LOW COST MOTOR CONTROL DEMO BOARD BASED ON RENESAS RX62T

Introduction

This document describes the Motor Control Reference Platform [MCRP07] based on the new powerful 32-bit Renesas MCU **RX62T**.

This platform drives a 3-phase Permanent Magnet Synchronous Motor (Brushless Motor) by using an advanced sensor-less Field Oriented Control algorithm.

The phase currents measurement is done via three shunts which offer a very low cost solution avoiding any expensive current sensor; single shunt current reading is also possible.

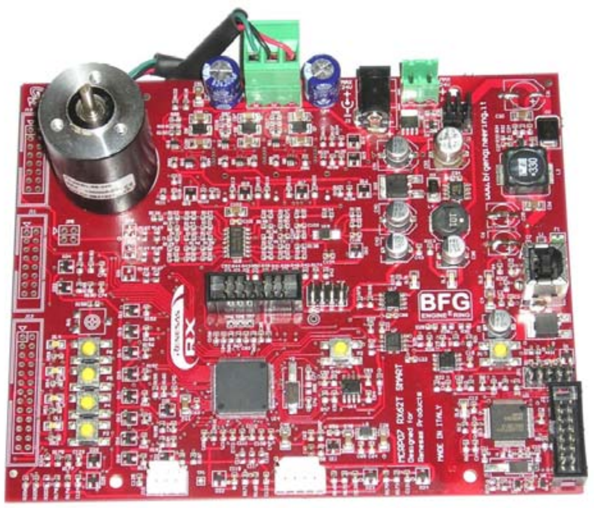
The main focus applications for this type of algorithm are compressors, air conditioning, fans, industrial drives, washer, etc.

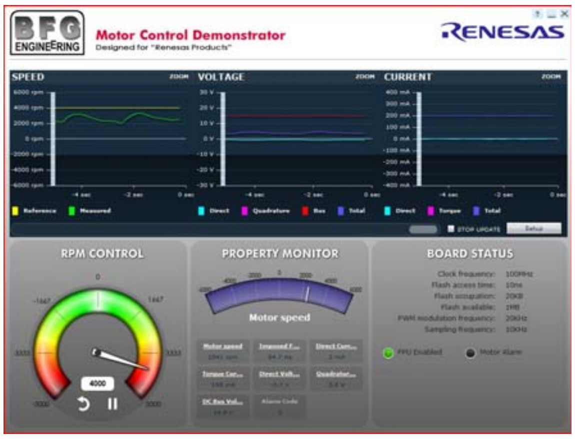
The platform allows easy custom applications development, including an on-board small motor and a user-friendly PC user interface.

The platform can be powered directly by USB, or it's possible to provide an external power supply and in this case the USB communication is optically insulated.

An external power stage can be managed, in order to control higher power motors.

With this demo system you can experience a new and easier way to deal with motor control, you can develop and test your own solutions, thanks to the power of the new Renesas MCUs.





Clock frequency: 100MHz
Flash access time: 10ns
Flash occupation: 20KB
Flash available: 1MB
PWM modulation frequency: 20KHz
Sampling frequency: 10KHz

Contents

Hardware overview

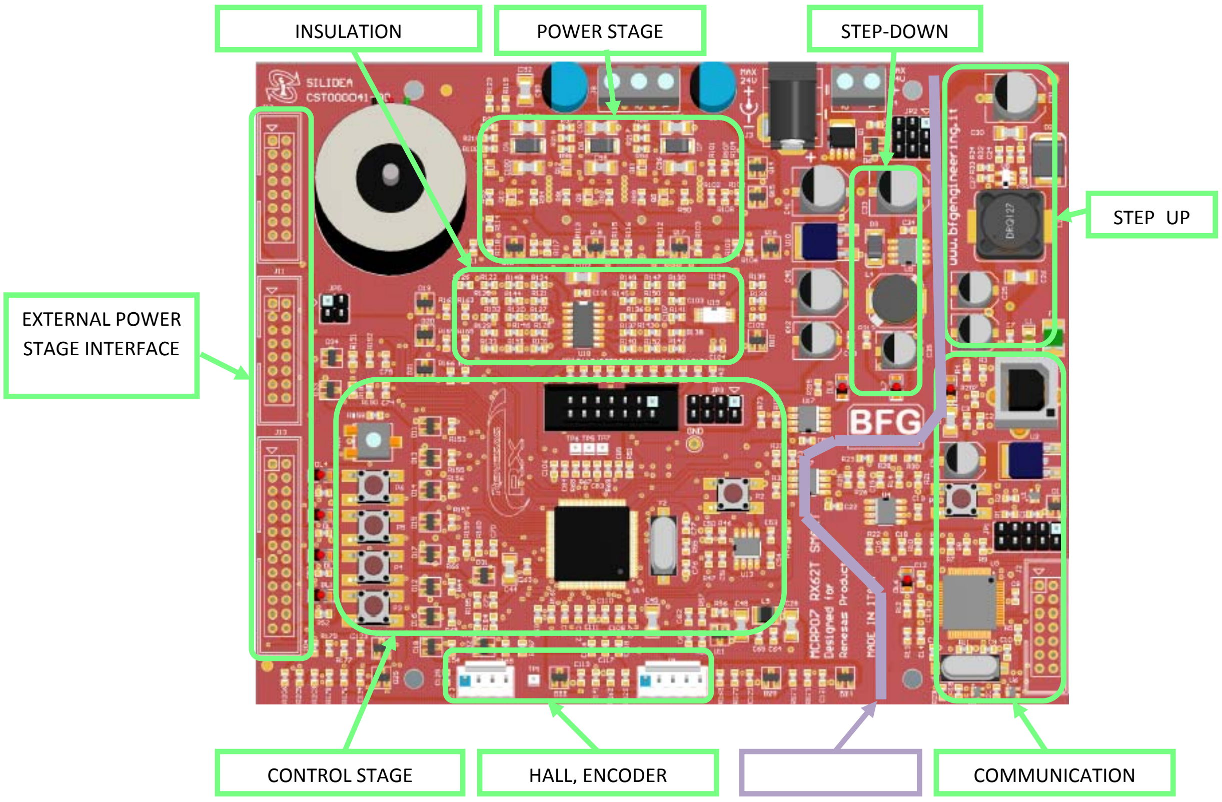
MCRP07 is a single board inverter, based on the new RX series microcontroller RX62T. The hardware includes a low-voltage MOSFETs power stage, and a communication stage.

To obtain the maximum flexibility, the demo board includes:

- A complete 3-phase inverter on-board with a low voltage motor, so it becomes easy to test the powerful sensor-less algorithm on the RX62T.
- USB communication with the PC via a H8S2212 MCU.
- Connectors for hall sensors and encoder connections.
- Compatibility with the existing Motor Control Reference Platforms MCRP05/06 power stage.

To achieve these aims, an independent communication stage was implemented, based on the Renesas microcontroller H8S2212, which performs the USB to serial conversion (the two serial lines RX and TX are easier to insulate).

This stage uses the PC USB power lines as power supply. Furthermore, the possibility to supply all the board using the PC USB port was added, realizing a step-up converter to obtain the inverter V_{BUS} necessary for the motor; obviously, if this feature is used, the system is no more insulated from the PC.



If external power supply is used for the inverter, the logic power supply is obtained through a step-down converter, in order to reduce heating and power consumption.

Please refer to the electrical drawings to get the hardware implementation details.

Power supply selection

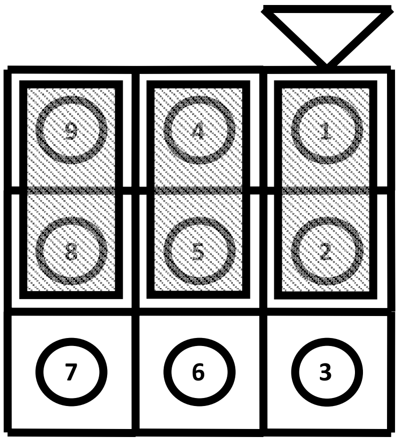
As stated before, there are two ways to supply power to the board.

One possibility is to use directly the PC USB supply, and in this case the current you can give to the motor is limited by the USB possibilities. A dual power USB cable is recommended to give enough power to the board.

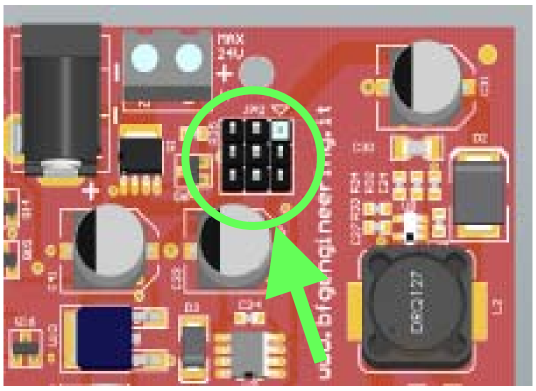
The second possibility is to use an external voltage DC source to supply the board.

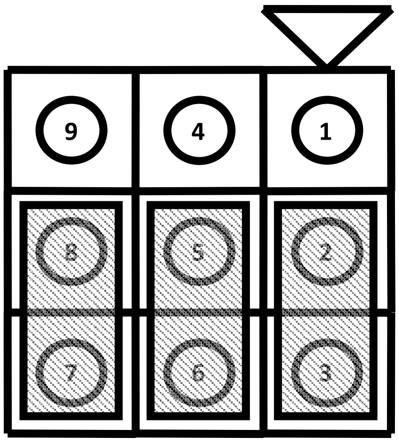
The recommended voltage values are between 12V_{DC} and 24V_{DC}. In this case the communication stage is insulated from the inverter.

The selection between the two possibilities is made through three jumpers in the J2 connector, as described in the following figure.



