

YROTATE-IT-RX23T

UM-YROTATE-IT-RX23T

Rev. 1.0

Rotate it! – Motor Control RX23T

October 6, 2015

Introduction

The Renesas Motor Control Kit called YROTATE-IT-RX23T, is based on the RX23T device from the powerful 32-bit RX microcontrollers family running at 40MHz and delivering up to 80DMIPs.

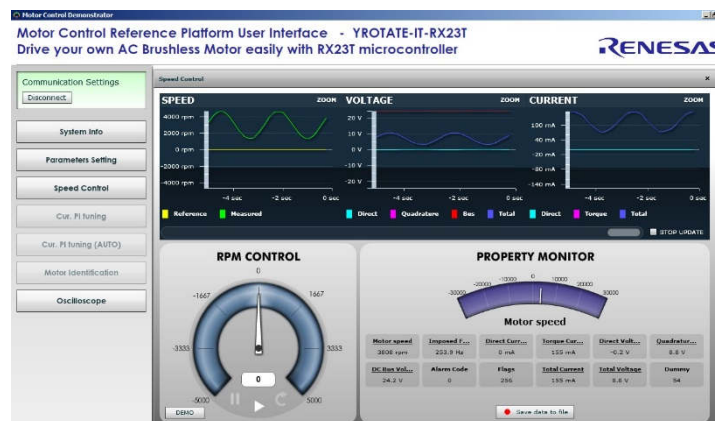
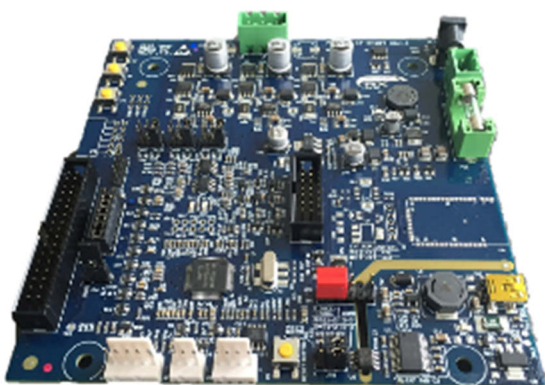
The kit enables engineers to easily test and evaluate the performance of the RX23T in a laboratory environment when driving any 3-phase Permanent Magnet Synchronous Motor (e.g. AC Brushless Motor) using an advanced sensorless Field Oriented Control algorithm. Typical applications for this type of solution are compressors, air conditioning, fans, air extractors, pumps, home appliances inverters and industrial drives.

The 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. A single shunt current reading method is also available to ensure an even more compacter bill of material.

The powerful user-friendly PC Graphical User Interface (GUI) gives real time access to key motor performance parameters and provides a unique motor auto-tuning facility. Furthermore, it becomes also possible to select the best switching frequency and control frequency (e.g. control loop) to adapt the control dynamics suitable to the application requirements.

The hardware is designed for easy access to key system test points and for the ability to hook up to an RX23T debugger known as E1. Although the board is normally powered directly from the USB port of a Host PC, connectors are provided to utilise external power supplies where required.

The YROTATE-IT-RX23T is an ideal tool to check out all the key performance parameters of your selected motor, before embarking on a final end application system design.



Target Device: RX23T Microcontrollers Family

Contents

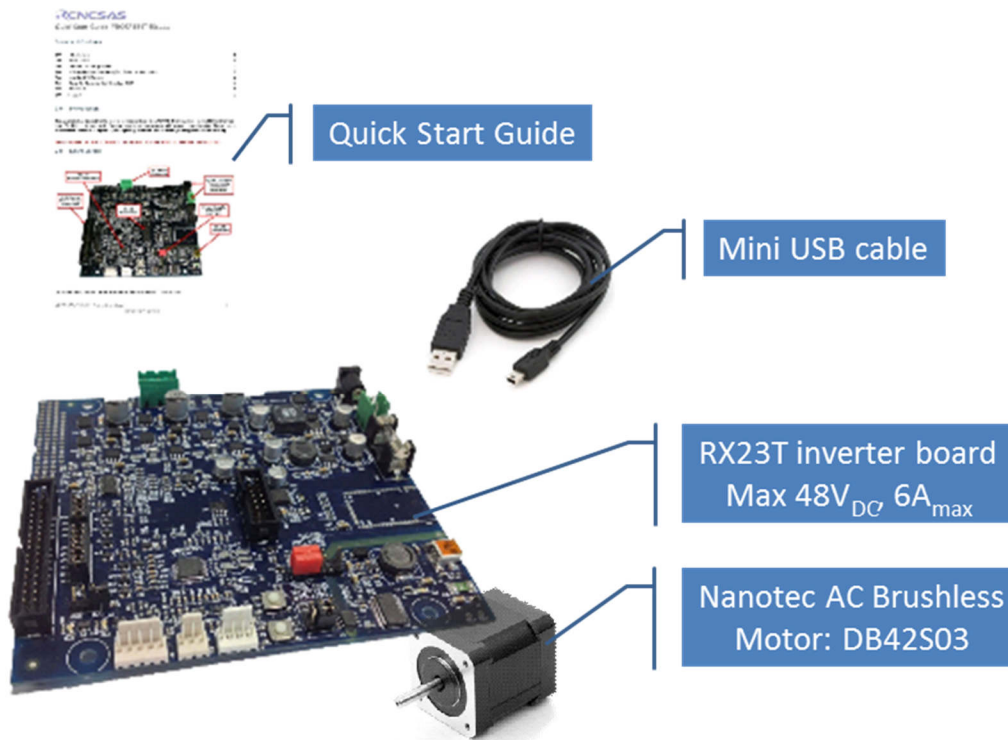
1. Specifications & Hardware overview	3
2. Connectors description	7
3. Power supply selection.....	8
4. LEDs functions description.....	10
5. Test points for debugging	11
6. Internal power stage description	12
7. Current reading methods selection	14
8. Interface to an external power stage	16
9. Single shunt current reading	20
10. Current reading timing in three shunts and single shunt configurations	21
11. Website material: www.renesas.eu/motorcontrol	24
12. Microcontroller RX23T short overview	25
13. Permanent Magnets Brushless Motor model.....	27
14. Sensorless Field Oriented Control algorithm	32
15. Software Tools used	33
16. Software description and Resources used.....	37
17. Start-up procedure – Embedded software	44
18. Reference system transformations in details	46
19. Rotor position estimation.....	47
20. PC Graphical User Interface in details	50
21. EEPROM parameters: detailed description	56
22. Motor Auto-calibration using the PC GUI.....	57
23. Updating the RX23T Flash memory using the Renesas Programming Flash Tool.....	68
24. Integration of user code to the motor control algorithm	73
25. Communication Protocol between the MCU and the PC GUI	74

1. Specifications & Hardware overview

ITEM	SPECIFICATIONS
TYPE OF MOTORS SUPPORTED	3-phase Permanent Magnet Synchronous (PMSM, PMAC, BLAC) 3-phase Brushless DC (BLDC)
KIT MOTOR PARTNAME	NANOTEC DB42S03, 24V _{DC} , 4000 RPM
KIT MAX INPUT RANGE	External power supply from: 20V _{DC} to 48V _{DC} , 6A _{peak}
TRANSISTOR USED	Renesas Mosfets: RJK0654DPB, 60V, 30A
POWER SUPPLY OPTION	Either USB connection or external supply: up to 48V _{DC}
CURRENT DETECTION	One or three shunts configuration (10mΩ)
USB IC USED ON THE BOARD	FT232R - USB UART IC from FDTI, 76.6Kbd communication speed
MICROCONTROLLER	RX23T (R5F523T5ADFM), 64-pin LQFP, 40MHz, 64KB Flash, 10KB RAM
MCU PERFORMANCE	40MHz, 80DMIPs, 166 CoreMark
KEY FEATURES	Floating Point Unit, 3-phase inverter Timer 12-bit A/D Converter, fast on-chip Comparators, Port Output Enable
MCU EMBEDDED FIRMWARE	Sensorless vector control algorithm (Field Oriented Control)
SWITCHING FREQUENCY	4KHz to 64KHz, 16KHz by default (PWM frequency)
CONTROL LOOP FREQUENCY (SAMPLING FREQUENCY)	4KHz to 16KHz, 8KHz by default
CONTROL LOOP TIMING	50μs including debug features and auto-tuning/self-identification 40μs including only the sensorless vector control algorithm
CODE SIZE IN FLASH / RAM	24KB / 3KB
TOOL USED, VERSION	e ² studio 4.0.2.008. , CS+ , RXC Toolchain, CC RX version v2.03.00
COMPILER OPTIMIZATION LEVEL	Maximum, optimize for speed, little-endian data, RX V2 Core enable
ENVIRONMENT STANDARDS	RoHS compliant including China regulations WEEE, RoHS

The inverter kit YROTATE-IT-RX23T is a single board inverter, based on the 32-bit RX series microcontroller RX23T and includes a low-voltage MOSFETs power stage and a communication stage. The PCB is a four layers board and ensure the management of Permanent Magnet Motors up to $48V_{DC}$ and up to $6A_{max}$.

Please find below the content of the YROTATE-IT-RX23T kit:

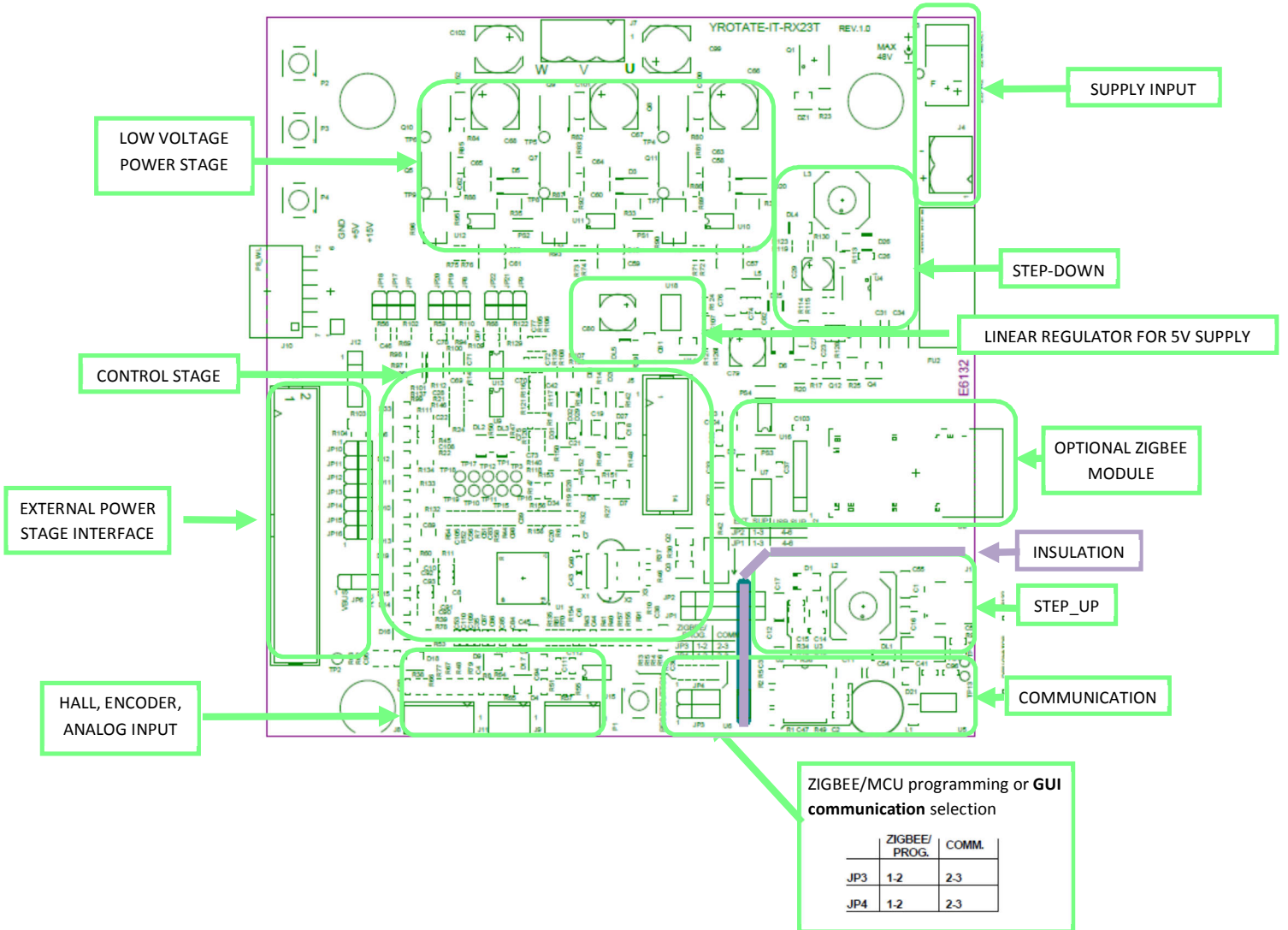


To obtain the maximum flexibility, the inverter reference kit includes:

- A complete 3-phase inverter on-board with a low voltage motor, so it becomes easy to test the powerful sensorless algorithm running on the Renesas 32-bit RX23T microcontroller (e.g. MCU).
- An insulated USB communication with the PC offering communication speed up to 76.6KBd.
- Connectors for hall sensors and encoder connections. Both encoder and hall sensors are not managed in the sensorless software but they can be supported under request.
- Compatibility with the existing Motor Control Reference Platform external power stages, the first delivering up to 1.5KW at $230V_{AC}$ and the second up to $60V_{DC}$, $60A_{DC}$.
- USB power supply possibility to avoid external power supplies where galvanic insulation is lost.

To achieve these aims, three different DC-DC converters are used:

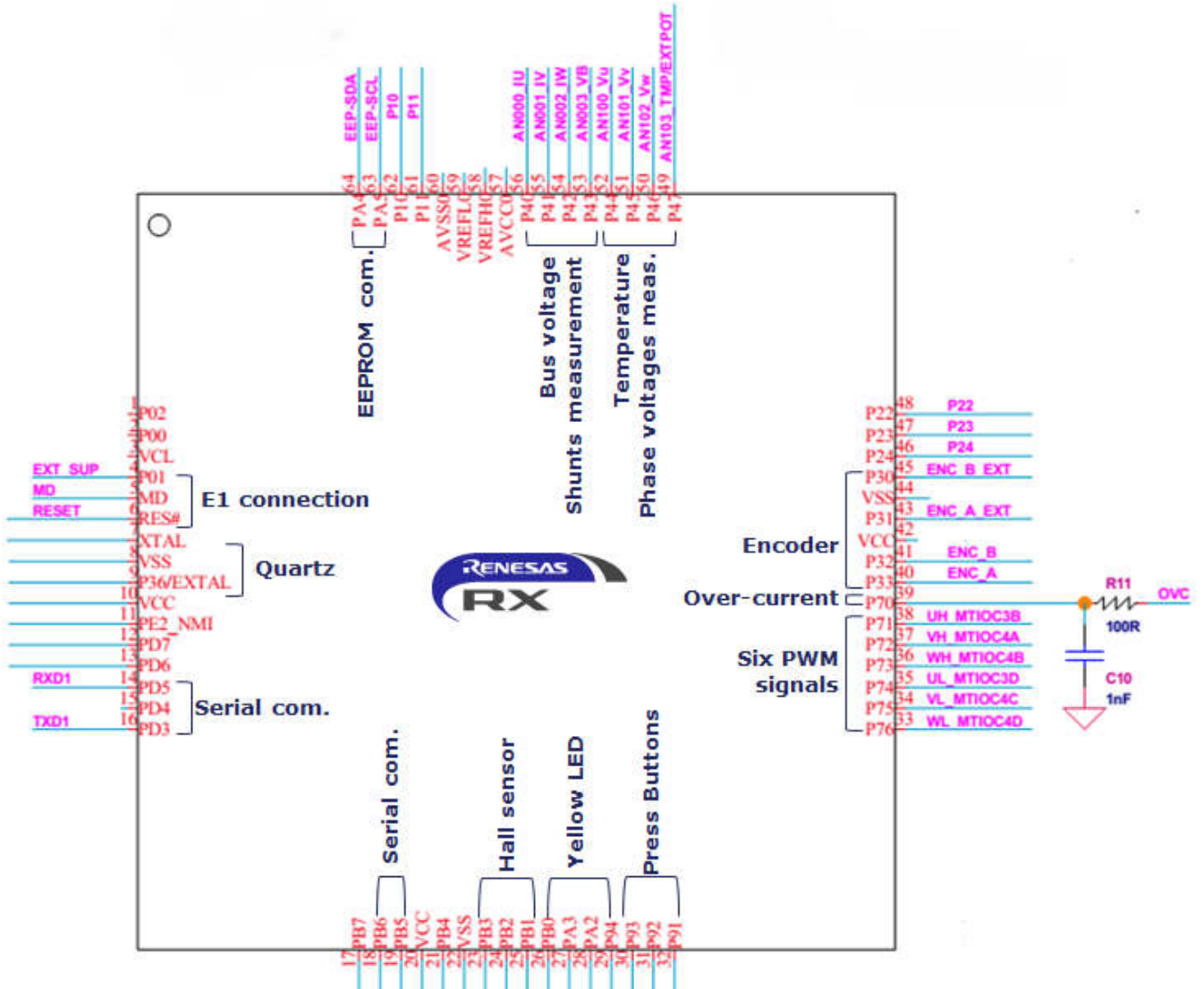
1. A step-up DC-DC converter to increase the voltage from the USB standard (5V) up to 13.5V_{DC}
2. A step-down converter and a low dropout linear converter, from the DC bus first to 15V and then to the CPU supply voltage: 5V, please find below the PCB overview.



The full schematics and CAD design files (e.g. Gerber) of the inverter kit are available on the website: www.renesas.eu/motorcontrol, in the section related to the YROTATE-IT-RX23T development kit.

In the YROTATE-IT-RX23T kit, the RX23T in a 64-pin package was selected to ensure the management of inverter, external communications, three shunts, the EEPROM communication, the E1 debugger, three voltages phases, the over-current detection, the Bus voltage monitoring, etc.

The picture below is showing the detailed I/O pins assignment of the RX23T to manage the complete kit.

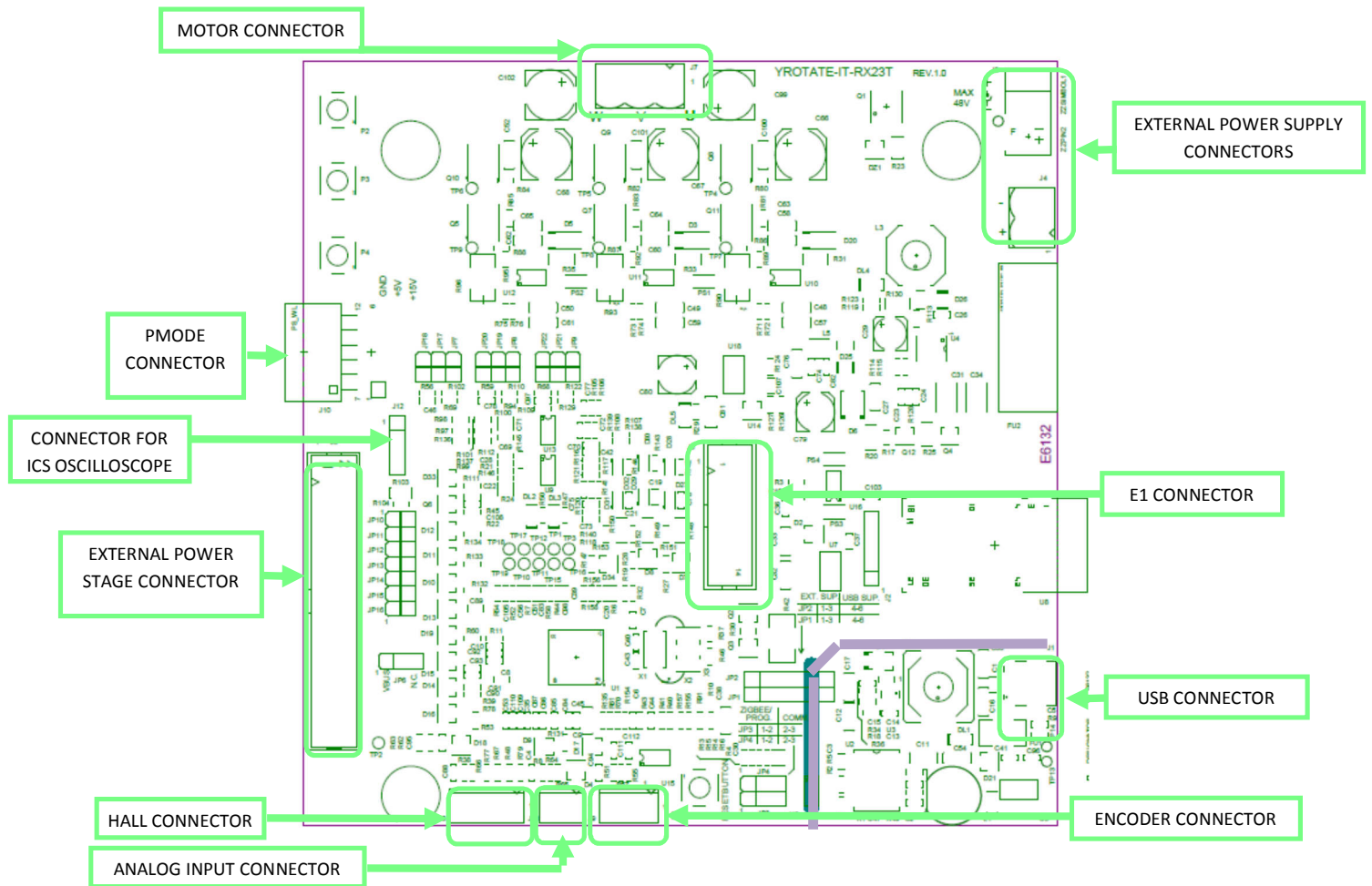


2. Connectors description

As in the following figure, you can find the position and the description of the connectors present on the board. Please refer to the board schematics for the full description of the connectors.

The E1 connector is used for the programming and the debugging of the software running on the RX23T. It can be connected either to the E2studio and the CS+ development environments.

The external power stage connector is compatible with the power stages, designed for Renesas inverter kits, which are able, the first one to drive 230V_{AC} motor up to 1.5KW, and the second one up to 60V_{DC}, 60A_{DC}. The schematics and Gerber file of the power stage are available on the website: www.renesas.eu/motorcontrol

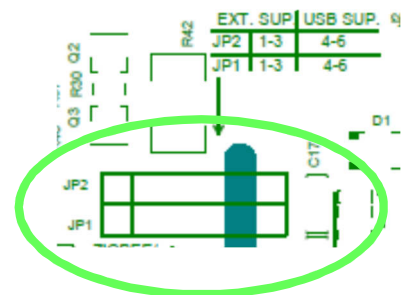
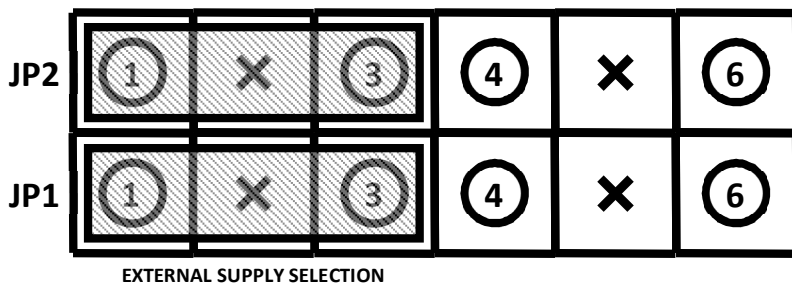
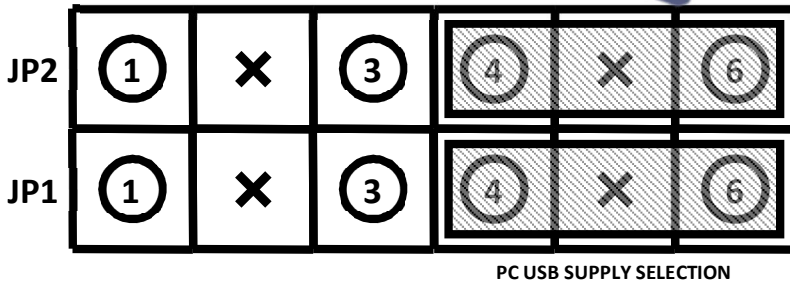
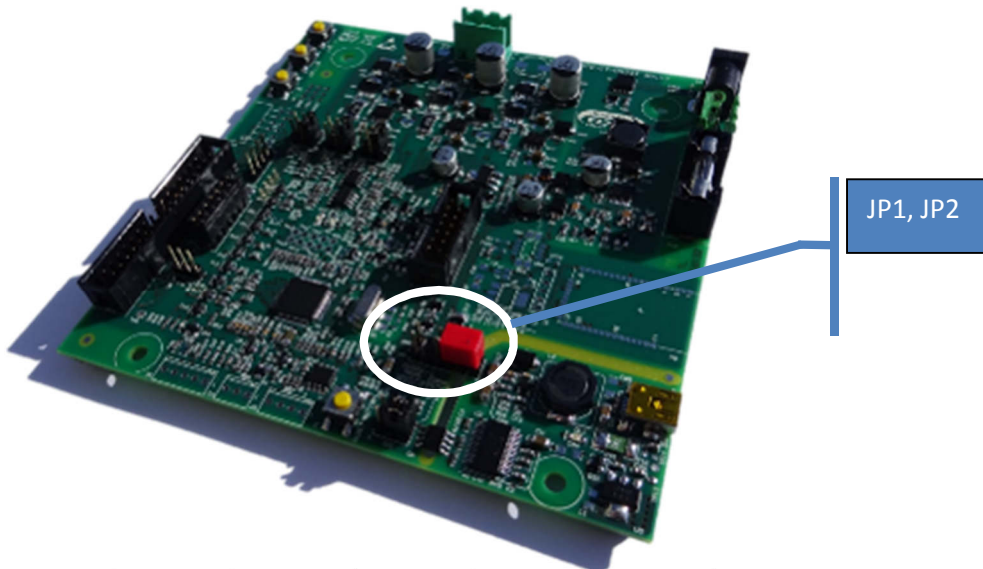


3. Power supply selection

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

1. The first possibility is to use directly the PC USB supply. In this case the current you can give to the motor is limited by the USB current capabilities.
2. The second possibility is to use an external voltage DC source to supply the board.

The recommended power supply voltage is between **20V_{DC}** to **48V_{DC}**. In this case the communication stage is insulated from the inverter. The selection between the two possibilities is made through two jumpers: **JP1** and **JP2**. Please find below the description:

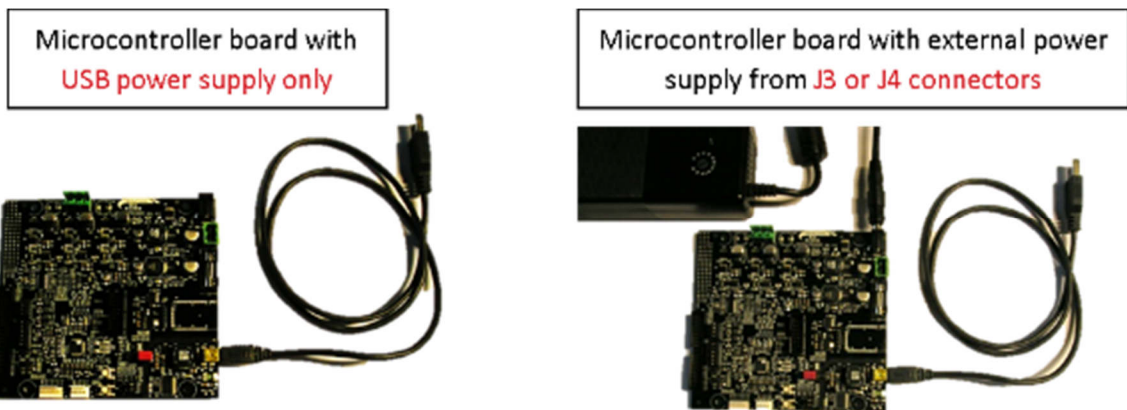


- 1) The first jumper configuration connects the USB ground to the inverter ground and the output of the step-up converter to the inverter DC link.

