RH850/F1KM-S1

R11AN0284ED0002
Rev.00.02
Feb.18.2019

Field Oriented Motor Control on a RH850/F1KM-S1 Starterkit V3

Introduction

This Application Note describes the software and parts of the used hardware for the field-oriented motor control, which comes with the Renesas RH850/F1KM-S1 Starter kit V3 (Y-BLDC-SK-RH850F1KM-S1).

The kit enables engineers to easily test and evaluate the performance of RH850/F1KM-S1 in a laboratory environment when driving a 3-phase Permanent Magnet Synchronous Motor (PMSM) using an advanced sensorless field-oriented control algorithm. Typical applications for this type of solution are compressors, air conditioning, fans, air extractors, pumps, home appliances inverters and industrial drives.

The Starter Kits are coming with a powerful user-friendly PC Graphical User Interface (GUI), which gives real time access to key motor performance parameters. But it is also possible to drive the motor stand-alone with no PC connected by using the mounted encoder and the potentiometer. Some parameters can also be shown on the OLED Display (not included).

The phase current measurement is done via three shunts, which offers a low-cost solution, avoiding the need for an expensive current sensor or hall sensor.

The Hardware of the starter kit are described in the related user manuals and quick start guides in detail:

- RH850/F1KM-S1 Starter Kit V3 [R12UT0004ED0101]
- QSG for RH850/F1KM-S1 Starter Kit V3 [R12QS0027ED0001]

You can find all software packages and documents under the following websites:

- http://www.renesas.eu/update?oc=Y-BLDC-SK-RH850F1KM-S1

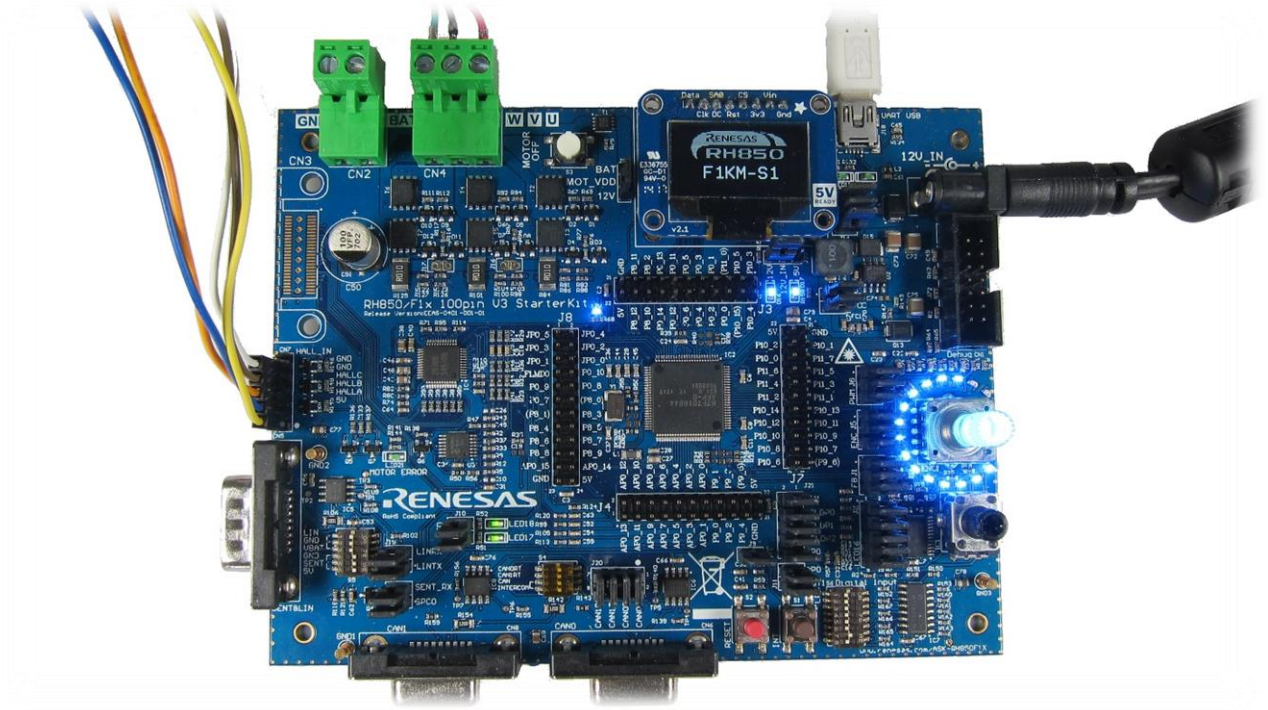


Figure 1. RH850/F1KM-S1 Starter kit

Table of Contents

1. Hardware Components Overview

In this section, some of the mounted hardware components are listed.

Table 1. Hardware components overview

ITEM	SPECIFICATIONS
Types of motors supported	• 3-phase Permanent Magnet Synchronous Motor (PMSM, PMAC, BLAC) • 3-phase brushless DC (BLDC)
KIT motor partname	Fulling Motor FL28BL26-15V-8006AF, 15VDC, 8000 RPM
Kit maximum input range	Included power supply: 12V External power supply: 5.3V to 18VDC
Transistor used	Renesas MOSFETs: NP75N04YUG, 40V, 75A
Transistor driver	Renesas MOSFET driver Intelligent Power Device: R2A25108KFP Operating voltage: 5.3V to 18V(VBAT), typically 12V
Provided Power Supply	$12V_{DC}$, 1A
Current detection	Three shunts configuration (10mΩ)
USB IC used on the board	FT232RQ-Tube – USB to UART IC from FTDI, 9.6 kBd communication speed
Microcontroller	• RH850/F1KM-S1 (R7F701684), 100-pin QFP, 120 MHz, 1024 kB Code Flash, 64kB Data Flash, 96kB LRAM, 32kB RRAM
MCU Performance	F1KM-S1, 120 MHz: 93 Automark, 343 CoreMark (v1.0), 271 DhrystoneMIPS (v2.1)
Key features used	General Purpose Timer Array Unit D (TAUD), Motor Control Function, 12-bit A/D Converter, LIN/UART module.
MCU embedded Firmware	Sensorless vector control algorithm (Field Oriented Control)
Switching frequency (PWM frequency)	10KHz to 20KHz, 20KHz by default
Control Loop Frequency (sampling frequency)	4KHz to 20KHz, 10KHz by default
Control loop timing	~17µs including the FOC loop and current and speed PI controllers (when above default settings are used and the STANDALONE define is commented out)
Code size in flash / RAM	~ 58kB ROM, ~ 6.5kB RAM
Tool used, version	• Green Hills MULTI v6.1.6 (AP 2015 1.7 Patch) • IAR Embedded Workbench V1.40.4 • CubeSuite+ for CC V6.00.00 • winIDEA Build 9.17.17 (73896)
Compiler optimization level	none
Environment standards	RoHS compliant including China regulations WEEE, RoHS

To obtain the maximum flexibility, the starter kit includes:

- A complete on-board 3-phase inverter with a low voltage motor (15V), so it becomes easy to test the powerful sensorless algorithm running on the Renesas 32-bit RH850/F1KM-S1 microcontrollers
- DC bus voltage provided via 12V external power supply or by connecting your own power supply
- Renesas low voltage MOSFET power devices. Power rating up to 3000W
- Three shunt current measurement resistors
- Renesas intelligent power device R2A25108KFP to pre-drive the power MOSFET inverter bridge, sense and amplifier the shunt current
    - Wide range operating: 5.3V to 18V(VBAT), 24V<60s
    - On-chip three phase pre-driver circuit
        - PWM control
        - Totem pole type MOSFET gate drive circuit, high drive capability:

          Ciss = 10000pF
        - Dead time control (adjustable)
    - On-chip current sense amplifier with reference bias buffer
- USB communication using FT232RQ USB to Serial UART/FIFO IC.
    - The interface is configured for the PC GUI to control Motor operation and for modifying motor and control parameters
    - Offering 9.6KBd communication speed with the PC GUI

2. Power Stage Description

The power board is a complete 3-phase bridge composed with discrete low voltage and high current MOSFETs. The MOSFETs are the Renesas **NP75N04YUG** n-channel power MOSFETs. Please refer to the data-sheet available on the Renesas website: www.renesas.eu for the switches characteristics and to the board schematics for the details on the driving circuit. The maximum current is **75A**, and the maximum voltage is **40V$_{DC}$**.

The inverter has the classical schema as shown in the following figure with the three shunts on the lower arms.

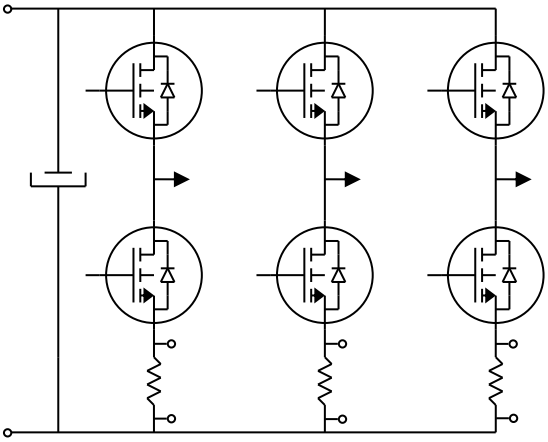


Figure 2. Classical Schema Three Shunts Bridges

Furthermore, the intelligent power device R2A25108KFP is adopted to pre-drive the six low voltage MOSFETS NP75N04YUG. See chapter “3 Renesas Intelligent Power Device R2A25108KFP Short Overview” for more details.

3. Renesas Intelligent Power Device R2A25108KFP Short Overview

The R2A25108KFP device is an Intelligent Power Device to drive the power MOSFET inverter of the three phase brushless motor.

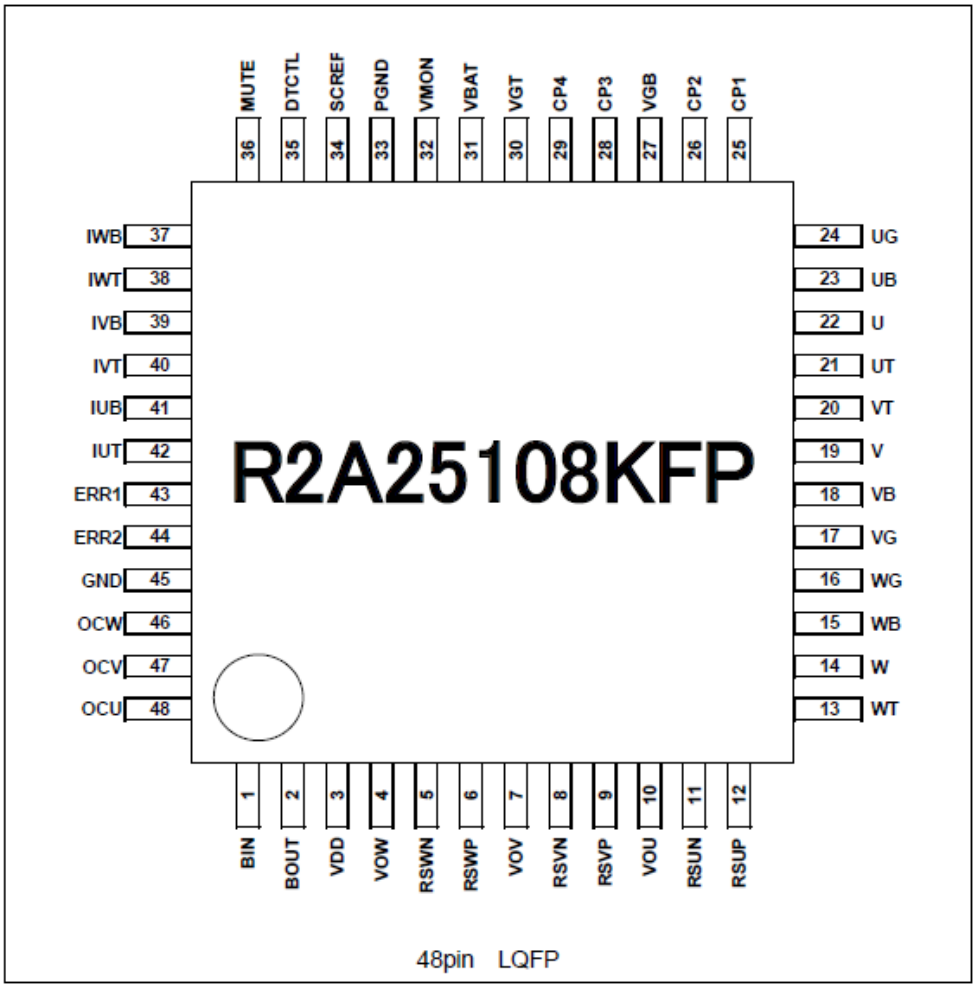


Figure 3. Top View of R2A25108KFP

This device contains three sets of MOSFET-drivers, charge pump circuit for the gate drive of high side and low side external power MOSFET, zero-crossing detector for detection of motor position, three channels of current sense amplifier and safety functions as over voltage detection circuit (OVD), low voltage detection circuit (UVD), thermal shut down circuit (TSD), short circuit protection etc.