PC USB SUPPLY SELECTION





EXTERNAL SUPPLY SELECTION

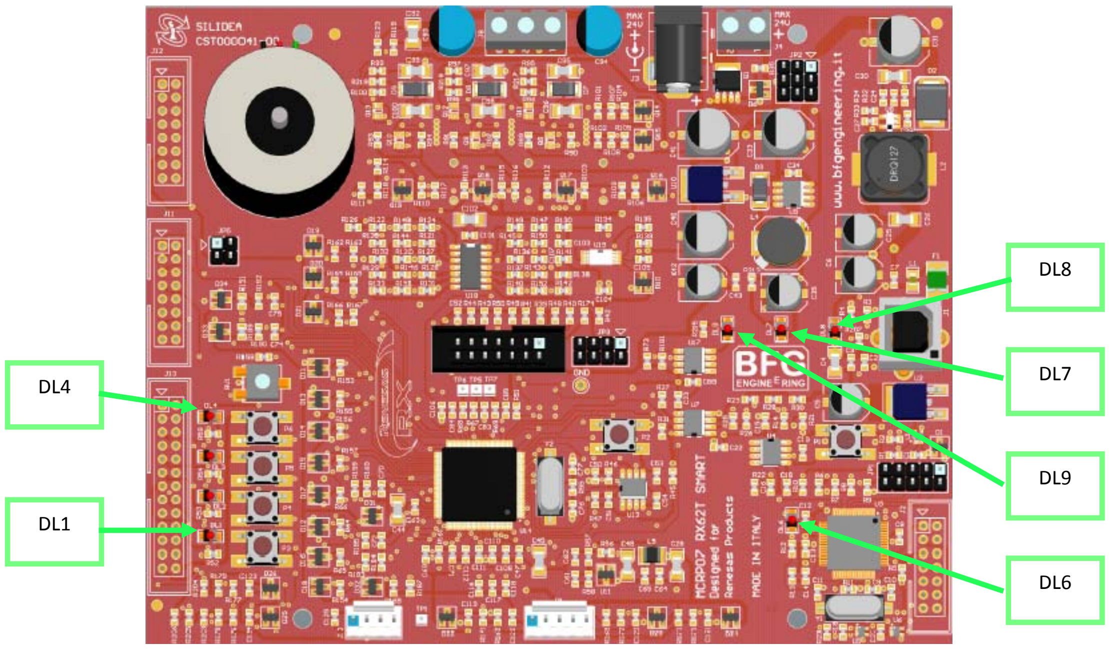
The first jumper configuration connects the USB ground to the inverter ground, the USB 5Vdc to the logic +5Vdc and the output of the step-up converter (around 13Vdc) to the inverter DC link.

The second jumper configuration connects the external power supply ground to the inverter ground, the output of the step-down converter (+5Vdc) to the logic +5Vdc and the external +Vdc (from 12 to 24 Vdc) to the inverter DC link.

LEDs function description

Three LEDs available on the board are directly connected to the hardware and allows the user to understand the status of the supply of the board. Please refer to the LED map for the following indications:

- DL8 is connected to the USB supply, so it indicates that the USB port is supplied (and, by consequence, all the communication section);
- DL7 is connected to the step-down converter output, and it is on only if an external power supply is connected;
- DL9 is connected to the logic supply, so it indicates that the control section is supplied.



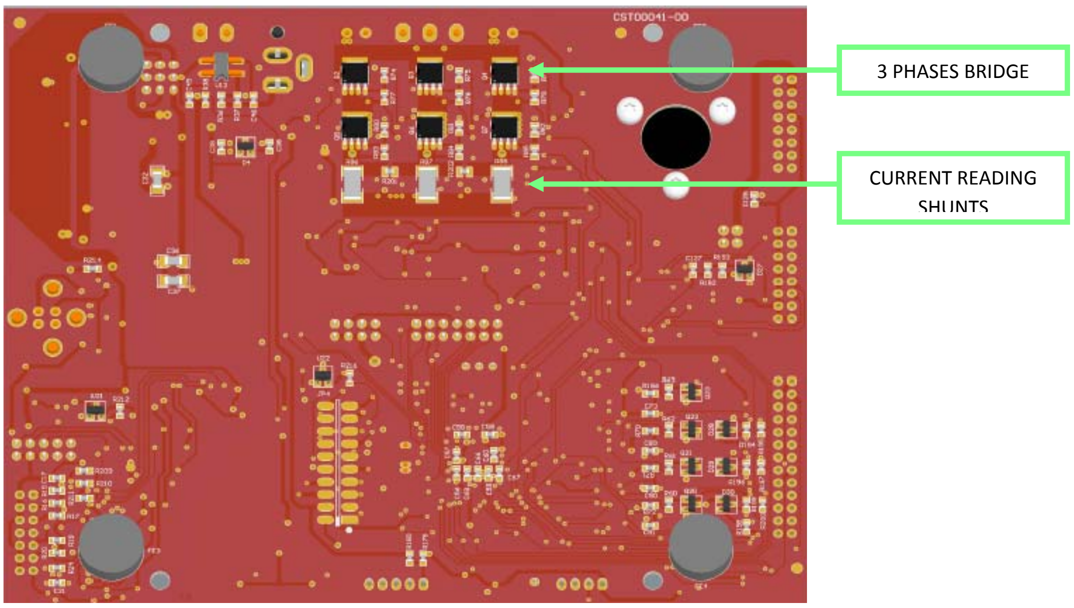
The other LEDs in the board are driven via software, in particular:

- DL6 is blinking if there is a communication between the PC and the board;
- DL1 is blinking if the control section MCU (RX62T) is running normally.
- DL4 is blinking (quickly) if an alarm has been detected.

Internal power stage brief description

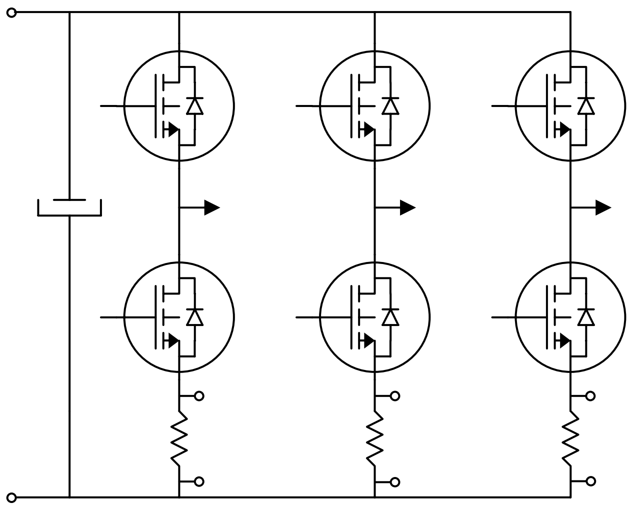
The power stage is a complete 3-phase bridge composed with discrete low voltage power MOSFETs, mounted on the bottom side of the board. The MOSFETs are the Renesas **RJK0654DPB** n-channel power MOSFETs (please refer to the data-sheet for the characteristics).

On the upper side of the board is mounted the MOSFETs driving circuit, composed with discrete elements (refer to the electric drawings).



The current reading shunts are also in the bottom side of the board, while the signal conditioning circuit is in the upper side.

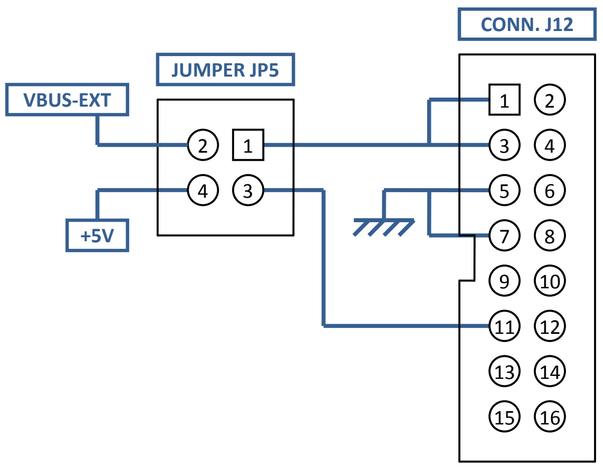
The inverter has the classical schema with the three shunts on the lower arms:

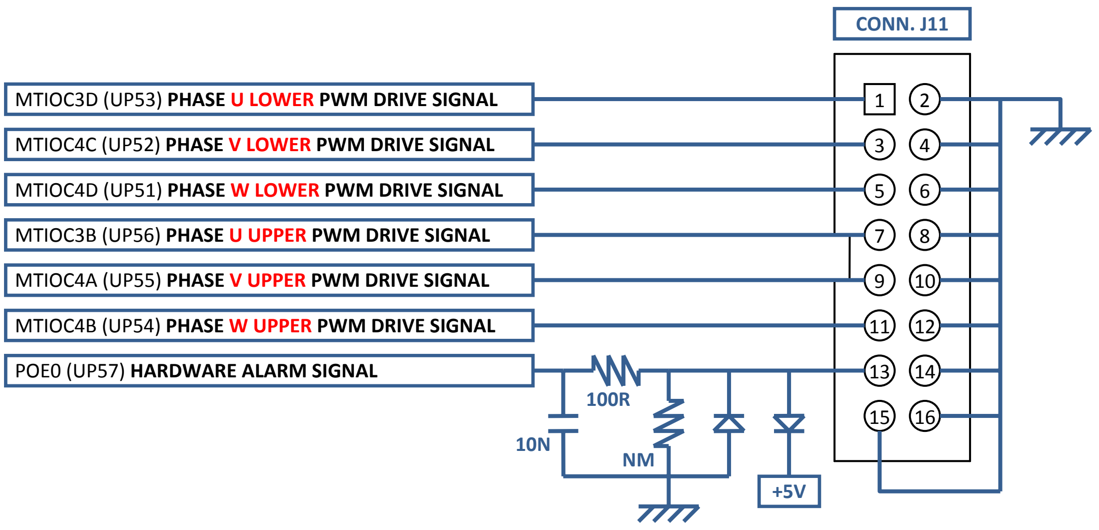


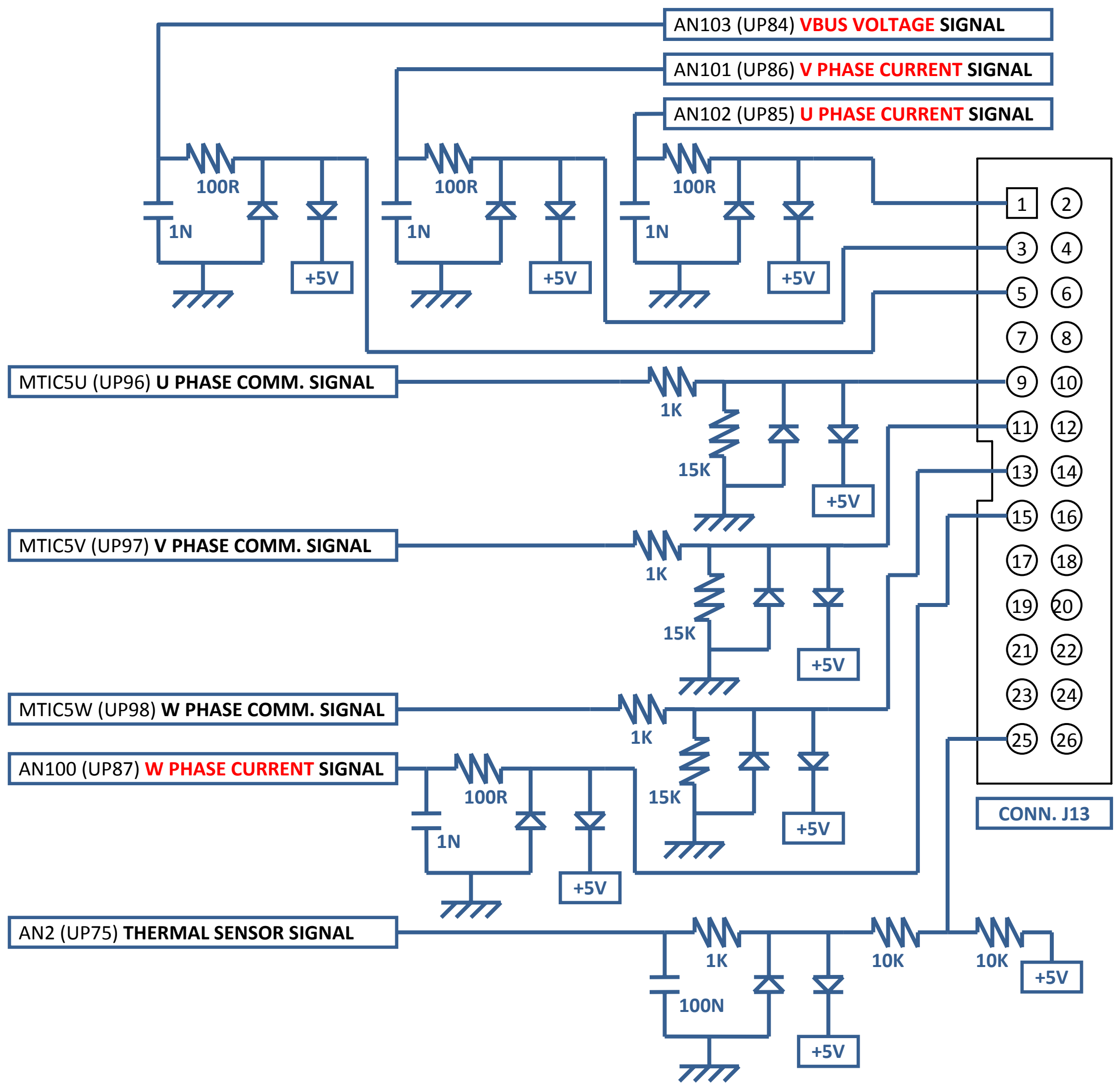
Interface with an external power stage

Since internal power stage allows only the management of small motors, an interface with an external power stage was added. This was made easy due to the presence in the microcontroller of several timer sections that make it possible to manage more than one motor.

Pls. find below the schematic of the connectors present in the board, used for connecting an external power stage.




CONN. J11
MTIOC3D (UP53) PHASE U LOWER PWM DRIVE SIGNAL
MTIOC4C (UP52) PHASE V LOWER PWM DRIVE SIGNAL
MTIOC4D (UP51) PHASE W LOWER PWM DRIVE SIGNAL
MTIOC3B (UP56) PHASE U UPPER PWM DRIVE SIGNAL
MTIOC4A (UP55) PHASE V UPPER PWM DRIVE SIGNAL
MTIOC4B (UP54) PHASE W UPPER PWM DRIVE SIGNAL
POE0 (UP57) HARDWARE ALARM SIGNAL
100R
10N
NM
+5V


AN103 (UP84) VBUS VOLTAGE SIGNAL
AN101 (UP86) V PHASE CURRENT SIGNAL
AN102 (UP85) U PHASE CURRENT SIGNAL
100R
1N
+5V
MTIC5U (UP96) U PHASE COMM. SIGNAL
MTIC5V (UP97) V PHASE COMM. SIGNAL
MTIC5W (UP98) W PHASE COMM. SIGNAL
AN100 (UP87) W PHASE CURRENT SIGNAL
AN2 (UP75) THERMAL SENSOR SIGNAL
1K
15K
10K
100N
CONN. J13

The interface between the board and an external power stage is organized as follows:

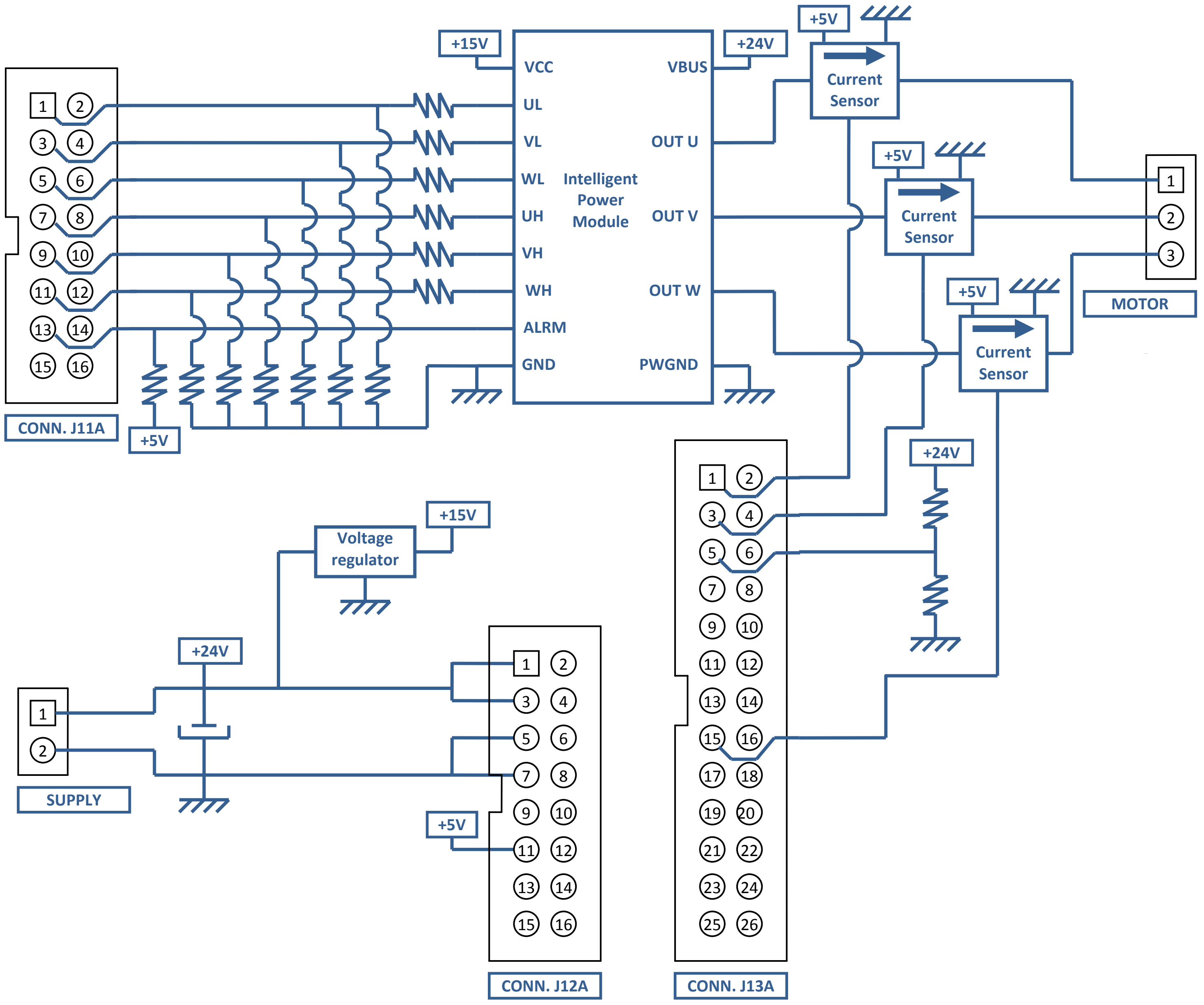
a) A 16 pins connector (J11) is used for the PWM drive signals; the signals are directly connected to the microcontroller output pins, and there is no pull-up or pull-down resistor connected, so the polarization has to be done in the power stage (note that in case of alarm, the microcontroller output pins can be placed in high impedance state, so the external polarization is necessary); these output commands are logic level signals, with limited current output capability, so an external driver is probably required. A further line is connected to the microcontroller: it is the external alarm signal, connected to the POE input pin; this pin is not polarized, so if the POE is enabled and the input is leaved unconnected, undesired alarms can occur. All the free pins of the connector are connected to the board ground to minimize the cross talking of the lines if a flat cable is used.

b) A 26 pins connector (J13) is used to collect some signals from the power stage, in particular the current readings and the DC link voltage reading; those signals are clamped and weakly filtered, then directly connected to the A/D converter input pins of the microcontroller, so the external power stage has to take care of the gain and the offset of these signals. An input is dedicated also to a thermal sensor, and a pull-up resistor is present. Three further signals are managed: they are the commutation signals from the output phases, useful if the hardware compensation of dead-times facility of the MTU is used; those signals are clamped with a diode directly connected with the microcontroller power supply, so a suitable series resistance is needed in the power stage to avoid damages to the board.

c) A further connector (J12) can be used to supply the board from the power stage or vice-versa (making a short circuit between the pins 1 and 2 of the jumper JP5); also the board 5V can be make available to the power board (making a short circuit between the pins 3 and 4 of JP5), but not vice-versa, because they are directly connected to the step-down switching supply of the board. The ground connection is always on, and it represents the reference for all the interface signals.