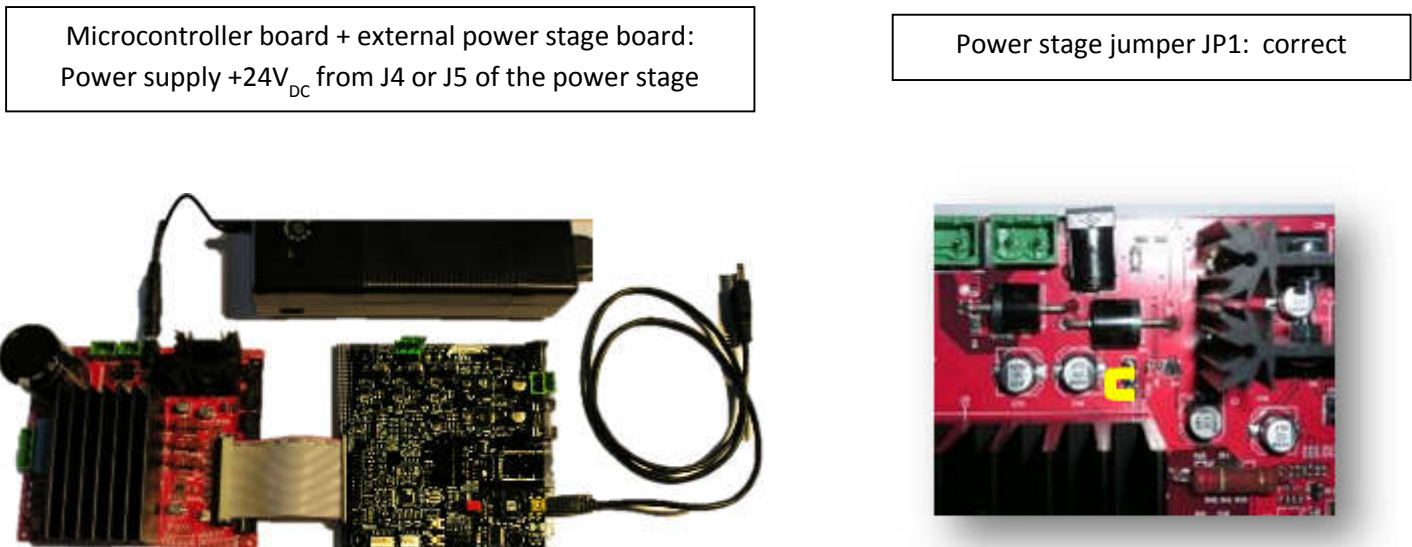
Please notice that in this case there is no galvanic insulation between the device connected to the USB and the board.

- 2) The second jumper configuration connects the external power supply ground to the inverter ground and the external + V_{DC} to the inverter DC link.

Please find below the configuration of the RX23T kit using the internal MOSFETs power stage.



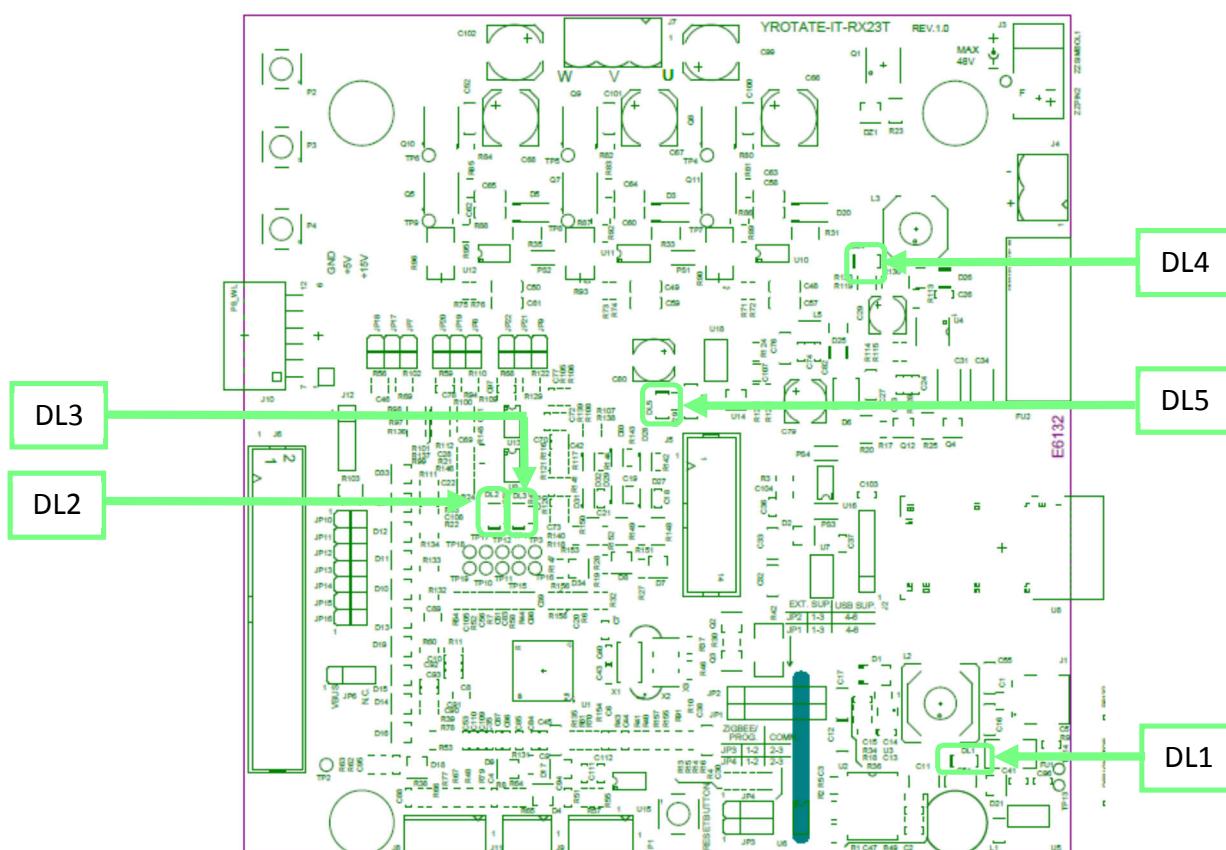
Please find below the configuration of the RX23T kit connected to the external 1.5KW, 230V_{AC} power stage.



4. LEDs functions description

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

- DL4 is connected to the output of the 15V step-down DC-DC converter and indicates the presence of the switches drive supply;
- DL5 is connected to the output of the 5V linear converter and indicates the presence of the 5V logic power supply.



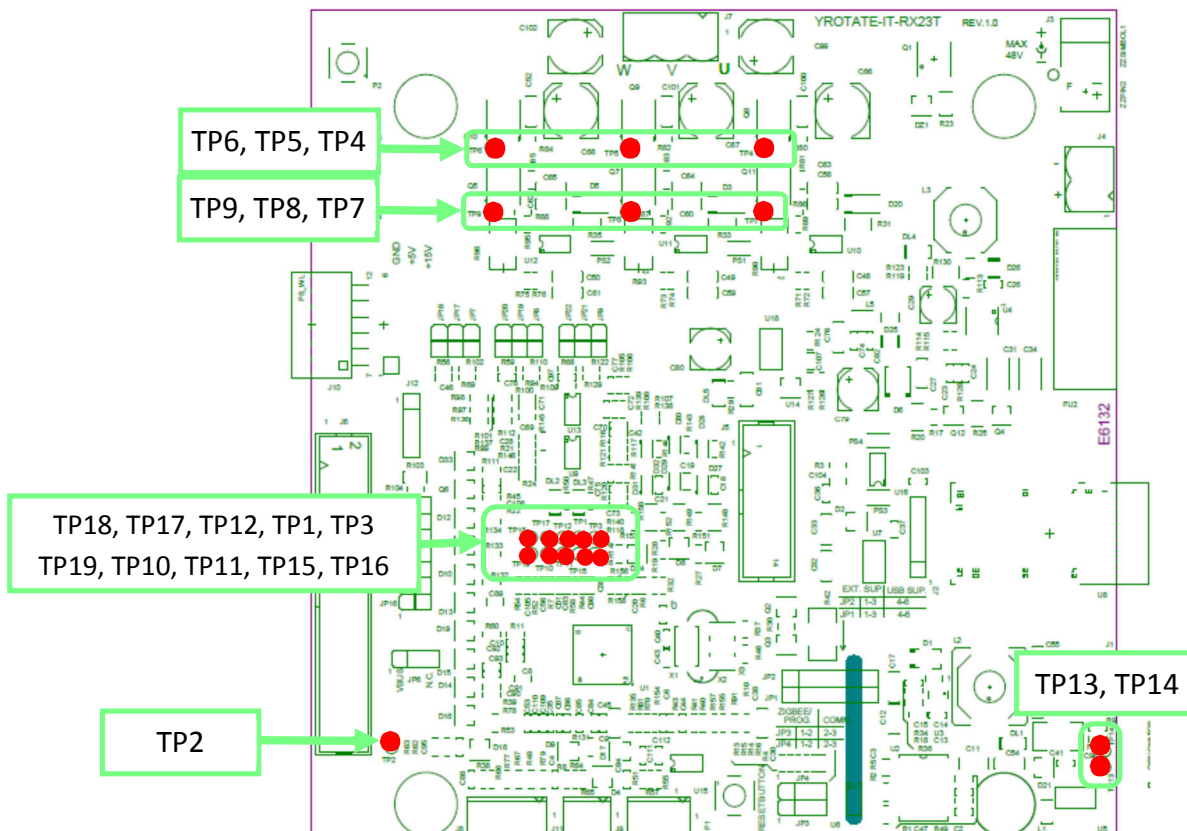
Other LEDs in the board are driven via software, in particular:

- DL1 is the USB communication indicator and blinks when there are data exchanges between the PC and the board.
- DL2 is blinking slowly if the control section of the RX23T microcontroller is running normally. In case of hardware or software alarms, the LED DL3 is blinking quickly.
- DL3 is free for the user and in the default software it is ON when the main control interrupt is active. It give to the user a quick and simple way to measure the timing of the control loop of the algorithm.

5. Test points for debugging

Several specific test points are available on the board to visualize with the oscilloscope the behaviour of some internal analog signals.

Furthermore, it is possible to visualize internal variables as analog waveforms using filtered PWM outputs. Finally, it is very useful during the tuning process for adapting the software to a new motor to use the test points.

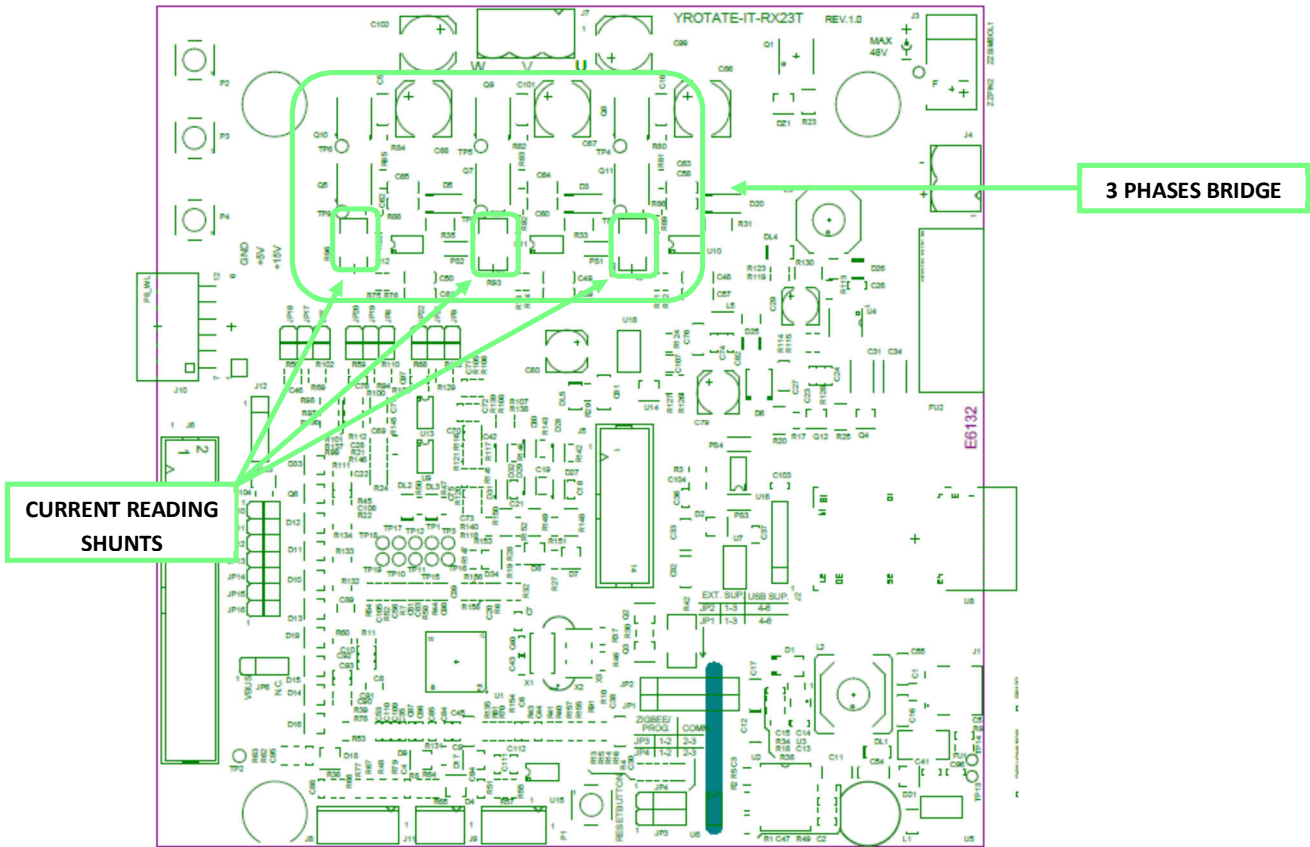


Please find below the description of the test points:

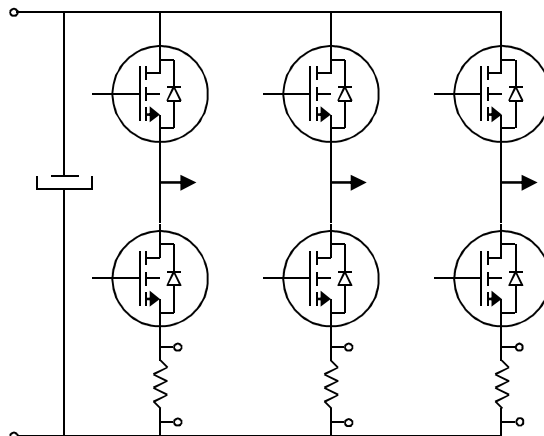
- **TP13, TP14:** are connected to the two USB communication signals, for debug purposes. Please refer to the board schematics for more details.
- **TP4, TP5, and TP6:** they are connected to the three inverter outputs (sources of the higher switches)
- **TP7, TP8, and TP9:** they are connected to the sources of the lower switches of the inverter
- **TP1, TP3, TP12, TP15, TP16:** they are connected to some microcontroller ports
- **TP17, TP18, TP19 are connected to the board GND**
- **TP10, TP11** are two filtered PWM outputs which can be used to visualize the behaviour of internal variables
- **TP2 not used**

6. Internal power stage description

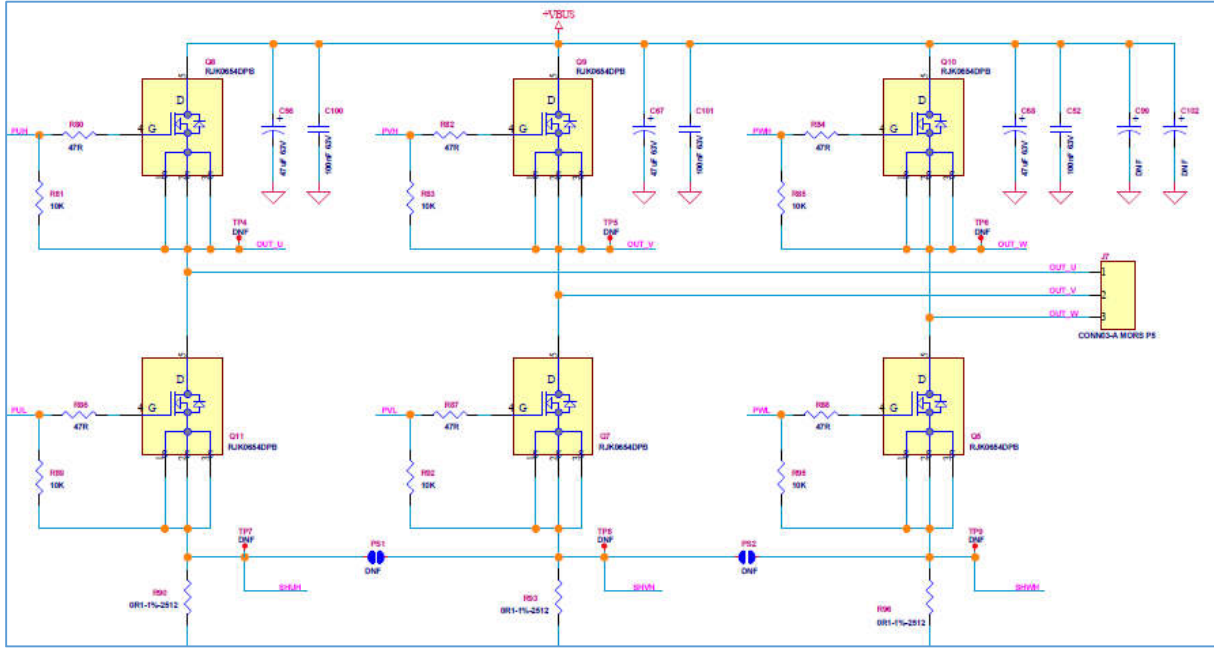
The power stage is a complete 3-phase bridge composed with discrete low voltage and high current MOSFETs. The MOSFETs are the Renesas **RJK0654DPB** n-channel power MOSFETs. Please refer to the data-sheet available on the Renesas website: www.renesas.eu for the switches characteristics and to the board schematics for the details on the driving circuit. The maximum current is **30A**, and the maximum voltage is **60V_{DC}**.



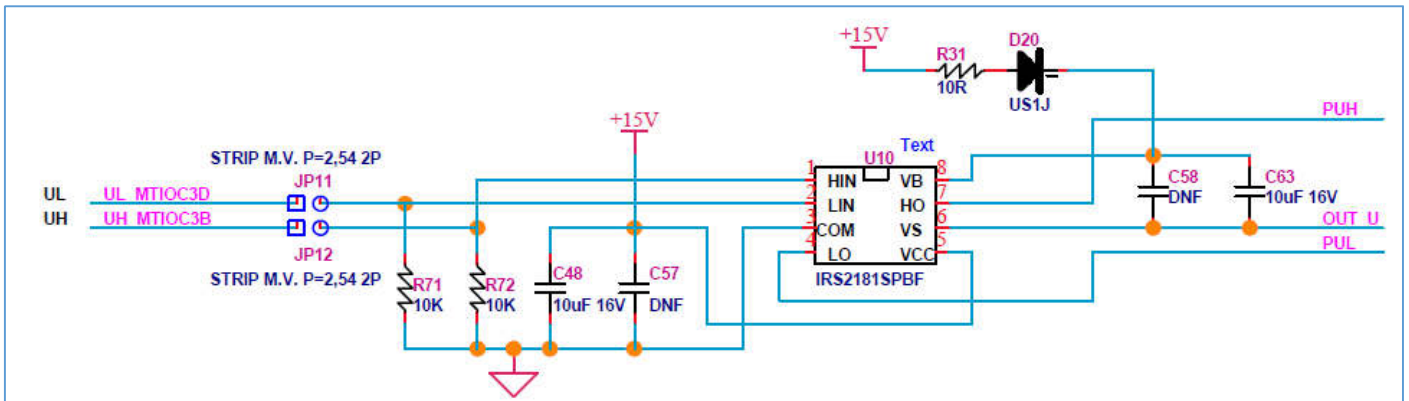
The inverter has the classical schema with the three shunts on the lower arms, with the possibility to use a single shunt by removing two of them.



Please find below the schematics extract showing the draying in mode details.

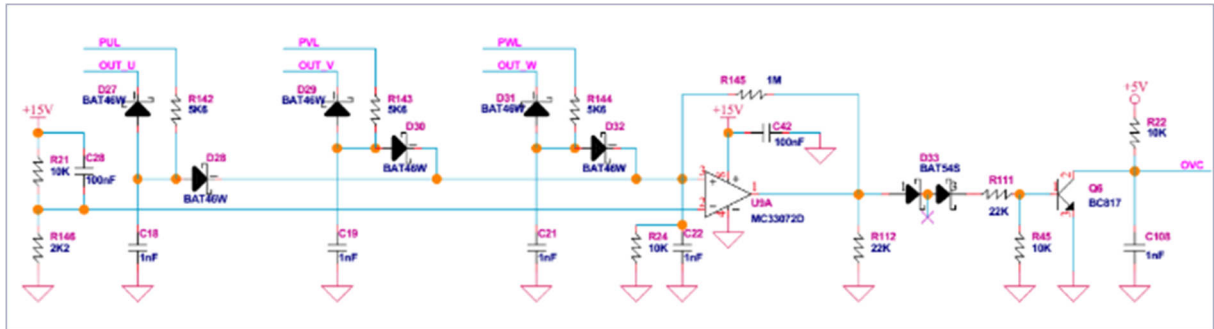


Furthermore, the three driving circuits for the six low voltage MOSFETS **RJK0654DPB** are described below.



Furthermore, the signals from the gate driver circuits called "PUL, PVL, PWL" are used to manage the risk of over-current in the board. The signals are directly linked to the RX23T Microcontroller module called the Port Output Enable (e.g. POE) to stop the PWM signal in hardware and protect the 3-phase power stage.

In case of over-current, the POE module will set in hardware the six PWM signals in high impedance mode to ensure safe-fail mechanism. Please find below the extract of the schematics regarding the over-current management.

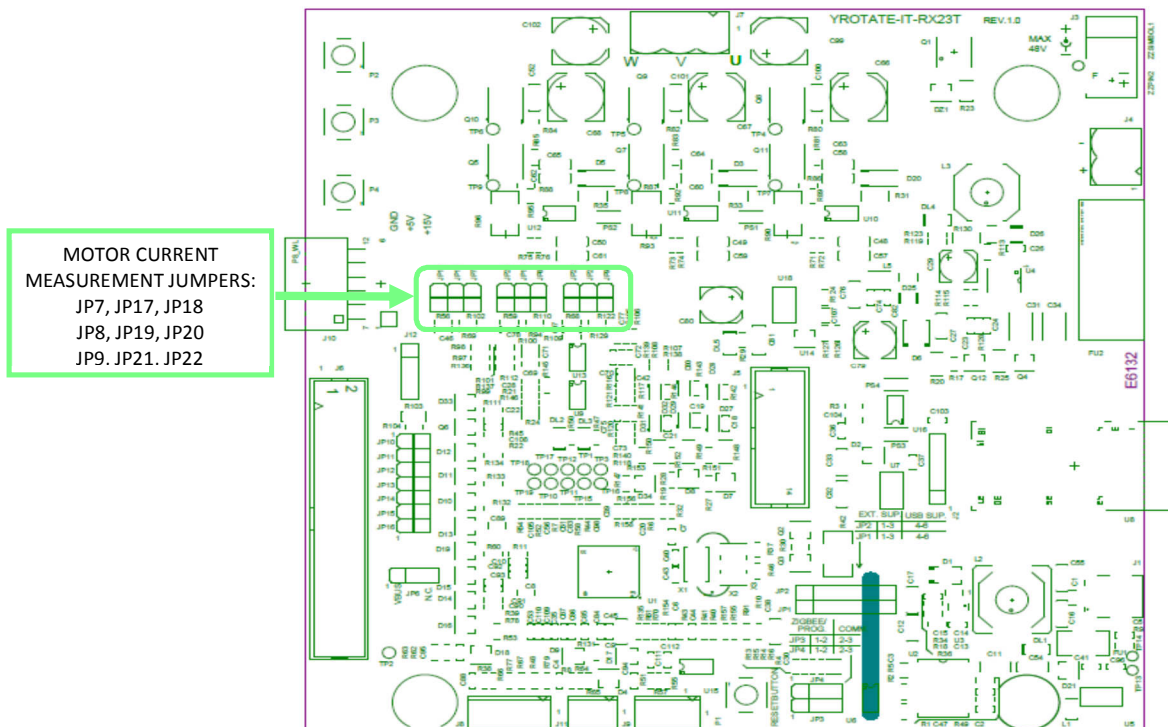


7. Current reading methods selection

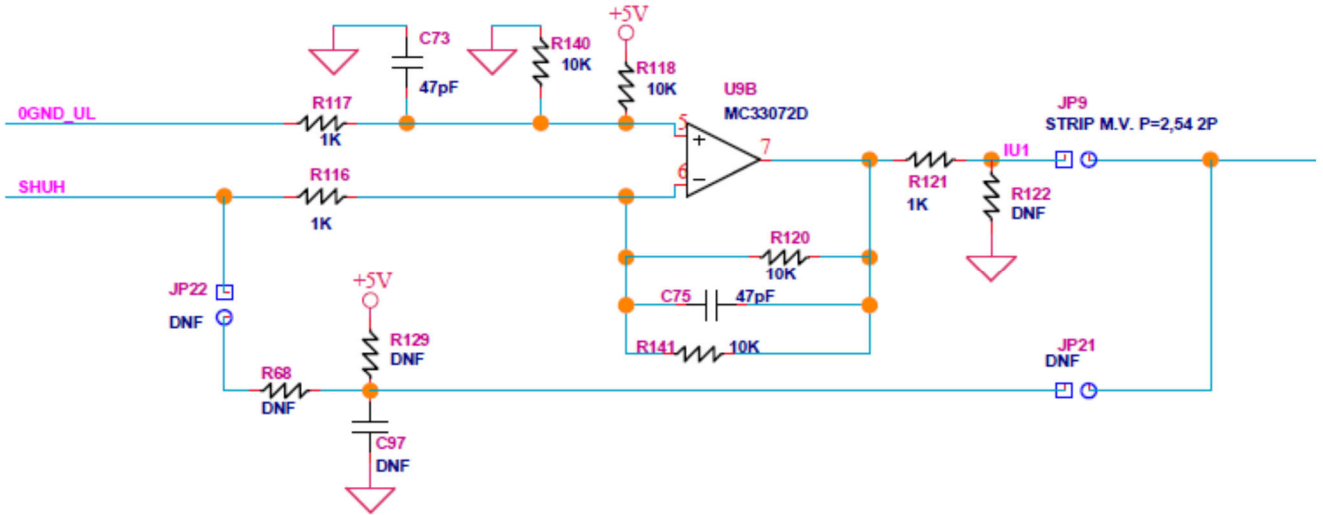
As shown in the following figure, some jumpers are placed on the board (see schematics): JP7, JP17, JP18, JP8, JP19, JP20, JP9, JP21 and JP22. The jumpers allows the reading of the motor current by using external operational amplifiers or internal one as shown here below:

- **Configuration 1: external op. amp.:** JP7, JP8, JP8 closed, JP17, JP18, JP19, JP20, JP21, JP22 open
- **Configuration 2: internal op. amp.:** JP7, JP8, JP8 open, JP17, JP18, JP19, JP20, JP21, JP22 closed

NOTE: the RX23T microcontroller family is not equipped with internal operational amplifiers. But the RX24T family is equipped with on-chip Op-amp, also called Programmable Gain Amplifier (PGA).