The main features of the R2A25108KFP are as follows:

- Wide range operating voltage: 5.3V to 18V (VBAT)
- On-chip three phase pre-driver circuit
  - PWM control
  - Totem pole type MOSFET gate drive circuit, high drive capability: Ciss = 10000pF
  - Dead time control (adjustable)
- On-chip charge pump circuit (for power supply of gate drive)

- On-chip zero-crossing detection circuit
- On-chip current sense amplifier with reference bias buffer
- On-chip safety functions
    - Low voltage detection circuit (LVD)
    - Over voltage detection circuit (OVD)
    - Thermal shut down circuit (TSD)
    - Short circuit protection; adjustable detection level
        - Function for the short to battery protection
        - Function for the short to GND protection
- Internal oscillation circuit; 175 KHz typ.
- 48 pin LQFP package
- Comply with AEC-Q100

For further details please refer to the data-sheet available on the Renesas website: [www.renesas.eu.](http://www.renesas.eu)

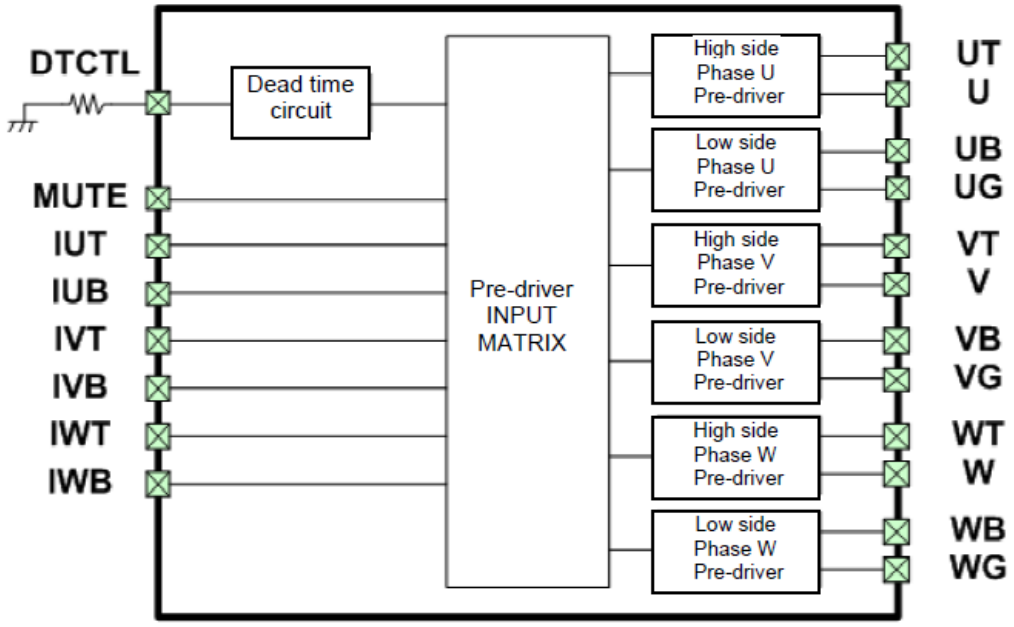


Figure 4. R2A25108KFP for Gate-Driver

An application example is shown below.

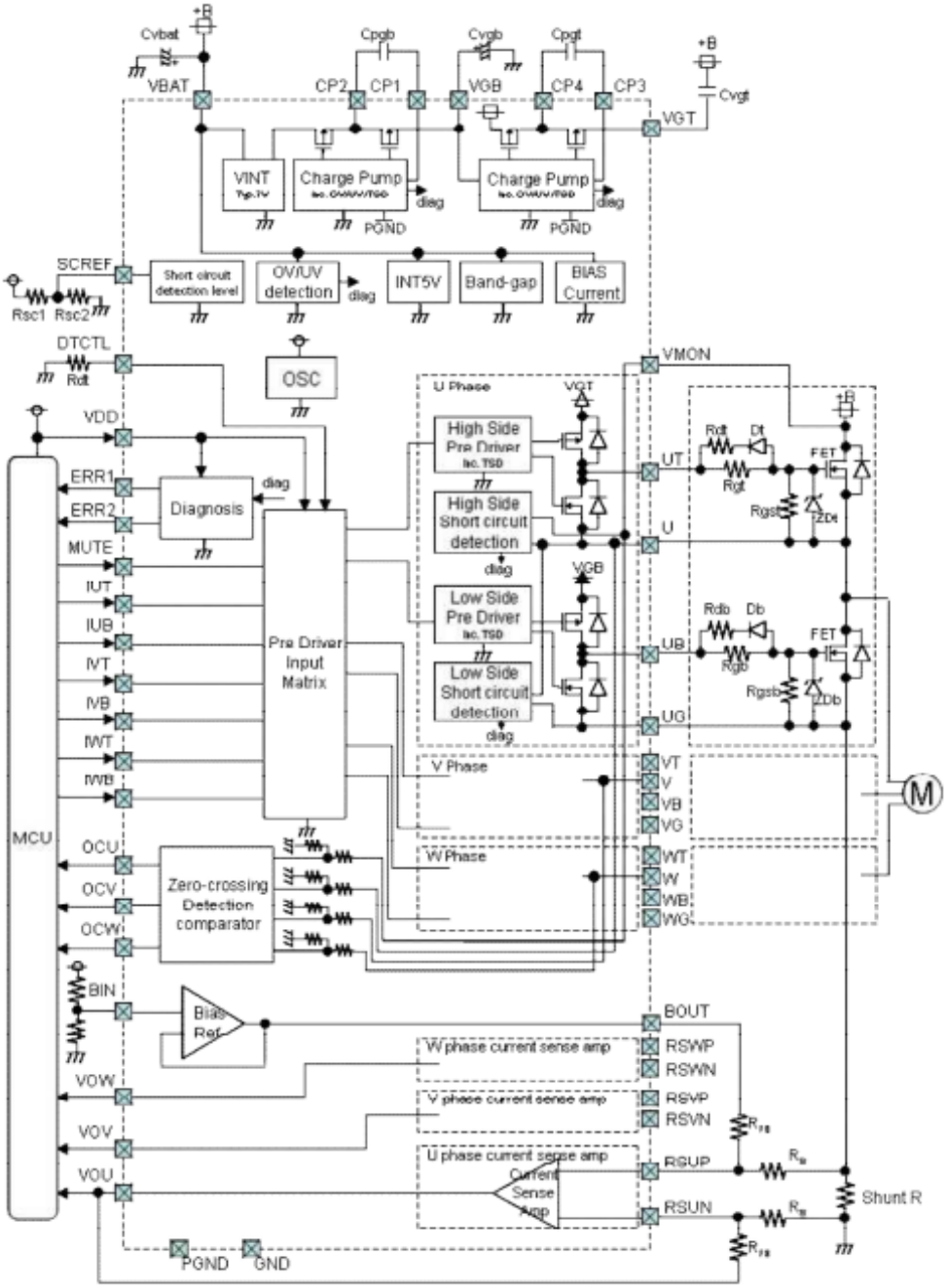


Figure 5. R2A25108KFP Application Example

4. Current Reading Timing in Three Shunt Configuration

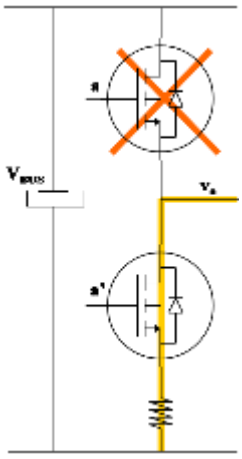


Figure 6. Three Shunt Current Reading

The figure aside shows the situation related to the three-shunt configuration.

Three shunts configuration (J3 and J4 are open)

In the three-shunt configuration the current in one shunt is equal to the corresponding phase current when the corresponding lower switch is ON.

The most suitable moment to read the current in this configuration is at the trough of the PWM.

By default, the Starter Kits are delivered in the three shunts configuration.

The amplifier circuit shown below is made to manage current up to 14.7A (17A) flowing through the shunts and the output of the amplifier is connected directly to the Microcontroller.

Please find below the equations related to the amplifier circuit that are useful to change the range of currents to be measured through the three shunts.

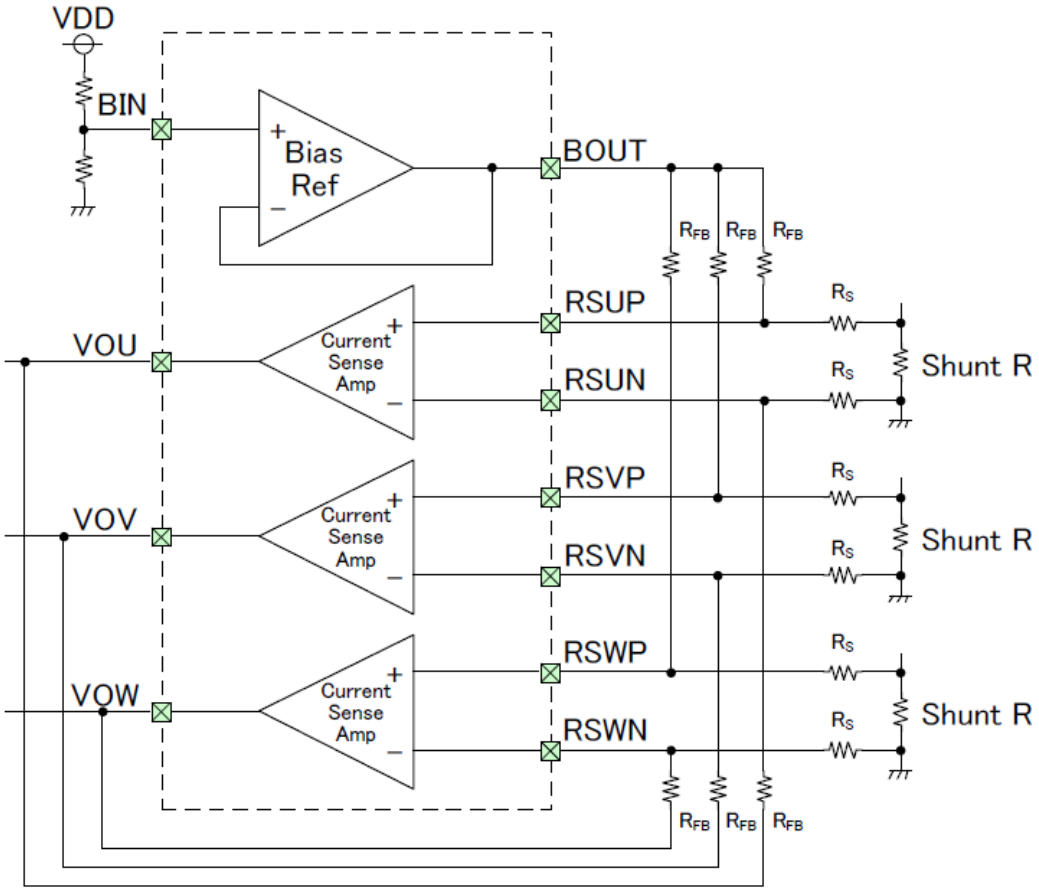


Figure 7. R2A25108KFP for Current Measurement

The amplification of the circuit is $R_{FB}/R_S = 15$.

So, the output voltage (V_{out} to MCU) is:

$$V_{out} = 2.5V - V_{shunt} \times R_{FB}/R_S, \text{ and } V_{shunt} = R_{shunt} \times I_{shunt}$$

So

$$V_{out} = 2.5V - (R_{shunt} \times I_{shunt}) \times R_{FB}/R_S$$

And the term

$$Delta\_V = (R_{shunt} \times I_{shunt}) \times R_{FB}/R_S$$

It is mandatory to keep between -2.5V and 2.5V with some margin (margin of 0.3V could be enough) so:

$$-2.2V < Delta\_V < 2.2V$$

So, the maximum current will be:

$$I_{shunt\ max\ dc} = 2.2V / (R_{shunt} \times (R_{FB}/R_S)$$

The default values are the following:

$R_{shunt} = 0.01\Omega$, $R_{FB} = 15K$, $R_S = 1K$, so $I_{shunt\ max\ dc}$ = **14.7Adc**

If there is no margin, then $I_{shunt\ max\ dc} = 2.5V / (R_{shunt} \times (R_{FB}/R_S)$ =**17Adc**

```
/* hardware settings */
#ifndef RSHUNT_OHM
        #define RSHUNT_OHM      ( 0.01 )        // shunt resistors [Ohm]
#endif  // ifndef RSHUNT_OHM
#ifndef RSGAIN
        #define RSGAIN          ( 15 )          // circuit gain
#endif  // ifndef RSGAIN
```

To manage different values, it is enough to change Rshunt, R_{FB} and R_S.

The shunt resistor value needs to be updated in the software itself in the file called: “const_def.h” as shown below:

5. Permanent Magnets Brushless Motor Model and Field Oriented Control

The basic concept of the FOC is to control the sinusoidal currents of the motor in a coordinate system, which rotates with the same frequency as the rotating magnetic field generated by the alternating currents through the stator windings. As the rotor always tends to be aligned with this rotating field, the motor currents look from the perspective of the rotor like DC currents and therefore can be easily controlled in this rotating coordinate system. So, in the FOC the AC-values imposed to the stator windings are transformed to DC-Values, which are controlled and then transformed back to AC-Values. To understand this, the internal motor relations are described first.