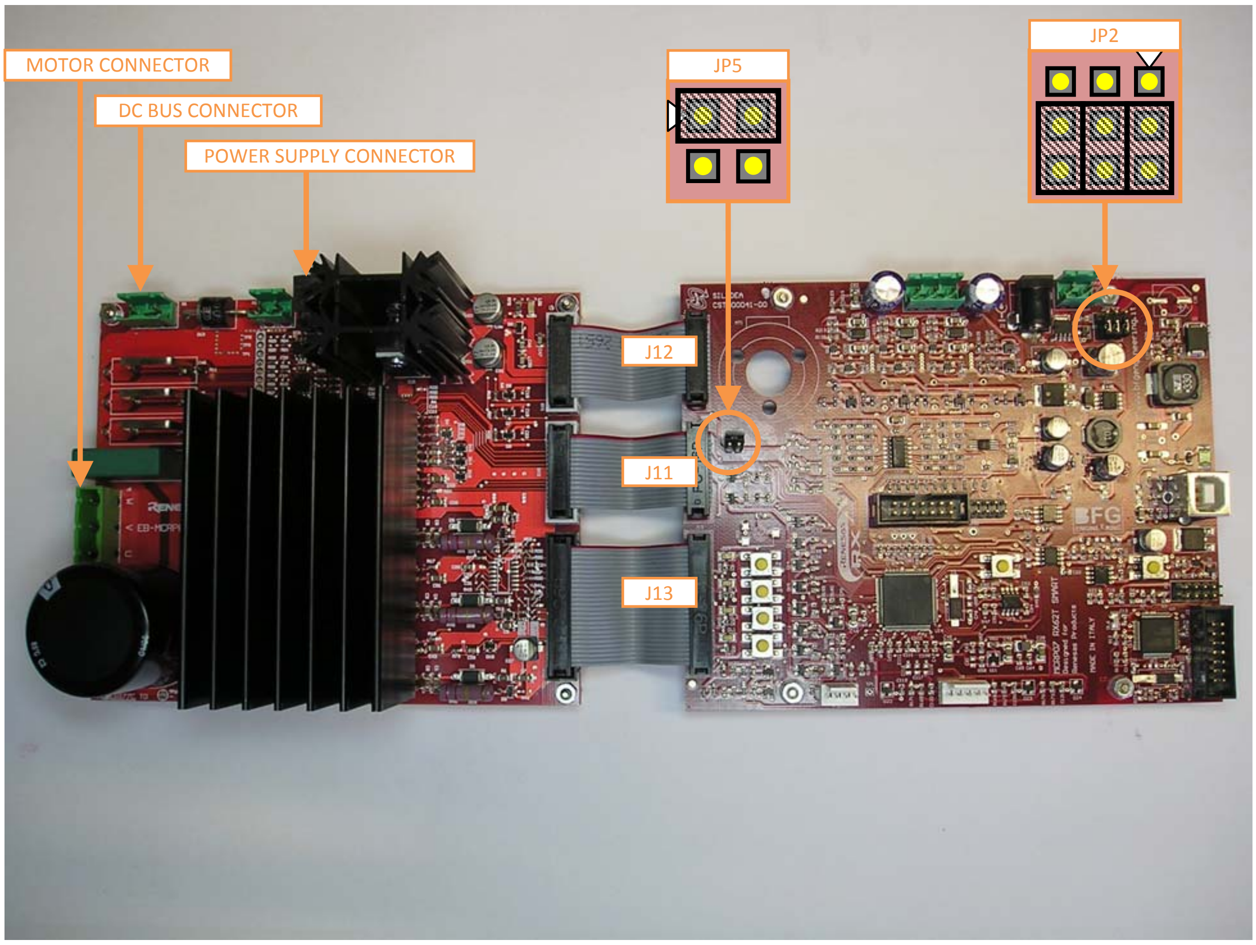
In the next figure a simple example regarding how the power board has to be arranged is presented: the power supply comes from the supply connector, and the supply for power module is derived from it. The external supply is also used to supply the microcontroller board through the connector J12A (and the jumper JP5 in microcontroller board); the 5V supply for current sensors and for the signal polarization is derived from the microcontroller board, through J12A (and JP5). The PWM drive signals are taken from J11A, while the current sensing signals and the bus voltage measurement are brought to J13A (the phase commutation signals and the temperature sensing signal are not reported for sake of simplicity).

Pls. refer to the complete schematics for further details.


+15V
+24V
+5V
VCC
VBUS
UL
VL
WL
UH
VH
WH
ALRM
GND
OUT U
OUT V
OUT W
PWGND
Intelligent Power Module
Current Sensor
Current Sensor
Current Sensor
MOTOR
CONN. J11A
CONN. J12A
CONN. J13A
Voltage regulator
SUPPLY
+24V
+5V
+15V

Example of connection with an external power stage

The interface for an external power supply was designed to be compatible with the power stages of previous Renesas motor control demo platforms MCRP05/06. The schematic of such a power stage has been included in the documentation of the present board. A picture of the two board connected can be found below.



The power supply (24Vdc) is furnished trough the power board, and comes to the microcontroller board through the jumper JP5, in which the pin 1 and 2 are short-circuited, while the 3 and 4 are leaved open. In the microcontroller board, the supply configuration jumper JP2 is configured in order to select the external supply (not the USB one). In the MCRP05 power board, a DC bus connector allows the user to have (only for the motor) a higher external DC voltage; in such way high voltage motors can be managed.

Control MCU overview

The RX62T Group is a set of MCUs featuring the high-speed, high-performance RX CPU as the 100MHz processor core.

Each basic instruction of the processor is executable in one cycle of the system clock. Calculation functionality is enhanced by the inclusion of a single-precision floating-point calculation unit as well as a 32-bit multiplier and divider. Additionally, code efficiency is improved by instructions with lengths that are variable in byte units to cover an enhanced range of addressing modes.

A multi-functional timer pulse unit 3 (for motor control), general PWM timer, compare match timers, watchdog timer, independent watchdog timer, serial communications interfaces, I2C bus interfaces, CAN module, serial peripheral interface, LIN module, 12-bit A/D converters with three-channel simultaneous sampling function, and 10-bit A/D converter are incorporated as peripheral functions which are essential to motor control devices. In addition, the 12-bit A/D converters include a window comparator and programmable gain amplifier for additional functionality.

Please find below the summary of the RX62T features:

RX600 CPU

- High-speed: 100MHz clock
- High performance: 1.65MIPS/MHz
- Low current consumption: only 50mA @ 100MHz
- Single-precision floating point unit FPU, barrel shifter, MAC, RMPA
- 256kB Flash/16kB RAM to 64kB Flash/8kB RAM
- Zero wait access to Flash memory
- 64pin – 112pin package options

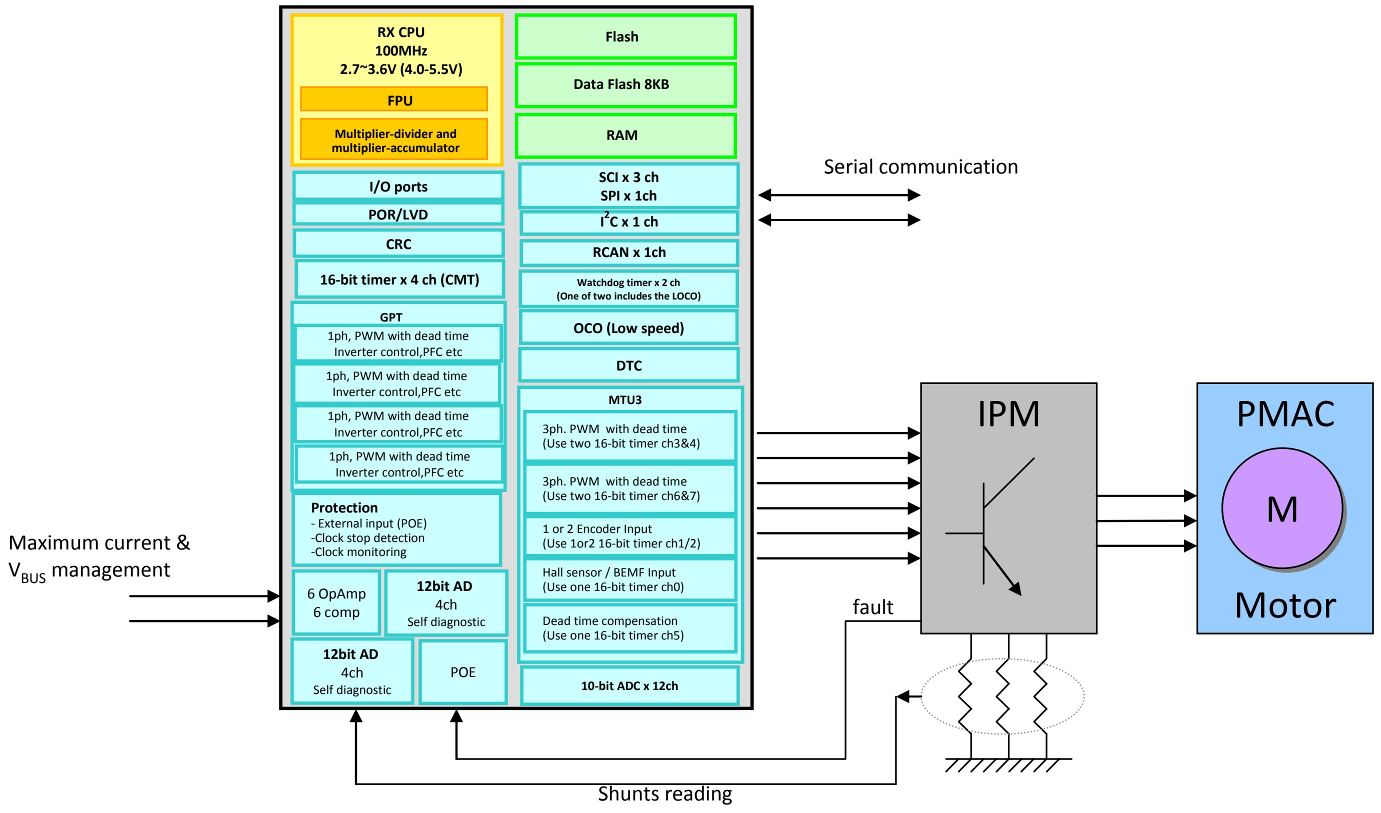
Functions

- Enhanced PWM resolution with MTU3, enhanced PWM functionality with GPT
- 12-bit A/D converter (1µs) : 4 channels x 2 unit , 10-bit A/D converter (1µs) : 12 channels x 1 unit
- Three S/H circuits for each unit: Three shunts control enable
- Double data registers for each unit: 1 shunt control enable
- Programmable gain Operational amplifier, Window comparator for Voltage monitoring
- CAN option

Large-capacity flash memory units capable of high-speed operation are included as on-chip memory, significantly reducing the cost of configuring systems.