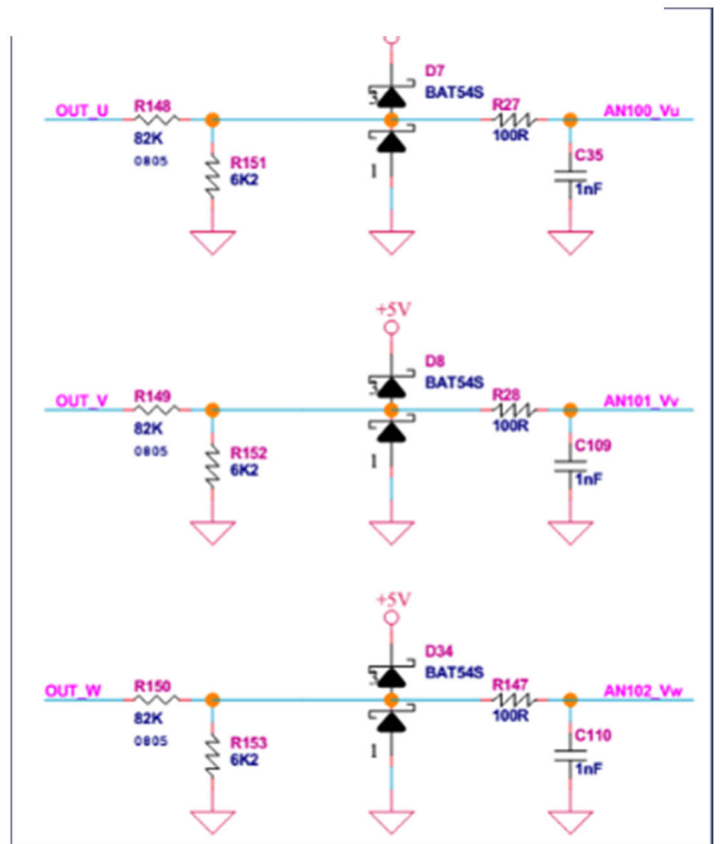
Please find below more details about the circuit used to read the current flowing through the shunts.



In case of 3-phase Brushless motors that need to be driven in sensorless block commutation mode, the RX23T kit integrates the possibility to manage the motor phases voltage.

These are measured for the back EMF signals detection using the A/D inputs: AN100 to AN102.

Please find on the right the detailed schematics made for the phase voltages reading.

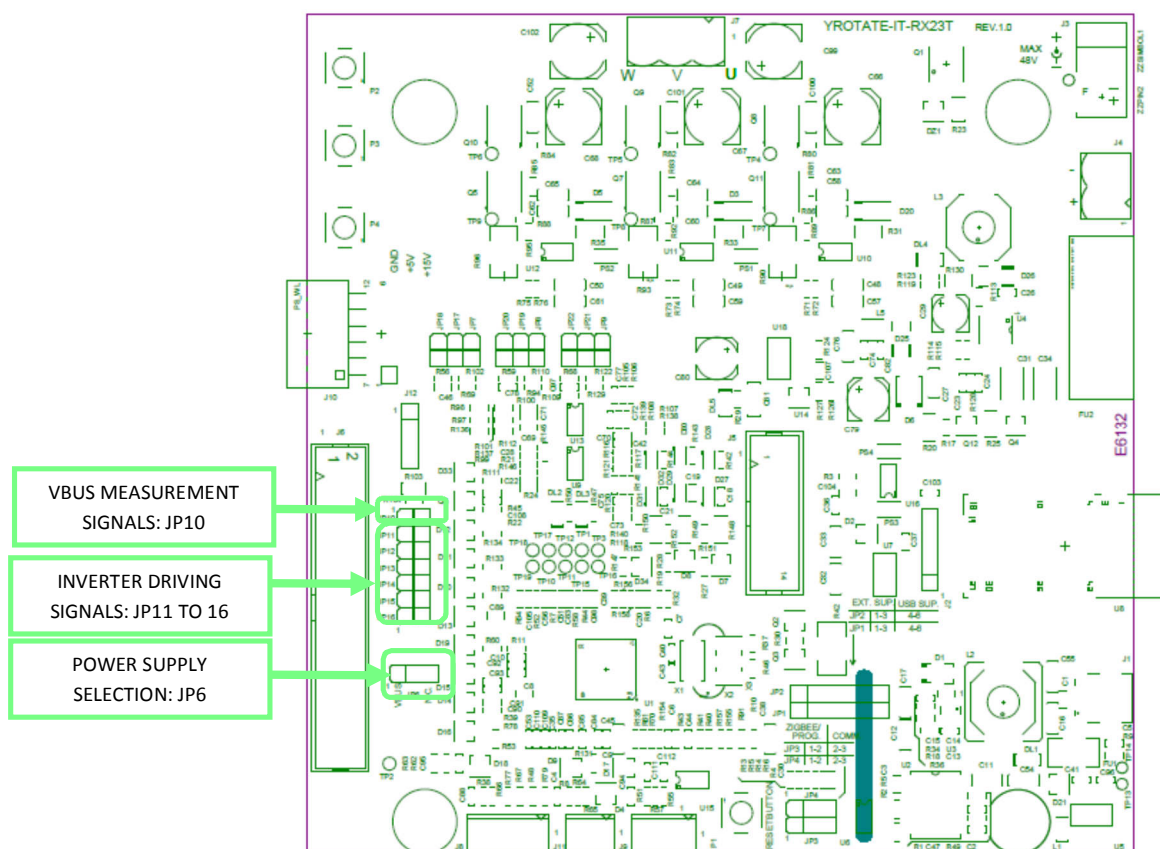


8. Interface to an external power stage

Since internal power stage allows only the management of low voltage motors, an interface with an external power stage has been developed.

The selection between the internal power stage and the external power stages is ensured by jumpers. It is a safe way to ensure that the right voltage and current signals are active. When the external power stage is connected to the kit, it is by default the **active** one. So the microcontroller pins are directly connected to the external power stage connector.

In this case the internal power stage should be disconnected, and this must be done by disconnecting the appropriate jumpers.



Please find below the jumpers description.

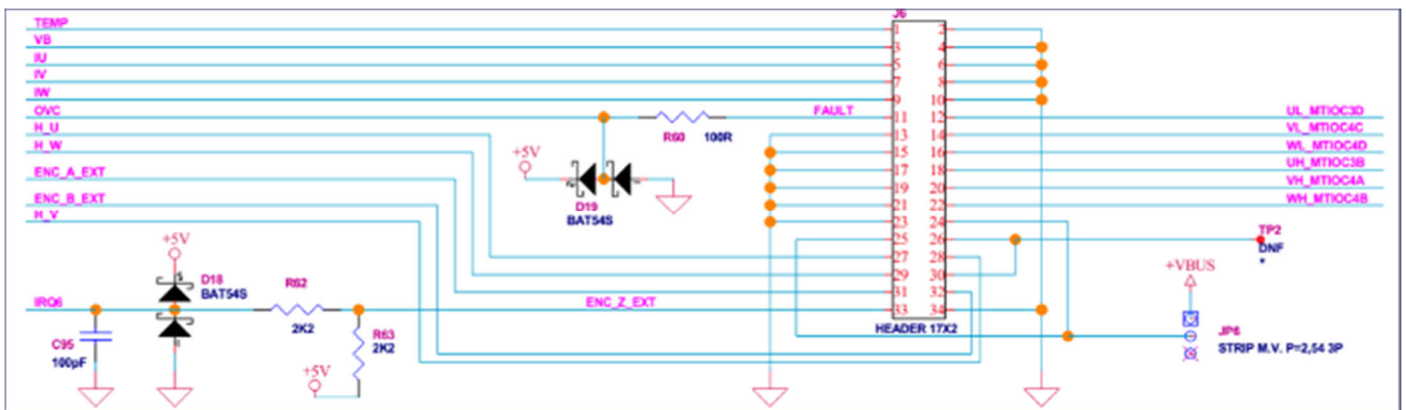
- JP10: if closed, then the internal power stage DC Bus voltage is connected to the opportune A/D converter pin.
- JP7, JP8, and JP9: if closed, then the internal power stage currents measurements (U, V, W) are connected to the opportune A/D converter pins.
- JP11 to JP16: if closed then the inverter driving signals are connected to the internal power stage drivers.
- JP6: it allows the following selection:

- if pins 1 and 2 are shorted, then the external power stage low voltage supply (between 15V to 24V) is connected to the internal DC Bus Voltage; in this case both the step-down converters of the board will work;
- If pins 2 and 3 are shorted, then the external power stage low voltage supply cannot be used to supply the board.

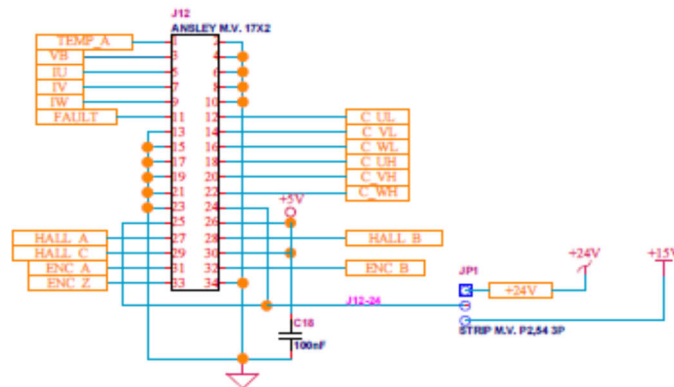
Please be careful to take into account the following precautions:

1. Please avoid to connect both the external power stage connector (J6) and to close the jumpers JP6 to JP16: this would produce short circuits between signals coming from different sources.
2. In JP6, chose the configuration with the pins 1 and 2 shorted, when external power supply board is used.

Please find below the drawing of the interface connector.



For a comparison, find below the drawing of the corresponding connector in the 1.5KW, 230V_{AC} external power stage (E6108A).



If using a different external power stage, please keep present the following notes:

- a) The PWM drive 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. 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 polarized with a 10K pull-up toward the logic supply.

- b) The analog measurement signals from power stage, in particular the current readings and the DC link voltage reading are clamped (with diodes from logic GND and to logic V_{cc}) 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.
- c) The ground connection is always active, and it represents the reference for all the interface signals.

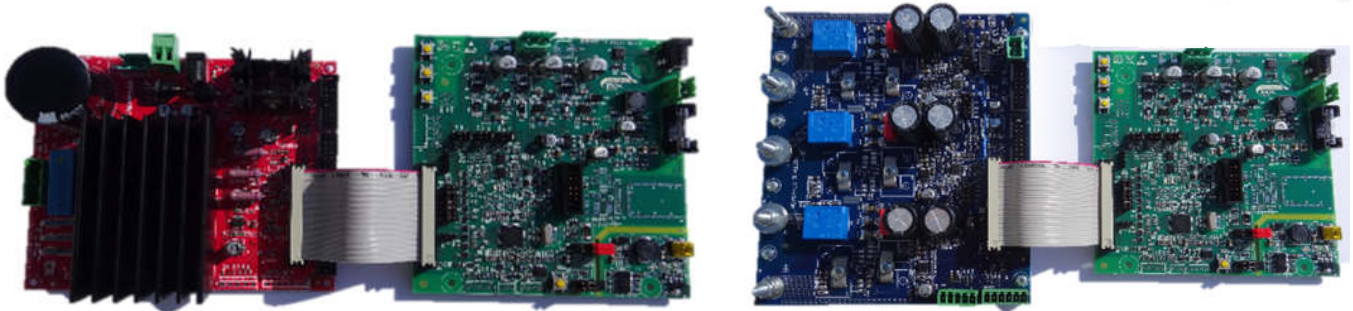
In the next figure a simple example regarding how the power board connections have to be arranged, is presented.

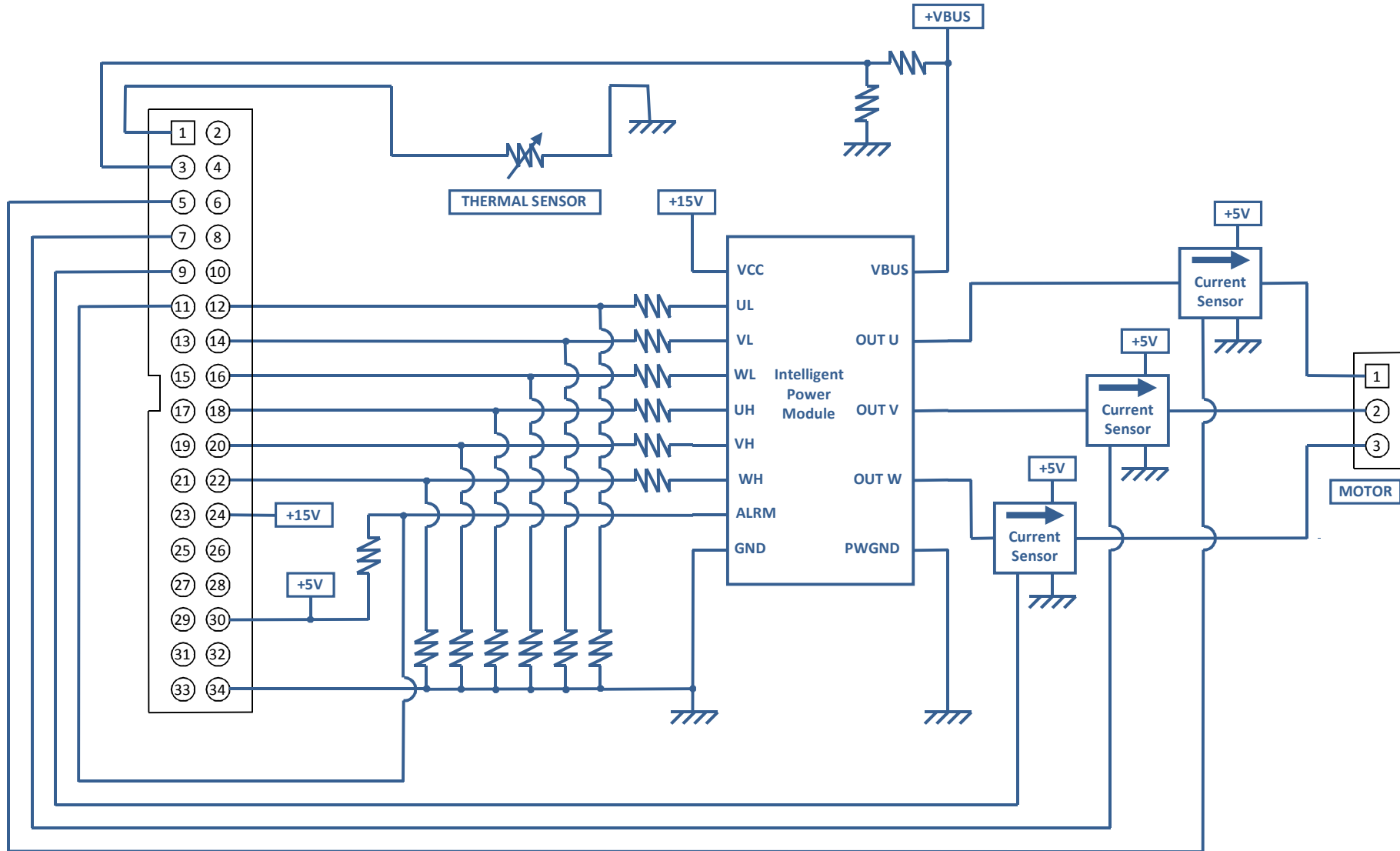
In this schema it is supposed that the power board has its own supply for the power module (+15V); +15 to +24V supply from power board is also used to supply the microcontroller (thanks to jumper JP6 in microcontroller board).

Please refer to the complete schematics for further details that are available on the website: www.renesas.eu/motorcontrol.

Two power stages are currently available under request, were successfully tested with the YROTATE-IT-RX23T kit.

On the left hand side, the power stage can manage 300V_{DC} for 20A and the right hand side 60V_{DC} for 100A.





9. Single shunt current reading

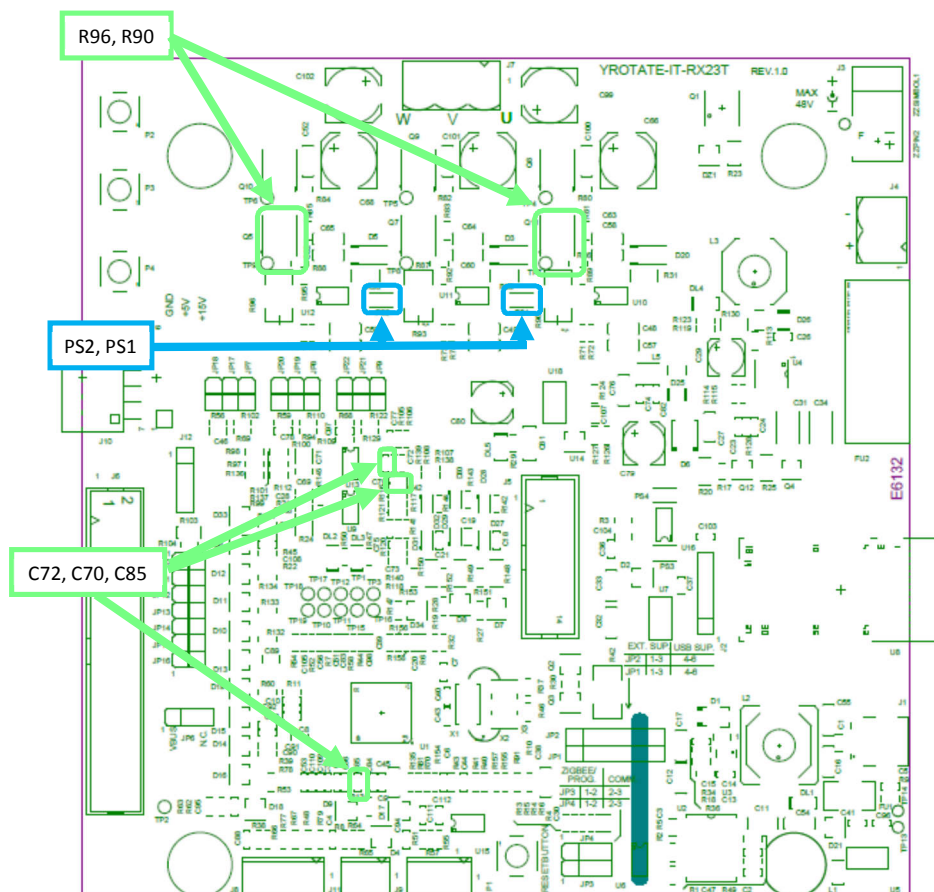
While the normal configuration of the board and the standard software are based on three shunts current reading, we also offer the possibility to configure the board for single shunt current reading.

Some hardware modifications are required and a different software version has to be program into the RX23T flash memory.

The required hardware modifications are the following (please refer to the board schematics):

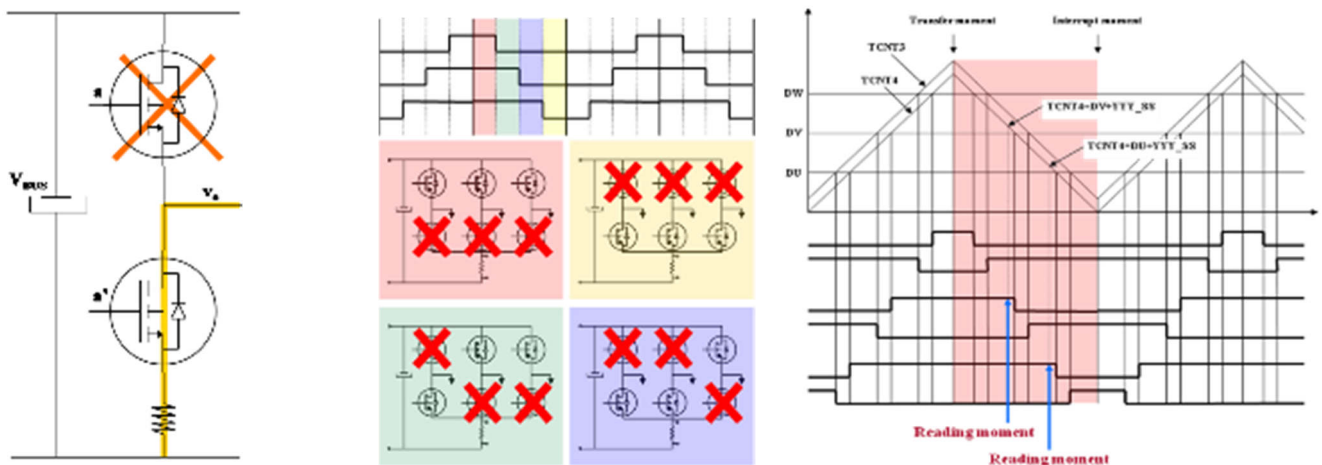
- Remove from the board the shunts R90 and R96 (they are the shunts related to the phases U and W)
- Close the soldering points PS1 and PS2 (those soldering points put the three inverter harms in common, below the lower switches and above the shunts)
- Remove the capacitors C70, C72, C85, it speeds up the current reading circuit

The components involved in the modifications are indicated in the figure below.



10. Current reading timing in three shunts and single shunt configurations

The figures below show the different situations related to the two configurations. The first figure is related to three shunts current reading, the other are related to the single shunt current reading.



Three shunts configuration

In the three shunts 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 YROTATE-IT-RX23T kit is delivered in the three shunts configuration.

Single shunt configuration

In the single shunt configuration, only when one or two of the lower switches are ON the current through the shunt is related with the phase current.

When only one of the lower switches is ON, the current in the shunt is equal to the current of the corresponding inverter phase.

When two of the lower switches are ON, the current in the shunt is equal to the sum of the currents of the corresponding phases that is it is minus the current of the third phase.

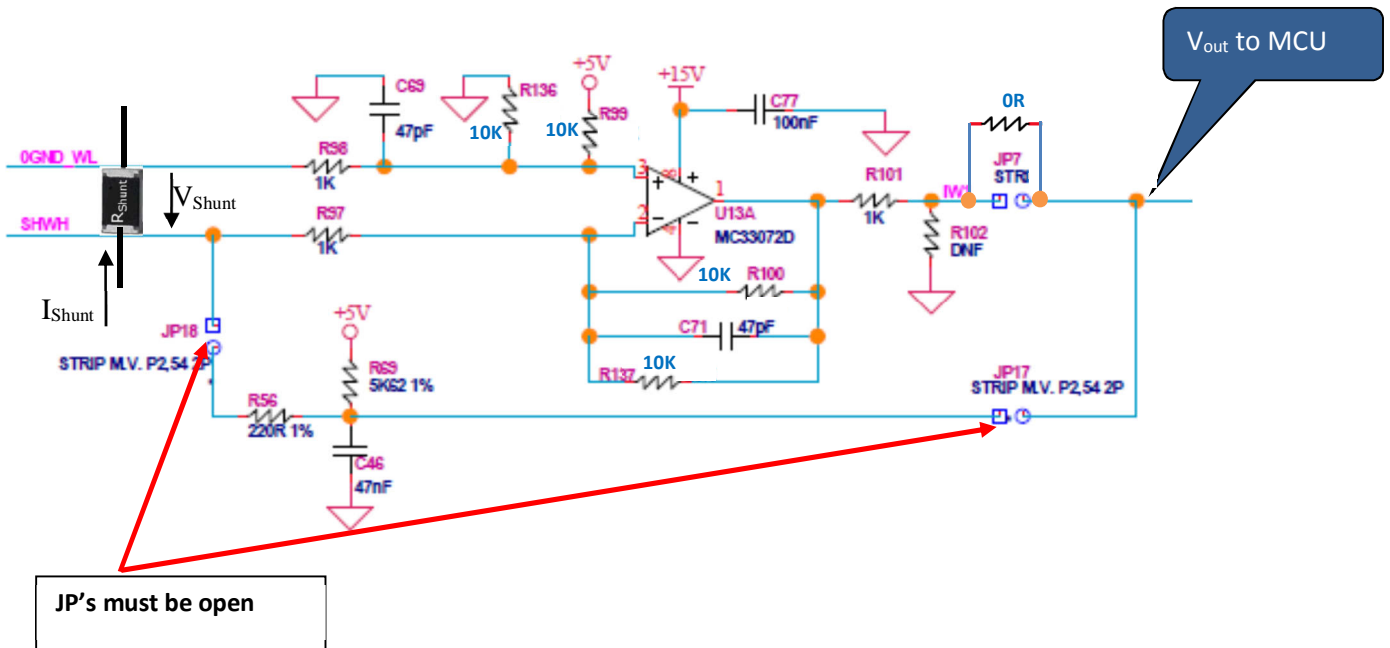
Important Note:

The software project available on the website: www.renesas.eu/motorcontrol is designed under e²studio environment and only for three shunts configuration. An additional version is available under the CubeSuite CS+ Environment.

The three shunts software project designed under for e²studio is called: "YRotateItRX23T_3s_V1.zip"

Furthermore, the circuit below is used to measure in real-time the motor currents flowing through the shunts resistors. In the Bill of Material, the shunt are called R₉₀, R₉₃, R₉₆ as described on the page 7 of the schematics.

The default value is **R₉₀ = R₉₃ = R₉₆ = 100mΩ**



The amplifier circuit is made to manage up to 5A flowing through the shunts and the output of the amplifier is connected directly to the Microcontroller RX23T.

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.

If R100 = R137 = R136 = R99 and R97 = R96, then the amplification of the circuit is (R100/2)/R97 = 5/1 = 5.