The synchronous permanent magnets motor (sinusoidal brushless motor) is widely used in the industry. More and more home appliance makers are now using such brushless motor, mainly because of the intrinsic motor efficiency.

The permanent magnet motor is made with few components:

- A ***stator*** formed by stacking sheared metal plates where internally the copper wiring is wound, constructing the stator winding
- A ***rotor*** in which permanent magnets are fixed
- Two covers with ball bearings that keep together the stator and the rotor; the rotor is free to rotate inside the stator

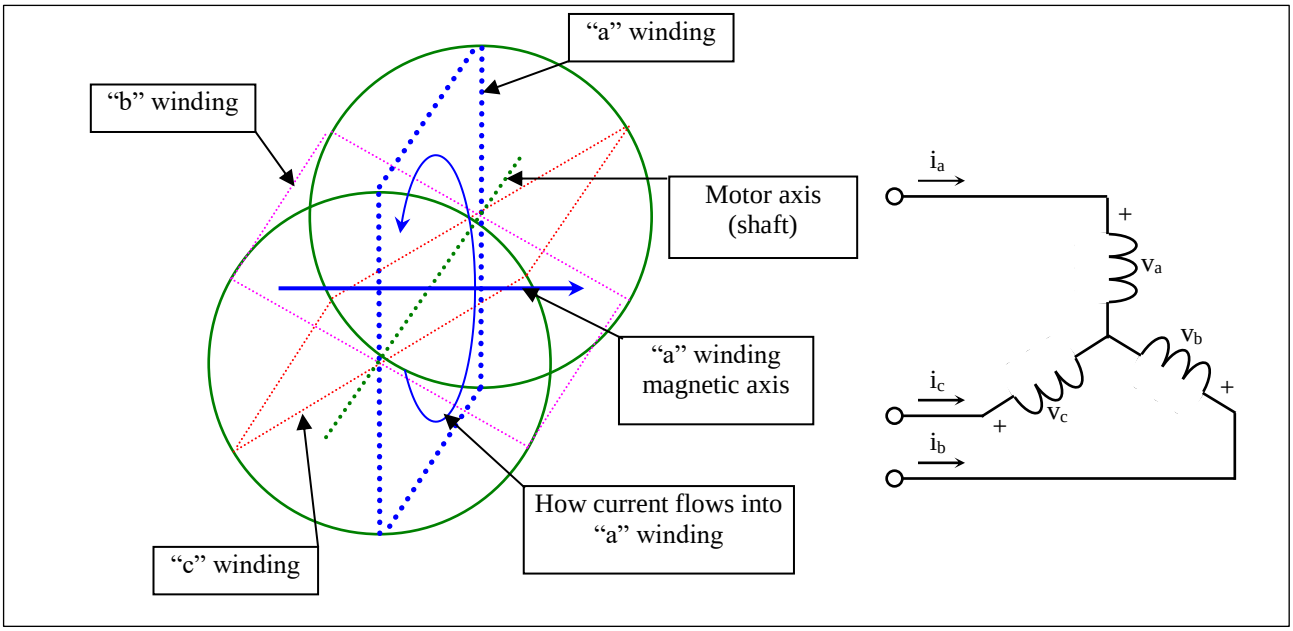


Figure 8. Stator Windings of BLDC Model

The working principle is quite simple: if we supply the motor with a three-phase system of sinusoidal voltages at constant frequency, in the stator windings flow sinusoidal currents, which create a rotating magnetic field.

The permanent magnets in the rotor tend to stay aligned with the rotating magnetic field, so the rotor rotates at synchronous speed.

The main challenge in driving this type of motor is to know the rotor position in real-time, so most implementations are using a position sensor or a speed sensor.

In our implementation, the system is using **three shunts** to detect the rotor position in real-time.

Let's analyze the motor from a mathematic point of view.

If we apply three voltages $v_a(t)$, $v_b(t)$, $v_c(t)$ to the stator windings, the relations between phase voltages and currents are:

$$v_a = R_S i_a + \frac{d\lambda_a}{dt}$$

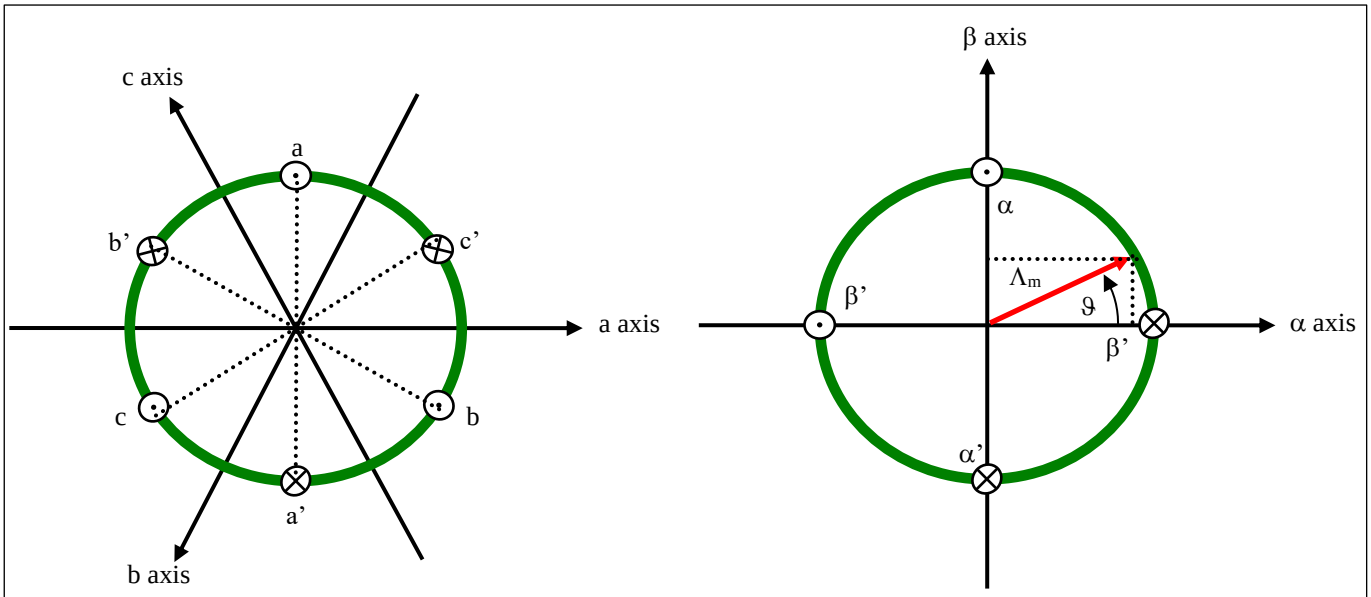
$$v_b = R_S i_b + \frac{d\lambda_b}{dt}$$

$$v_c = R_S i_c + \frac{d\lambda_c}{dt}$$

- λ_i is the magnetic flux linkage with the i-th stator winding

- R_S is the stator phase resistance (the resistance of one of the stator windings)

The magnetic flux linkages λ_i are composed by two items, one due to the stator currents, one to the permanent magnets.



Real axes (a, b, c) and equivalent ones (α, β); a fixed amplitude vector can be completely determined by its position respect the (α, β) system (angle ϑ)

Figure 9. Reference System Transformation from ab to dq axis

The permanent magnet creates a magnetic field that is constant in amplitude and fixed in position in respect to the rotor. This magnetic field can be represented by vector Λ_m whose position in respect to the stator is determined by the angle ϑ between the vector direction and the stator reference frame.

The contribution of the permanent magnets in the flux linkages depends on the relative position of the rotor and the stator, represented by the mechanical-electric angleϑ.

It is, in every axis, the projection of the constant flux vector Λ_m in the direction of the axis:

$$\lambda_a = Li_a + \Lambda_m \cos(\vartheta)$$

$$\lambda_b = Li_b + \Lambda_m \cos(\vartheta - 2\pi/3)$$

$$\lambda_c = Li_c + \Lambda_m \cos(\vartheta - 4\pi/3)$$

Supposing that the rotor is rotating at constant speed ω (that is: ϑ(t) = ωt) the flux linkages derivatives can be calculated, and we obtain:

$$v_a = R_S i_a + L\frac{di_a}{dt} - \omega\Lambda_m \sin(\vartheta)$$

$$v_b = R_S i_b + L\frac{di_b}{dt} - \omega\Lambda_m \sin(\vartheta - 2\pi/3)$$

$$v_c = R_S i_b + L\frac{di_b}{dt} - \omega\Lambda_m \sin(\vartheta - 4\pi/3)$$

A “three phase system” may be represented by an equivalent “two phase system”. So, by using specific transformations, our three equations system is equivalent to a two equations system. It is basically a mathematical representation in a new reference coordinates system.

In the two phases (α,β) fixed system the above equations become:

$$v_\alpha = R_S i_\alpha + \frac{d\lambda_\alpha}{dt}$$

$$v_\beta = R_S i_\beta + \frac{d\lambda_\beta}{dt}$$

For the magnetic field equations, we got:

$$\lambda_\alpha = Li_\alpha + \lambda_{\alpha m} = Li_\alpha + \Lambda_m \cos(\vartheta)$$

$$\lambda_\beta = Li_\beta + \lambda_{\beta m} = Li_\beta + \Lambda_m \sin(\vartheta)$$

After performing the derivation:

$$\frac{d\lambda_\alpha}{dt} = L\frac{di_\alpha}{dt} - \omega\Lambda_m \sin(\vartheta) = L\frac{di_\alpha}{dt} - \omega\lambda_{\beta m}$$

$$\frac{d\lambda_\beta}{dt} = L\frac{di_\beta}{dt} + \omega\Lambda_m \cos(\vartheta) = L\frac{di_\beta}{dt} + \omega\lambda_{\alpha m}$$

Finally, we obtain for the voltages in (α,β) system:

$$v_\alpha = R_S i_\alpha + L\frac{di_\alpha}{dt} - \omega\lambda_{\beta m}$$

$$v_\beta = R_S i_\beta + L\frac{di_\beta}{dt} + \omega\lambda_{\alpha m}$$

A second reference frame is used to represent the equations as the frame is turning at the rotor speed. So, the “d” axis is chosen in the direction of the magnetic vector Λ_m, and with the “q” axis orthogonal to the “d” axis. The new reference system is (d, q).

The reference frame transformations from the (α,β) system to the (d, q) system depends on the instantaneous position angle ϑ

We can see

$$v_d = R_S i_d + L\frac{di_d}{dt} - \omega\lambda_{qm} \qquad \text{and} \qquad \lambda_{dm} = Li_d + \Lambda_m$$

$$v_q = R_S i_q + L\frac{di_q}{dt} + \omega\lambda_{dm} \qquad \lambda_{qm} = Li_q$$

So, we obtain two inter-dependent equations in the (d, q) system:

$$v_d = R_S i_d + L\frac{di_d}{dt} - \omega L i_q$$

$$v_q = R_S i_q + L\frac{di_q}{dt} + \omega L i_d + \omega\Lambda_m$$

These two equations represent the mathematical motor model as shown in the following figure.

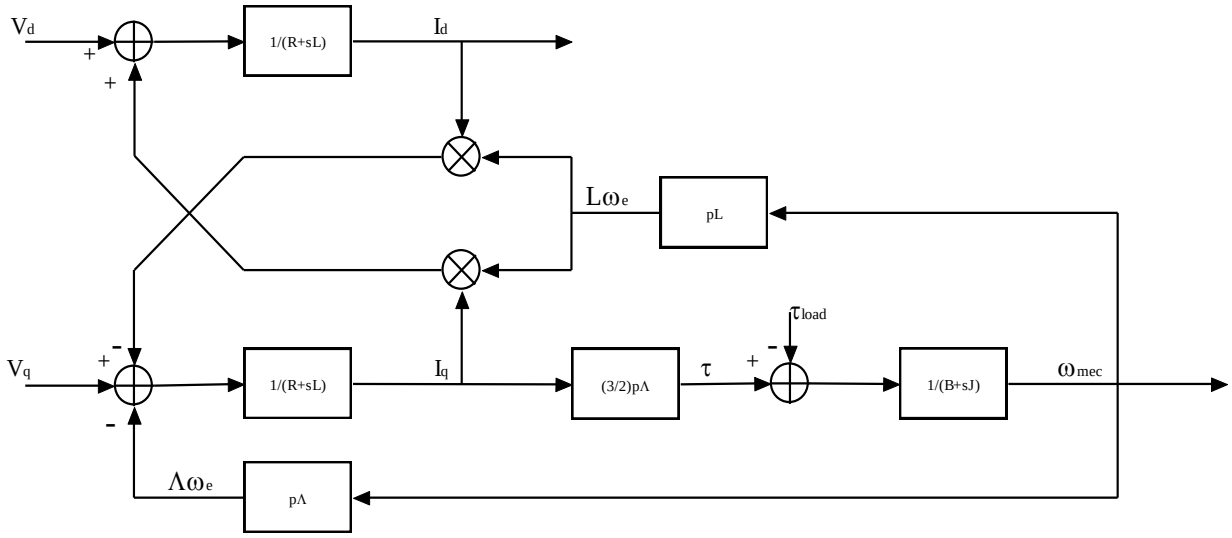


Figure 10. The mathematical motor model

A control algorithm which wants to produce determined currents in the (d, q) system must impose voltages given from the formulas above.

