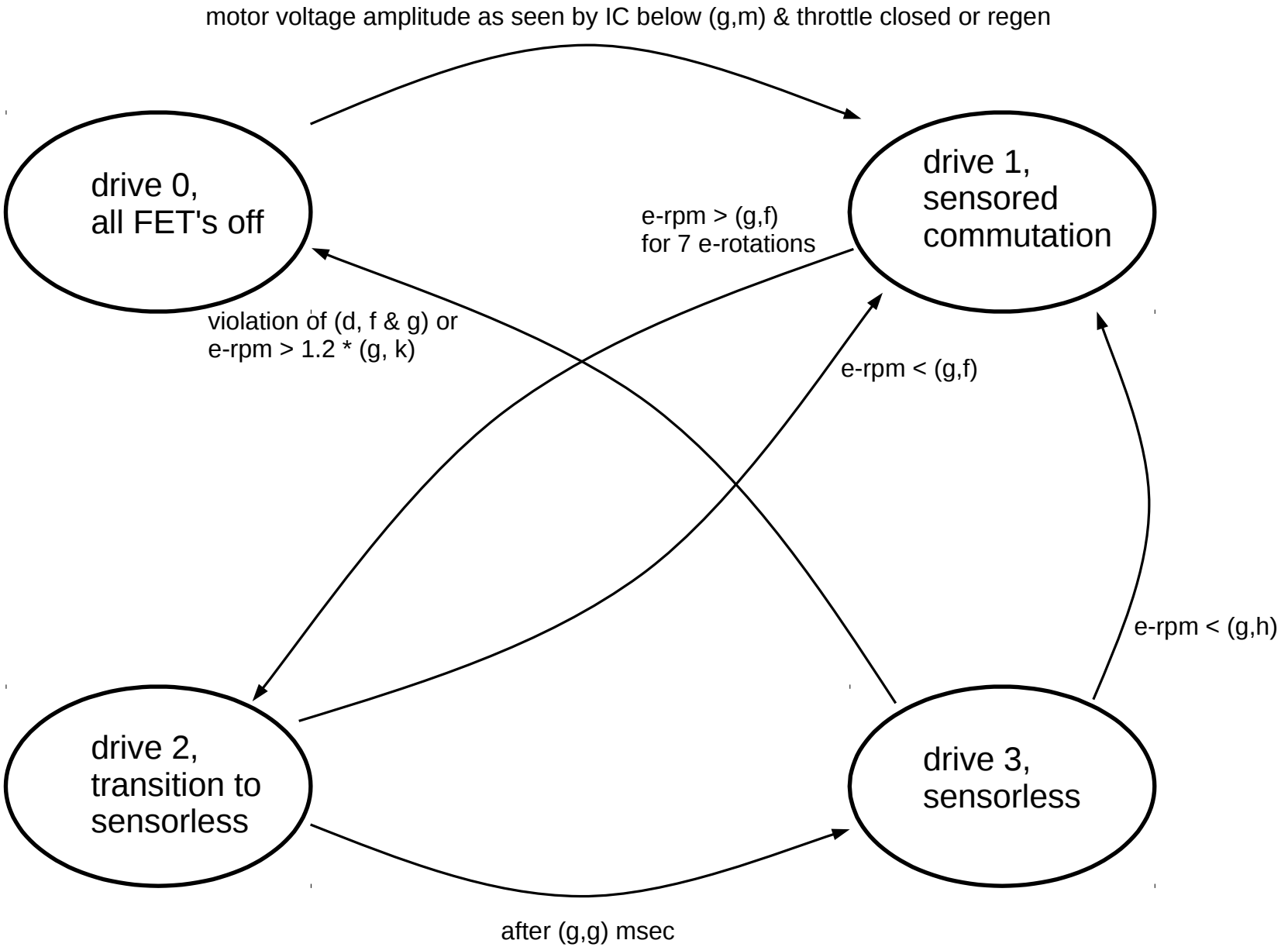
```
                                                                g) running modes
a) sensored or sensorless: SENSORLESS
b} sensorless startup: PUSH START
c} e-rpm limit sensorless self start: 585
d} minimum current push start: 0.9 A
e} push start current, error allowed: 10 %
f] erpm sensored to sensorless transition: 600
g] transition time sensored to sensorless: 199 milli-sec
h) return to motor start below 200 erpm
i) controlled slowdown for direction change: NO
j) phase current for controlled slowdown: 0.0 A
k) motor maximum, forward: 6.99 k-erpm
l) motor maximum, reverse: 6.99 k-erpm
m) motor standstill voltage threshold: 0.49 V
n) enable low side pulsing in drive 0: YES
o) low side pulsing rate: 20 Hz
p) low side pulsing width: 20 usec
z) return to main menu

------>
```