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

$$V_{out} = 2.5V - V_{shunt} \times (R100/2)/R97, \text{ and } V_{shunt} = R_{shunt} \times I_{shunt}$$

So

$$V_{out} = 2.5V - (R_{shunt} \times I_{shunt}) \times (R100/2)/R97$$

And the term

$$\Delta V = (R_{shunt} \times I_{shunt}) \times (R100/2)/R97$$

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:

$$I_{shunt \text{ max dc}} = 2.2V / (R_{shunt} \times (R100/2)/R97)$$

The default values are the following:

$$R_{\text{shunt}} = 0.1\text{ohm}, R_{100} = 10\text{K}, R_{97} = 1\text{K}, \text{ so } I_{\text{shunt max dc}} = 4.4\text{Adc}$$

To manage different values it is enough to change R_{shunt} R100 and R97 **but it is important to keep the following conditions:**

$$R_{100} = R_{137} = R_{136} = R_{99}$$

$$R_{96} = R_{97}$$

Finally, the shunt resistor value needs to be updated in the software itself in the file called: "pws_E6132.h" as shown below:

```

/*****
*   File:      pws_E6132.h
*   Project:   YRotateIt-RX23T (three shunts)
*   Revision:  1.0
*   Date:      01-09-15 (DD-MM-YYYY)
*   Author:    Andrea Ghirardello
*   Contents:  Internal power stage (E6132) related definitions
*****/

#ifndef _PWS_E6132_H
#define _PWS_E6132_H

#include "typedefine.h"

// current conversion
// NOTE: the current is considered positive if going out from the inverter
#define RSHUNT_OHM      ( 0.1 )    // shunt value [Ohm]
#define RSGAIN          ( 5.0 )    // circuit gain

```

11. Website material: www.renesas.eu/motorcontrol

The complete kit material is available on-line at the address: www.renesas.eu/motorcontrol

The latest updates of the downloadable material for the YROTATE-IT-RX23T kit is listed below:

Items	Description of Resources
Auto-tuning Video-Tutorial	Short video explaining how to easily tune any Brushless AC motor in 45 seconds using just the intuitive PC Graphical User Interface
Drivers YROTATE-IT-RX23T Drivers	Drivers and setup files for the PC Graphical User Interface
Embedded Software e ² Studio Source Code CubeSuite CCS+ Source Code	Source files for code flashed by default into the Renesas microcontroller Royalty-Free software
Manuals Kit Motor Specifications Renesas Datasheets Tuned Motors Specifications	Relevant documentation for the kit, the motor and the microcontroller, the MOSFETs. Languages: English, Japanese, Korean, Chinese
Schematics-Gerber-BoM External Power Stages Main Board	Schematics, Gerber files of Bill of Materials for both the main kit and the two external power stages (not included as part of the kit)

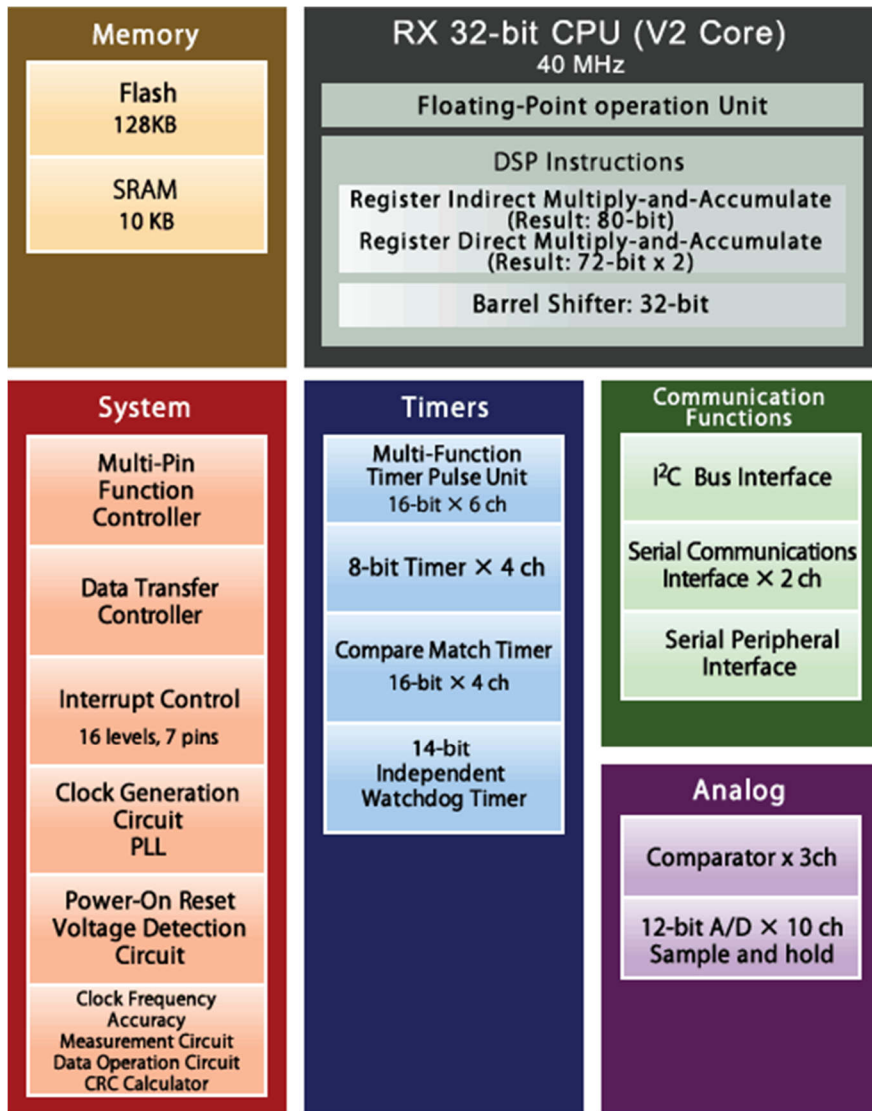
12. Microcontroller RX23T short overview

The RX23T Group is 32-bit microcontroller and suited for single inverter control and has a built-in FPU (floating-point processing unit) that enables it to easily program complex inverter control algorithms. This helps to greatly reduce the man-hours required for software development and maintenance. Furthermore, thanks to the RX200 core the current consumed in software standby mode with RAM retention is a mere 0.45 μ A. RX23T MCUs operate in a broad voltage range from 2.7 V to 5.5 V, which is useful for inverter control, and are highly compatible with the RX62T Group at the pin arrangement and software level.

The main specifications of the RX23T microcontrollers are as follows:

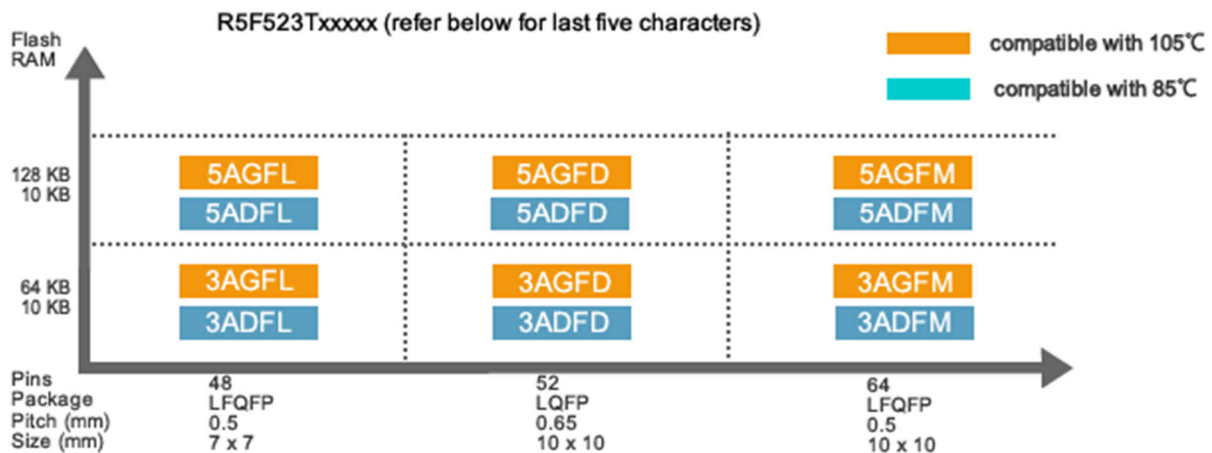
Item	RX23T Group	
CPU core	RX CPU v2 core running at 40MHz, delivering 2DMIPs/MHz General registers: 32-bit x 16, 32-bit multiplier & Divider Two types of Multiply-accumulator memory-to-memory operations and register-to-register operations	
Power supply voltage	2.7 to 5.5V	
Floating-point operation unit	Single-precision floating-point operation unit	
Flash / RAM memory	Max.128 KB / 10KB	
On-chip peripheral functions	Transfer	Data transfer controller (DTCa)
	Timers	Multi-function timer pulse unit 3: 16-bit x 6 channels (MTU3c) Port output enable 3 (POE3b) 8-bit timer (TMR): 8-bit x 2 channels x 2 units Compare match timer (CMT): 16-bit x 2 channels x 2 units
	Communications	Serial communications interface (SCIg): 2 channels I2C bus interface (RIICa): 1 channel Serial peripheral interface (RSPIa): 1 channel
	Analog	12-bit A/D converter (S12ADE): 10 channels Comparator C (CMPC): 3 channels D/A converter for generating ref. voltage for comparator C (DA)
	Safety	Clock frequency accuracy measurement circuit (CAC) Data operation circuit (DOC) 14-bit independent watchdog timer (IWDTa) A/D converter self-diagnostic/open-circuit detection function CRC calculator (CRC) Register write protection
	Clock generation circuit	Main clock oscillator Low-speed on-chip oscillator (LOCO) PLL frequency synthesizer Dedicated on-chip oscillator for the IWDT
	Other	Multi-function pin controller (MPC) Power-on reset circuit (POR) Voltage detection circuit (LVDAb)
Packages	PLQP0048KB-A(48-pin,LFQFP,7 x 7mm, 0.5mm pitch) PLQP0052JA-A(52-pin,LQFP,10 x 10mm, 0.65mm pitch) PLQP0064KB-A(64-pin,LFQFP,10 x 10mm, 0.5mm pitch)	

Please find below the RX23T microcontroller block diagram.



Please find below the memory line-up including the part-names.

RX23T Group Memory vs. Package Lineup

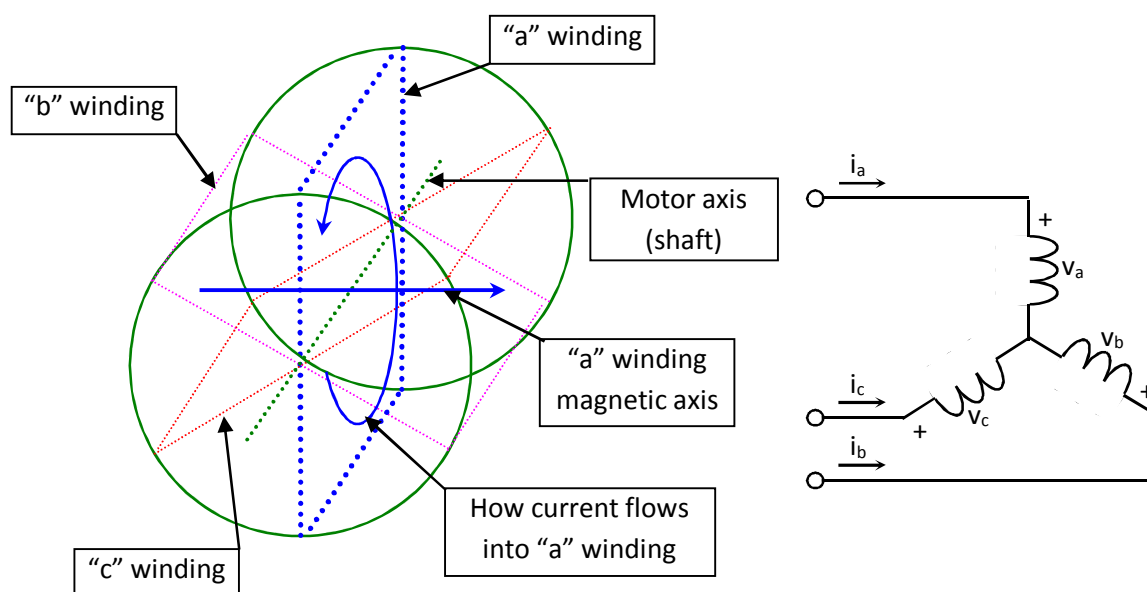


13. Permanent Magnets Brushless Motor model

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:

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

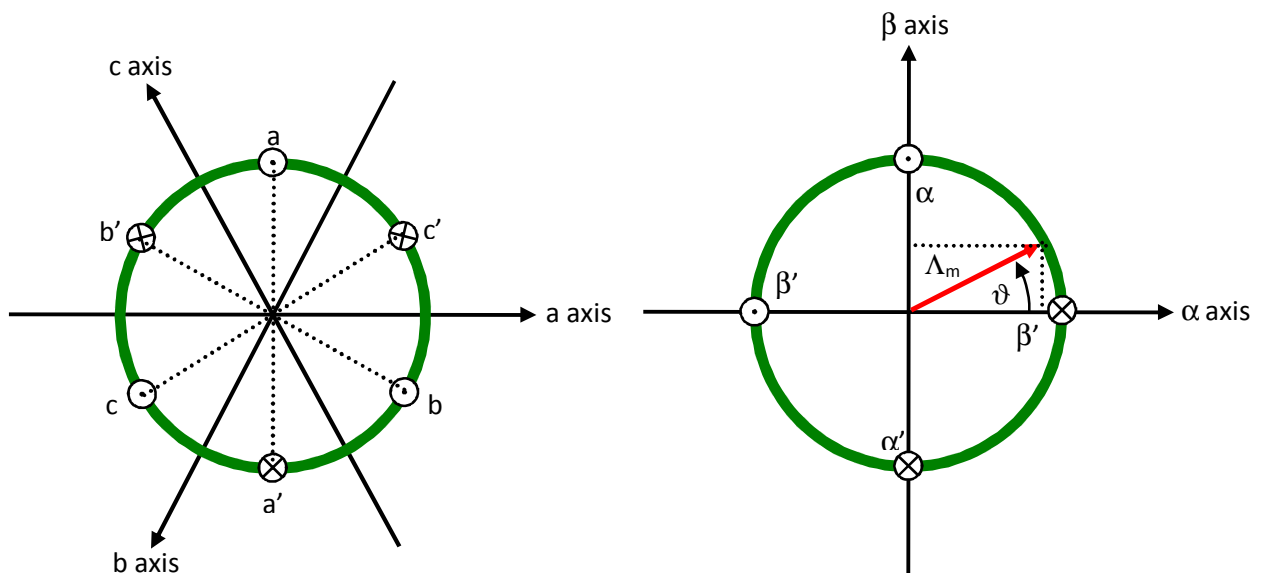
$$v_b = R_s i_b + \frac{d\lambda_b}{dt}$$

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

- 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:

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

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

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

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).

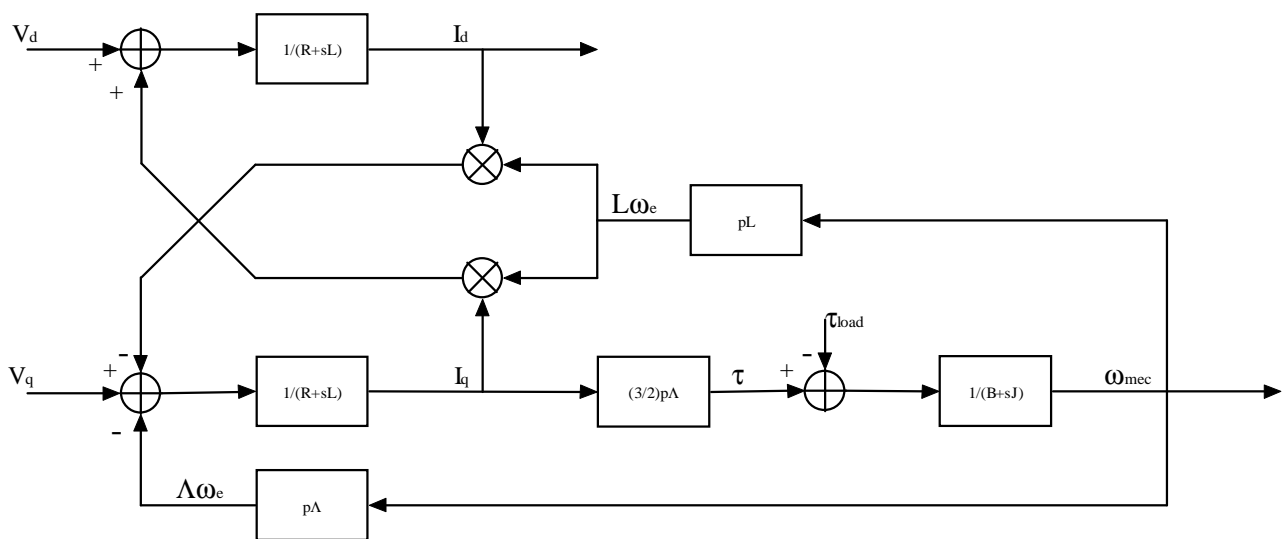
The reference frame transformations from the (α, β) system to the (d, q) system depends on the instantaneous position angle ϑ

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:

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

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

Furthermore:

$$\Lambda_m \cos(\vartheta) = \lambda_\alpha - Li_\alpha$$

$$\Lambda_m \sin(\vartheta) = \lambda_\beta - Li_\beta$$

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

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

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

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

$$\vartheta = \arctan\left(\frac{x}{y}\right)$$

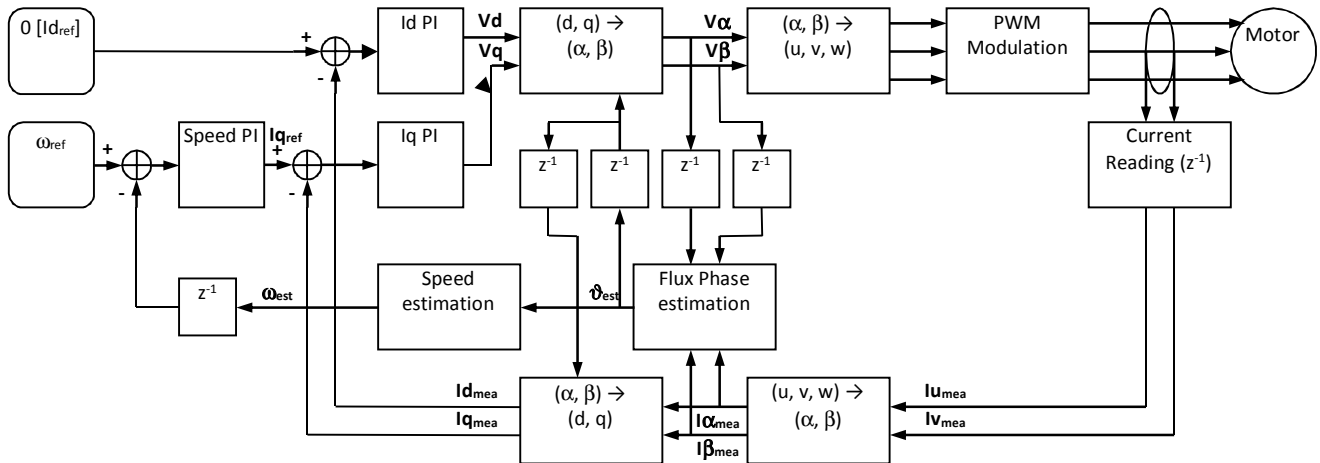
The system speed ω can be obtained as the derivative of the angle ϑ .

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

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.

14. Sensorless Field Oriented Control algorithm

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



The main 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 in principle, even if the blocks order has been adjusted.

15. Software Tools used

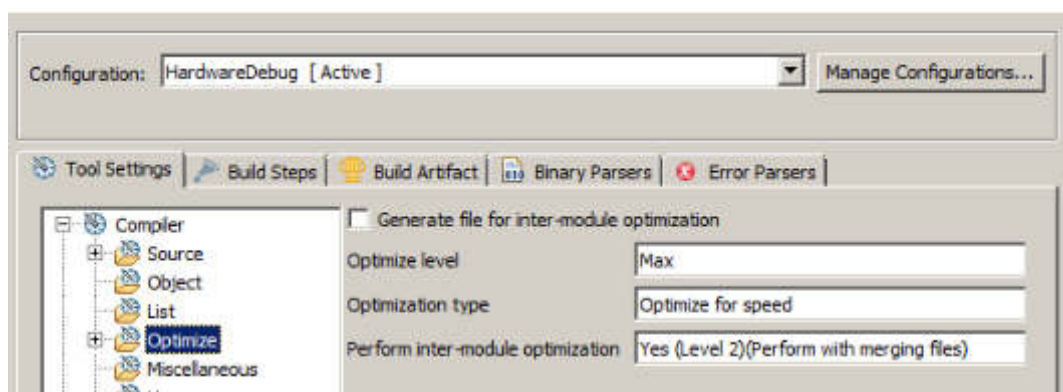
15.1 IDE and e2studio Tool Chain used

The embedded software delivered in the YROTATE-IT-RX23T kit is developed under e²studio integrated development tool. Please find below the details of the IDE used.



The software was designed under e2studio Version: **4.0.2.008**. Furthermore the e2studio IDE is based on Eclipse SDK 4.4.2 (Luna SR2) and CDT version 8.6.0.

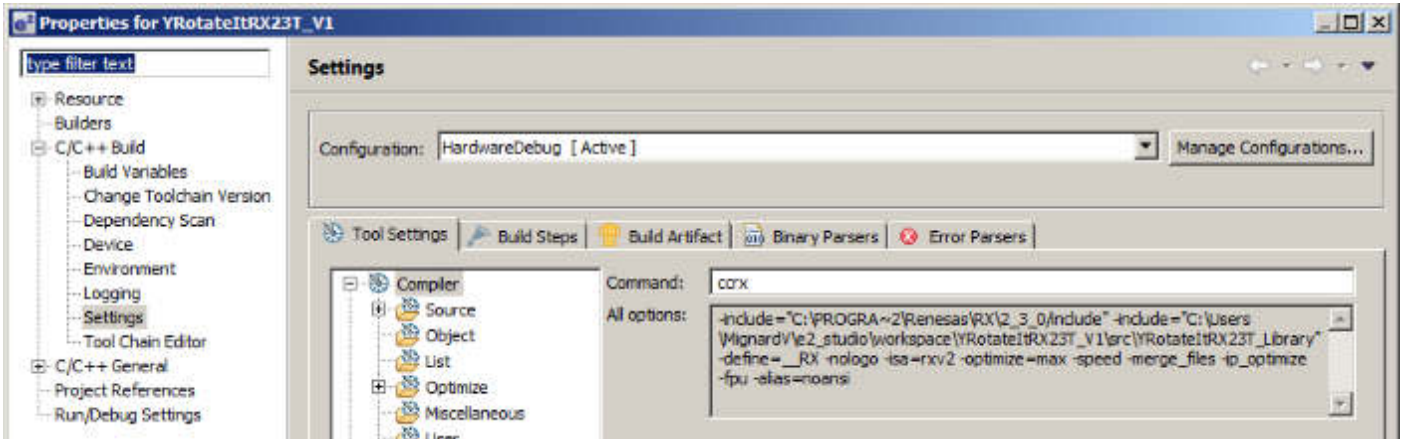
The Renesas RXC Toolchain, version v2.03.00 is used. The compiler used is the “Renesas CCRX”. The Optimisation level is “Max” and the optimization type is “Speed”, as visible on the picture below.



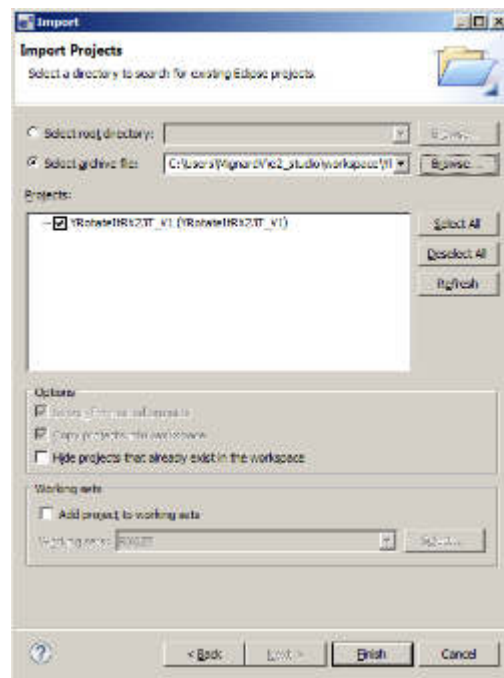
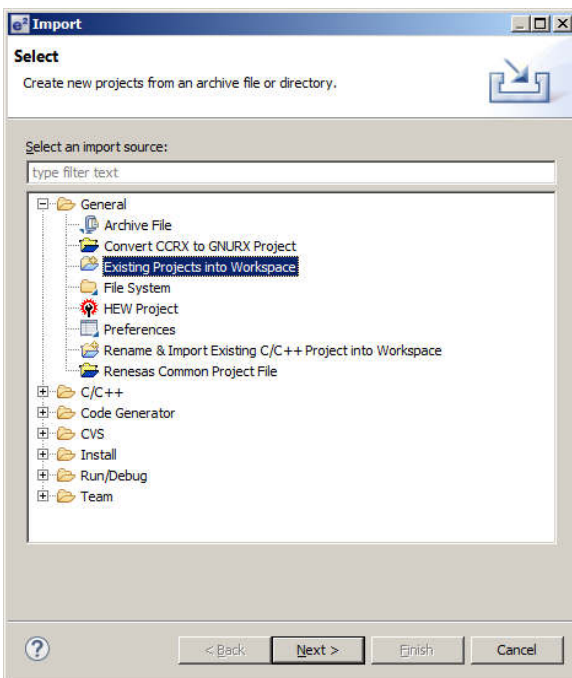
The device selected in the program is: R5F523T5AxFM, it means the RX23T with 128KB flash, 10KB RAM in a 64-pin package. The Instruction Set Architecture selected must be “RXv2 Architecture” to ensure that the microcontroller is delivering 80DMIPS at 40MHz CPU clock.

Finally, the Data endian selected is “little-endian data”.

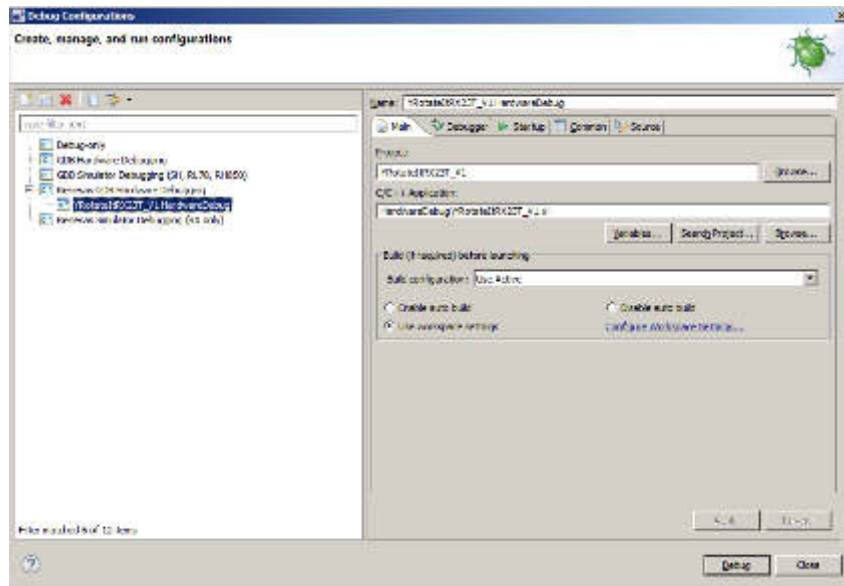
Please find below the list of options for the compiler:



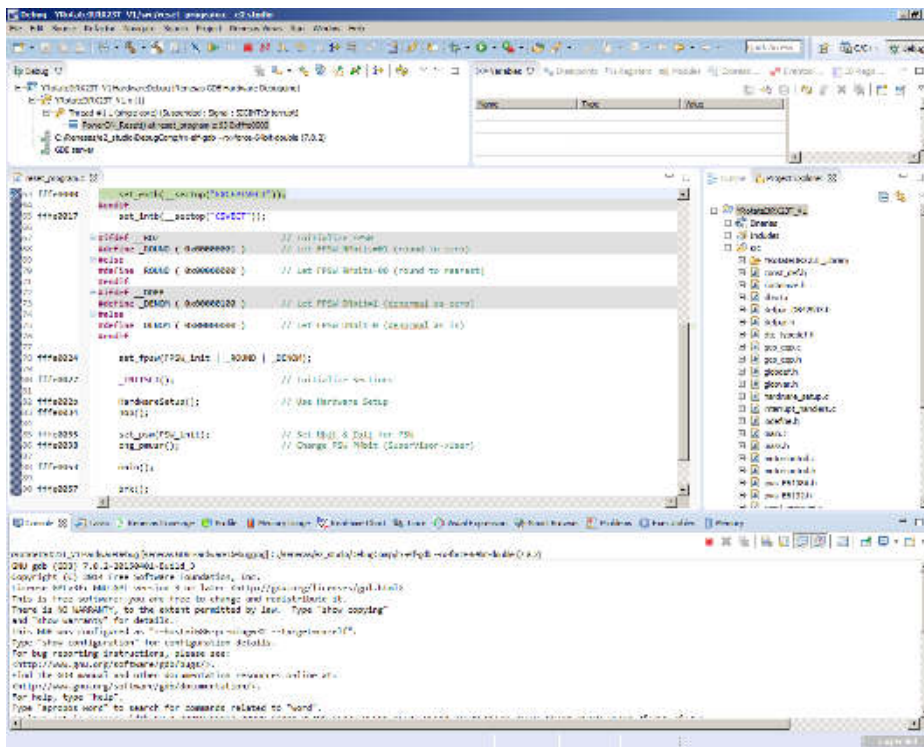
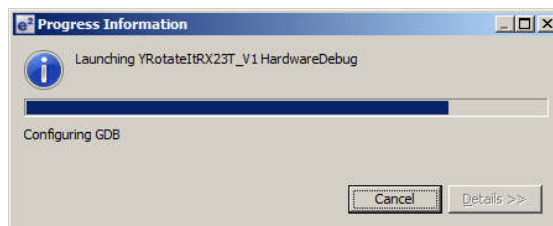
15.2 Project importation into e²studio



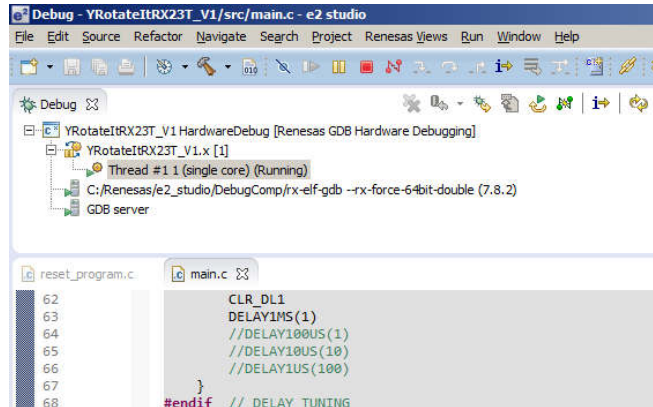
Launch the debugger by clicking on “Debug configuration” and the window below will appear:



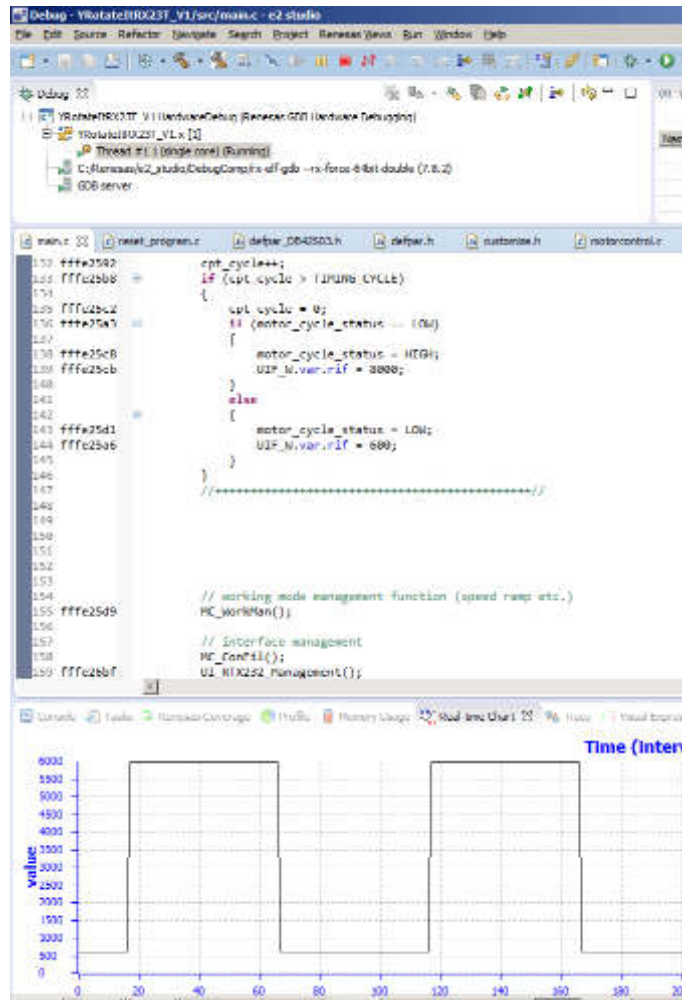
After connection the E1 debugger to the target board powered by USB or external power supply, please click on “Debug” to start the debugging and load the program into the microcontroller flash memory. The windows below appears:



The program is loaded after clicking twice on the green button “Resume” to load and run the program. The window below appears once the program is running from the MCU itself. The E1 debugger can be removed to run the software on its own.