This is ensured by closed loop PI control on both axis “d” & “q” (Proportional Integral).

Since there is a mutual influence between the two axes, decoupling terms can be used.

In the block scheme the mechanic part is included, where “p” is the number of pole pairs, while “B” represents friction, “J” the inertia, “τ_{load}“ the load torque and “τ” the motor torque, “Λ” is “Λ_m”:

$$\tau = \frac{3}{2} \times p \times \Lambda \times i_q$$

Thus, the torque can be controlled directly by the current i_q only. This approach is similar to the DC motor equation in which the torque is proportional to the winding current.

The angular speed ω is represented in the scheme as ω_e to distinguish the electrical speed from the mechanical one ω_{mec}.

Let's now consider the equations we have seen in (α,β) system:

$$v_\alpha = R_S i_\alpha + \frac{d\lambda_\alpha}{dt}$$

$$v_\beta = R_S i_\beta + \frac{d\lambda_\beta}{dt}$$

These equations show that magnetic flux can be obtained from applied voltages and measured currents simply by integration:

$$\lambda_\alpha = \lambda_{\alpha 0} + \int_0^t (v_\alpha - R_S i_\alpha) dt$$

$$\lambda_\beta = \lambda_{\beta 0} + \int_0^t (v_\beta - R_S i_\beta) d$$

Furthermore:

$$\Lambda_m \cos(\vartheta) = \lambda_\alpha - L i_\alpha$$

$$\Lambda_m \sin(\vartheta) = \lambda_\beta - L i_\beta$$

If the synchronous inductance L is small, the current terms can be neglected, if not they have to be considered. In general:

$$x = \Lambda_m \cos(\vartheta) = \lambda_\alpha - L i_\alpha = \lambda_{\alpha 0} + \int_0^t (v_\alpha - R_S i_\alpha) dt - L i_\alpha$$

$$y = \Lambda_m \sin(\vartheta) = \lambda_\beta - L i_\beta = \lambda_{\beta 0} + \int_0^t (v_\beta - R_S i_\beta) dt - L i_\beta$$

So, in the (α,β) system phase we obtain from the flux components:

$$\vartheta = \arctan(y/x)$$

The system speed ω can be obtained as the derivative of the angle ϑ.

$$\omega = \frac{d}{dt}\vartheta(t)$$

Based on this, a sensorless control algorithm was developed to give the imposed phase voltages, to measure phase currents, to estimate the angular position ϑ and finally the system speed.

Please, find below the sensorless vector control algorithm block diagram.

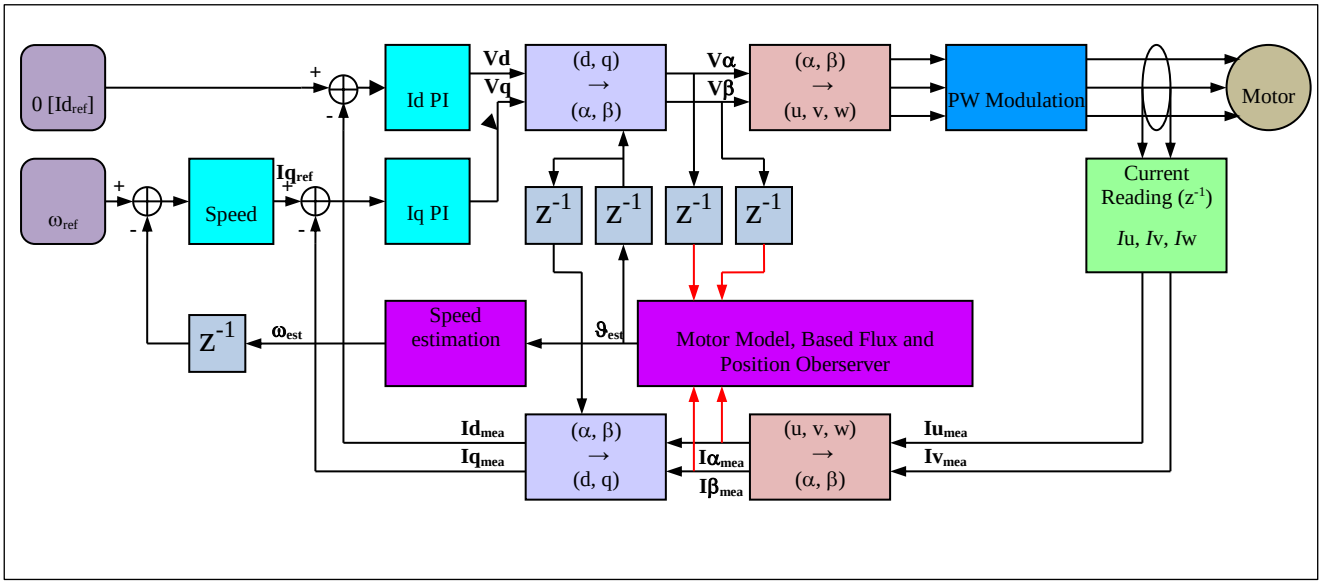


Figure 11. FOC Algorithm Block Diagramd

6. Software Description and Resources Used

The software delivered in the starter kits, previously described, is working on the RH850 microcontroller clocked at 120MHz and its operating voltage is 5V which guarantee a high noise immunity.

Using the interrupt skipping function, it is possible to regulate separately the PWM frequency (Pulse Width Modulation) and the sampling frequency (also called control loop frequency). For instance, if the PWM frequency is set to 20KHz and the control loop is set to 10KHz, the ratio is 2 which means that the full vector control algorithm is processed every two PWM cycles.

Finally, the main interrupt is called every 100µs which leaves enough time to perform the sensorless vector control algorithm and the system control if needed.

Please find below detailed information related to the software blocks of the motor control embedded software:

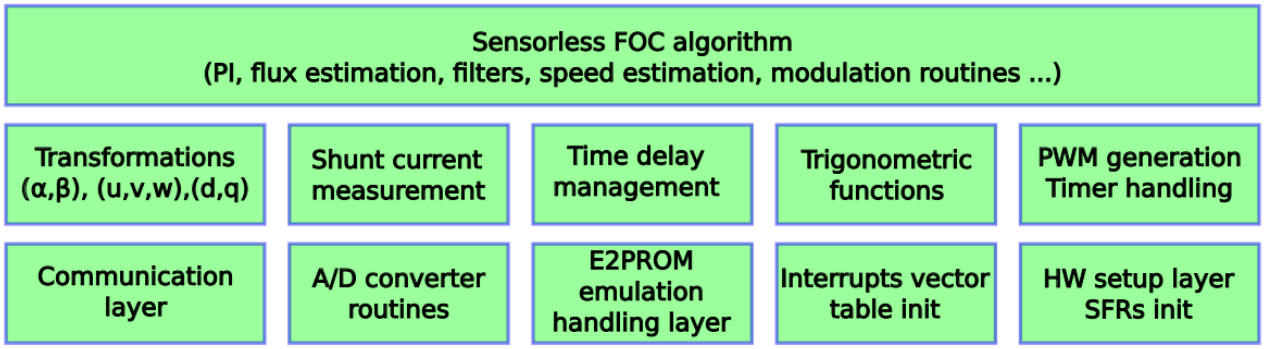


Figure 12. Software blocks of FO motor control embedded software

The complete software uses the resources below in the three shunts configuration. It includes the serial communication interface, the board management and of course the complete sensorless vector control algorithm.

F1KM-S1 FLASH memory usage: ~58KB and RAM memory usage: ~6.5kB

CPU-Load

The control loop of the field oriented control algorithm (e.g. sampling frequency, “SAM_FRE_CUSTOM_HZ”) is set to 10KHz by default. The Pulse Width Modulation (PWM, “PWM_FRE_CUSTOM_HZ”) frequency is set by default to 20KHz. The parameters are visible in the module name “customize.h”.

Note: By default, the macro “STAND_ALONE” in “const_def.h” is defined in F1KM-S1 Starter kit. As the information for the display need to be prepared and the encoder and potentiometer management needs CPU time, the load with STAND_ALONE defined takes more time than without.

F1KM-S1 Main Loop: ~ 12us (35us, every 4th because of display update) Control interrupt: ~ 20us

As the sampling period is 100µs (10KHz) and the control loop for the inverter control takes maximum of 20µs, the CPU load of the motor control algorithm is: 20/100 = 20%. Additionally, every 10 ms the Main Loop is performed, which takes about 4 us: 0.004/10 = 0.04% CPU Load. So, the CPU Load in regularly operation is ~ 20%.

It leaves ~ 80% of the CPU to perform additional tasks, like board management, system control, display, etc.

The following flowcharts show the software implementation of the motor control part of the software.

Please find below the flowchart for the main loop and the interrupt service routine.