Most of the options in this menu set the conditions on which the main state machine changes state. There are 2 state machines, one for sensored and one for sensorless (see the next two slides). The states are called drive 0, drive 1, drive 2 and drive 3. The current state is indicated by the LEDs on pins 14-17. The lit up LED indicates the current state when the motor is in forward, in reverse all LEDs light up except the one indicating the state. Drive 3 is the main operating mode in which the motor runs sensorless with automatic timing advance, the setup should be aimed at reaching this drive mode as soon as possible.

Options k & l are the maximum e-rpm's allowed in forward and reverse. Make sure to set k & l realistically ! When k or l is violated the controller automatically closes the throttle until e-rpm has dropped under the set limits. A second rpm limit 20% above option k causes total shutdown if violated (transition to state 0). This implies also that l should be less than 120% of k.

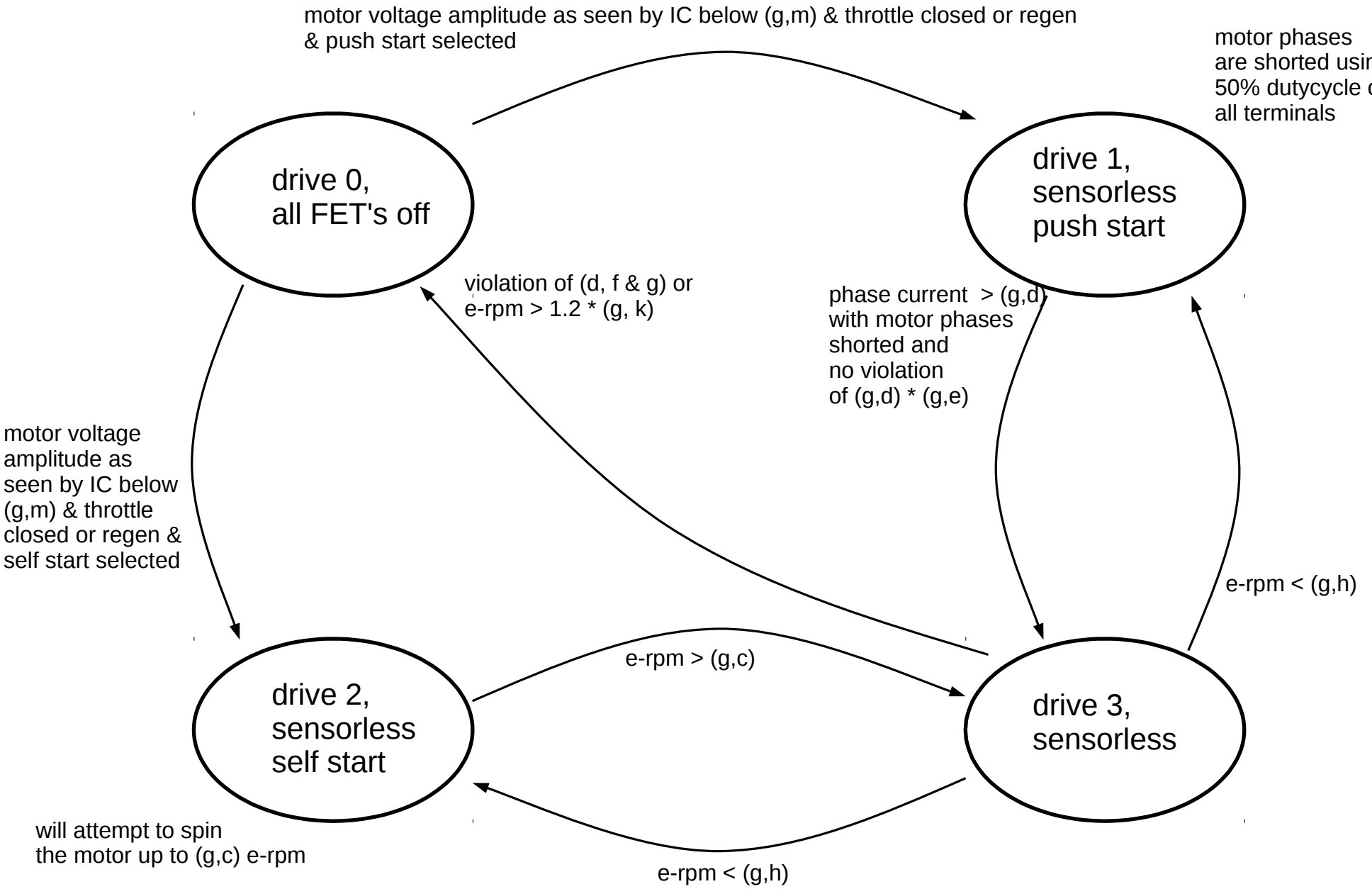
State machine, sensored operation

(x,y): menu x, option y



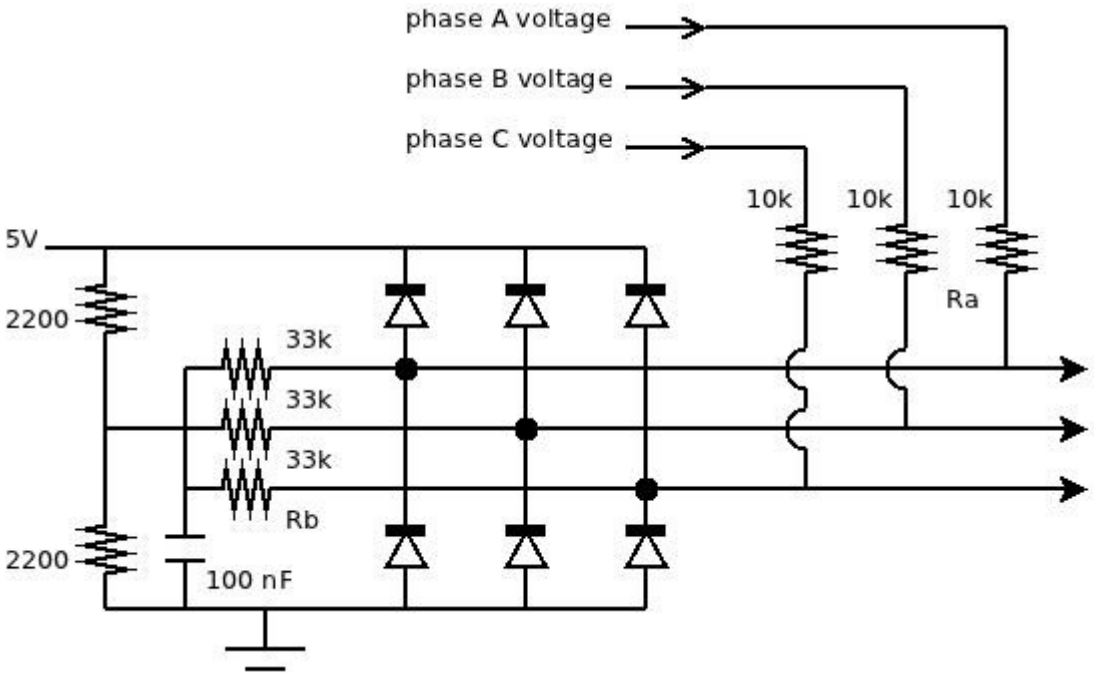
State machine, sensorless operation

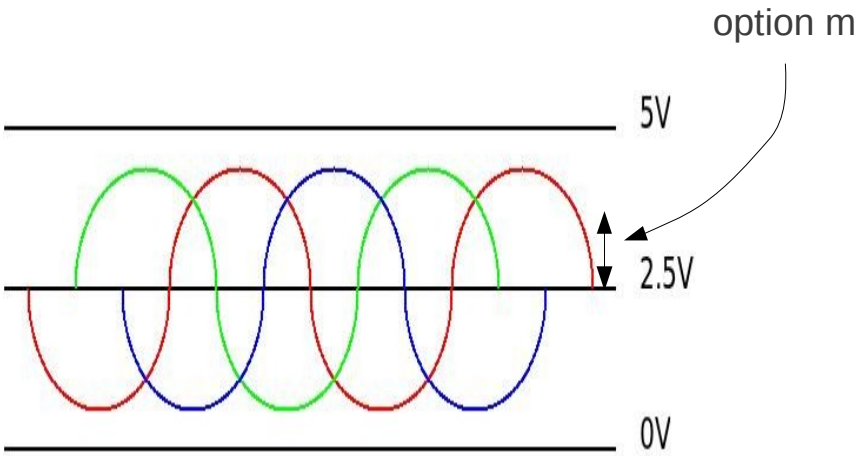
(x,y): menu x, option y



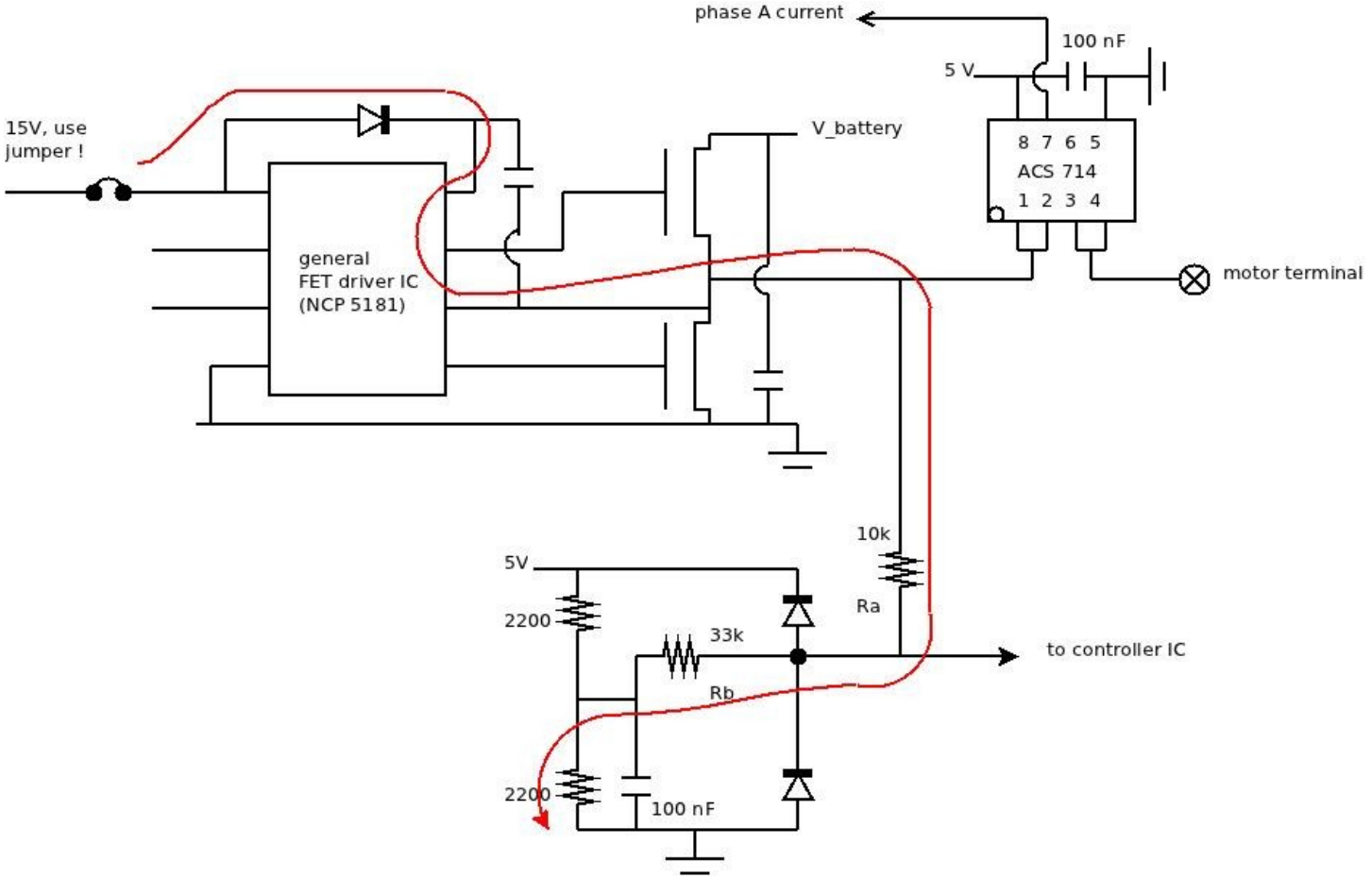
Options i and j have to do with the way in which reverse is implemented. The state of the reverse switch is treated as a request, not as a command. To prevent damage to the output stage a change in direction is only executed when the motor e-rpm is low enough. In sensored operation the direction change request is granted in drive 0 and 1, in sensorless operation it's granted in drive 0, 1 and 2. When the direction request is different from the actual direction, the throttle is closed automatically with the controller only responding to regen. Options i & j allow to automatically apply a set amount of regen when a direction change request is received.
Note that the state LEDs indicate the actual state of the motor direction, throwing the reverse switch will only invert the LEDs once the reverse request is granted.