In the debugger, it’s also possible to display the measured speed as shown below.



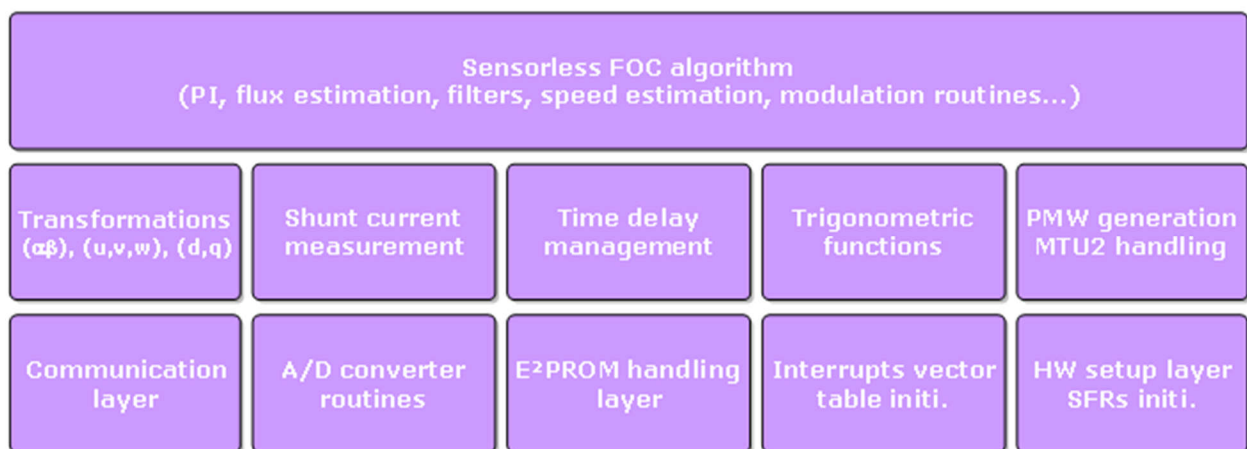
16. Software description and Resources used

The software delivered in the YROTATE-IT-RX23T kit, previously described, is working on the RX23T microcontroller clocked at 40MHz 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 **16KHz** and the control loop is set to **8KHz**, so 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:



The complete software uses the resources below in the three shunts configuration. It includes the serial communication interface, the board management, the LED management, the EEPROM management, the auto-tuning algorithm and self-identification and of course the complete sensorless vector control algorithm.

- **FLASH memory usage: 24KB and RAM memory usage: 3KB**

The embedded software package is called **"YRotateItRX23T_3s_VX.zip"** running under e2studio environment. The same source code is ported under CS+ (e.g. CubeSuite) environment and the software package is called: **"YRotateItRX23T_VX_E2CS+.zip"**.

The control loop of the field oriented control algorithm (e.g. sampling frequency) is set to **8KHz** by default. The Pulse Width Modulation (PWM) frequency is set by default to **16KHz**. The parameters are visible in the module name "customize.h".

The parameter **"SAM_FRE_DEF"** is the sampling frequency in Hertz and is set to **8000** (e.g. 8KHz).

The parameter **"F_RATIO_DEF"** is the ratio between the sampling frequency and the PWM and is set to **2**, so it means $2 \times 8000 = 16000$ Hz (16KHz).

Such parameters can be modify dynamically using the PC GUI without recompiling the overall project changing the parameters below and resetting the board. The PC GUI provide a direct access to the parameters below.

19	SAM_FRE_DEF	Set the sampling frequency [Hz] of the control loop
20	F_RATIO_DEF	Set the ratio between the PWM frequency and sampling frequency, e.g. if 8000 is set in the parameter #19 and 2 in the parameter #20, the PWM frequency is 16KHz.

The parameter #19 is setting the control loop speed. By default, 8KHz is selected by entering the value "8000". The PWM frequency can be set to four different values depending on the motor and the applications either **8KHz**, **16KHz**, **24KHz** or **32KHz** by entering either **1, 2, 3, 4** as ratio between the two frequencies.

As the sampling period is **125µs** (8KHz), and the control loop for the inverter control takes maximum of **50µs**, the CPU load of the motor control algorithm with all the option enable is: $50/125 = 40\%$. The CPU clock is running at 40MHz.

It leaves 60% of the CPU to perform additional tasks, like board management, system control, display, etc.

Please find below more details information regarding the timing of the control loop according to the selected options:

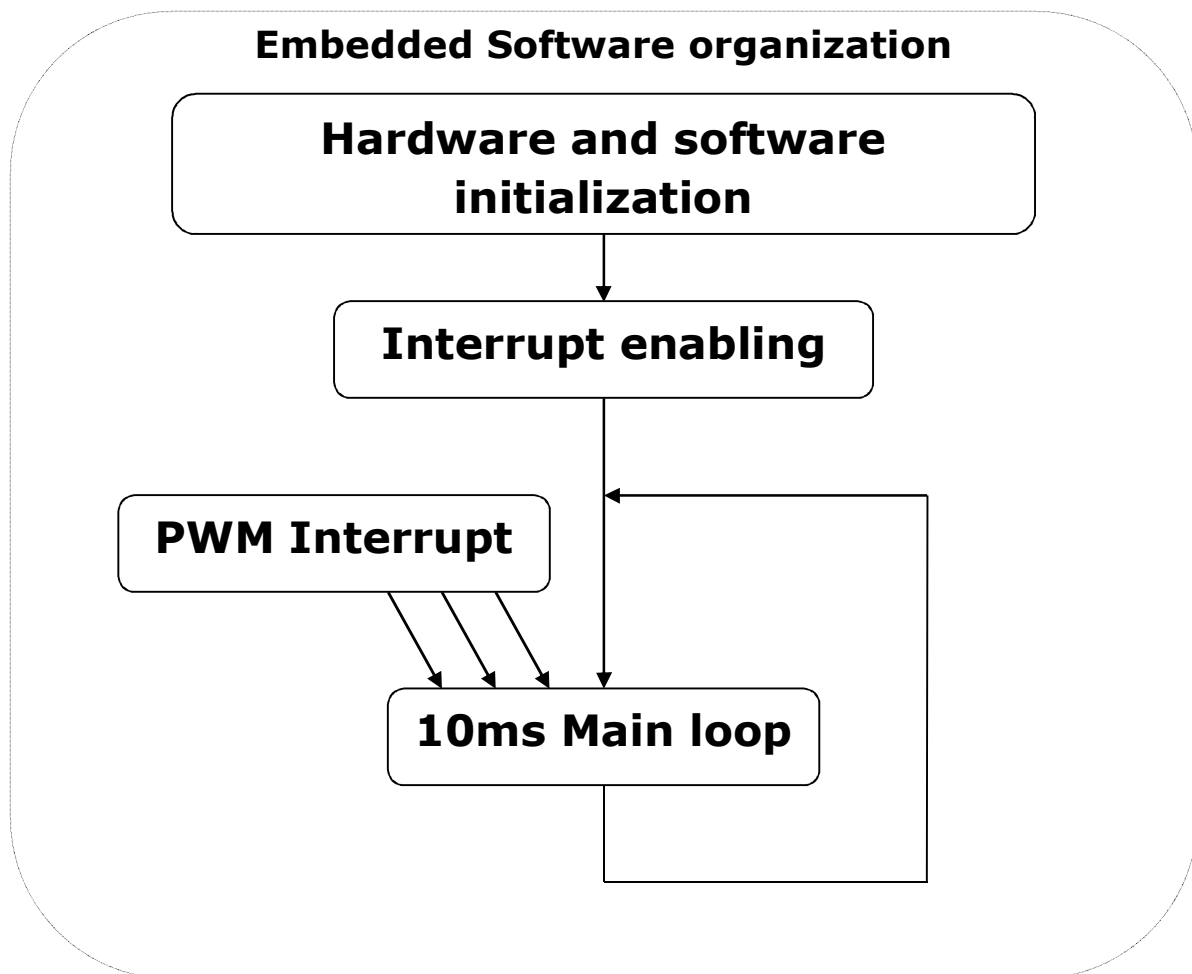
Option	Development Version	Mass Production
Communication management	Enable	Enable
EEPROM management	Enable	Disable
PI current auto-tuning	Enable	Disable
Motor self-identification	Enable	Disable
Oscilloscope window	Enable	Disable
Internal Variables debug	Enable	Disable
Control loop Timing	50µs	40µs
CPU load [8KHz sampling frequency]	40%	32%
CPU load [14KHz sampling frequency]	70%	55%

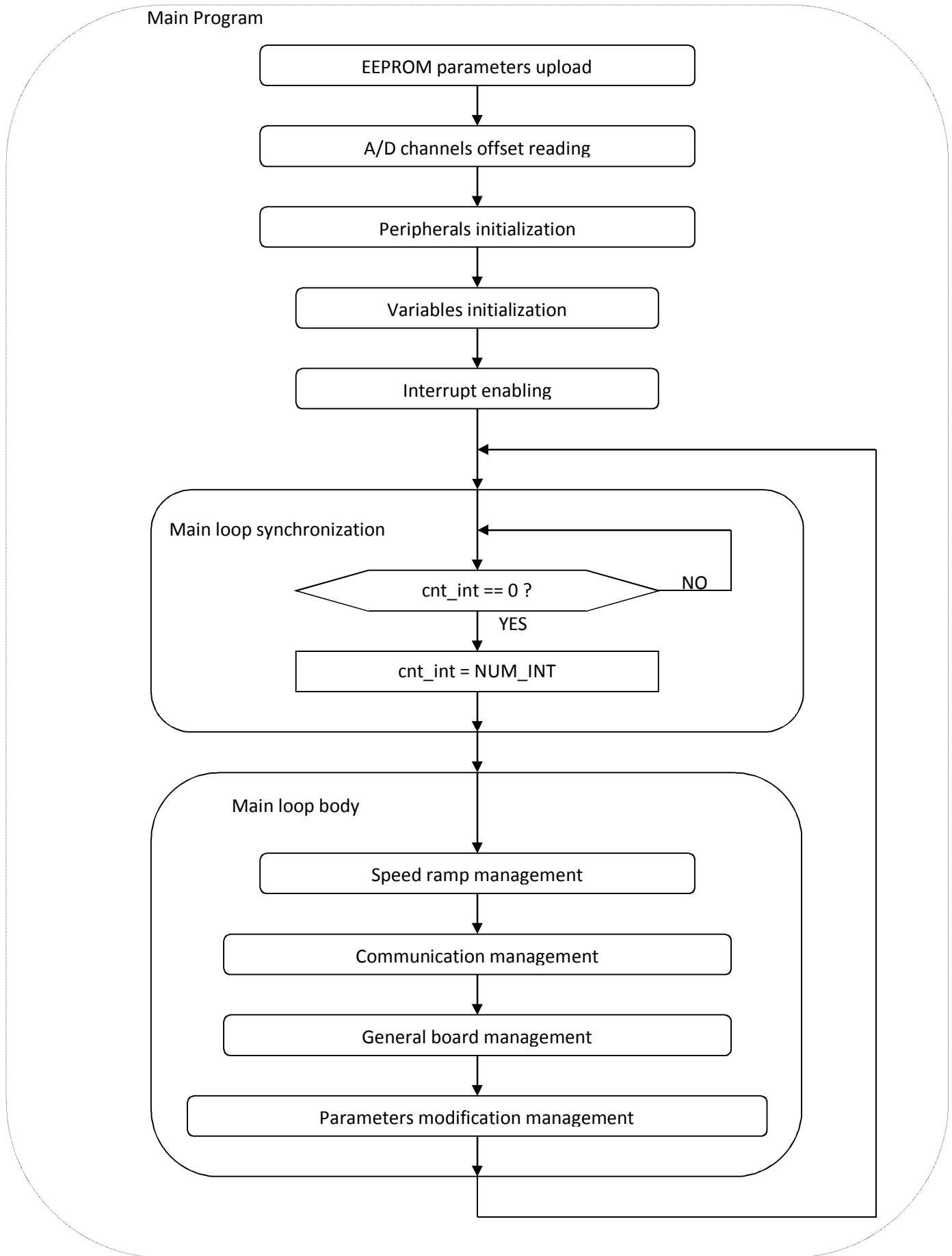
Furthermore, the execution time of the motor control interrupt including all the, the Speed PI block, the Current PI block and the complete vector control algorithm and the auto-calibration mechanisms: is **50µs with**

The default development tools used is the e²studio tool chain: **RX Family C/C++ Compiler**.

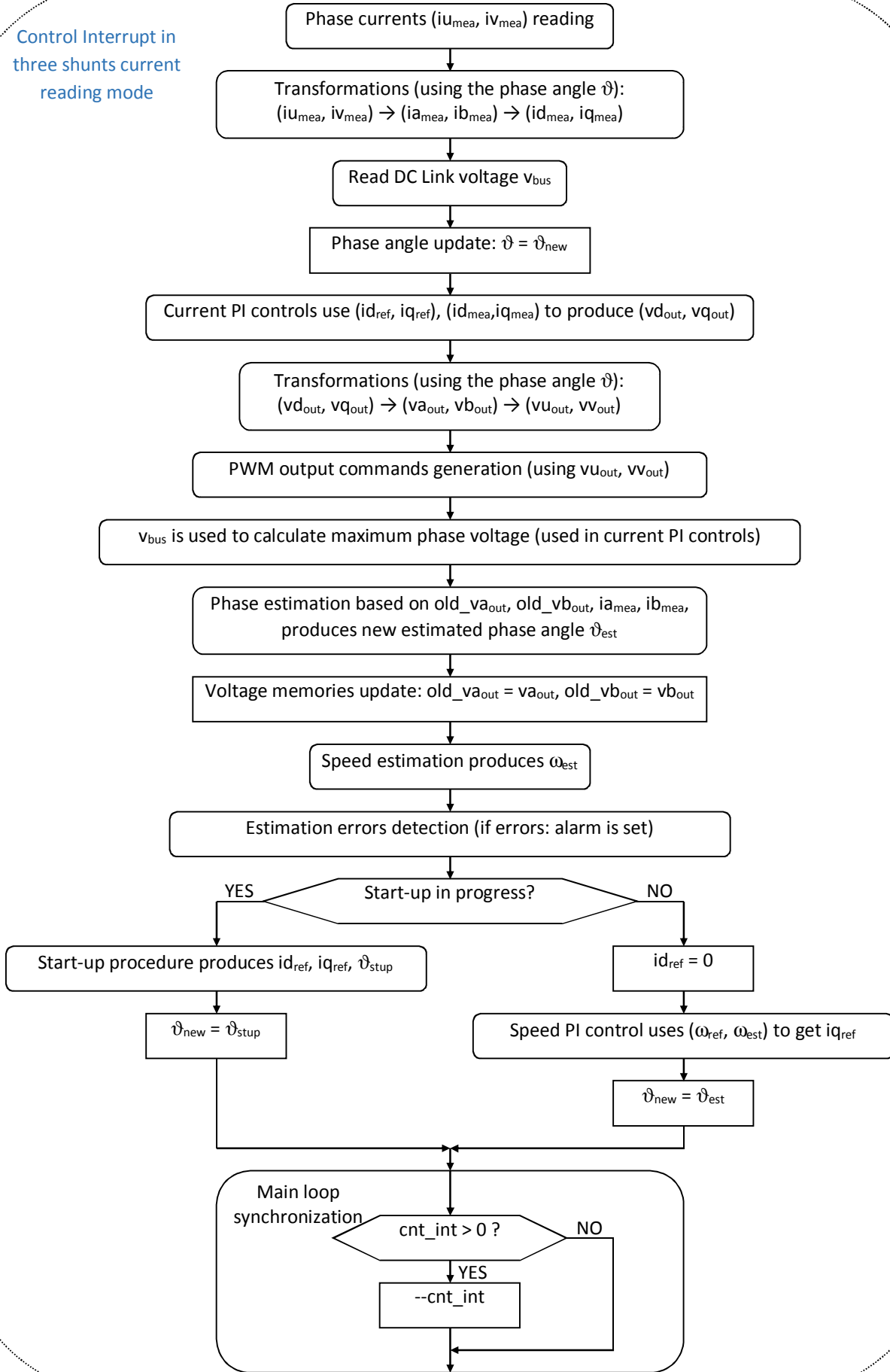
The following flowcharts show the software implementation of the motor control part of the software.

Please find below the flowchart for the main loop, the interrupt service routines and the Automatic Tuning.

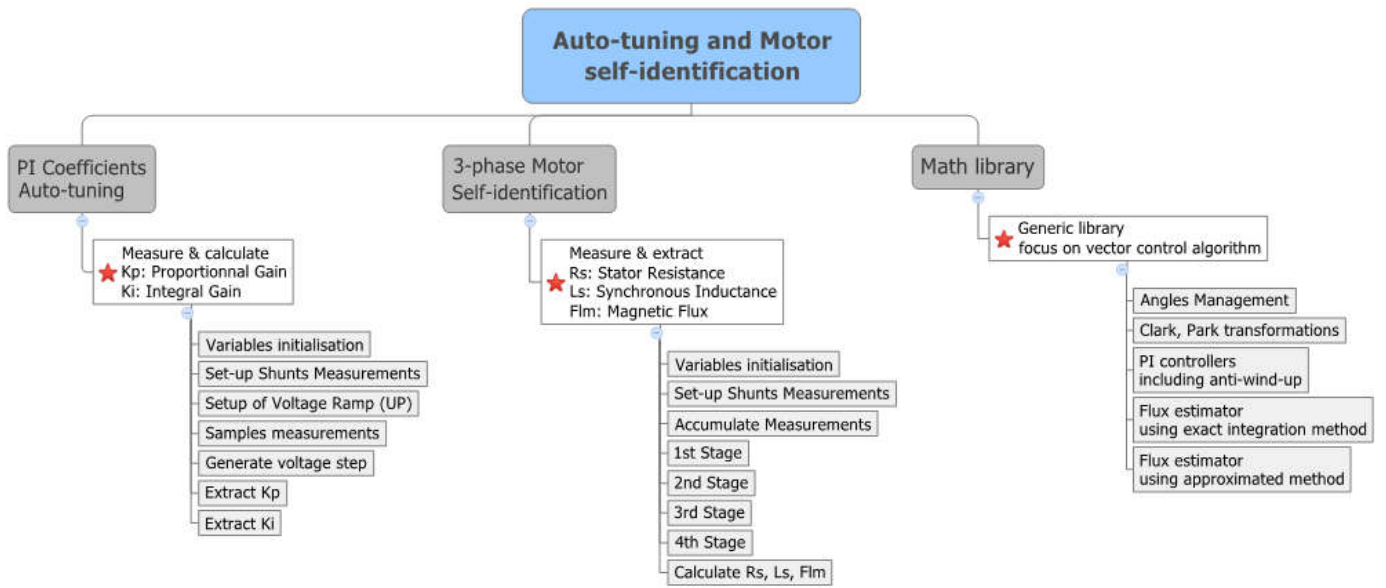




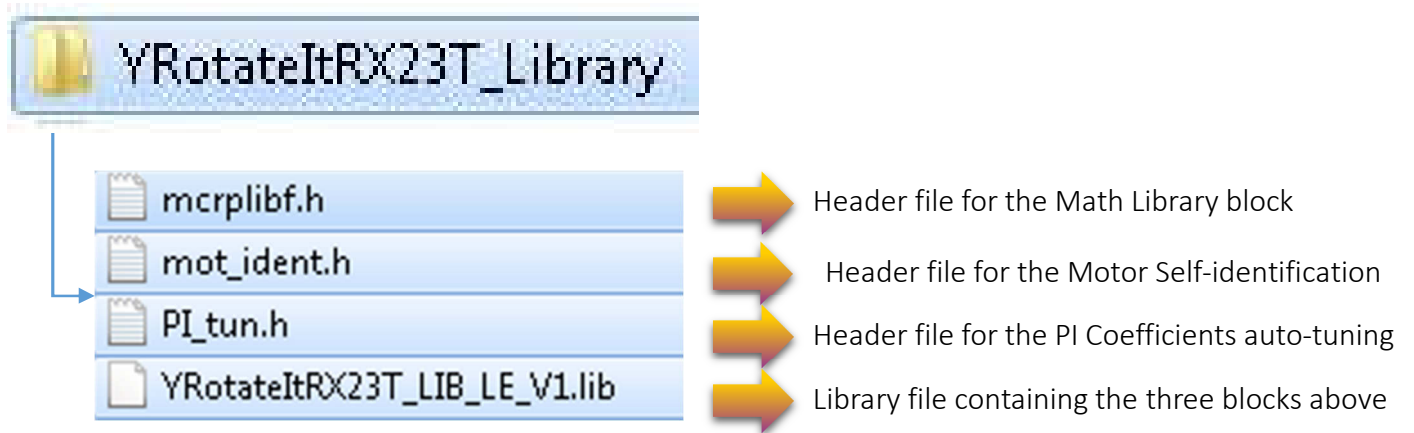
Control Interrupt in three shunts current reading mode













The auto-tuning process and the self-identification mechanisms are fully independent from the main sensorless vector control software and can be used in the 1st phases of evaluation and configuration of the software.



















The three blocks mentioned above are located in the library called: "YRotateItRX23T_LIB_LE_V1.lib" located in the folder called: "YRotateItRX23T_Library" as shown below:



The complete project source code under e2studio is described below. For each C module, a specific header file is associated.

 YRotateItRX23T_Library	➔	Contains PI auto-tuning, Motor identification and estimators routines
 dbstc.c	➔	Renesas Project Generator: setting of B, R sections
 ges_eqp.c	➔	External EEPROM management functions: read, write via I ² C up to 64 parameters. By default, 21 are used by the PC GUI
 hardware_setup.c	➔	Hardware setup definition for each MCU pins and SFRs
 interrupt_handlers.c	➔	Interrupt Service Routines definition: linked to the A/D converter to launch the control loop
 main.c	➔	Main loop including initialisation of PWM, measurements, enable interrupt, serial communication, LED management...
 motorcontrol.c	➔	Complete FOC algorithm in interrupt, including POE management, modulation, etc.
 reset_program.c	➔	Reset procedure
 userif.c	➔	Serial protocol used to manage the PC GUI communication
 vector_table.c	➔	Define addresses for each interrupt service routines

Please find below the important Header files included into the project folder.

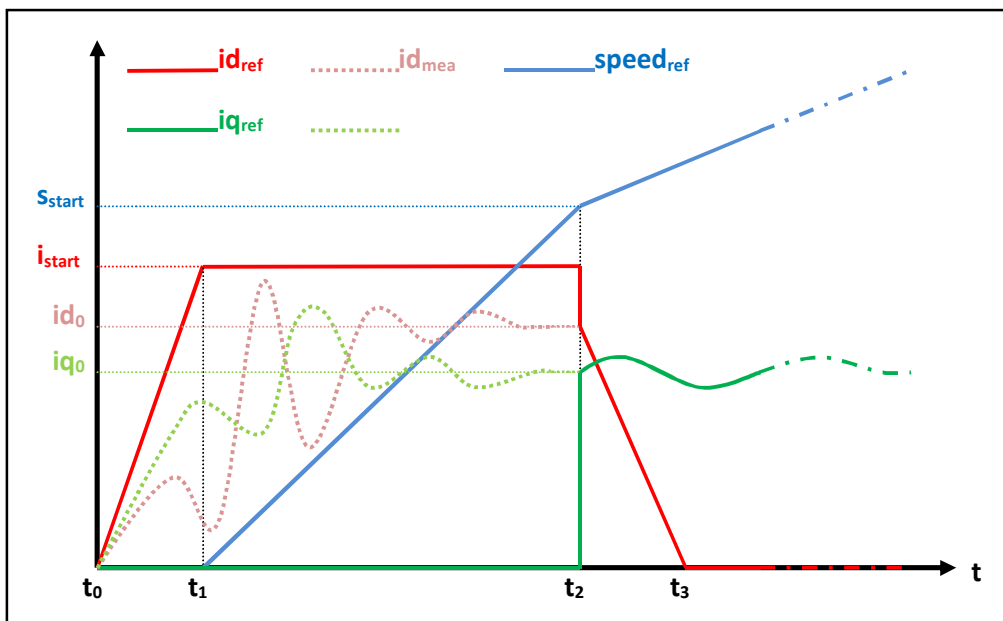
 const_def.h	➔	Definition of constants: Base Timing, PWM parameters, dead-time, Duty cycles, trigger delay for A/D, conversions constants, alarms, interrupts prio.
 customize.h	➔	Definition application parameters, switches to enable Auto-tuning, over-modulation, estimators selection, flux weakening, etc.
 defpar.h	➔	3-phase Permanent Magnet Motor – Default parameters used by the kit: NANOTEC DB4253, 24V, 4000RPM nominal speed.
 dtc_typedef.h	➔	Data Transfer Control (DTC): data structure definition
 ges_eqp.h	➔	EEPROM management functions, variables and constants
 globdef.h	➔	Global Definitions
 globvar.h	➔	Global Definitions
 iodefine.h	➔	Microcontroller I/O definitions
 mask.h	➔	Mask definitions used
 motorcontrol.h	➔	Definition of functions used in the Interrupt service routines
 pws_E6108A.h	➔	Parameters used to drive the External power stage using IPM
 pws_E6132.h	➔	Parameters used to drive the internal power stages using MOSFETS
 stacksct.h	➔	Stack and data types definition
 typedefine.h	➔	Stack and data types definition
 userif.h	➔	Parameters used in the serial protocol
 vect.h	➔	Vectors definition

17. 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 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 start-up procedure (in case of three shunts current reading) is described below.