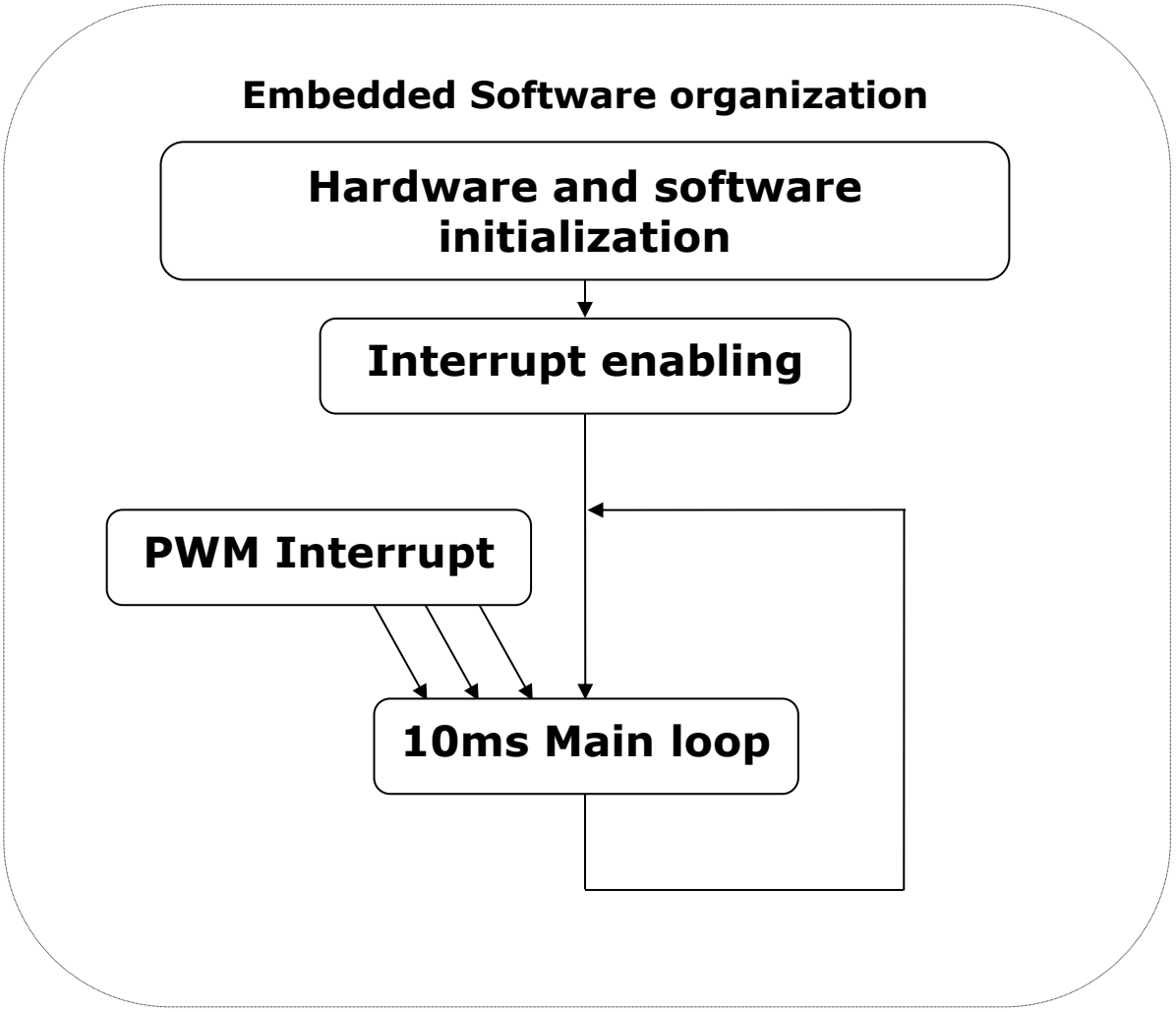


Figure 14. Flow chart for the main function

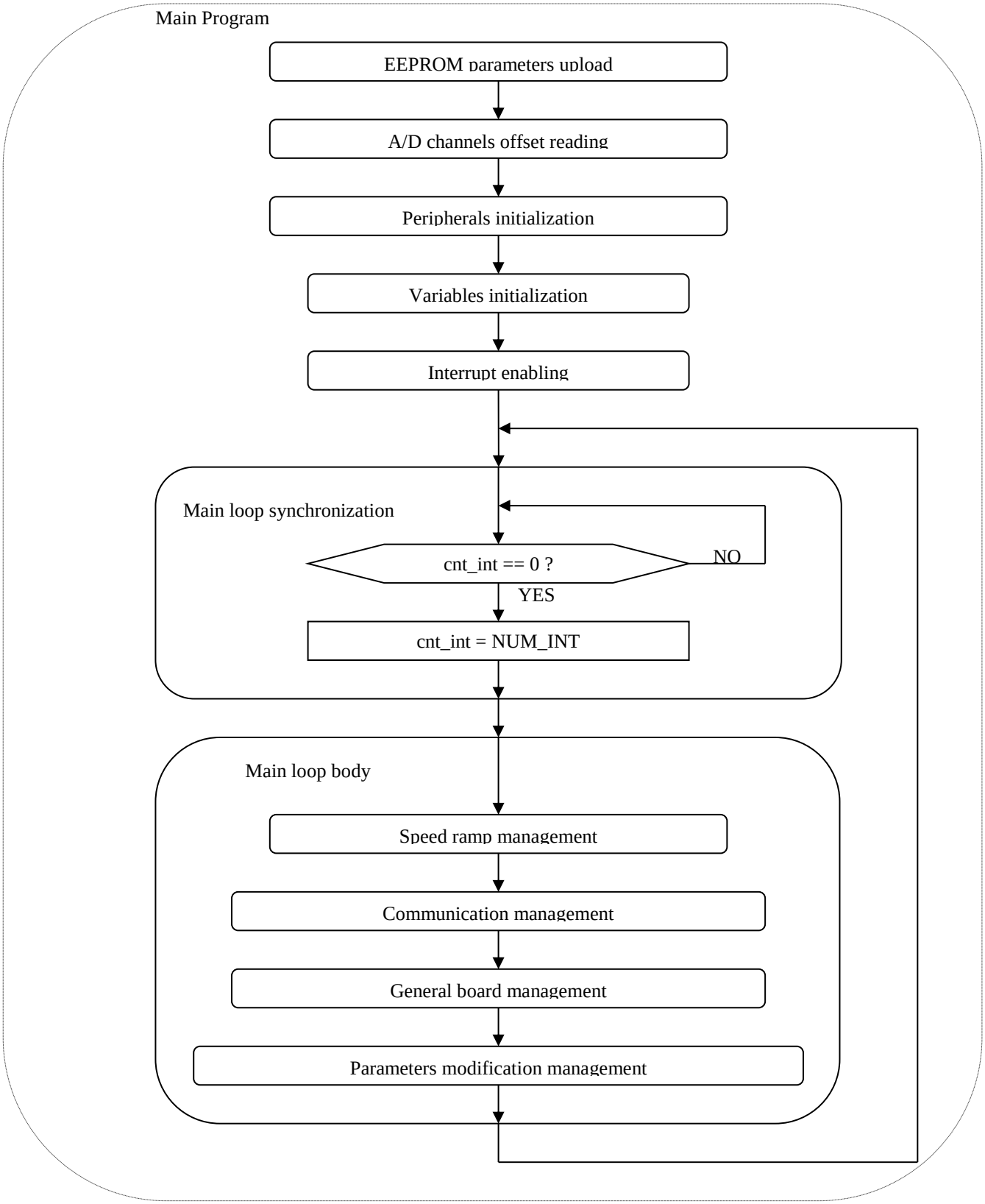


Figure 15. Flow chart for the main loop in detail

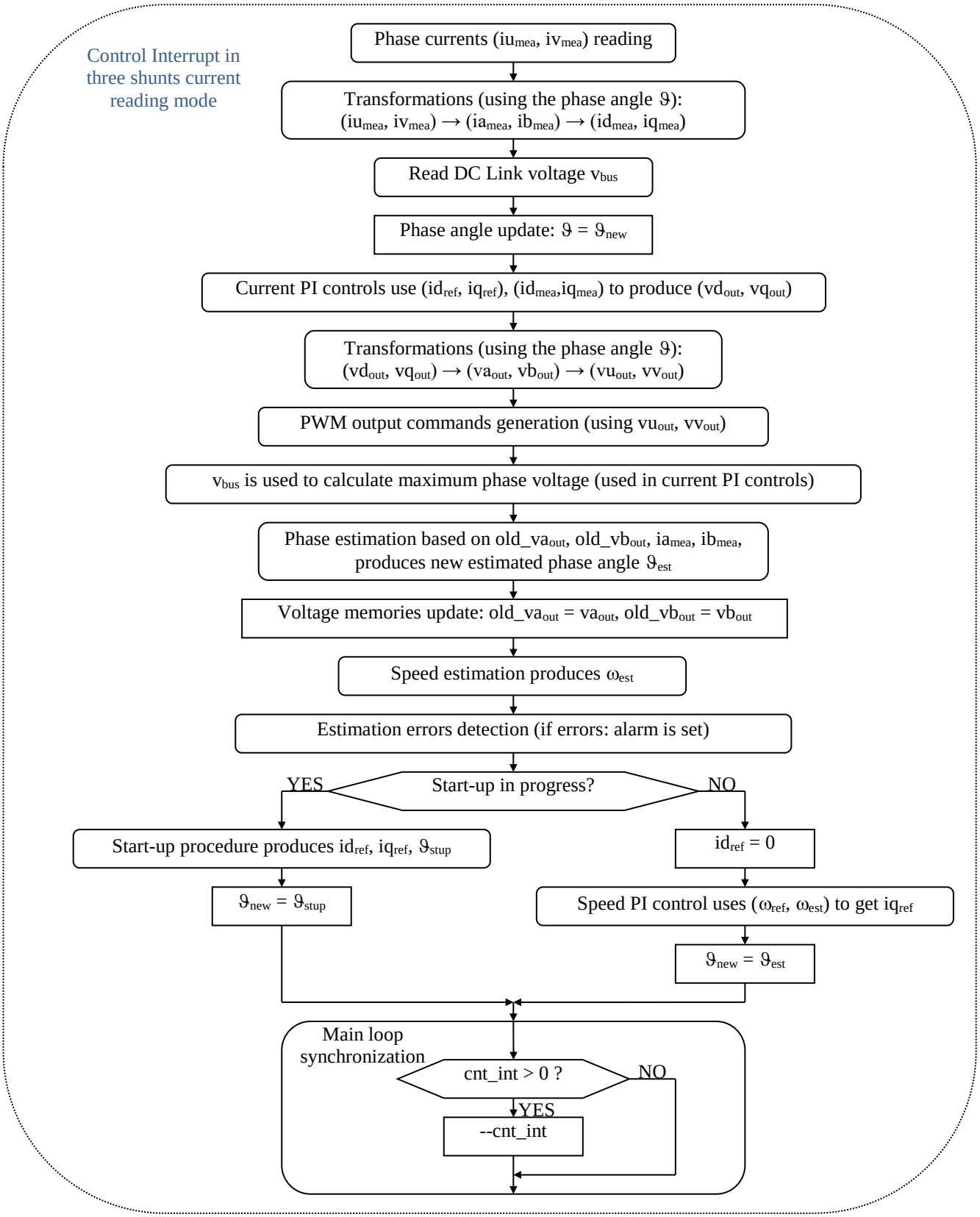


Figure 16. Control interrupt in three shunt current reading model

The whole motor control software is splitted into different source files which are in the module called: motorcontrol, located in the source folder and are described below:

Table 2 C-modules of project source code

Source Files	Functions Descriptions
motorcontrol.c	Contains the FOC algorithm, current reading management, PW-Modulation, parameter management and estimators routines
ini.c	Basic peripheral initialization
main.c	The main program loop
userif.c	Communication routines (for the GUI)
par_tab.c	The Parameter management definitions and tables
globvar.c	Global variable definitions
mcrplib.c	Math functions (sin, cos, Park, Clarke …)

Please find below the header files included into the project folder.

Table 3. Header files of project source code

Header Files	Functions Descriptions
customize.h	Basic motor parameters (Not modifiable through the GUI)
const_def.h	Definition of the basic numerical constants
par_tab.h	Parameter definitions, function prototypes and references
ini.h	function prototypes
globvar.h	Global variable definitions and references
mask.h	General support definitions and references
userif.h	General support definitions, references and function prototypes
mcrplib.h	Header file for the Math Library block
motorcontrol.h	Header file for complete FOC algorithm in interrupt service routines

The following table shows the assembler modules in the project.

Table 4. Assembler modules of project source code

Assembler Modules	Functions Descriptions
dr7f701684_startup.850*	startup file

7. Start-up Procedure – Embedded Software

When the motor is in stand-still, the phase of the permanent magnet flux vector cannot be detected with the used algorithm. So, an appropriate start-up procedure must be applied.

The idea is to move the motor in feed-forward (with higher current than that required to win the load), till a speed at which the estimation algorithm can work. Then the system can be aligned to the estimated phase, and the current can be reduced to the strictly necessary quantity.

The following graph illustrates the strategy used (the suffix "$_{ref}$" stands for *reference*, the suffix "$_{mea}$" stands for *measured*).

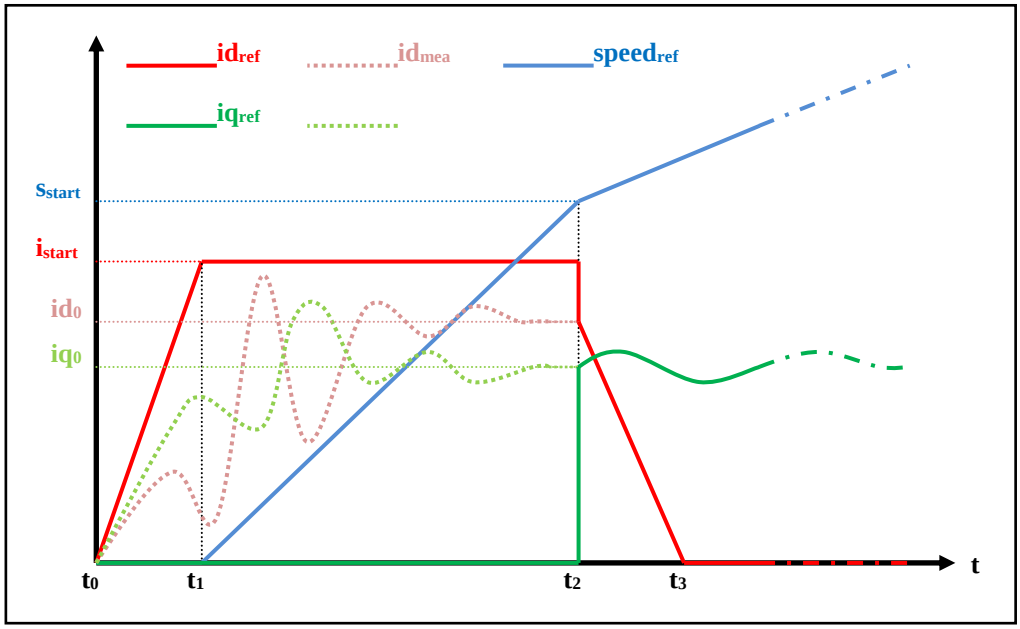


Figure 17. Startup procedure in embedded software

Referring to the graph above, the start-up procedure (in case of three shunts current reading) is described below.

a) At the beginning t_0, the system phase is unknown. No current is imposed to the motor; the system phase is arbitrarily decided to be ϑ_a=0. All the references: id_{ref}, iq_{ref} and $speed_{ref}$ are set to zero.

b) From the moment t_0, while the iq_{ref} and the $speed_{ref}$ are maintained to zero, id_{ref} is increased with a ramp till the value i_{start} is reached at the moment t_1.

The references are referred to an arbitrary (d_a, q_a) system based on the arbitrary phase ϑ_a. From this moment, the phase estimation algorithm begins to be performed, and the estimated phase ϑ_{est} is used to calculate the components of the measured current, referred to the (d, q) system based on the estimated phase, id_{mea} and iq_{mea}. The components of the current referred to the arbitrary (d_a, q_a) system are controlled to follow the references by the current PI controllers. On the other hand, since the phase ϑ_{est} is still not correctly estimated, id_{mea} and iq_{mea} have no physical meaning. Even if they are not shown in the graph, the applied voltages are subjected to the same treatment (vd_{mea} and vq_{mea} are calculated in the algorithm).

c) At t = t_1, while iq_{ref} is maintained to zero and id_{ref} is maintained to its value i_{start}, $speed_{ref}$ is increased with a ramp till the value s_{start} is reached at t = t_2. The system phase $\vartheta_a(t)$ is obtained simply by integration of $speed_{ref}$; in the meanwhile, the phase estimation algorithm begins to align with the real system phase. Furthermore, id_{mea} and iq_{mea} begin to be similar to the real flux and torque components of the current. The real components are supposed to be id_0 and iq_0 (those values are obtained applying a low-pass filter to id_{mea} and iq_{mea}).

The interval (t_2-t_1) is the start-up time, and it is supposed to be large enough to allow the estimation algorithm to reach the complete alignment with the real phase of the system.

d) At $t = t_2$, the phase estimation process is supposed to be aligned. At this point a reference system change is performed: from the arbitrary (d_a, q_a) reference to the (d, q) reference based on the estimated phase ϑ_{est}.

The current references are changed to the values id_0 and iq_0, and all the PI controllers are initialized with these new values. The speed PI integral memory is initialized with the value iq_0, while the current PI integral memories are initialized with the analogous voltage values vd_0 and vq_0, obtained from vd_{mea} and vq_{mea}.

e) After $t > t_2$, the normal control is performed, based on the estimated phase ϑ_{est}; the speed reference is increased with the classical ramp; the id current reference is decreased with a ramp, till it reaches the value zero at the moment t_3; then it is maintained to zero; the iq current reference is obtained as output of the speed PI controller.