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 m.



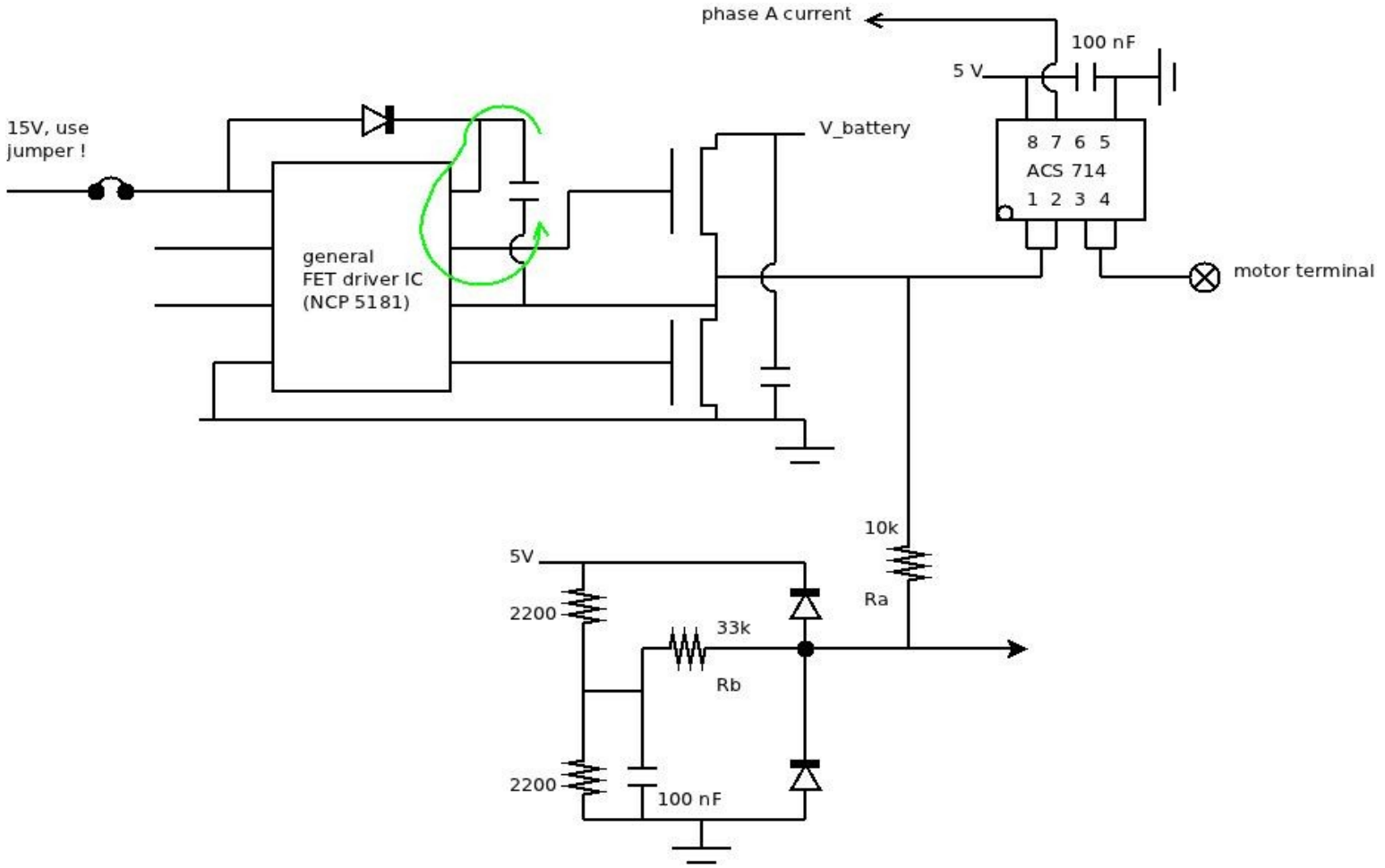


Dependent on the type of output driver a bias current can pass through resistors Rb in which case the detected voltages will never fall below 2.5V+option m.



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 m 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.

Note that the same mechanism can influence the back-emf waveform measurements, disconnect the 15V FET driver supply (remove jumper) to prevent this !



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 m.

The pulsing of the low side FET during drive 0 is turned on or off with option n. Options o and p 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.

```
                                                          h) CAN bus 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. A slave controller that does not receive valid throttle information for about 50 msec will transide to drive 0 and release the motor.

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

```
                                                    i) Field Oriented Control
 a) use Field Oriented Control: YES

  Before automatic measurement of the motor parameters
  the controller must be supplied with the same voltage
  as in the vehicle. PWM frequency and deadtime must be
  initialised. ADC's must be properly set up and calibrated
  or calibration must have been restored to default.
  The following parameters must be set at their final value:

  loop sample frequency: 40.00 kHz
  current sensor transimpedance: 100.00 mV/A

 b) amplitude measurement current: 11.9 A
 c) impedance measurement frequency: 24.97 k-erpm
 d) determine motor impedance

 e) battery voltage (for inductance display only): 78 V
    measured inductance (star configuration): 36.3 uH
    total system 'resistance' (star configuration): 635.8 mOhm

 z) return to main menu
```

Option a turn FOC on or off. When off only options a and z are shown, not the rest of the menu.

Determining the motor impedance for FOC should be performed last, as the controller determines not so much the motor inductance as more an internal variable it needs to run the motor with FOC. When, among others, variables like loop sample frequency or current sensor impedance are changed the internal variable must be updated. Changing a variable without updating the FOC internal variable will result in incorrect motor operation. Therefore, perform FOC calibration last when all other variables are fixed.