- 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 $\vartheta_a=0$. All the references: $i_{d,ref}$, $i_{q,ref}$ and $speed_{ref}$ are set to zero.
- From the moment t_0 , while the $i_{q,ref}$ and the $speed_{ref}$ are maintained to zero, $i_{d,ref}$ 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, $i_{d,mea}$ and $i_{q,mea}$. 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, $i_{d,mea}$ and $i_{q,mea}$ have no physical meaning. Even if they are not shown in the graph, the applied voltages are subjected to the same treatment ($v_{d,mea}$ and $v_{q,mea}$ are calculated in the algorithm).

- At $t = t_1$, while $i_{q,ref}$ is maintained to zero and $i_{d,ref}$ 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 $i_{d_{mea}}$ and $i_{q_{mea}}$ begin to be similar to the real flux and torque components of the current. The real components are supposed to be i_{d_0} and i_{q_0} (those values are obtained applying a low-pass filter to $i_{d_{mea}}$ and $i_{q_{mea}}$).

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 i_{d_0} and i_{q_0} , and all the PI controllers are initialized with these new values. The speed PI integral memory is initialized with the value i_{q_0} , while the current PI integral memories are initialized with the analogous voltage values v_{d_0} and v_{q_0} , obtained from $v_{d_{mea}}$ and $v_{q_{mea}}$.

- 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 i_d 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 i_q current reference is obtained as output of the speed PI controller.

18. Reference system transformations in details

Find below the detailed equations used for the coordinates transformations in the embedded software for the RX23T 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)

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

19. Rotor position estimation

The rotor position estimation method which has been chosen is the direct integration of the back EMF. Such method is enable by default in the RX23T inverter kit.

Please find below the fundamental equations:

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

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

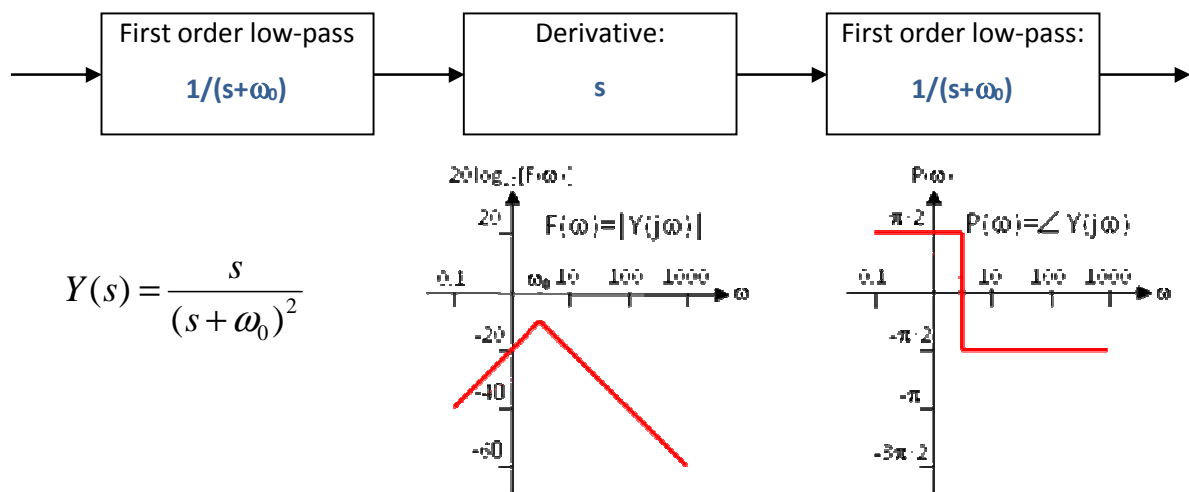
$$\vartheta = \arctan(x/y)$$

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

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 high-pass filter, as in the following scheme:



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

That's why, a specific parameter related to the estimator can be tuned via the PC GUI. It is the parameter n°18 "App. FE Time Constant" in "ms" which is the Filter time constant used in the approximate integration flux estimation method. Please find below more technical details about it.

Regarding the Filter time constant, the BEMF integral is approximated with a function composed by two low pass filters and a derivative. In discrete time domain, the expression for the low pass filter is:

$$y(n) = ((k-1)/k) * y(n-1) + (1/k) * x(n)$$

Where $(k * T_c)$ is the so called time constant of the filter and T_c is the sampling period

In the new software release for RX23T: the parameter to be specified is the time constant on the filter in ms, which is much more flexible and also more immediate to understand.

In fact the cut-off frequency of the filter is directly:

$$1/(2 * \pi * \text{time_const}[s]) = 1000 / (6.28 * \text{parameter_value}[ms]).$$

Internally k is calculated as $1000 * \text{parameter_value}[ms] / T_c$, and the filter is implemented performing a division, so there's no restrictions on the value and the sampling frequency is now variable.

For example, if the `parameter_value` = 32, so the `time_constant[s]`=0,032 and the `cutoff_freq`=4,97Hz, internal $k=0,032 * \text{sampling_freq}$.

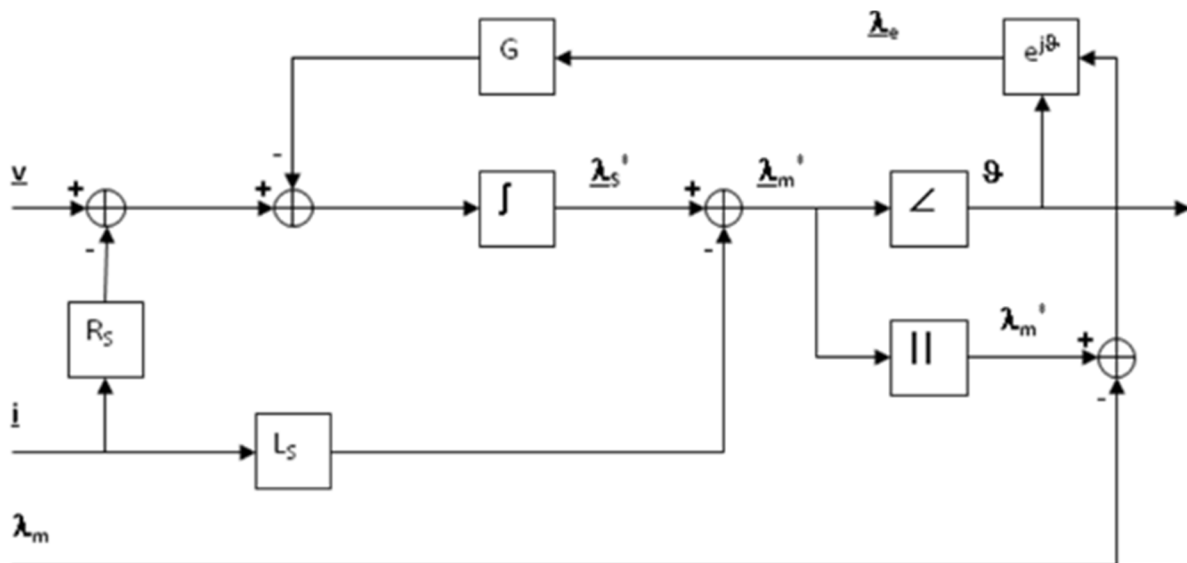
That's why, at 8KHz, please find below some calculation about the possible values to be used to tune the estimators:

Para. 18 in ms App. FE Time Constant	Cut-off Frequency
128	1.25Hz
64	2.5Hz
32	5Hz
16	10Hz

Obviously high values of the cut-off frequency produce poor behaviour at low speed and good behaviour at high speed, and vice-versa.

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:



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 dependant.

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 YROTATE-IT-RX23T kit, this **second** strategy is selected. The choice to test the first one is left to the user thanks to the setting of the macros in the source code. Such modifications required a compilation of the embedded software. Within the software module "customize.h", please uncomment the define APP_INT to enable the approximation method.

```
// Flux estimation method selection
// #define APP_INT      // uncomment this if you prefer approximated integration
```

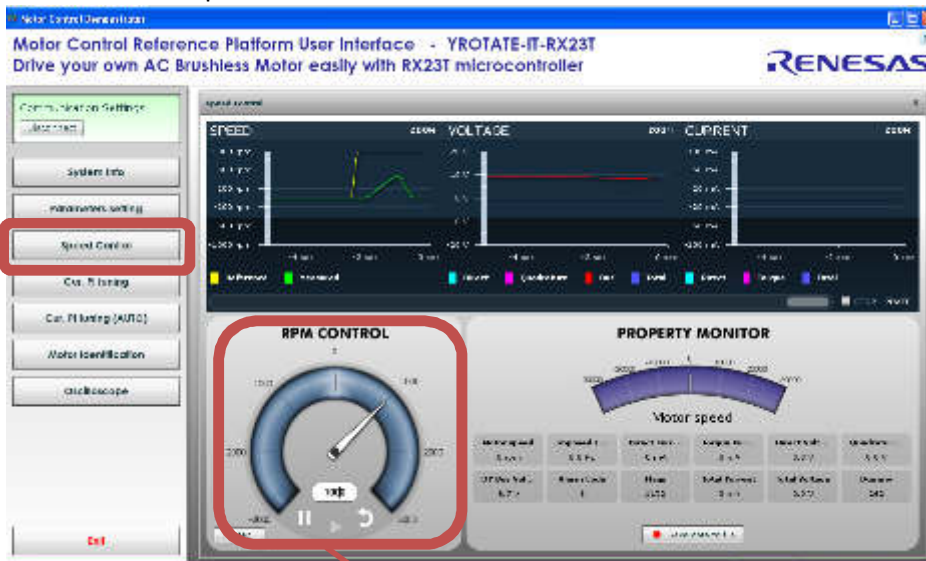
20. PC Graphical User Interface in details

Please install the Motor Control PC GUI on your machine by following the instructions of the Quick Start Guide delivered in the YROTATE-IT-RX23T kit. After connecting the Nanotec Motor (DB42S03, 24V_{DC}, 4000RPM), please connect the board RX23T and select the COM port or use the Auto-detection mechanism.

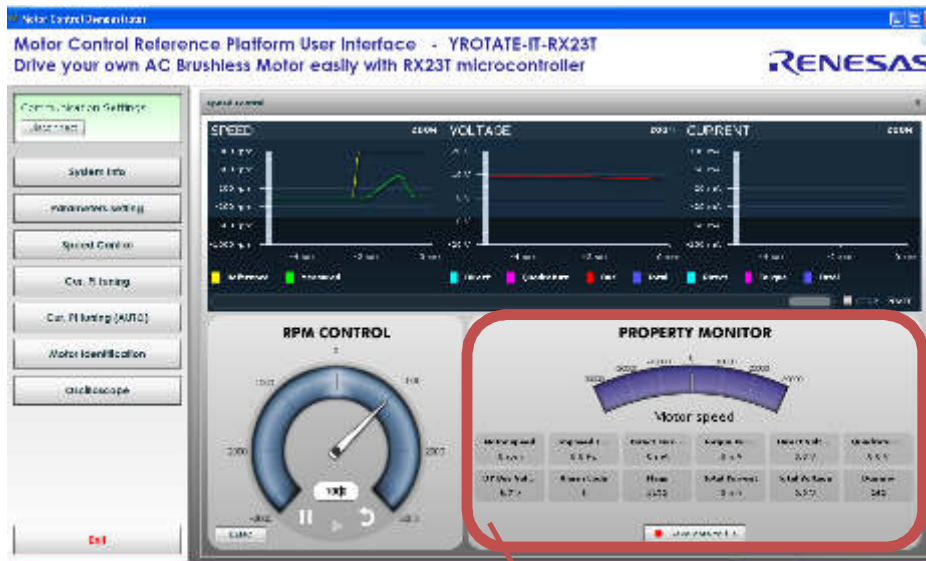
The PC Graphical User interface supports such Operating Environment:

- 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 description of the PC GUI tabs and windows.

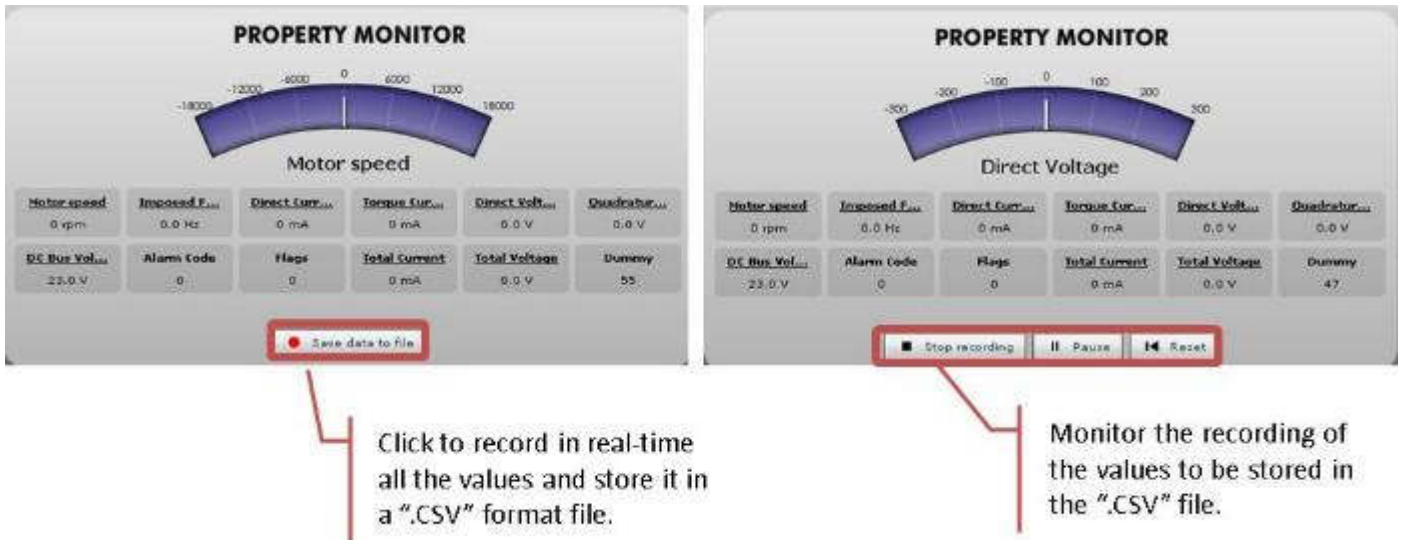


Set motor speed, stop it, and reverse

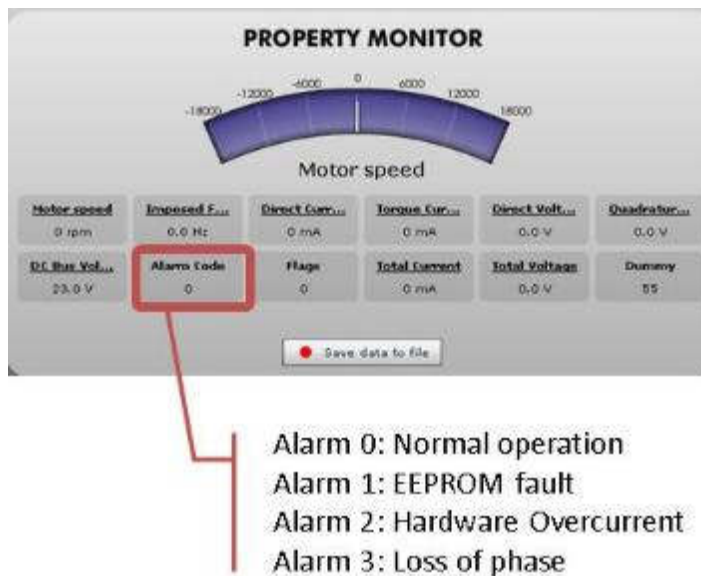


Display internal quantities (Motor speed, torque and direct current, frequency, etc)

By clicking on the button “Save data to file, it becomes possible to record regularly all the values display in real-time in a file, as describe below:



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



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 LED DL2 is quickly blinking on the main board to indicate that an alarm is set.

The maximum and minimum values are specified in the two constants tables called: "par_max[]" "par_min[]" in the "ges_eqp.h" header 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 or switching off the power supply.

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 “hardware overcurrent” and described in the software by “FAULT_ALL”. This alarm is produced by the MCU peripheral called Port Output Enable (POE) in case of external overcurrent signal. The hardware overcurrent is producing a falling edge input on the POE pin. Furthermore, if the hardware level of the PWM output pin is not coherent with the level imposed by software, the alarm 2 will also be triggered.

The LED DL2 is quickly blinking on the main board to indicate that an alarm is set.

The only way to clear the alarm is to reset the board by using the reset button on the PCB or by switching off the supply and on again.

Finally, one of the root causes of the Alarm 2 is a hardware defect or a wrong behaviour of the current control. So please also check the setting of the current PI coefficients that are stored in EEPROM or used in real-time.

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.

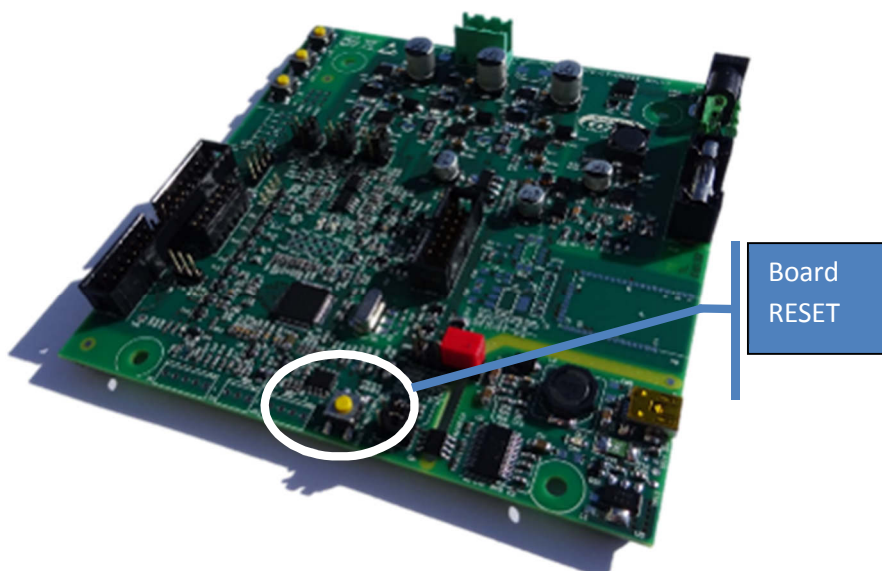
The LED DL2 is quickly blinking on the main board to indicate that an alarm is set.

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

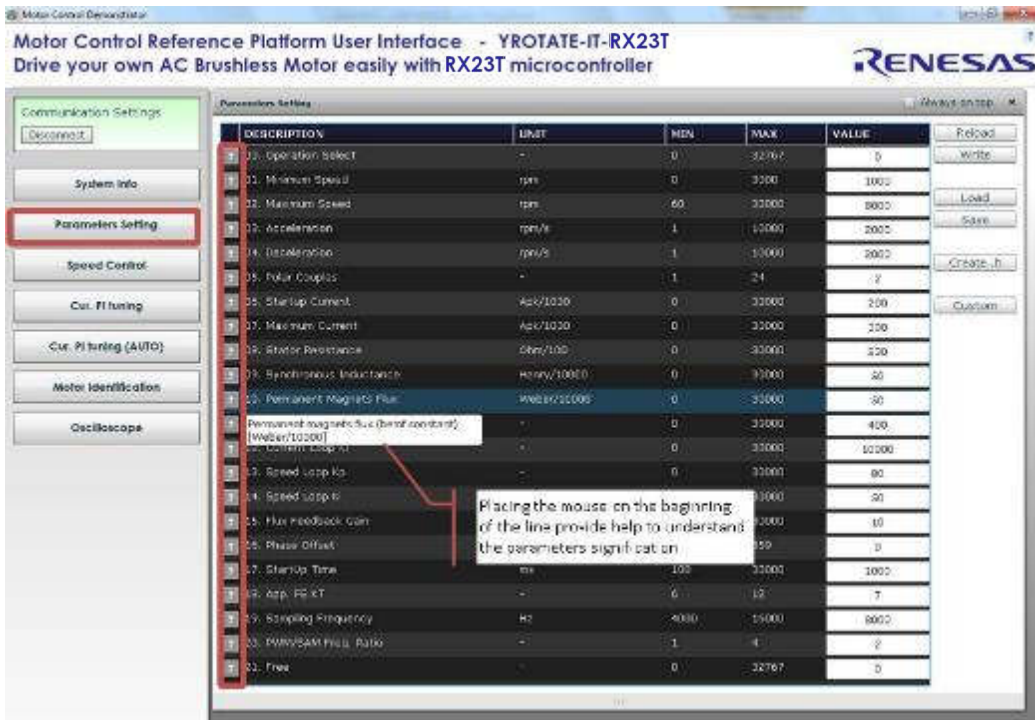
Please find below an extract of the header file “const_def.h”.

```
// Alarms
#define EQP_ALL          ( 1 )    // EEPROM alarm code
#define FAULT_ALL       ( 2 )    // Overcurrent hardware alarm code (POE)
#define TRIP_ALL        ( 3 )    // Loss of phase alarm code
```

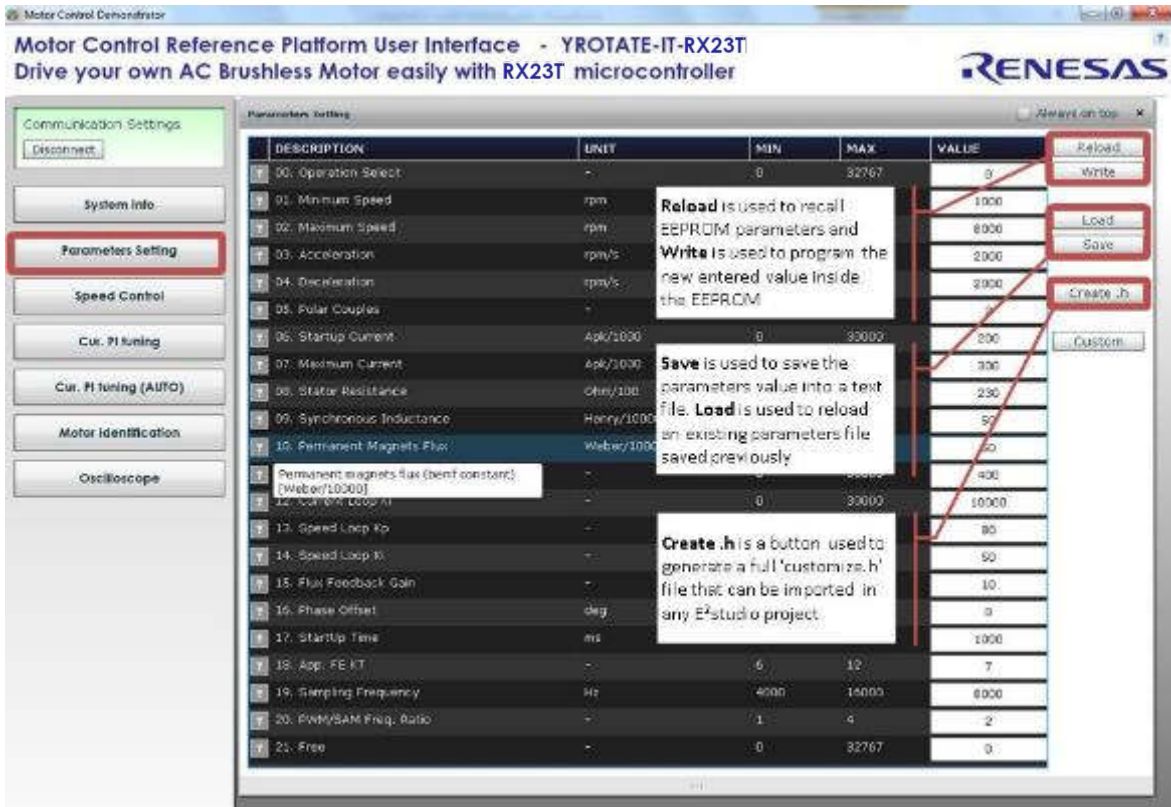
In order to reset the board in case of Alarm code 2 “Overcurrent”, please push the button as described below:



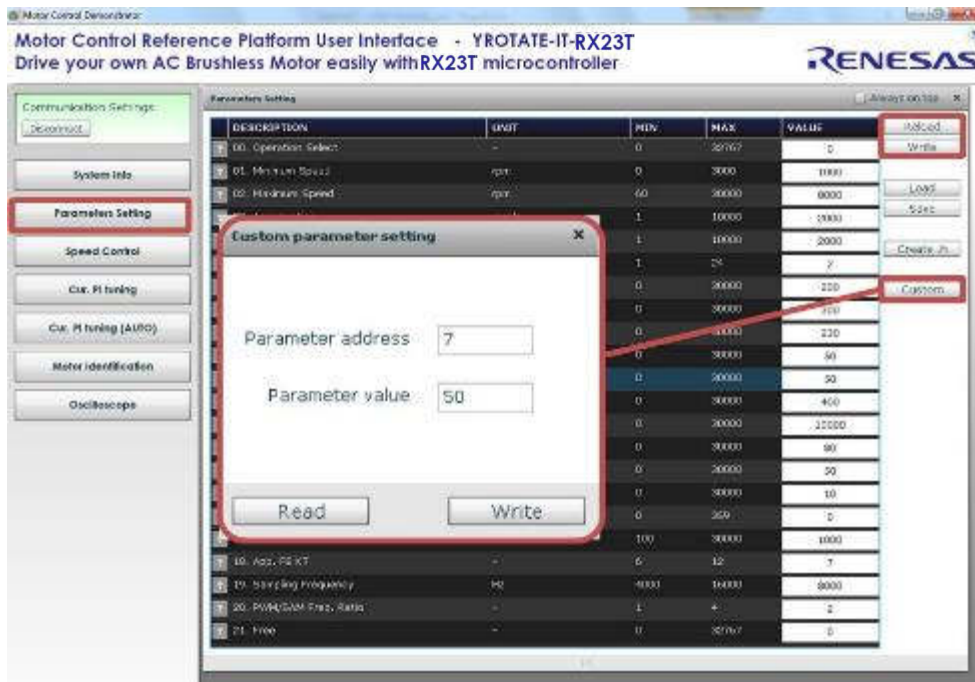
Then by clicking on the “Parameters Setting” button, the important window can be displayed showing all the parameters of the system that can be changed in real-time without having to recompile the embedded software.



The detailed description of each parameter is display when pointing the mouse on the question mark. Each parameters unit is displayed. To change one value in real-time, simply enter the new value and click on “Write” to program the new value into the EEPROM.



Furthermore, it is possible to change only one parameter at a time during fine tuning, when the motor is rotating by using the “Custom” button and enter the number of the parameter and the value inside the window as shown below.



All the parameters can be changed on the fly and after pushing the “Write” button, it’s automatically set.