8. Reference System Transformations in Details

Find below the detailed equations used for the coordinates transformations in the embedded software for the RH850/F1KM-S1 microcontroller.

$$g_\alpha = \frac{2}{3}(g_u - \frac{1}{2}g_v - \frac{1}{2}g_w) = g_a$$
$$g_\beta = \frac{2}{3}(\frac{\sqrt{3}}{2}g_v - \frac{\sqrt{3}}{2}g_w) = \frac{1}{\sqrt{3}}(g_v - g_w) = \frac{1}{\sqrt{3}}(g_u + 2g_v)$$

(u, v, w) → (α, β)

$$g_u = g_\alpha$$
$$g_v = -\frac{1}{2}g_\alpha + \frac{\sqrt{3}}{2}g_\beta = (-g_\alpha + \sqrt{3}g_\beta)/2$$
$$g_w = -\frac{1}{2}g_\alpha - \frac{\sqrt{3}}{2}g_\beta = (-g_\alpha - \sqrt{3}g_\beta)/2$$

(α, β) → (u, v, w)

$$g_d = g_\alpha \cos(\vartheta) + g_\beta \sin(\vartheta)$$
$$g_q = -g_\alpha \sin(\vartheta) + g_\beta \cos(\vartheta)$$

(α, β) → (d, q)

$$g_\alpha = g_d \cos(\vartheta) - g_q \sin(\vartheta)$$
$$g_\beta = g_d \sin(\vartheta) + g_q \cos(\vartheta)$$

(d, q) → (α, β)

$$\left\{\begin{array}{l} v_u = V\cos(\omega t + \varphi_0) \\ v_v = V\cos(\omega t + \varphi_0 - 2\pi/3) \\ v_w = V\cos(\omega t + \varphi_0 - 4\pi/3) \end{array}\right\} \leftrightarrow \left\{\begin{array}{l} v_\alpha = V\cos(\omega t + \varphi_0) \\ v_\beta = V\sin(\omega t + \varphi_0) \end{array}\right\} \leftrightarrow \left\{\begin{array}{l} v_d = V\cos(\varphi_0) \\ v_q = V\sin(\varphi_0) \end{array}\right\}$$

9. Rotor Position Estimation

The rotor position estimation method which has been chosen is the direct integration of the back EMF. Such method is enabled by default in the starter kits.

Please find below the fundamental equations:

$$x = \Lambda_m \cos(\vartheta) = \lambda_\alpha - Li_\alpha = \lambda_{\alpha 0} + \int_0^t (v_\alpha - R_S i_\alpha)dt - Li_\alpha$$

$$y = \Lambda_m \sin(\vartheta) = \lambda_\beta - Li_\beta = \lambda_{\beta 0} + \int_0^t (v_\beta - R_S i_\beta)dt - Li_\beta$$

$$\vartheta = \arctan(x/y)$$

$$\omega = \frac{d}{dt}\vartheta(t)$$

The challenges in this approach are the calculation of the integrals which is well known as a problematic issue in a numeric context, and the choice of the initial conditions, which are not known in general. There are two possibilities to overcome these difficulties:

1. To use a so-called "**approximated integration**", which means that instead of using an integral (1/s), a special transfer function is chosen, which is very similar to the integral in certain conditions.
2. To correct the result of the integration with a sort of feedback signal, obtained combining the estimated phase with the real flux amplitude, known as a parameter of the system.

In the **1st case**, we choose an integral approximation function which has a limited memory of the errors and with a zero DC gain. The goal is to reject any low frequency component, preventing the result to diverge, and automatically forgetting the errors (noise, etc.). This is obtained by combining a low-pass filter with a second low-pass filter, as in the following schemes in **Error! Reference source not found.** and **Error! Reference source not found.**:

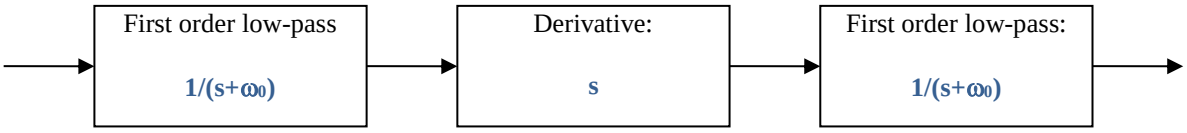


Figure 18. Filter Diagram of approximated Integration in Rotor Position Estimation

$$Y(s) = \frac{s}{(s+\omega_0)^2}$$

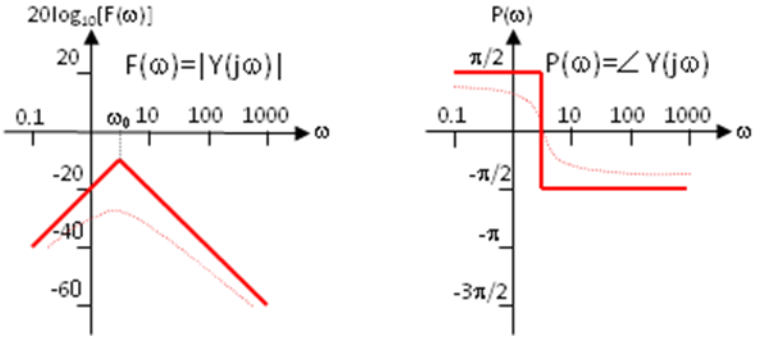


Figure 19. Corresponding S-domain Functions for Filters in Figure 18

It is evident the relationship between Y(s) and the integral I(s)=1/s for $s=j\omega$, when $\omega >> \omega_0$.

In the **2nd case**, to prevent the integral to diverge, and the errors related to wrong initial conditions are rejected, by the correcting action of the feedback.

The block scheme of the exact BEMF integration method for flux position estimation is the following:

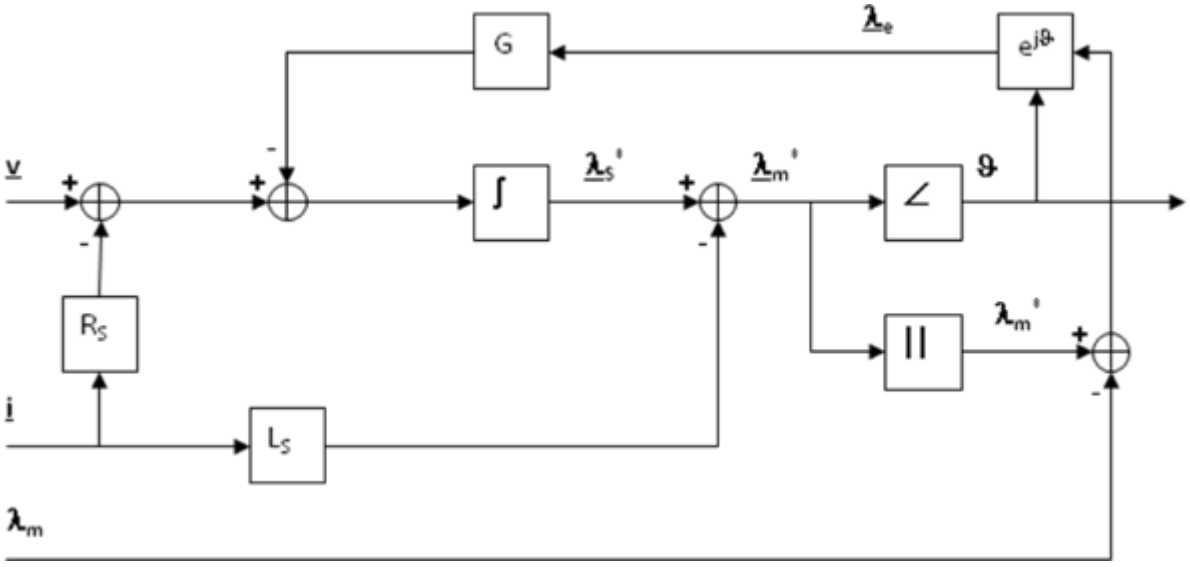


Figure 20. Block Scheme of the Exact BEMF Integration Method for Flux Position Estimation

The inputs of the system are the imposed voltage vector v and the measured current vector i. The motor phase resistance R_s, the synchronous inductance L_s and the permanent magnet flux amplitude λ_m are known as parameters and motor dependent.

The integral operation is corrected with a signal obtained modulating accordingly with the estimated phase the error between the estimated flux amplitude and the amplitude of the permanent magnets flux.

The gain of this correction is indicated with G. It is this feedback which avoids the integral divergence due to the errors or offsets. The higher G is, the higher is the relationship between the estimated amplitude and the theoretical one, but the larger can be the induced phase error.

The choice of G is a trade-off, in order to guarantee that the integral remains close to its theoretical value, but free enough to estimate the correct system phase.

In the **default embedded software** delivered on the starter kits, this **first** strategy is selected.

10. PC Graphical User Interface in Detail

Please install the Motor Control PC GUI on your machine by following the instructions of the Quick Start Guide delivered in the starter kits. After connecting the Fulling motor (FL28BL26-15V-8006AF, 15V_{DC}, 8000RPM), please connect the board RH850/F1KM-S1 and select the COM port.

The PC Graphical User Interface supports the following Operating Environments:

- Windows® 10 (32-bit, 64-bit)
- Windows® 8.1 (32-bit, 64-bit)
- Windows® 8 (32-bit, 64-bit)
- Windows® 7 (32-bit, 64-bit)

Please find below the detailed descriptions of the PC GUI tabs and windows.

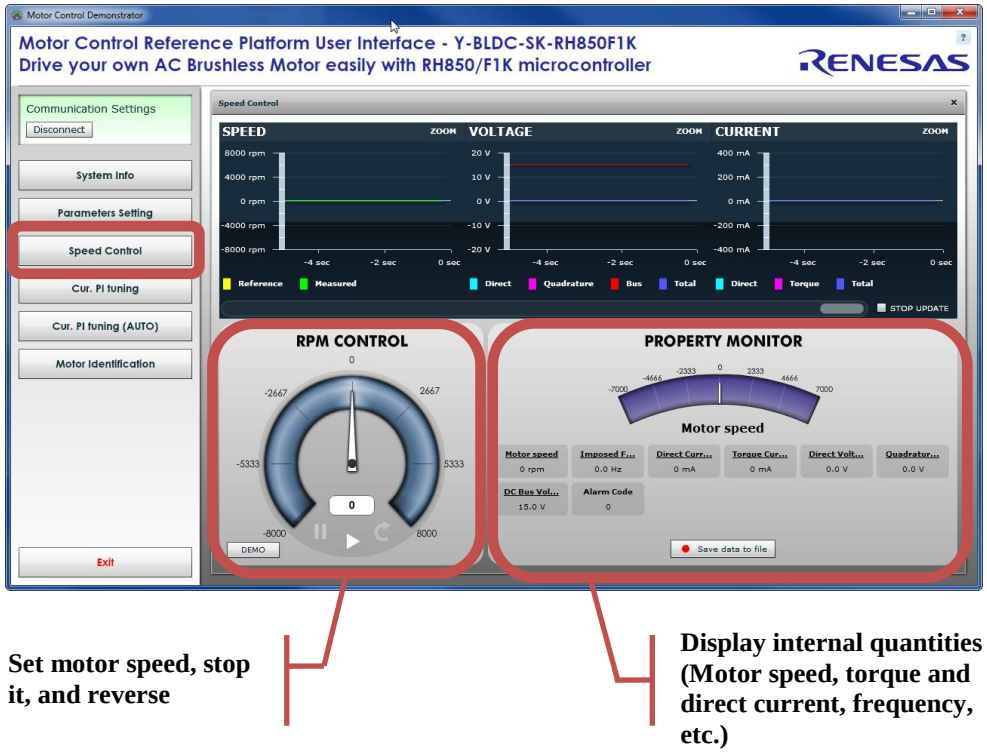


Figure 21. Detail descriptions of PC GUI

By clicking on the button “Save data to file”, it is possible to record regularly all the values displayed in real-time in a file, as described in figure below:

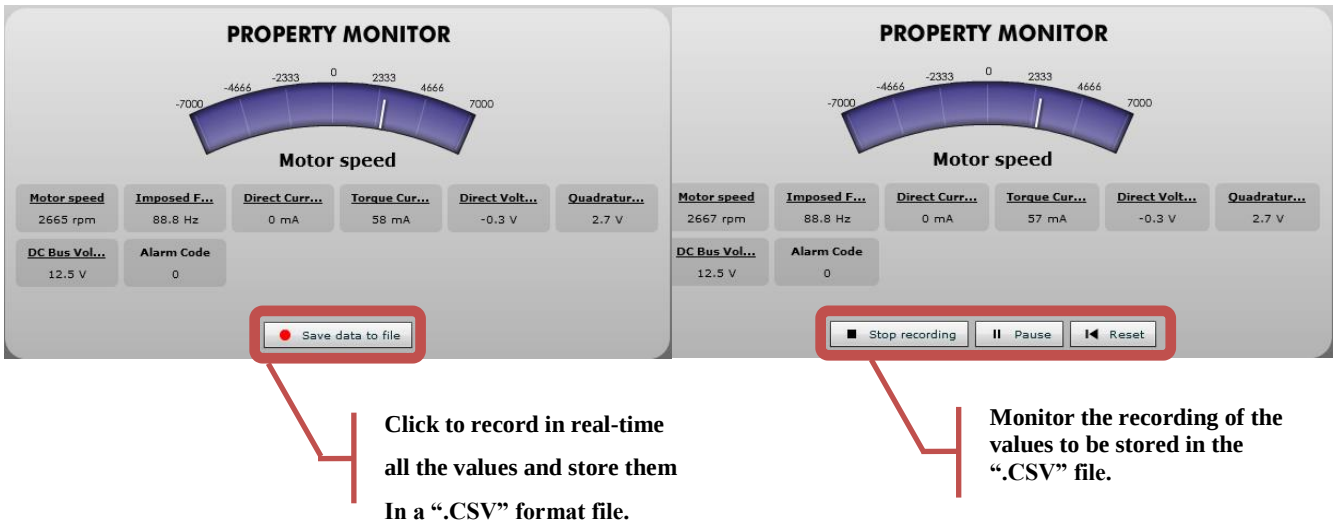


Figure 22. Save Data to File in Real-Time

Furthermore, the Speed control window displays the Alarm codes status of the board itself:

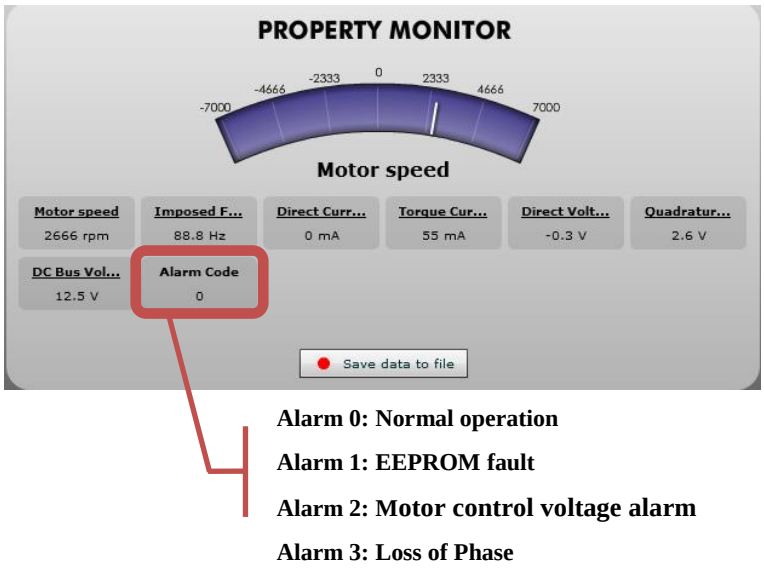


Figure 23. Alarm Codes Status of the Board

Alarm code 1:

The alarm 1 is called “EEPROM alarm” and described in the software by “EQP_ALL”. This alarm is set when one or more EEPROM parameters are higher than the maximum allowed value or lower than the minimum allowed value.

The maximum and minimum values are specified in the two constants tables called: "par_max[]" "par_min[]" in the "par_tab.c" file. Another root cause for the alarm 1 is the EEPROM hardware failure when the error is accessed in read or write mode.

When this alarm is active, the access to the EEPROM is restricted. To reset the alarm the default parameters set should be reloaded in the EEPROM. By using the PC GUI and the parameters setting window, it becomes possible to clean the EEPROM content. *The first step is to write the magic number "33" in the first parameter n°00. The second step is to reset the board by pressing the "reset" button on the PCB, then click the "Reload" button in the parameters setting window.*

At this point a coherent set of parameters is loaded and the alarm should disappear.

Finally, if the alarm is produced by a hardware failure of the EEPROM itself, then the board needs to be repaired.

Alarm code 2:

The alarm 2 is called "motor control voltage alarm" and described in the software by "MOT_VDD_ALL". This alarm is set when the voltage of the motor control part is below ~4,7V. To guarantee a correct function, the voltage for the predriver must be over 5.3V. If it is under 4.7V, the predriver will not work at all, therefore a use of the motor control unit wouldn't make sense.

This alarm can be reset if you apply a higher voltage to the motor control part. After the voltage is changed, a reset is mandatory.

It is also possible that the Jumper setting is wrong. For example, if you want to power the motor control part via the DC Jack, the Jumper have to be set to <MOT_VDD - 12V>. Otherwise the motor control part isn't power supplied.

Note: Always set jumpers, while the power supply is off. A reset is mandatory if you have changed the voltage.

Alarm code 3:

The alarm 3 is called "loss of phase" and described in the software by "TRIP_ALL". This alarm is produced when the sensorless position detection algorithm is producing inconsistent results. It means that the rotor position is unknown due to a lack of accuracy, so the motor is stopped.

This alarm can be reset by setting the speed reference to zero.

Please find below an extract of the header file "const_def.h".

```
/* alarms */
#define EQP_ALL     ( 1 )          // eeprom alarm code
#define MOT_VDD_ALL ( 2 )          // MOT_VDD too low alarm code
#define TRIP_ALL    ( 3 )          // loss of phase alarm code
```

Then by clicking on the "Parameters Setting" button, the important window can be displayed showing all the parameters of the system that can be changed in real-time without having to recompile the embedded software.

The detailed description of each parameter is displayed when pointing the mouse on *the question mark*, see figure above. Each parameters unit is displayed.

All the parameters can be changed by pushing the "Write" button.

Note: It is recommended to change the parameters in standstill to avoid malfunction.

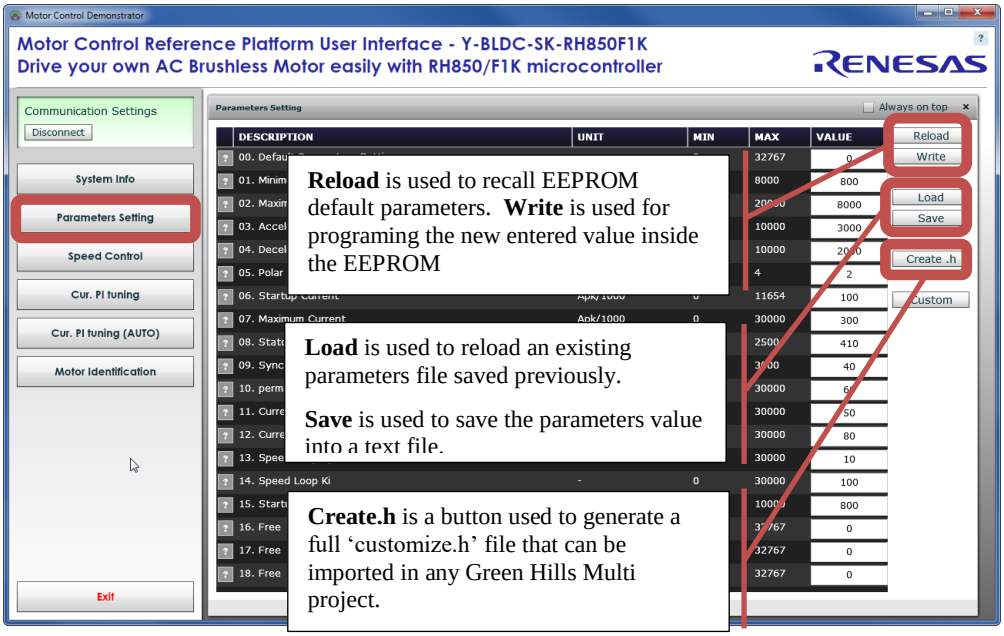


Figure 24. Question Mark Functions in Parameters Window

Speed range limitations

The Y-BLDC-SK-RH850F1K Starter Kit is driving a BLDC using a sensorless vector control algorithm. So it means that there is a **minimum** speed to reach in order to run the motor properly using the three shunts current measurement methods. In the case of the Fulling Motor FL28BL26-15V-8006AF delivered with the kit, the minimum speed is **500RPM**. Below this speed, the current flowing through the three shunts are too low to be detected.

If only 12V is applied to the FL28BL26-**15V**-8006AF brushless motor, it is not able to reach its maximum speed of **8000RPM** (the motor needs a power supply of 15V to reach its maximum speed).

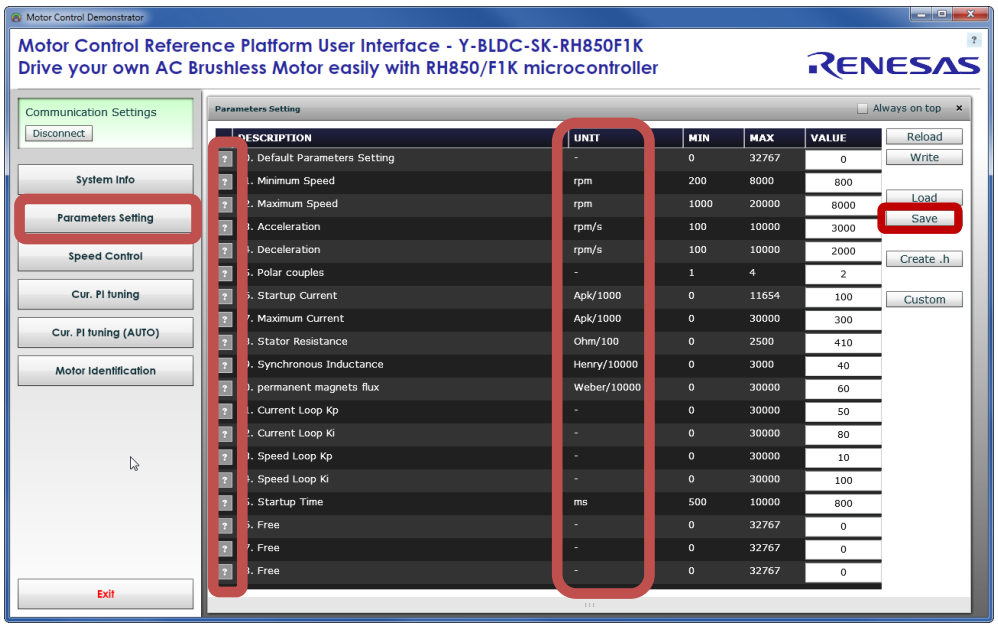


Figure 25. Buttons Function-Descriptions in Parameters Window

11. Software parameters: detailed description

Please find below the software parameters list including their full description. Each parameter located in the "customize.h" header file can be tuned by the user directly by the Graphic User Interface, without re-compiling the program.

Table 5. Full descriptions of software parameters in EEPROM

Parameter number	Short name	Description
0	SEL_OP	default parameters setting, used to perform special operations, like default parameter set re-loading
1	RPM_MIN	Set the Minimum Speed in RPM
2	RPM_MAX	Set the Maximum Speed in RPM
3	R_ACC	Set the acceleration [RPM/s]
4	R_DEC	Set the deceleration [RPM/s]
5	C_POLI	Set the number of polar couples
6	I_START	Set the start-up current (peak) [Ampere/AMP_DIV]. Used to specify the peak phase current value to be used during the start-up
7	I_MAX	Set the maximum phase current (peak) [Ampere/AMP_DIV]
8	R_STA	Set the stator resistance [Ohm/OHM_DIV]
9	L_SYN	Set the synchronous inductance [Henry/HEN_DIV]
10	PM_FLX	Set the permanent magnets flux [Weber/WEB_RES]. This value is only used when the exact integration flux estimation algorithm is selected. By default, it's not needed as the approximated integration is selected.
11	KP_CUR	Set the Current loop Proportional coefficient: KP
12	KI_CUR	Set the Current loop Integral coefficient: KI
13	KP_VEL	Set the Speed loop Proportional coefficient: KP
14	KI_VEL	Set the Speed loop Integral coefficient: KI
15	STP_TIM	Set the Start-up acceleration time [ms]

12. Communication Protocol Between the MCU and the PC GUI

After the introduction of the auto-tuning, a new set of information is exchanged between the GUI and the board. To distinguish between the software versions the answer to the check request is used. In the previous software version, the answer to a check com request ("c"), was the uppercase ("C").

The serial communication speed tested is 9.6 KBd.

****

*** MULTIPOINT MASTER-SLAVE SERIAL COMMUNICATION SIMPLIFIED PROTOCOL ***

****

ASCII: '!'=0x21, '#'=0x23, '?'=0x3F, 'C'=0x43, 'W'=0x57, 'c'=0x63, 'w'=0x77

Master String:

l i s o a n D1 .. Dm k

l = frame total length (1 byte)

i = master string identification ('?' = question)

s = station address (1 byte)

o = operation code (1 byte)

a = data address (1 byte)

n = data number (1 byte)

Dx = x-th data byte (1 byte)

k = checksum (1 byte)

Master operation codes:

'c' = check request

'w' = word reading (1 word = 2 bytes)

'W' = word writing (1 word = 2 bytes)

Possible master frames (questions):

check:	l (l=5)	?	s	c	k					
word read.:	l (l=7)	?	s	w	a	n	k			
word writ.:	l	?	s	W	a	n	D11	D10	..	Dn1
Dn0	k	(l=7+2*n)								

Slave string:

l i s o a n D1 .. Dm k

l = frame total length (1 byte)

i = slave string identification ('!' = OK answer, '#' = NOK answer)

s = station address (1 byte)

o = operation code (1 byte)

a = data address (1 byte)

n = data number (1 byte)

Dx = x-th data byte (1 byte)

k = checksum (1 byte)

Slave operation codes:

'C' = check answer

'w' = word reading answer (word = 2 byte)

'W' = word writing answer (word = 2 byte)

Possible slave frames (answers):

nok:	l (l=5)	#	s	o	k					
check:	l (l=5)	!	s	C	k					
word read.:	l	!	s	w	a	n	D11	D10	..	Dn1
Dn0 k		(l=7+2*n)								
word writ.:	l (l=5)	!	s	W	k					

ADDRESSES:

If the address "a" specified in the question is a < NUM_PAR (number of EEPROM parameters), then an EEPROM parameter (GUI table) is read or written. Otherwise if a > = NUM_PAR_EQP, then a parameter in the RAM table (Cf. module “userif.h”) is read or written. Its address in the RAM table is a-NUM_PAR.