The main application fields of this MCU are: industrial equipment, household electrical appliances, machines requiring motor control, and inverter-powered machines.

Please find below the block diagram of the RX62T and the role of each peripherals.

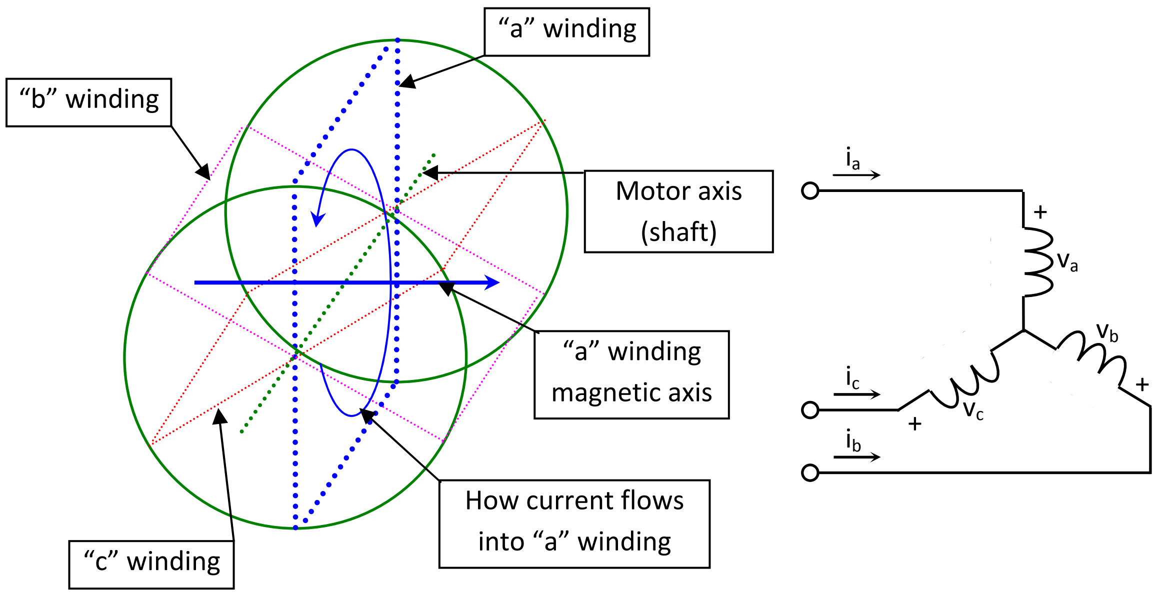

RX CPU
100MHz
2.7~3.6V (4.0-5.5V)
FPU
Multiplier-divider and multiplier-accumulator
Flash
Data Flash 8KB
RAM
I/O ports
POR/LVD
CRC
16-bit timer x 4 ch (CMT)
GPT
1ph, PWM with dead time Inverter control,PFC etc
1ph, PWM with dead time Inverter control,PFC etc
1ph, PWM with dead time Inverter control,PFC etc
1ph, PWM with dead time Inverter control,PFC etc
Protection
- External input (POE)
-Clock stop detection
-Clock monitoring
6 OpAmp
6 comp
12bit AD 4ch Self diagnostic
12bit AD 4ch Self diagnostic
POE
SCI x 3 ch
SPI x 1ch
I²C x 1 ch
RCAN x 1ch
Watchdog timer x 2 ch (One of two includes the LOCO)
OCO (Low speed)
DTC
MTU3
3ph. PWM with dead time (Use two 16-bit timer ch3&4)
3ph. PWM with dead time (Use two 16-bit timer ch6&7)
1 or 2 Encoder Input (Use 1or2 16-bit timer ch1/2)
Hall sensor / BEMF Input (Use one 16-bit timer ch0)
Dead time compensation (Use one 16-bit timer ch5)
10-bit ADC x 12ch
Serial communication
Maximum current & V_BUS management
IPM
PMAC
M
Motor
fault
Shunts reading

Permanent magnets brushless motor model

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:

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



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 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 mainly implementation are using a position sensor or a speed sensor.

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

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

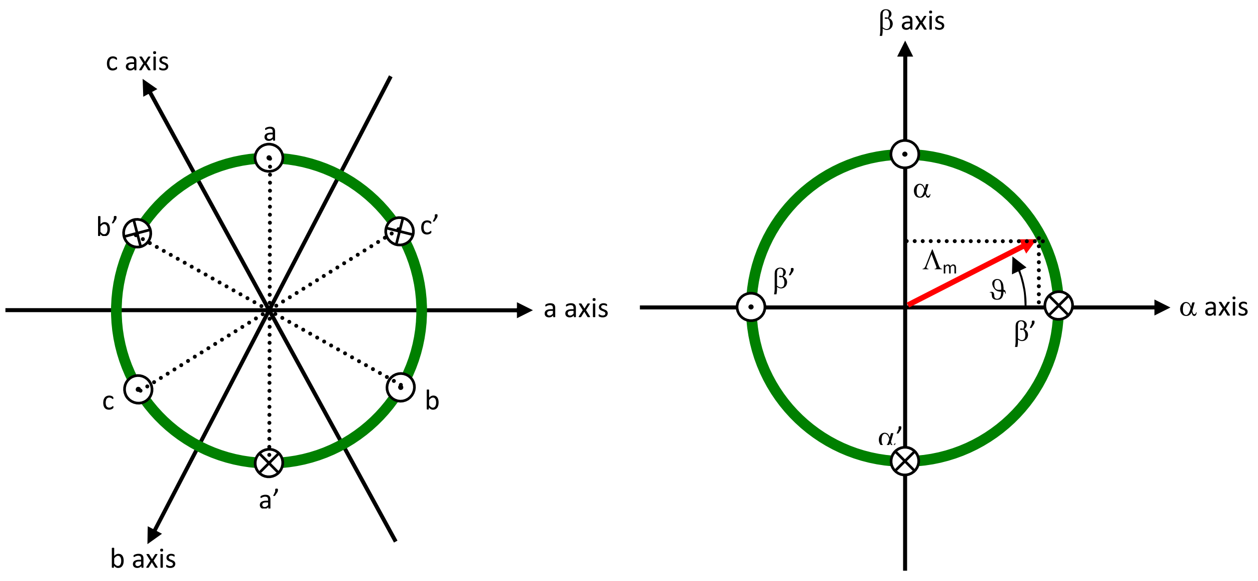
If we apply three voltages $v_a(t)$, $v_b(t)$, $v_c(t)$ to the stator windings, the relations between phase voltages and currents are:

$$v_a = R_S i_a + \frac{d\lambda_a}{dt}$$
$$v_b = R_S i_b + \frac{d\lambda_b}{dt}$$
$$v_c = R_S i_c + \frac{d\lambda_c}{dt}$$

- 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)

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 angle ϑ.

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: $\vartheta(t) = \omega t$) 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 phases system" may be represented by an equivalent "two phases system". So the 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:

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

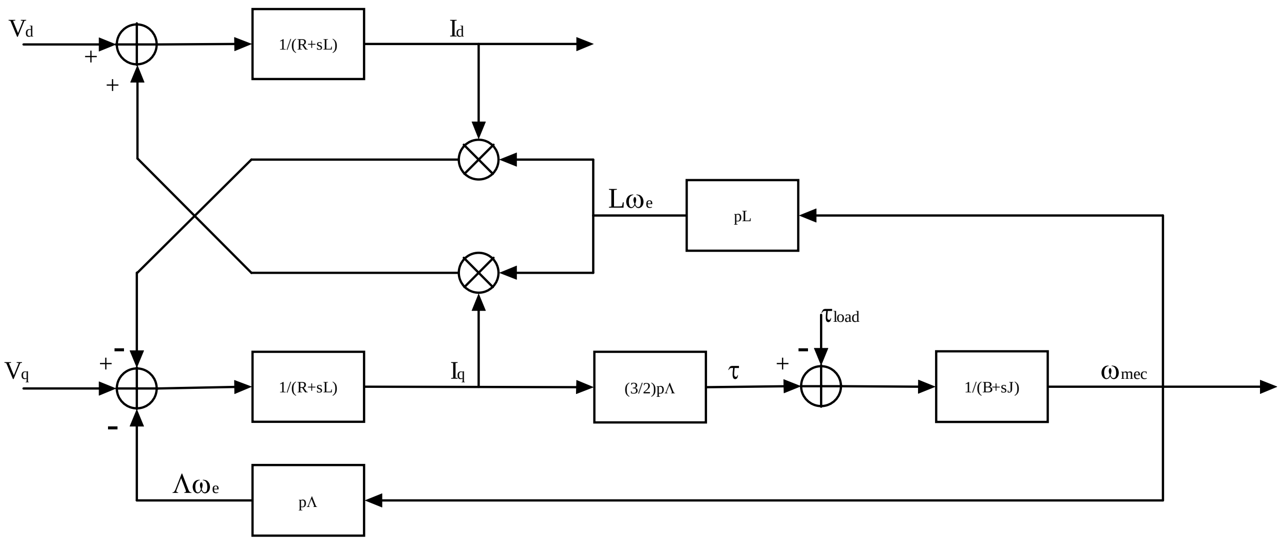
A second reference frame is used to represent the equations as the frame is turning at the rotor speed. So the "d" axis is chosen in the direction of the magnetic vector Λ_m, and with the "q" axis orthogonal to the "d" axis. The new reference system is (d, q).