But there are two exceptions related to the parameters #19 and #20:

19	SAM_FRE_DEF	Set the sampling frequency [Hz] of the control loop
20	F_RATIO_DEF	Set the ratio between the PWM frequency and sampling frequency, e.g. if 8000 is set in the parameter #19 and 2 in the parameter #20, the PWM frequency is 16KHz.

The parameter #19 is setting the control loop speed. If 8KHz is selected by entering the value “8000”, the PWM frequency can be set to four different values depending on the motor and the applications either 8KHz, 16KHz, 24KHz or 32KHz.

=> It’s basically done by entering the ratio value in the parameter #20. Please find below the possible values that can be entered.

Parameter 19: Sampling freq.	Parameter 20: Ratio = 1	Parameter 20: Ratio = 2	Parameter 20: Ratio = 3	Parameter 20: Ratio = 4
4KHz	PWM freq.: 4KHz	PWM freq.: 8KHz	PWM freq.: 12KHz	PWM freq.: 16KHz
8KHz	PWM freq.: 8KHz	PWM freq.: 16KHz	PWM freq.: 24KHz	PWM freq.: 32KHz
10KHz	PWM freq.: 10KHz	PWM freq.: 20KHz	PWM freq.: 30KHz	PWM freq.: 40KHz
12KHz	PWM freq.: 12KHz	PWM freq.: 24KHz	PWM freq.: 36KHz	PWM freq.: 48KHz
14KHz	PWM freq.: 14KHz	PWM freq.: 28KHz	PWM freq.: 42KHz	PWM freq.: 56KHz
16KHz	PWM freq.: 16KHz	PWM freq.: 32KHz	PWM freq.: 48KHz	PWM freq.: 64KHz

In order to change the values of the parameters #19 and #20, please follow the description below.



After entering the new values, in this case 8KHz of sampling frequency and 16KHz of PWM frequency, click on “write” and Push the Reset button of the board. It’s only after the Reset of the board that the new PWM and loop frequencies will be set in the embedded software.

Important Note: After setting up the new values for the parameters #19 or #20, it’s recommended to run the Auto-calibration procedure described below. It ensure the software to use the best intrinsic values and the most adapted values of the current PI coefficients.

Speed range limitations

The YROTATE-IT-RX23T kit is driving any 3-phase Permanent Magnet Motors 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 Nanotec Motor DB42S03 delivered with the kit, the minimum speed is **600RPM**. Below this speed, the current flowing through the three shunts are too low to be detected. Furthermore, when the board is supplied only via the USB cable, the maximum current provided to the board is limited by the **500mA** of the USB PC port and the voltage generating by the board which is **12V**.

It means that, the first tests using the 3-phase Brushless AC motor DB42S03 from Nanotec will work properly in a specific speed range: from **600RPM** up to **3500RPM**.

The DB42S03 brushless motor is able to reach its maximum speed of 6200RPM (without load) when the power supply is 24V and up to 1A is provided. After changing the jumpers as described above and providing 24V to the board, the Nanotec motor from the YROTATE-IT-RX23T kit reach easily 6200RPM, its maximum rated speed without load. Of course, the embedded software enabling flux weakening technics, by providing more current to the board, the motor can reach **8000RPM**.

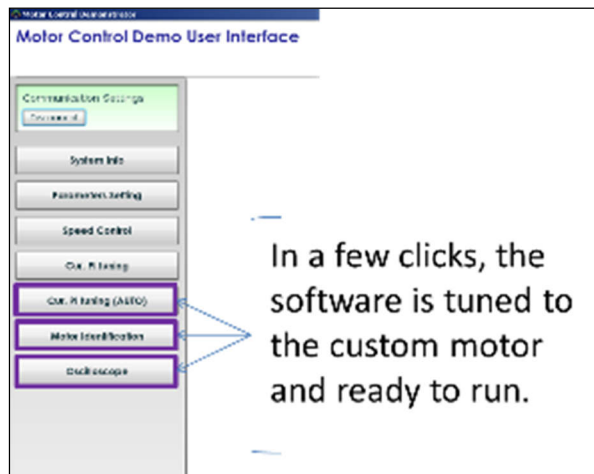
21. EEPROM parameters: detailed description

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

Parameter number	Short name	Description
0	SEL_OP	default parameters setting, Used to perform special operations, like default parameter set re-loading, or current PI tuning working mode setting
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_RES]. 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_RES]
8	R_STA	Set the stator resistance [Ohm/OHM_RES]
9	L_SYN	Set the synchronous inductance [Henry/HEN_RES]
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	FB_GAIN	Set the flux amplitude feedback gain. 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
16	PHA_OFF	Set the phase offset [deg]. It is used to add a phase offset to the phase estimation, to reach better alignment
17	ST_TIM	Set the Start-up acceleration time [sec/SEC_RES]
18	Not used	
19	SAM_FRE_DEF	Set the sampling frequency [Hz] of the control loop
20	F_RATIO_DEF	Set the ratio between the PWM frequency and sampling frequency, e.g. if 8000 is set in the parameter #19 and 2 in the parameter #20, the PWM frequency is 16KHz.

22. Motor Auto-calibration using the PC GUI

The full calibration of any 3-phase AC Brushless motor can be performed automatically using the PC Graphical User Interface. Three specific buttons are now available for and shown below:



In terms of AC Brushless motor driven in sinusoidal mode and FOC algorithm, the most important parameters to tune are:

1. Current PI parameters: **Proportional K_p** and **Integral K_i**
2. Motor parameters: **Stator resistance R_s** , the **synchronous inductance L_s** , and **Permanent Magnet flux Λ_m** .

Please find below the auto-tuning process step by step of the Nanotec Motor DB42S03 delivered with the YROTATE-IT-RX23T kit. The DB42S03 motor is a low voltage Permanent Magnet Synchronous Motor. The auto-tuning procedure will be performed using the kit running the sensorless vector control algorithm.

Important Note: The auto-tuning embedded software is working only on the three shunts version.

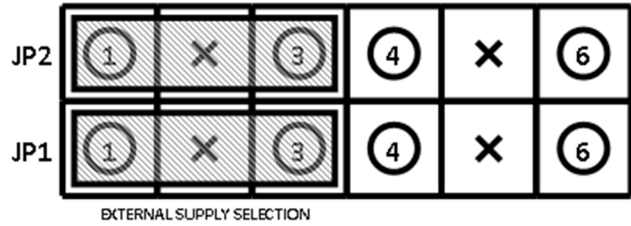
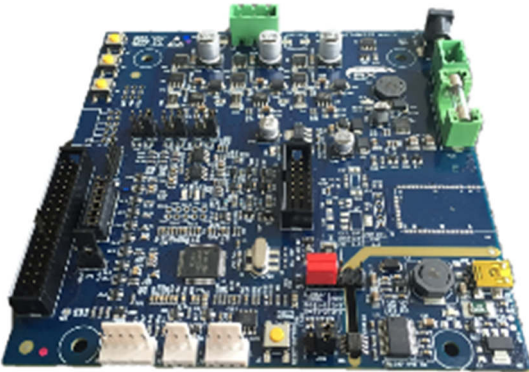
a) Please find below the specifications of the Motor delivered by the motor maker:

- Motor Manufacturer: **NANOTEC** www.nanotec.com
- Motor type: 3-phase AC Brushless **DB42S03**
- Maximum current: **5.4A**
- Bus Voltage: **24V**
- Speed rated: **4000 RPM**
- Number of pole pairs: **4**

SPECIFICATION	CONNECTION	DELTA
NO. OF POL./PHASE		8/3
VOLTAGE RATED (VDC)		24
CURRENT NO LOAD/RATED/PEAK (AMP)		0.2/1.79/5.4
RESISTANCE/PHASE TO PHASE (Ohms) @25°C		1.5±15%
INDUCTANCE/PHASE TO PHASE (mH) @1KHz		2.1±20%
TORQUE RATED/PEAK (Nm) [lb-in]		0.0625/0.19 [0.553/1.68]
TORQUE/VOLTAGE CONSTANT (Nm/A)/(Vrms/KRPM)		0.035/2.78=BACK EMF
POWER RATED (W)		26
SPEED RATED/NO LOAD (U/min)		4000/6200



b) Let's setup the Motor control kit for 24V_{DC} external power supply: the jumper JP1 and JP2 needs to be set to 1-3 position as explained in the "Chapter 3 Power Supply selection".

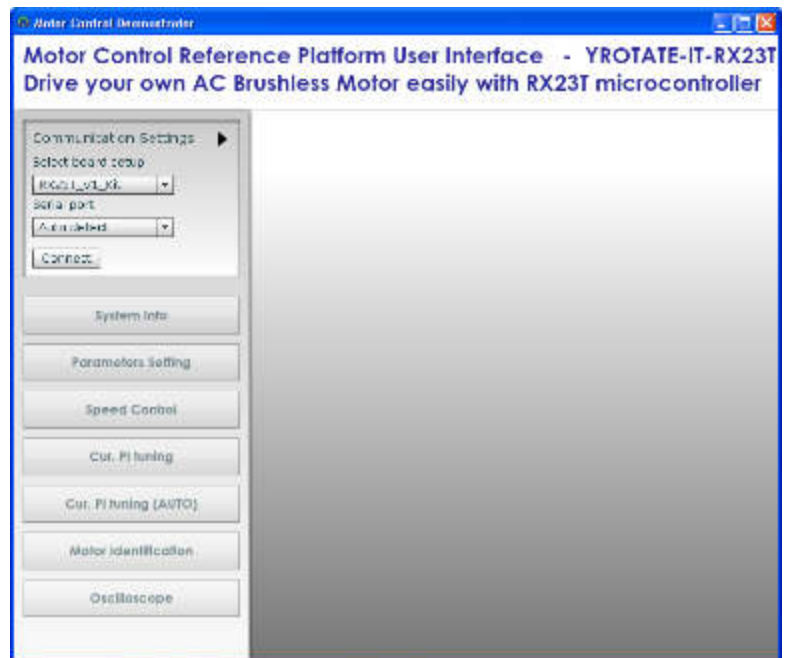
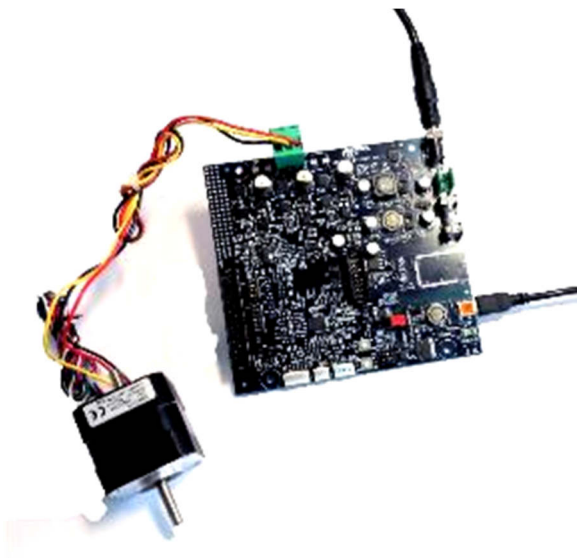


c) Let's connect the 24V_{DC} Power supply to the RX23T motor control reference kit.

The following LEDs: **DL3**, **DL4**, **DL5** are ON and the LED **DL2** is blinking to indicate that the MCU is running fine.

Important Note: The procedure below is also working with the USB power supply, but it is recommended to connect 24V_{DC} external power supply to the inverter kit. It ensure a higher resolution during the auto-tuning procedure and the extraction of the intrinsic motor parameters as the level of energy is higher.

d) Now, connect the USB cable to the PC and the Kit and connect the 24V to the kit and the motor to the kit:



e) Launch the PC GU from the folder: "C:\Program Files\MCDemo" launch: "MotorController.exe" The LED **DL1** of the RX23T board is blinking rapidly showing communication between the board and the PC.

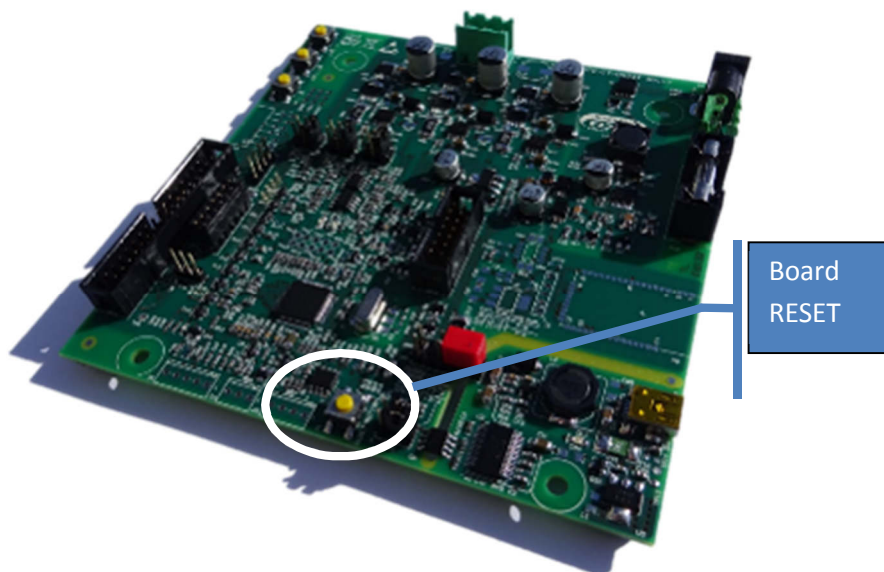
Click on the “setup” button and select “RX23T Kit” and select “Auto detect” and click on “Connect” to ensure the PC GUI is connected to the RX23T kit.

On the left hand side, the new buttons appears: “Cur. PI tuning”, “Cu. PI tuning (AUTO)”, “Motor Identification” and “Oscilloscope” which are needed for the self-calibration of the motor.

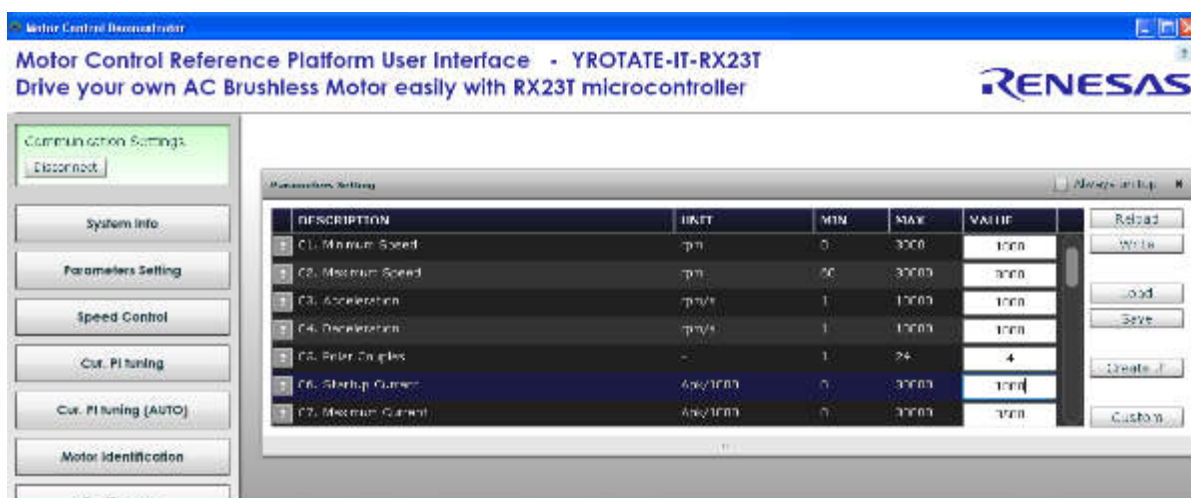
f) Clean the EEPROM content and start with the default parameters in the EEPROM.

The first thing to do is to ensure that the inverter board is the default state and the default parameters are written inside. The procedure below ensures it:

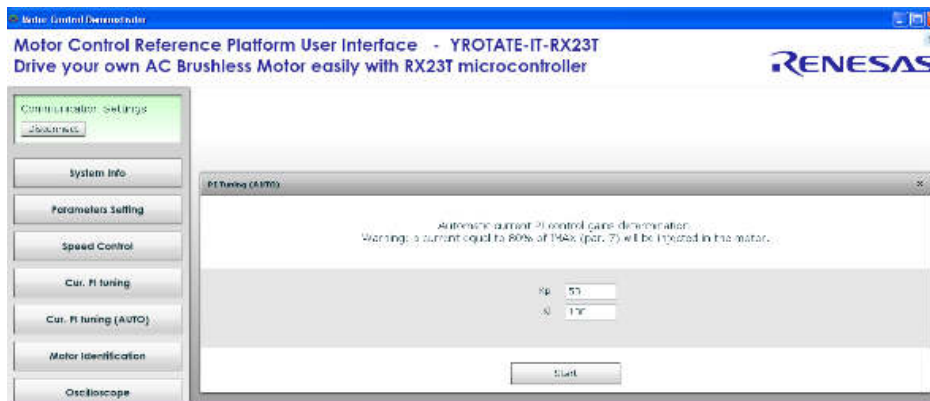
Click on the “Parameters Setting” button and enter the magic value “33” in the “00. Operation Select” and push the RESET button on the board as shown:



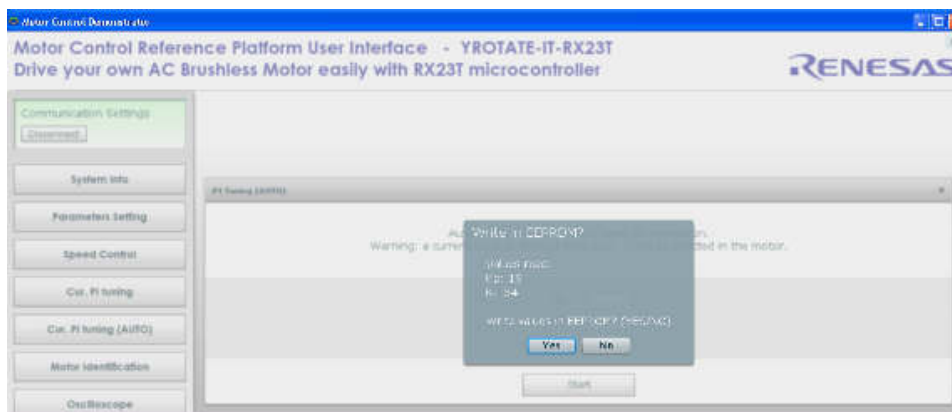
g) Set the **maximum current** (parameter n°07) as it will influence all the next steps: Click on “Parameters settings” Enter the value: **3500** (the unit is in mA) and click on “Write” to save the parameter into the EEPROM and close the parameter setting window. The maximum current parameter is fundamental for the auto-calibration. The maximum value allowed by the motor must be used to guarantee the highest resolution.



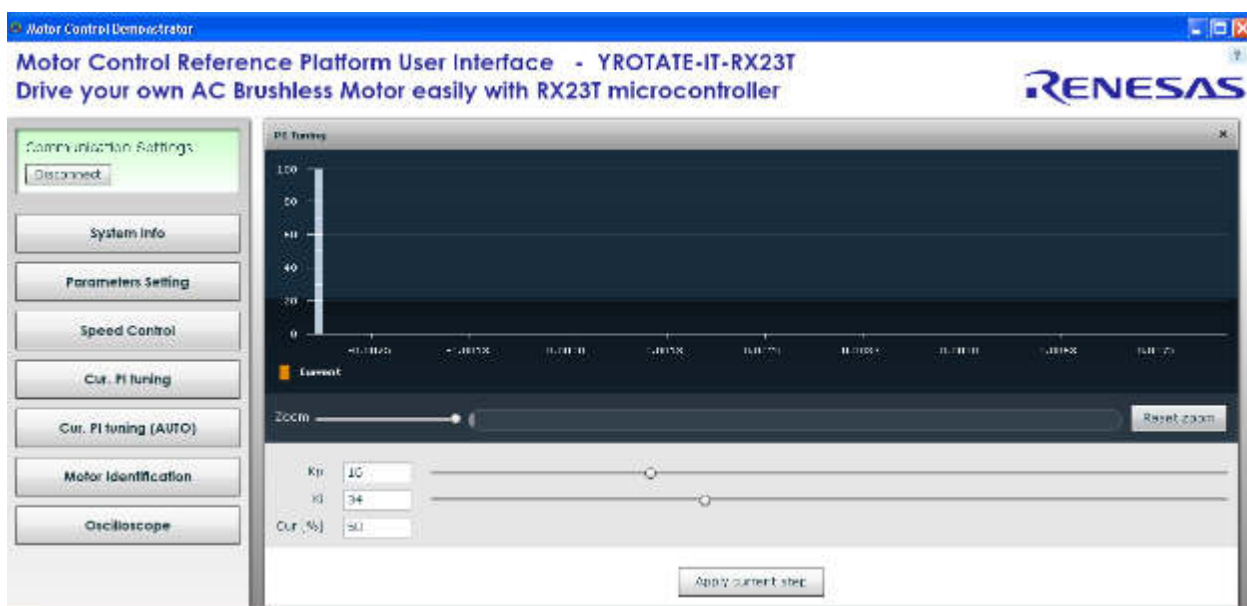
h) Click now on “Cu. PI tuning (AUTO)” button and press “start” to perform an automatic Current PI tuning. The two coefficients of the PI current block will be extracted thanks to the embedded software able to generate a step voltage and measuring the motor response.



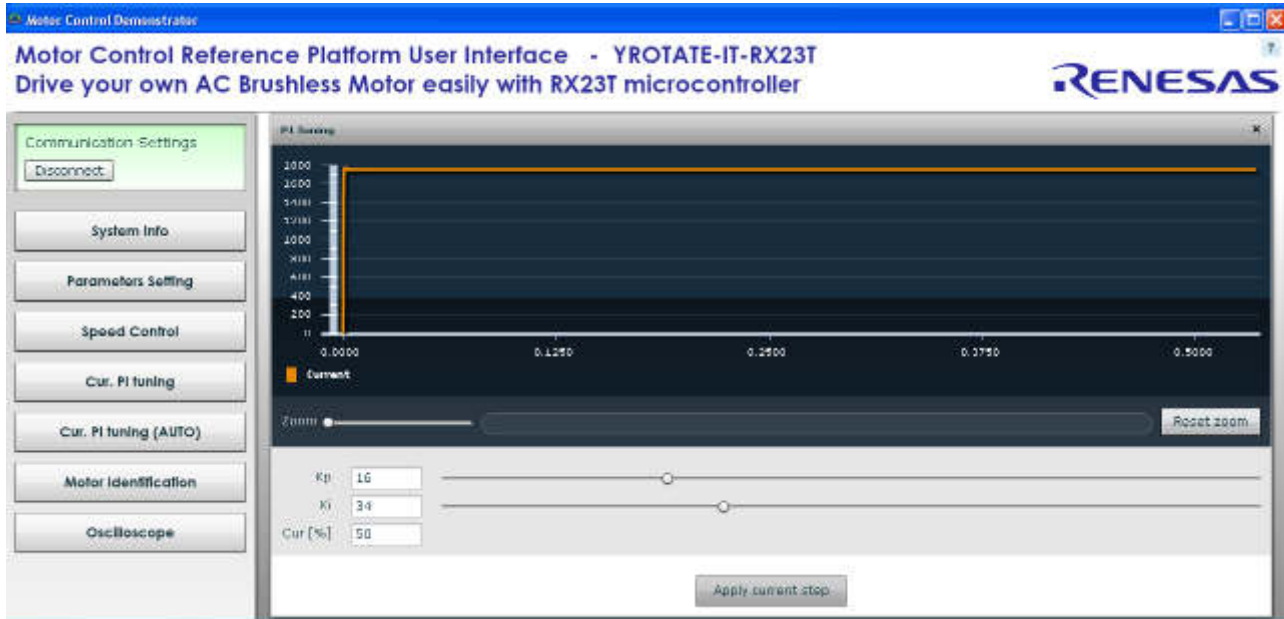
And click on “Yes” to accept the results to be programmed into the EEPROM as shown below.



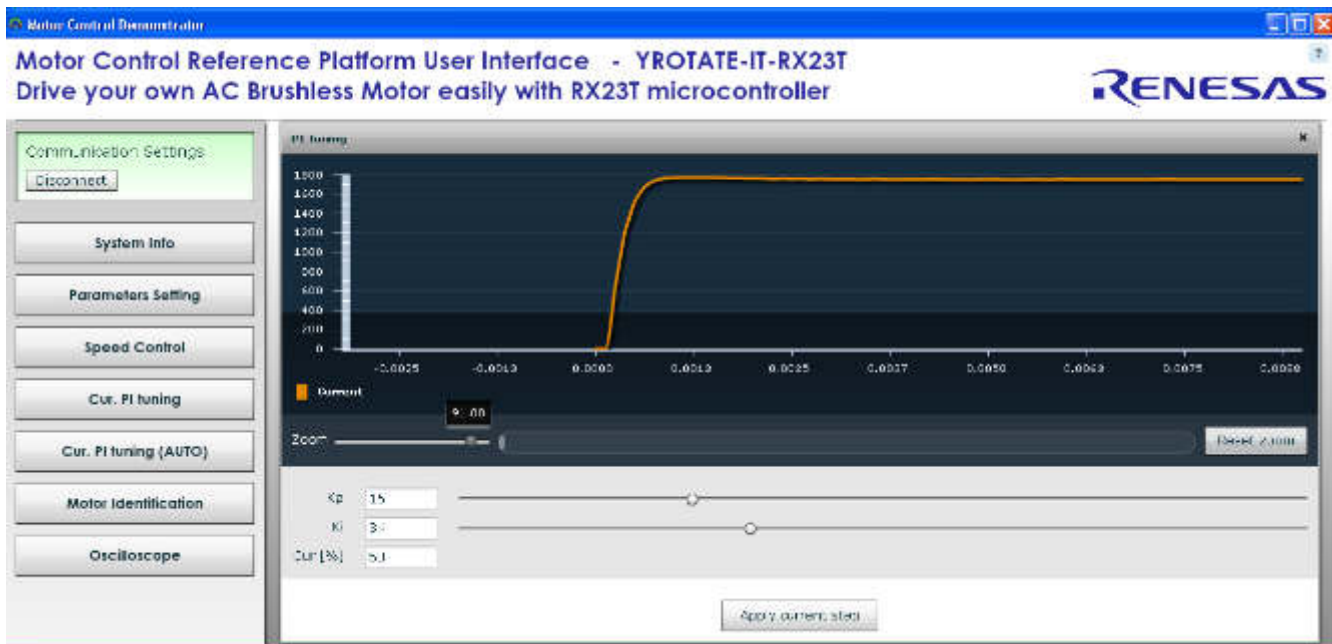
i) Now click on the button “Cu. PI tuning” to open the manual current PI tuning window and check the step answer by clicking on “Apply current step” button.



Depending on the motor, the parameters found by the automatic procedure can be too fast or too slow.