Same goes for the main battery supply, the FOC calibration should be performed with the vehicles battery voltage !!!! This because the internal FOC variable depends on the battery voltage.

```
                                                    i) Field Oriented Control
 a) use Field Oriented Control: YES

  Before automatic measurement of the motor parameters
  the controller must be supplied with the same voltage
  as in the vehicle. PWM frequency and deadtime must be
  initialised. ADC's must be properly set up and calibrated
  or calibration must have been restored to default.
  The following parameters must be set at their final value:

  loop sample frequency: 40.00 kHz
  current sensor transimpedance: 100.00 mV/A

 b) amplitude measurement current: 11.9 A
 c) impedance measurement frequency: 24.97 k-erpm
 d) determine motor impedance

 e) battery voltage (for inductance display only): 78 V
    measured inductance (star configuration): 36.3 uH
    total system 'resistance' (star configuration): 635.8 mOhm

 z) return to main menu
```

Option b sets the current used for the motor impedance measurement, option c the frequency.

It is recommended to set option b quite high, around half the maximum motor phase current (menu d, option c). For option c I recommend the maximum motor e-rpm.

Option d starts the impedance measurement. During the measurement the motor must be standing still. It will make quite some noise, this is normal.

As mentioned on the previous page, the measurement determines an internal variable. To convert this into an inductor/resistor value the controller must know the battery voltage, set this with option e.

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

   a) write variables to EEPROM
   b] reverse direction and write variables to EEPROM
   z) return to main menu

   ------>
```

Except for the high/low active of the PWM output signals all variable updates as entered in the menu options are made in RAM only. With this menu they are stored in EEPROM so that they are retained even when the power is turned off.

Option a writes the data to EEPROM, as is.

Option b reverses the motor direction before writing the data to EEPROM. This allows for the sampling of the hall sensor positions and back-emf voltages in reverse. This can be useful when the motor has a freewheel. Reversing the direction can only be done for sensored operation, for sensorless two motor wires have to be inter changed.