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

So we obtain two inter-dependant 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.



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.

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

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

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:

$$\vartheta = \arctan({}^{x}\!/_{y})$$

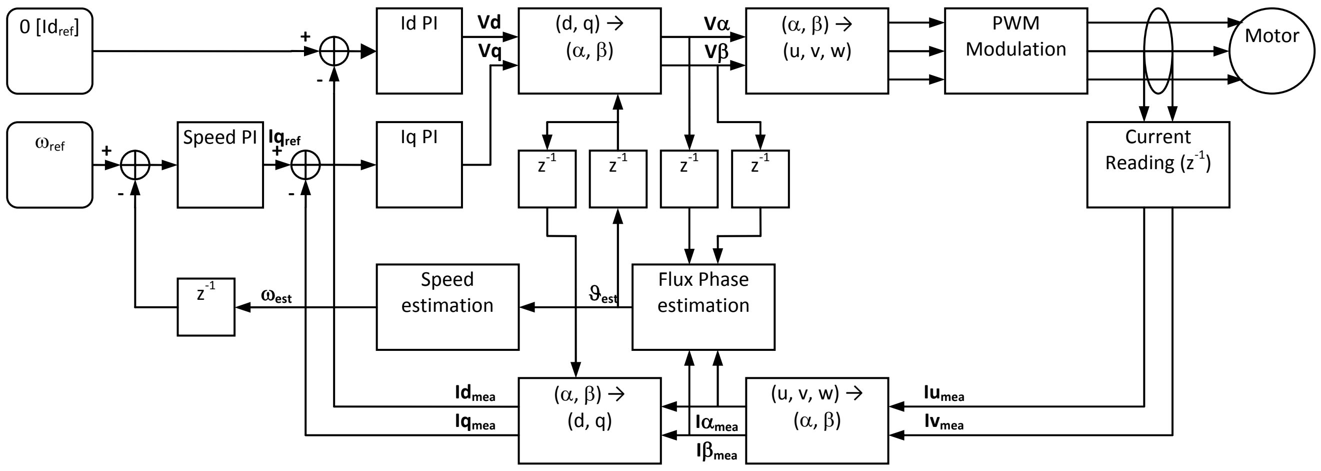
The system speed ω can be obtained as the derivative of the angle ϑ.

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

Based on this, a sensor-less 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.

Sensor-less F.O.C. algorithm

Please, find below the sensor-less FOC algorithm block diagram.

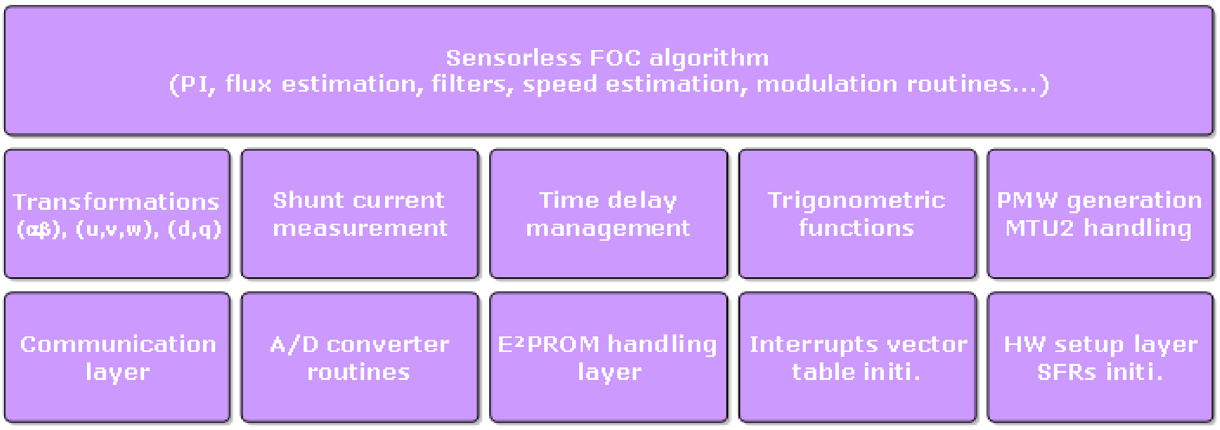


The only difference between the three shunts configuration and the single shunt one is in the "Current Reading" block, the rest of the algorithm remains the same.

Software description

On MCRP07 this software is working on RX62T [100MHz], which is a huge microcontroller for this kind of algorithm. This allows the user to realize virtually what he wants in addition.

MCRP05: Software blocks

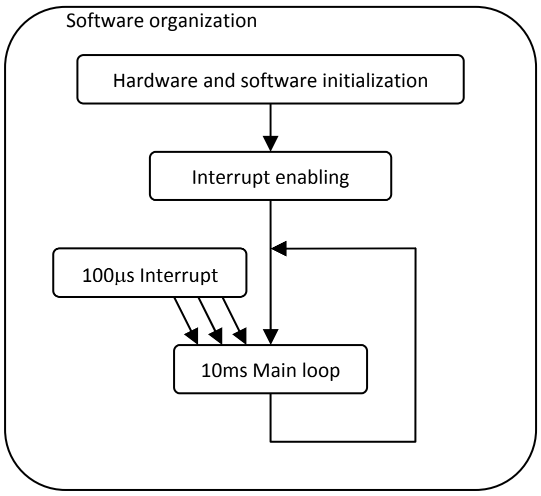

Sensorless FOC algorithm
(PI, flux estimation, filters, speed estimation, modulation routines...)
Transformations (αβ), (u,v,w), (d,q)
Shunt current measurement
Time delay management
Trigonometric functions
PMW generation MTU2 handling
Communication layer
A/D converter routines
E²PROM handling layer
Interrupts vector table initi.
HW setup layer SFRs initi.

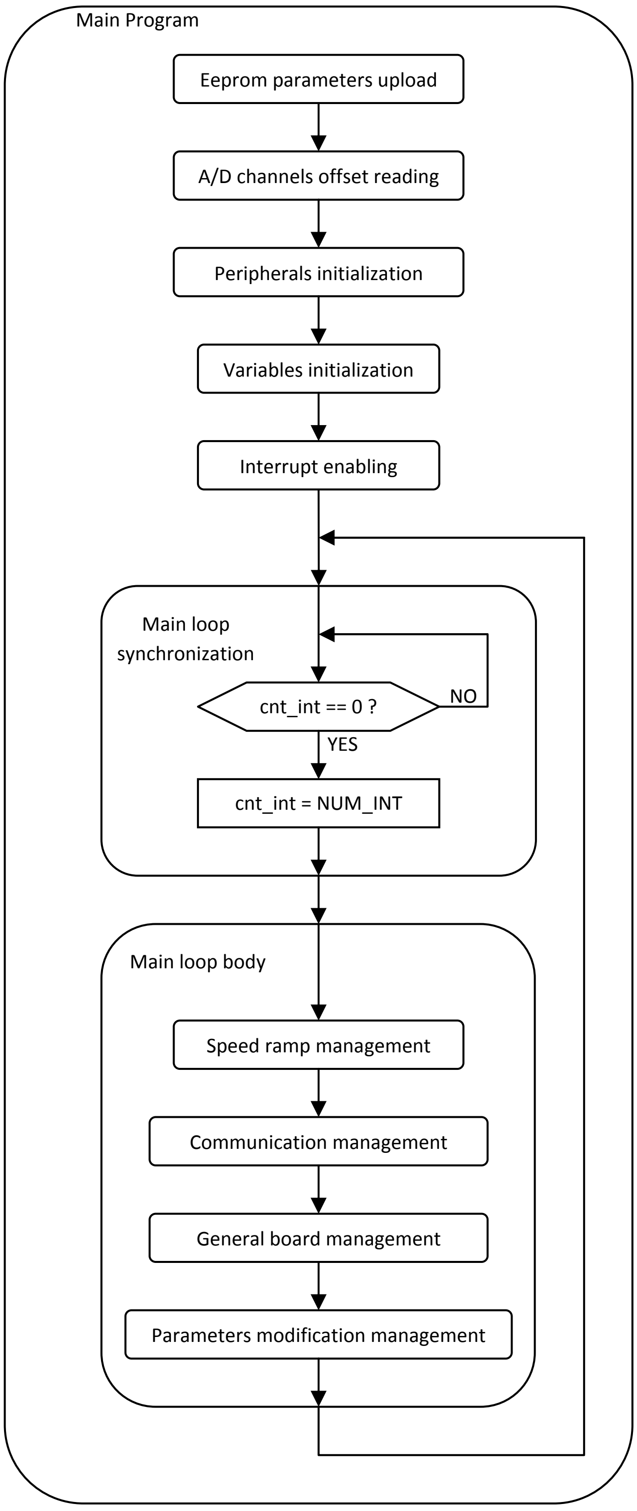
The total software uses the following resources:

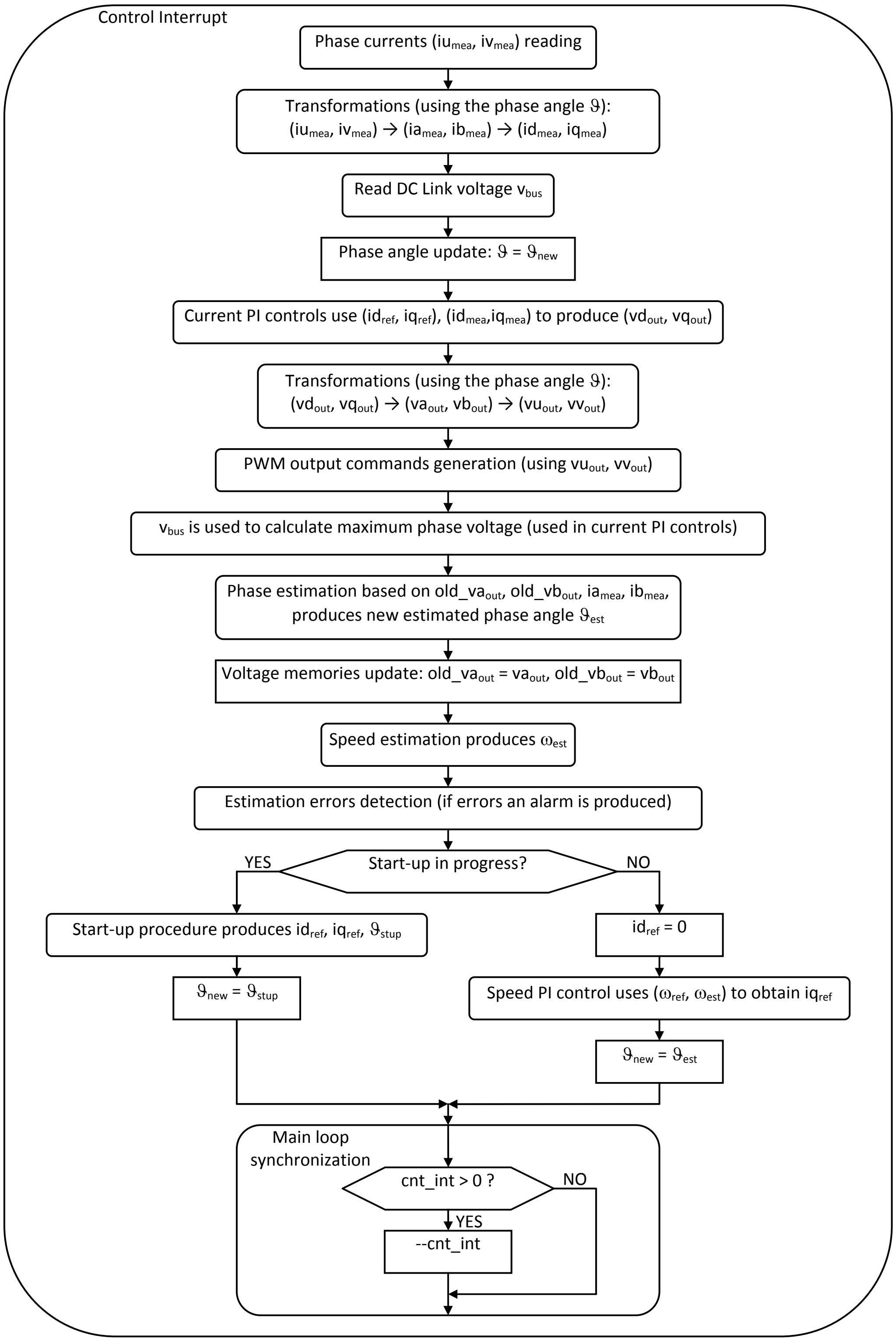
1) FLASH	:	< 18Kbytes
2) RAM	:	< 3Kbytes

Pls. Note that these data include also the communication interface and the demo board management.

The following flow charts show the software implementation of the motor control part of the software




Main Program
Eeprom parameters upload
A/D channels offset reading
Peripherals initialization
Variables initialization
Interrupt enabling
Main loop synchronization
cnt_int == 0 ?
NO
YES
cnt_int = NUM_INT
Main loop body
Speed ramp management
Communication management
General board management
Parameters modification management

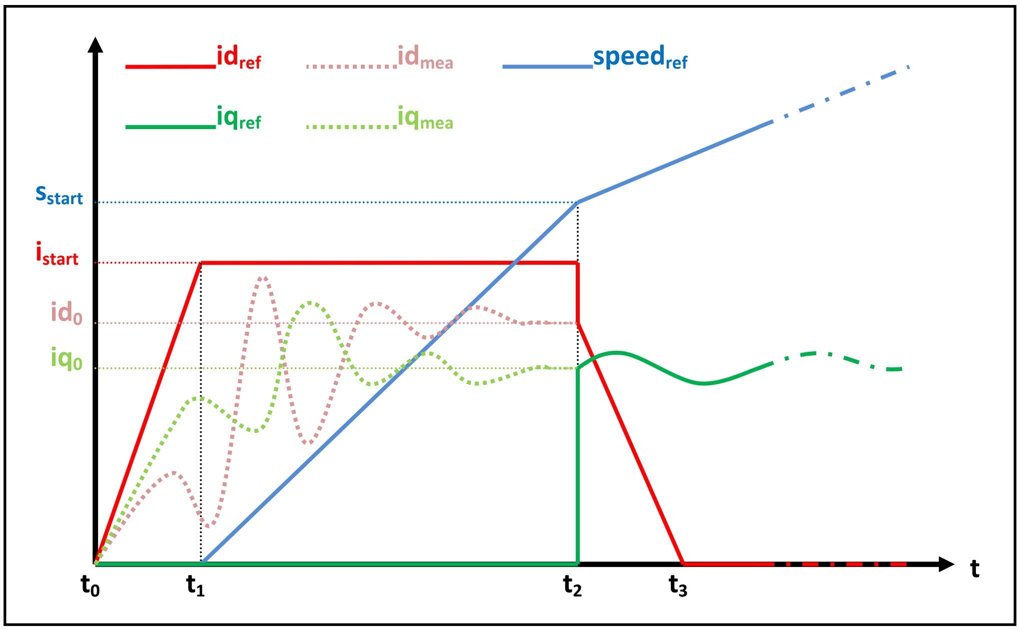

Control Interrupt
Phase currents (iu_{mea}, iv_{mea}) reading
Transformations (using the phase angle ϑ): (iu_{mea}, iv_{mea}) → (ia_{mea}, ib_{mea}) → (id_{mea}, iq_{mea})
Read DC Link voltage v_{bus}
Phase angle update: $\vartheta = \vartheta_{new}$
Current PI controls use (id_{ref}, iq_{ref}), (id_{mea},iq_{mea}) to produce (vd_{out}, vq_{out})
Transformations (using the phase angle ϑ): (vd_{out}, vq_{out}) → (va_{out}, vb_{out}) → (vu_{out}, vv_{out})
PWM output commands generation (using vu_{out}, vv_{out})
v_{bus} is used to calculate maximum phase voltage (used in current PI controls)
Phase estimation based on old_va_{out}, old_vb_{out}, ia_{mea}, ib_{mea}, produces new estimated phase angle ϑ_{est}
Voltage memories update: $old\_va_{out} = va_{out}$, $old\_vb_{out} = vb_{out}$
Speed estimation produces ω_{est}
Estimation errors detection (if errors an alarm is produced)
Start-up in progress?
YES
NO
Start-up procedure produces id_{ref}, iq_{ref}, ϑ_{stup}
$\vartheta_{new} = \vartheta_{stup}$
$id_{ref} = 0$
Speed PI control uses (ω_{ref}, ω_{est}) to obtain iq_{ref}
$\vartheta_{new} = \vartheta_{est}$
Main loop synchronization
cnt_int > 0 ?
NO
YES
--cnt_int

Start-up procedure

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 has to 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*).



Referring to the graph, the startup 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 the 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.

Reference system transformations in details

Find below the detailed equations used for the coordinates transformations.

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

$(u, v, w) \rightarrow (\alpha, \beta)$

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

$(\alpha, \beta) \rightarrow (u, v, w)$

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

$(\alpha, \beta) \rightarrow (d, q)$

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

$(d, q) \rightarrow (\alpha, \beta)$

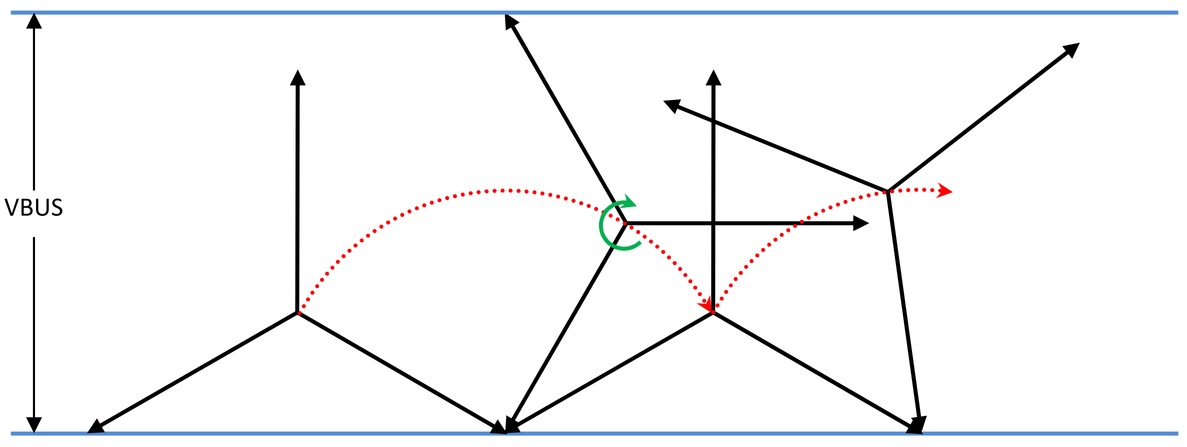
$$\begin{Bmatrix} v_u = V\cos(\omega t + \varphi_0) \\ v_v = V\cos(\omega t + \varphi_0 - 2\pi/3) \\ v_w = V\cos(\omega t + \varphi_0 - 4\pi/3) \end{Bmatrix} \leftrightarrow \begin{Bmatrix} v_\alpha = V\cos(\omega t + \varphi_0) \\ v_\beta = V\sin(\omega t + \varphi_0) \end{Bmatrix} \leftrightarrow \begin{Bmatrix}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

PWM modulation technique

Among the various possibilities, a particular form of PWM modulation was chosen. In this modulation technique, the voltages to be imposed are shifted in order to have in every moment one of the three phases of the motor connected to the system ground. This allows reducing the commutations of the power bridge of one third, in respect to other modulation techniques. In fact the phase that is connected to the system ground doesn't require any commutation, having the lower arm always on and the upper arm always off.

The method is based on the fact that, having no neutral connection, we are interested only in phase-to-phase voltages, or in the voltage differences between the phases, not in the voltage level of the single phases. This allows us to add or subtract an arbitrary quantity to the phase voltages, on condition that this quantity is the same for all the three phases. So, obtained from the algorithm the three phase voltages requests, the minimum is chosen and it is subtracted to all the three requests.

With this method, the applied voltage star centre is not at a fixed level, but it is moving.