Operation example:

Example 1

PC request of reading 16 words from the structure UIF_R, starting from the second one (UIF_R.ram_tab[1], .., UIF_R.ram_tab[16]):

0	07	Number of bytes in the frame
1	3F	Master string indicator "?"
2	00	Station address (it is always 0 in our boards)
3	77	word reading operation "w"
4	41	data start address (1(address in UIF_R.ram_tab) + 40h (offset to add for ram reading/writing))
5	10	number of data (10h=16dec)
6	39	checksum

Board answer:

0	27	Number of bytes in the frame (27h=39dec)
1	21	Slave string indicator "!"
2	00	Station address (it is always 0 in our boards)
3	77	word reading operation "w"
4	41	data start address (1(address in UIF_R.ram_tab)+40h(offset to add for ram reading))

5	10	number of data (10h=16dec)
6	00	MSB of the 1st word of data (UIF_R.ram_tab[1]=UIF_R.var.rpm, speed)
7	00	LSB of the 1st word of data
8	00	MSB of the 2nd word of data (UIF_R.ram_tab[2]=UIF_R.var.fre, imposed frequency)
9	00	LSB of the 2nd word of data
10	00	MSB of the 3rd word of data (UIF_R.ram_tab[3]=UIF_R.var.id, d axis current)
11	00	LSB of the 3rd word of data
12	00	MSB of the 4th word of data (UIF_R.ram_tab[4]=UIF_R.var.iq, q axis current)
13	00	LSB of the 4th word of data
14	00	MSB of the 5th word of data (UIF_R.ram_tab[5])
15	00	LSB of the 5th word of data
16	00	MSB of the 6th word of data (UIF_R.ram_tab[6])
17	00	LSB of the 6th word of data
18	00	MSB of the 7th word of data (UIF_R.ram_tab[7]=UIF_R.var.vb, bus voltage)
19	18	LSB of the 7th word of data
20	00	MSB of the 8th word of data (UIF_R.ram_tab[8])
21	00	LSB of the 8th word of data
22	00	MSB of the 9th word of data (UIF_R.ram_tab[9]=UIF_R.var.all, alarm)
23	01	LSB of the 9th word of data
24	00	MSB of the 10th word of data (UIF_R.ram_tab[10])
25	00	LSB of the 10th word of data
26	00	MSB of the 11th word of data (UIF_R.ram_tab[11])
27	00	LSB of the 11th word of data
28	00	MSB of the 12th word of data (UIF_R.ram_tab[12])
29	00	LSB of the 12th word of data
30	00	MSB of the 13th word of data (UIF_R.ram_tab[13])
31	00	LSB of the 13th word of data
32	00	MSB of the 14th word of data (UIF_R.ram_tab[14])
33	00	LSB of the 14th word of data
34	00	MSB of the 15th word of data (UIF_R.ram_tab[15])
35	00	LSB of the 15th word of data
36	00	MSB of the 16th word of data (UIF_R.ram_tab[16])
37	00	LSB of the 16th word of data
38	69	checksum

Example 2

PC request of writing 4 words in the structure UIF_W, starting from the third one (UIF_W.ram_tab[2], .., UIF_W.ram_tab[5]):

0	0F	Number of bytes in the frame (0Fh=15dec)
1	3F	Master string indicator "?"
2	00	Station address (it is always 0 in our boards)
3	57	word writing operation "W"
4	42	data start address (2(address in UIF_W.ram_tab)+40h(offset to add for ram reading/writing))
5	04	number of data
6	03	MSB of the 1st word of data (value (03E8h=1000dec) to be written in UIF_W.ram_tab[2] = UIF_W.var.rif, speed ref)
7	E8	LSB of the first word of data
8	00	MSB of the second word of data (value to be written in UIF_W.ram_tab[3], not used)
9	00	LSB of the second word of data
10	00	MSB of the third word of data (value to be written in UIF_W.ram_tab[4], not used)
11	00	LSB of the third word of data
12	00	MSB of the fourth word of data (value to be written in UIF_W.ram_tab[5], not used)
13	00	LSB of the fourth word of data
14	E7	checksum

Board answer (indicates that the request is received and processed):

0	05	Number of bytes in the frame
1	21	Slave string indicator "!"
2	00	Station address (it is always 0 in our boards)
3	57	word writing operation "W"
4	E6	checksum

Note: Four new operation codes have been added:

'y'(=0x79): word reading (EEPROM minimum value)

'z'(=0x7A): word reading (EEPROM maximum value)

'k'(=0x6B): word reading (measurement samples vector)

'j'(=0x6A): word reading (EEPROM default value)

In these cases, the address is a < NUM_PAR.

To understand in more details the software implementation, please find below the extract of the “userif.h“ module, part of the embedded software.

```
typedef union
{
        unsigned short          ram_tab[N_RAM_READ];
        struct
        {
                signed short    rif;            //  0   internal speed reference
                signed short    rpm;            //  1   measured/estimated speed
                signed short    fre;            //  2   applied frequency
                signed short    id;             //  3   direct current
                signed short    iq;             //  4   quadrature current
                signed short    vd;             //  5   direct voltage
                signed short    vq;             //  6   quadrature voltage
                signed short    vb;             //  7   bus voltage
                signed short    all;            //  8   alarm
                signed short    flg;            //  9   status flags
                signed short    toti;           // 10   current vector amplitude
                signed short    totv;           // 11   voltage vector amplitude
                signed short    du0;            // 12
                signed short    du1;            // 13
                signed short    du2;            // 14
                signed short    du3;            // 15
                signed short    mod;            // 16   mode
                signed short    rs;             // 17   stator resistance
                signed short    ls;             // 18   synchronous inductance
                signed short    fl;             // 19   permanent magnet flux
                signed short    kpi;            // 20   current loop kp
                signed short    kii;            // 21   current loop ki
                signed short    pwmf;           // 22   pwm frequency [Hz]
                signed short    sf;             // 23   sampling frequency [Hz]
                signed short    ena;            // 24   extra features enable flags
                signed short    du4;            // 25
                signed short    du5;            // 26
                signed short    du6;            // 27
                signed short    du7;            // 28
                signed short    du8;            // 29
                signed short    du9;            // 30
                signed short    du10;           // 31

        }                                       var;
}       UIF_R_t;
```

```
typedef union
{
        unsigned short  ram_tab[N_RAM_WRITE];

        struct
        {
                signed short    trg;            //  0   trigger
                signed short    mod;            //  1   mode
                signed short    rif;            //  2   speed reference
                signed short    cra;            //  3   current ratio
                signed short    sel;            //  4   variable and time scale selection
                signed short    du0;            //  5
                signed short    du1;            //  6
                signed short    du2;            //  7
        }                                       var;
}       UIF_W_t;
```

```
/*****************************************************************************
Variable Externs
*****************************************************************************/
#ifdef _USERIF_C
UIF_R_t                         UIF_R;
UIF_W_t                         UIF_W;
OUTB_t                          outbuf, outbuf1;
uint16_t                        *wbuf = ((uint16_t *)(outbuf1.ss));
uint16_t                        *rbuf = ((uint16_t *)(outbuf.ss));
uint16_t                        *pbuf = ((uint16_t *)(outbuf.ss));

#else   // _USERIF_C
extern UIF_R_t    UIF_R;
extern UIF_W_t    UIF_W;
extern OUTB_t           outbuf, outbuf1;
extern uint16_t         *wbuf;
extern uint16_t         *rbuf;
extern uint16_t         *pbuf;
#endif
```

```
/*      --- Special Working Modes ---

        Special working modes are controlled trough UIF_W.var.mod:
        if it is 0 then normal mode is used, other values stay for
        special modes. A feedback of the working mode is given in UIF_R.var.mod,
        which is equal to the working mode.
        If the working mode requires a trigger from the GUI to execute some
        operations, UIF_W.var.trg bit0 is used (it is automatically cleared after
        being detected); the feedback regarding the requested operation status
        is given by some bits in UIF_R.var.flg: bit9=1 means BUSY, while if it is 0 means
        READY, bitA=1 means last operation was ENDED_NOK, while if it is 0 means ENDED_OK.
        Trigger requests are not accepted if UIF_R.var.sta is not ready. It means that the
        GUI has to stay in polling for this flag after a trigger.
        Other important flags are bit7 (alarm), and bit8 which indicates that the motor
        is driven.
        Working status can be changed only when the motor in not driven (UIF_R.var.flg
        has bit8 equal to 0).
*/

#define ENA_CURPI_TUN   UIF_R.var.ena |= WSET0;
#define ENA_CURPI_AUT   UIF_R.var.ena |= WSET1;
#define ENA_AUTO_IDEN   UIF_R.var.ena |= WSET2;
//#define ENA_OSCI_WIND UIF_R.var.ena |= WSET3;

#define NORM_MODE_CODE  ( 0 )   // normal inverter behavior
#define CURPI_TUN_CODE  ( 1 )   // current PI gains manual tuning mode
#define CURPI_AUT_CODE  ( 2 )   // current PI gains automatic detection mode
#define AUTO_IDEN_CODE  ( 3 )   // motor parameters auto-identification mode

#define NORM_MODE_REQ   ( NORM_MODE_CODE == UIF_W.var.mod )
#define CURPI_TUN_REQ   ( CURPI_TUN_CODE == UIF_W.var.mod )
#define CURPI_AUT_REQ   ( CURPI_AUT_CODE == UIF_W.var.mod )
#define AUTO_IDEN_REQ   ( AUTO_IDEN_CODE == UIF_W.var.mod )
```

```
#define NORM_MODE       ( NORM_MODE_CODE == UIF_R.var.mod )
#define CURPI_TUN       ( CURPI_TUN_CODE == UIF_R.var.mod )
#define CURPI_AUT       ( CURPI_AUT_CODE == UIF_R.var.mod )
#define AUTO_IDEN       ( AUTO_IDEN_CODE == UIF_R.var.mod )

#define NOT_NORM_MODE   ( NORM_MODE_CODE != UIF_R.var.mod )
#define NOT_CURPI_TUN   ( CURPI_TUN_CODE != UIF_R.var.mod )
#define NOT_CURPI_AUT   ( CURPI_AUT_CODE != UIF_R.var.mod )
#define NOT_AUTO_IDEN   ( AUTO_IDEN_CODE != UIF_R.var.mod )

#define SET_NORM_MODE   UIF_R.var.mod = NORM_MODE_CODE;
#define SET_CURPI_TUN   UIF_R.var.mod = CURPI_TUN_CODE;
#define SET_CURPI_AUT   UIF_R.var.mod = CURPI_AUT_CODE;
#define SET_AUTO_IDEN   UIF_R.var.mod = AUTO_IDEN_CODE;

#define ALRM_ON                 (UIF_R.var.flg & WSET7)
#define ALRM_OFF                (! ALRM_ON)
#define COM_ON                  (UIF_R.var.flg & WSET8)
#define COM_OFF                 (! COM_ON)
#define STA_BSY                 (UIF_R.var.flg & WSET9)
#define STA_RDY                 (! STA_BSY)
#define END_NOK                 (UIF_R.var.flg & WSETA)
#define END_OK                  (! END_NOK)
#define ICOM_ON                 (UIF_R.var.flg & WSETB)
#define ICOM_OFF                (! ICOM_ON)

#define SET_ALRM_ON             UIF_R.var.flg |= WSET7;
#define RES_ALRM_ON             UIF_R.var.flg &= WCLR7;
#define SET_COM_ON              UIF_R.var.flg |= WSET8;
#define RES_COM_ON              UIF_R.var.flg &= WCLR8;
#define SET_STA_BSY             UIF_R.var.flg |= WSET9;
#define RES_STA_BSY             UIF_R.var.flg &= WCLR9;
#define SET_END_NOK             UIF_R.var.flg |= WSETA;
#define RES_END_NOK             UIF_R.var.flg &= WCLRA;
#define SET_ICOM_ON             UIF_R.var.flg |= WSETB;
#define RES_ICOM_ON             UIF_R.var.flg &= WCLRB;

#define GUI_TRI                 (UIF_W.var.trg & WSET0)
#define RES_GUI_TRI             UIF_W.var.trg &= WCLR0;
#define GUI_STP                 (UIF_W.var.trg & WSET1)
#define RES_GUI_STP             UIF_W.var.trg &= WCLR1;
```

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Summary
0.01	February 22, 2018	-	Preliminary edition
0.02	February 18, 2019	-	Minor failure corrections. Preliminary edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.