The maximum phase-to-phase voltage that can be obtained (without distortion of the sinusoidal waveform) with this method is equal to the DC Link voltage, as in other methods (like Space Vector Modulation).

List of variables used in "motorcontrol.c"

The file "motorcontrol.c" includes the motor control algorithm routines. Please find below the description of the variables used.

Label(s)	**Type**	**Description**	**Unit**
ium_off, ivm_off, iwm_off	float	A/D conversion offsets of measured u, v, w phase currents; the value is around 2048, that corresponds to one half of the A/D converter supply voltage (5Vdc) (12bit A/D).	
vol_ref	float	A/D conversion result of the reading of the reference voltage (4.25V); used for compensate the effects of the power supply variations in the A/D conversions; the ideal value is 870 (10bit A/D), if the A/D converter supply voltage is exactly 5V.	
kadi, kadv	float	Current and voltage conversion constants; they are corrected on the grounds of vol_ref, and they are used to convert the A/D results in the used measurement units; multiplying the A/D result by the conversion constant, the current (voltage) in Ampere (Volt) is obtained (ex.: iu=kadi*(iuad-ium_off))	
r_sta	float	Stator resistance	ohm
l_sync	float	Synchronous inductance	henry
c_poli	float	Number of polar couples	
krpmocp, ukrpmocp	float	Conversion constant between mechanical speed and electrical speed, and its reciprocal (ukrpmocp=1/krpmocp).	(rad/s)/rpm, rpm/(rad/s)
vstart	float	Startup voltage in single shunt operation; during startup, first a voltage ramp at zero speed is performed, then a voltage and speed ramp; vstart is the actual value.	volt
vs_off	float	Offset startup voltage in single shunt operation; vs_off is the total starting value (total voltage at zero speed).	volt
vs_inc	float	It is the quantity added at every zero speed ramp step to obtain vs_off.	volt

Label(s)	**Type**	**Description**	**Unit**
vs_del	float	Total voltage quantity added during startup in single shunt operation; added to vs_off, it gives the voltage applied when the voltage and speed ramp is finished.	volt
vs_dela	float	Voltage quantity added at every voltage and speed ramp step during startup in single shunt operation.	volt
istart	float	Startup current in three shunts operation; during startup, first a current ramp at zero speed is imposed, then a speed ramp with constant current (istart).	ampere
is_inc	float	Startup current increment at every step.	ampere
omegae_s	float	Electrical speed during startup (instant value)	rad/s
delta_om	float	Speed quantity added at every step during startup ramp.	rad/s
om_chg	float	Speed to reach during the startup; when this speed is reached, the startup ramp ends.	rad/s
startup_phase	float	Electrical phase during startup.	rad
delta_ph	float	Phase variation at every step during startup.	rad
vdx, vqx, vdxf, vqxf	float	D and q axis voltages (instant and filtered) during startup.	volt
idx, iqx, idxf, iqxf	float	D and q axis currents (instant and filtered) during startup.	ampere
SystemPhase	float	Imposed electrical phase.	rad
Phase_est	float	Estimated electrical phase.	rad
vbus, vbusf	float	DC link voltage, instant value and filtered one.	volt
xvbf	float	DC link voltage, minimum ripple value, used for voltage clamping.	volt
vfmax	float	Maximum allowed phase voltage (star).	volt
vdmax, vdmax	float	Maximum d and q axis allowed voltages.	
i_max, iq_max	float	Maximum allowed total current, maximum allowed q axis current.	ampere
vdc, vqc, vdcf, vqcf	float	D and q axis imposed voltages, instant and filtered values.	volt

Label(s)	Type	Description	Unit
vac, vbc	float	Alpha and beta axis voltages.	volt
vuc, vvc, vwc	float	Phase voltages (star).	volt
old_va, old_vb	float	Previous step alpha and beta axis voltages.	volt
ium, ivm, iwm	float	Measured phase currents.	ampere
iam, ibm	float	Measured alpha and beta axis currents.	ampere
idm, iqm, idmf, iqmf	float	Measured d and q axis currents (instant and filtered values).	ampere
idr, iqr	float	D and q axis reference currents.	ampere
id_dec	float	After the startup, the d axis current residual is decreased till zero; id_dec is the variation at every step.	ampere
idint, iqint	float	Current PI integral memories.	volt
idimem, iqimem	float	Current PI integral memories; this values are used in single shunt operation to stop the integral action when the current reading is not possible.	volt
errint	float	Speed PI integral memory	ampere
kp_cur, ki_cur	float	Proportional and integral constant in current PI controllers.	volt/ampere
kp_vel, ki_vel	float	Proportional and integral constant in speed PI controller.	ampere/(rad/s)
freq	float	Electrical frequency.	hertz
mec_rpm	float	Mechanical speed.	rpm
rpmrif_x	float	Reference speed (speed ramp input value).	rpm
rpmrif_y	float	Reference speed (speed ramp output value).	rpm
rpmrif_abs	float	Absolute value of rpmrif_y.	rpm
r_acc, r_dec	float	Acceleration ramp, deceleration ramp.	rpm/s
rpm_min, rpm_max	float	Minimum and maximum allowed speed.	rpm
min_speed, max_speed	float	Minimum and maximum electrical speed.	rad/s

Label(s)	**Type**	**Description**	**Unit**
min_speed_trip, max_speed_trip	float	Minimum and maximum electrical speed (values used for estimation error detection).	rad/s
Speed_est	float	Estimated electrical speed.	rad/s
omrif, f_omrif	float	Reference electrical speed (instant and filtered values).	rad/s
omegae, omegae_f, omf	float	Imposed electrical speed (instant and filtered values).	rad/s
maxerr	float	Maximum electrical speed error.	rad/s
vbus_ulpkt_slow, vbus_ulpkt_fast	float	One divided by K, where K is the time constant of the vbus low-pass filter (slow and fast).	1/s
speedref_ulpkt	float	One divided by K, where K is the time constant of the speed reference low-pass filter.	1/s
startup_ulpkt	float	One divided by K, where K is the time constant of the startup low-pass filter.	1/s
off_ulpkt	float	One divided by K, where K is the time constant of the current offsets low-pass filter.	1/s
vr_ulpkt	float	One divided by K, where K is the time constant of the board reference voltage low-pass filter.	1/s
duty_u, duty_v, duty_w	signed short	PWM duty cycles for the three phases.	MTU pulses
vbus_ad	signed short	A/D conversion result of the DC link voltage reading.	
iss_off	signed short	A/D conversion offsets of measured single shunt current; the value is around 2048, that corresponds to one half of the A/D converter supply voltage (5Vdc) (12bit A/D).	
iaad, ibad	signed short	A/D conversion result of the first and the second single shunt current reading.	
deadtim	unsigned short	Dead-time.	MTU pulses
semiper	unsigned short	PWM half period.	MTU pulses
semiperdead	unsigned short	PWM half period plus dead-time.	MTU pulses

Label(s)	**Type**	**Description**	**Unit**
cr_ss	unsigned short	Status variable for single shunt current reading.	
trip_cnt	unsigned short	Counter for estimation error detection.	
startup_cnt	unsigned short	Counter for startup.	
startup_val	unsigned short	Startup time.	N° of sampling periods
stp_tim	unsigned short	Startup time.	ms
XXXXXX_ep	unsigned short	Many variables with suffix “_ep”: they are copies of various parameters, used for eeprom management.	
enc_ind	unsigned short	Index in encoder filter table.	
enc_sam	unsigned short	Encoder sample.	
enc_ang	unsigned short	Encoder angular position.	2PI is 65536
mec_ang	float	Mechanical position.	rad
ele_ang	float	Electrical angular position.	rad
off_ang	float	Electrical position offset.	rad
tele_ang	float	Corrected electrical position.	rad
om_mec	float	Mechanical angular speed.	rad/s
om_eme	float	Electro-mechanical angular speed.	rad/s
enc_buf[]	float	Encoder filter buffer.	rad

Cautions

Please note that the use of the demonstration board software is subjected to the following disclaimer:

```
/********************************************************************************
*      DISCLAIMER
*
*      THIS SOFTWARE IS PROVIDED "AS IS" AND BFG ENGINEERING MAKES NO WARRANTIES
*      REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING
*      BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
*      PURPOSE AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY DISCLAIMED.
*
*      WE (BFG ENGINEERING) DO NOT WARRANT THAT THIS SOFTWARE IS FREE FROM CLAIMS
*      BY A THIRD PARTY OF COPYRIGHT, PATENT, TRADEMARK, TRADE SECRET OR ANY OTHER
*      INTELLECTUAL PROPERTY INFRINGIMENT.
*
*      TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, BFG ENGINEERING
*      SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR
*      CONSEQUENTIAL DAMAGES FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF BFG
*      ENGINEERING HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*      PARTICULARLY, UNDER NO CIRCUMSTANCES ARE WE LIABLE FOR ANY OF THE FOLLOWING:
*      - THIRD PARTY CLAIMS AGAINST YOU FOR LOSSES OR DAMAGES;
*      - LOSS OF, OR DAMAGE TO YOUR RECORDS OR DATA; OR
*      - ECONOMIC CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) OR
*        INCIDENTAL DAMAGES, EVEN IF WE ARE INFORMED OF THEIR POSSIBILITY.
*
*      BFG ENGINEERING DOES NOT WARRANT UNINTERRUPTED OR ERROR_FREE OPERATION OF
*      THIS SOFTWARE.
*      BFG ENGINEERING HAS NO OBLIGATION TO PROVIDE SERVICE, DEFECT CORRECTION OR
*      ANY MANTEINANCE OF THIS SOFTWARE; OR TO SUPPLY ANY UPDATES OR ENHANCEMENTS
*      EVEN IF SUCH ARE, OR LATER BECOME, AVAILABLE.
*
*      IF YOU DOWNLOAD OR USE THIS SOFTWARE YOU AGREE TO THESE TERMS, AND DO IT AT
*      YOUR OWN RISK.
********************************************************************************/
```

The same disclaimer has to be applied also to the whole documentation, which is NOT guaranteed to be error-free; we decline any responsibility regarding the use that can be done with this manual and with the rest of the documentation included in the demo board package.