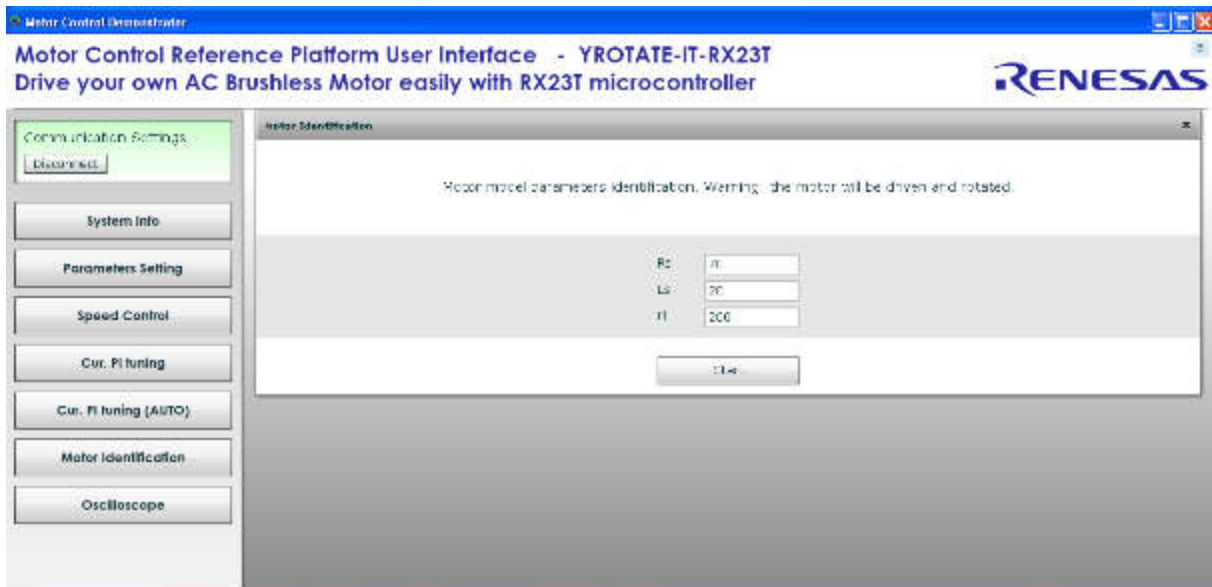
Please use the Zoom function to check the beginning of the step:



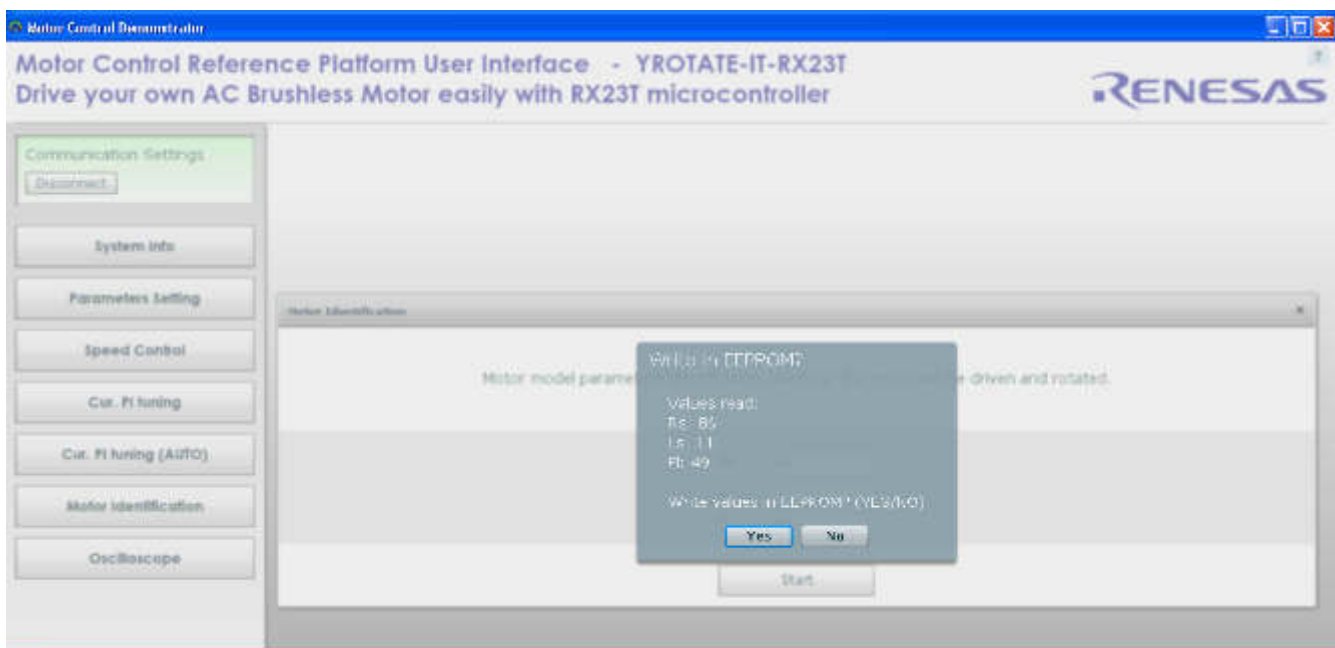
You can adjust manually the parameters to obtain an even better step response and also increase the step current level by increasing the percentage of “Cur. [%]” to 90%. The default value is 50%.

Once it’s done, the window can be closed as the proportional and integral coefficients of the PI current are tuned.

j) Perform an auto-identification of the motor parameters by clicking on “Motor Identification” and click “start”:



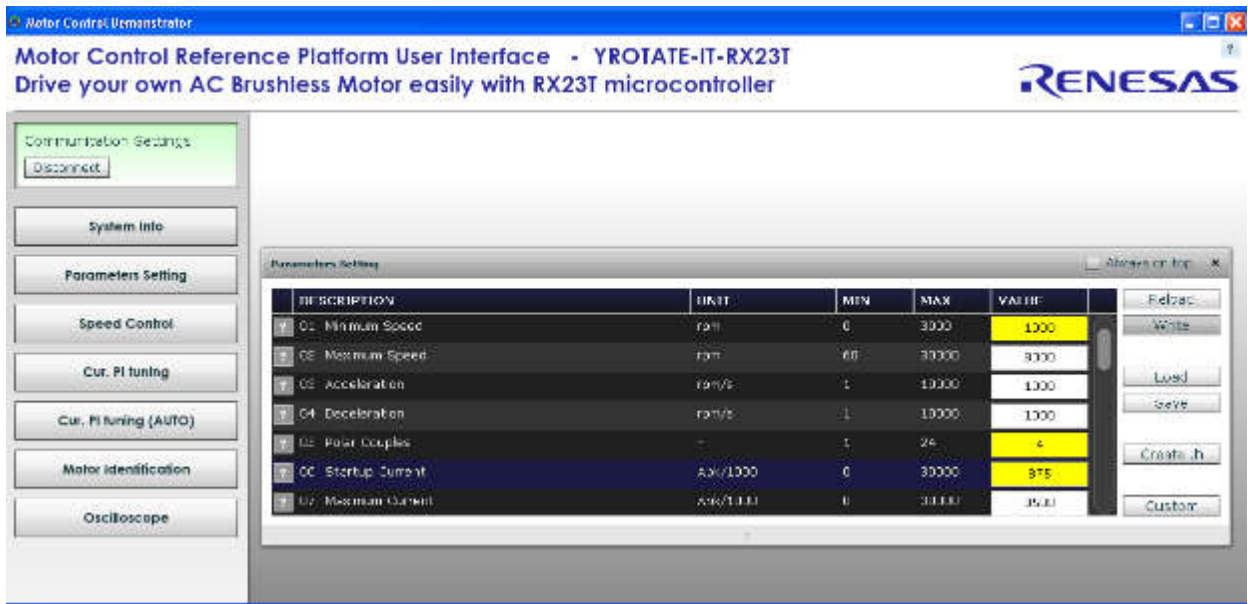
During this process the rotor should start rotating, please leave the rotor free and no loaded. And finally accept the results to store them into the EEPROM by clicking on “yes”.



The stator resistance, the synchronous inductance and the Permanent Magnet flux have been measured and tuned.

k) Now please click on “parameters settings” and enter the number of pole pairs: **4** (parameter n°5) and enter a minimum speed or 1000 RPM

l) Set a start-up current equal to 25% of the maximum current. In our case 25% of 3.5A is 0.875A. Please enter the value 875 into the parameter n°6 and click on the “write” button on the right hand side.



Then let's close the window.

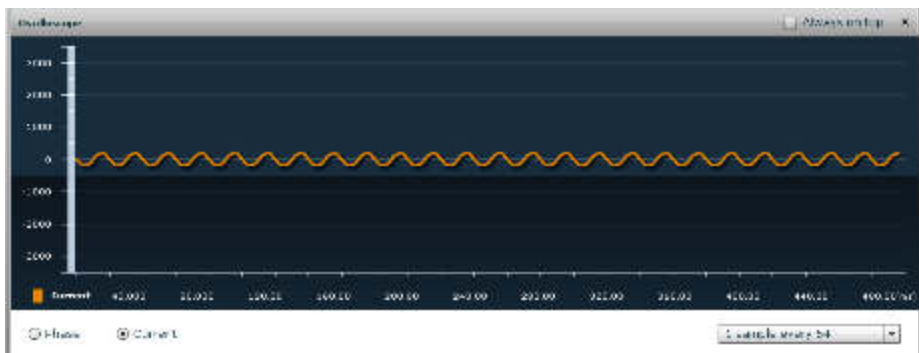
j) Now, let's try to run the motor. Please click on the button: "Speed Control":



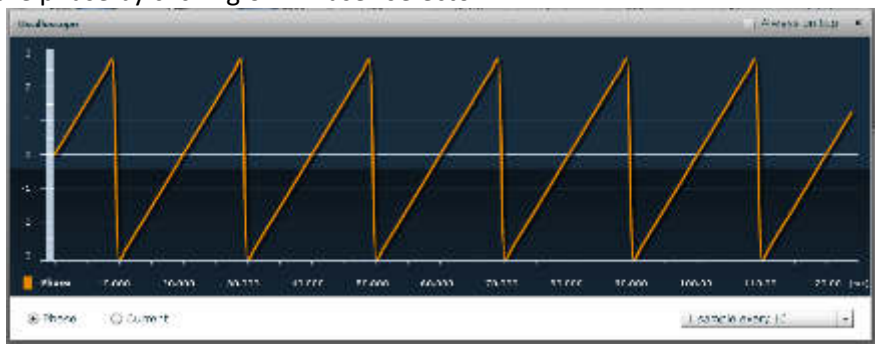
To start the motor, let's enter a speed which is 1.5 times the minimum speed, in this case **1500 RPM**



Please click on the “Oscilloscope” button to see the motor waveforms with the current in Y-axis and the time in x-axis.



You can also display the phase by clicking on “Phase” selector:



For the oscilloscope window, use an opportune time scale: “1 sample every 1” should be used for extremely fast phenomena when running at very high speed.

The setting “1 sample every 128” should be used for extremely low phenomena when running at very low speed. Let’s start with an intermediate value and adjust it in order to see some periods of the current or the phase.

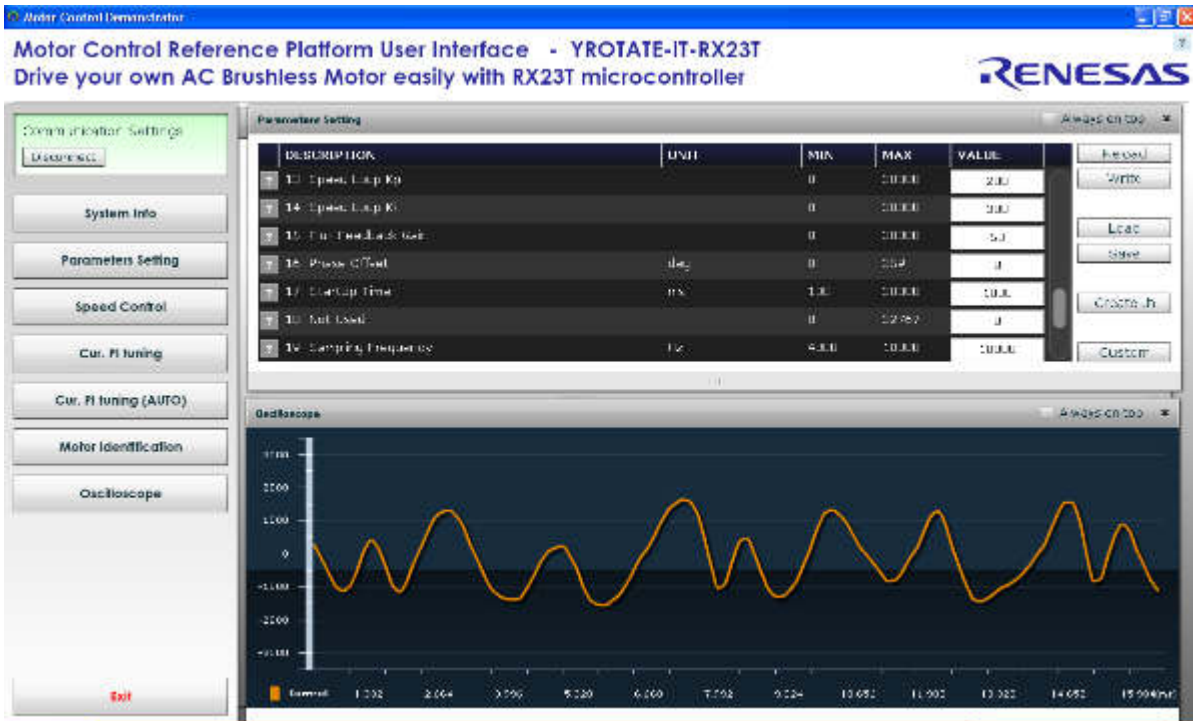
k) When the motor is running, you can adjust the two speed PI parameters: the proportional and integral terms: #13 and #14

Please follow the procedure: while running at a medium speed range: 2 times the minimum speed. In our example, the speed is set to 2000 RPM, the PWM frequency is set to 30KHz and the sampling frequency to 10KHz. The board was RESETTED and the auto-tuning procedure redone.

Please open the Oscilloscope window and the Parameters Setting windows as shown below.



To tune the coefficients, start by increasing the Parameter n°13 (Kp) until the instability that can be display in the current or phase waveform window. Add a step of "1000" and click "write" to see the effect and keep on increasing it.



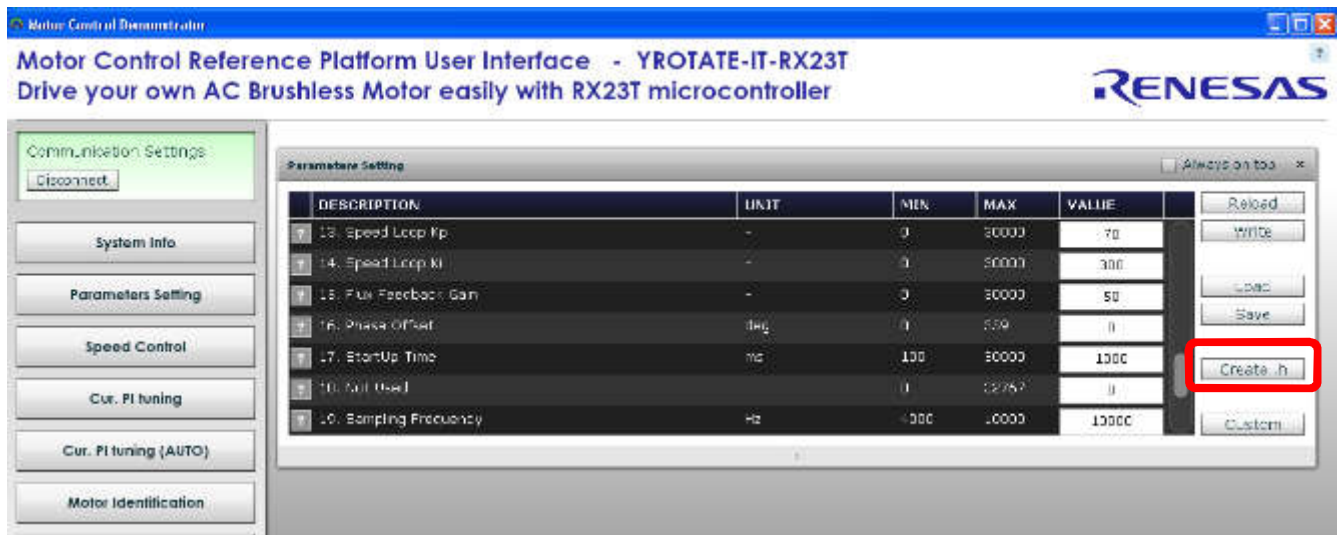
In our case, at **200** it started to be very unstable, but the motor is still running. Set the speed to "0". Then use less than half of the found value: **70** in our case, click on "write" and set the speed to 2000 RPM.

Do the same for the **parameter n°14** (integral coefficient) which is the speed loop Ki parameter. Increase it until it becomes unstable.

In our case the critical value is reached at **600** for Ki, so the value to be used is: **300** (half of the value found).



- l) Test the parameters found in all the speed ranges and different rotations.
- m) Finally the parameters list can be saved in a file in .CSV ("Save" button) or .h file ("Create.h" button) format for further used and can also be uploaded later on:



Troubleshooting:

At the stage j) if the motor doesn't start or generate an alarm n°3, please set the speed to "0" to clear the alarm which indicates that the software lost the phase. One first test is to increase or decrease the start-up current and the minimum speed or the speed PI gains

When the motor is running, you can verify the number of pole pairs taking measurement of the effective speed, and comparing it with the imposed frequency: the number of pole pairs n is: $n = \text{freq} * 60 / \text{speed}$; if you change the number of pole pairs, remember to adjust also the minimum (and maximum) speed values.

For some motors, the no-load start-up is easier if the inductance parameter is set to 0 (parameter #9)

All the procedure is tuned to manage motors which maximum current is close to the inverter capability, which is around 6A for the external power stage (shunt value is 0.05 Ohm) and 3Arms for the internal power stage (shunt value is 0.1 Ohm).

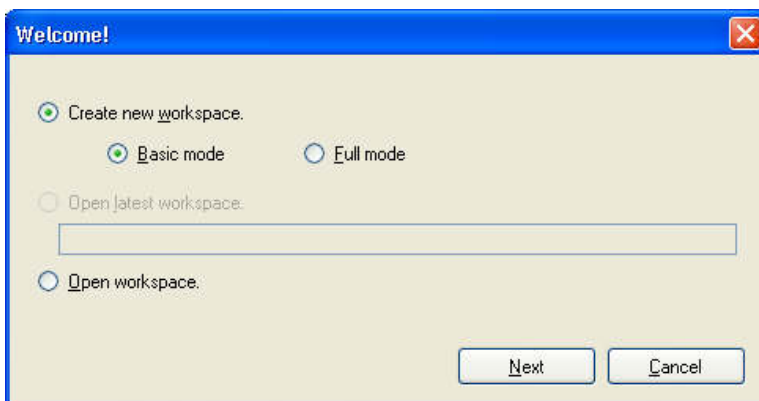
If you try to use it for very different motors, the results will be influenced by the losses in current reading resolution.

23. Updating the RX23T Flash memory using the Renesas Programming Flash Tool

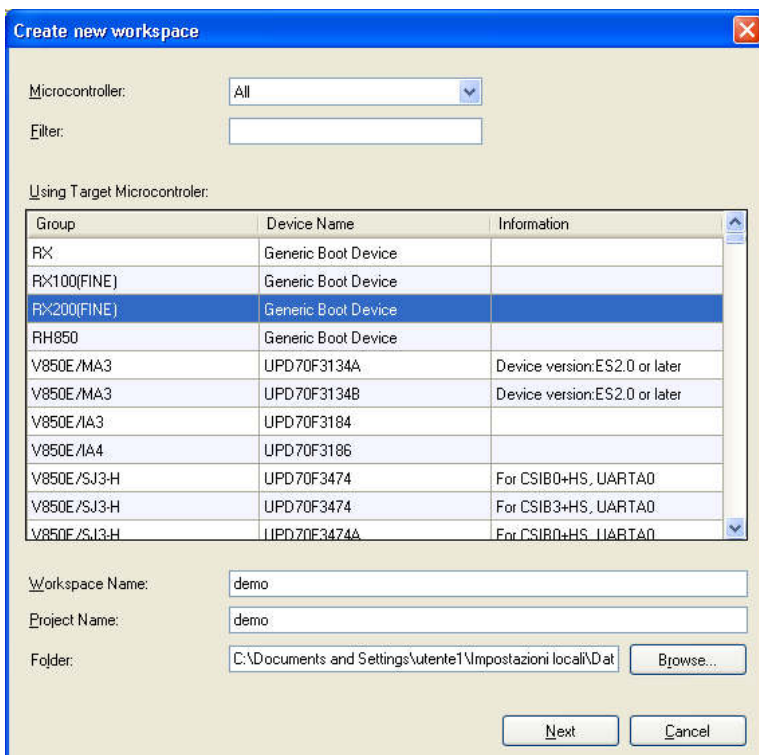
The procedure below explain in details how to re-program the RX23T flash memory using the Renesas Flash Programming tool, called the RFP. It could be used to update the RX23T kit with the latest Firmware version downloaded from the website: www.renesas.eu/motorcontrol . The RFP must be used with the E1 debugger and there's no need to install the full version of the development environment either e2studio or CS+.

Please have a look at the specific Flash Programming Tool website to us the latest software version: www.renesas.eu/products/tools/flash_prom_programming/index.jsp

Please follow the 13 steps to update the flash memory of your RX23T MCU using the RFP tool.



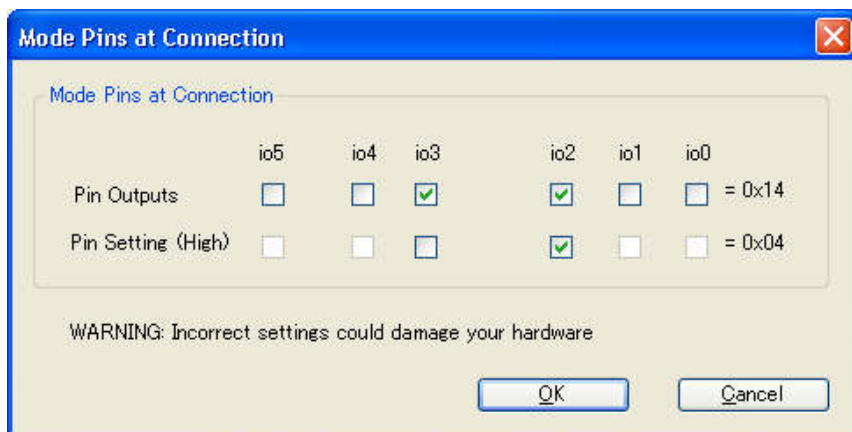
Step1: Open **RFP software v2.05**. The “**Welcome**” windows appears. Then click on “**Create new workspace**”



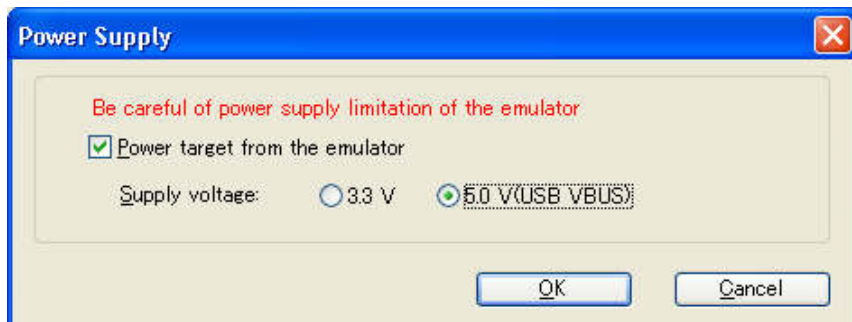
Step2: Select **RX200(FINE)** and a workspace name, then click “**Next**” button



Step3: Select the E1 debugger and then click “Avanti” or “next” button.



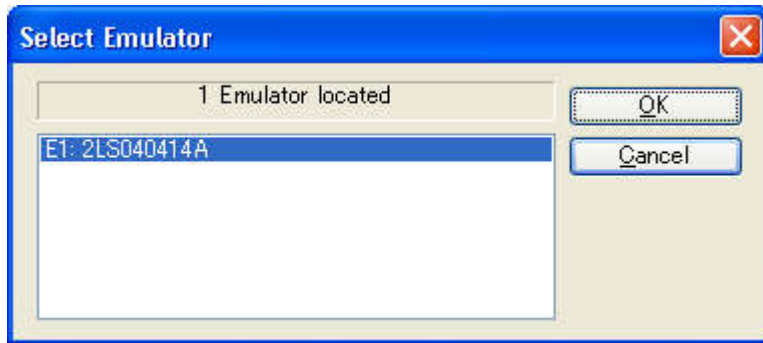
Step4: Select the options as shown and click “OK” button.



Step5: Click on “Power target from emulator”, then select 5.0V (USB VBUS) and then click “OK” button.



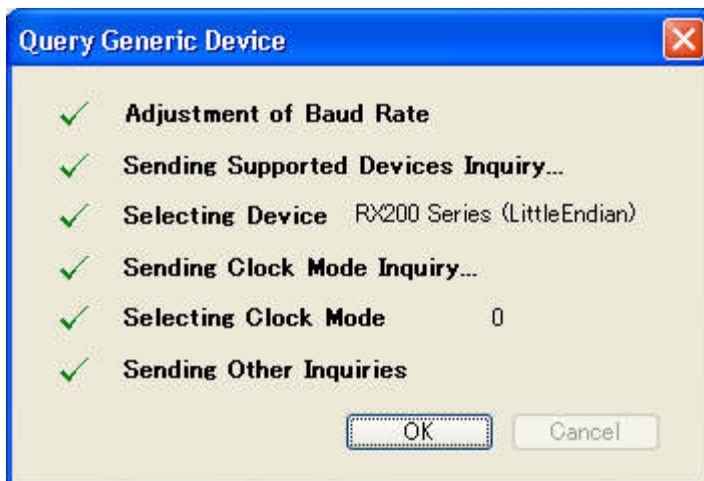
Step6: Click on "OK" button.



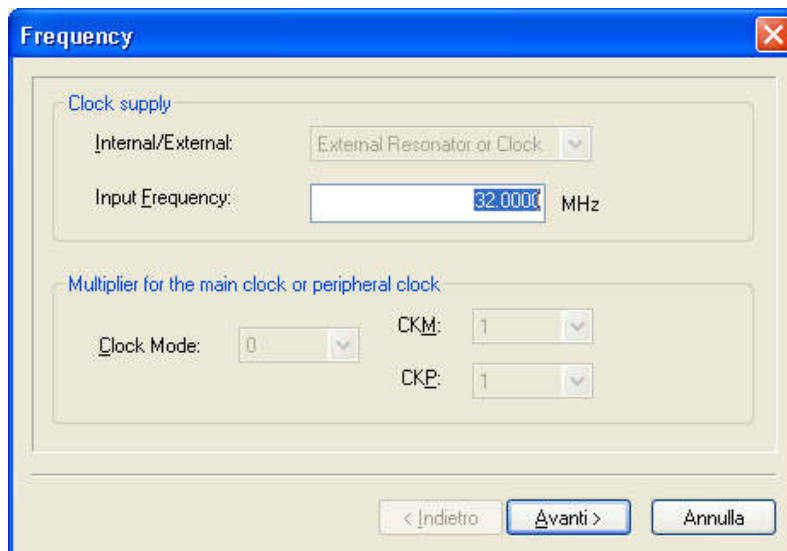
Step7: After E1 Emulator has been located press "OK" button



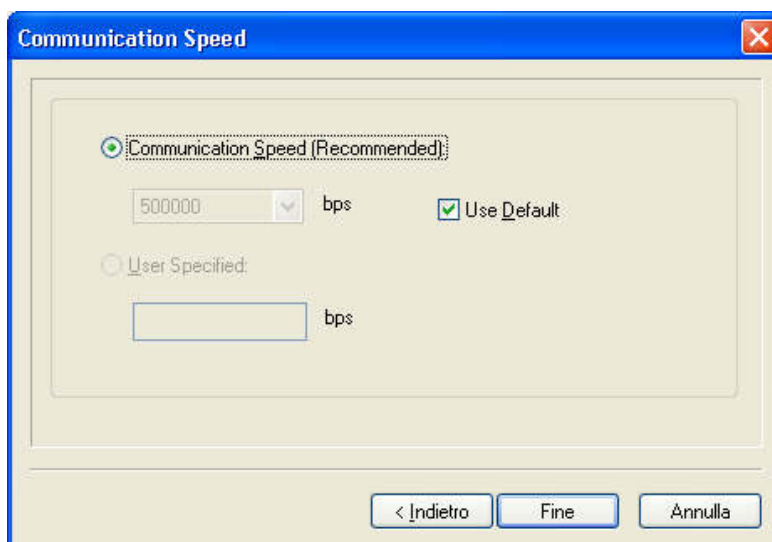
Step8: Select **RX200 Series (Little Endian)** and press "OK" button



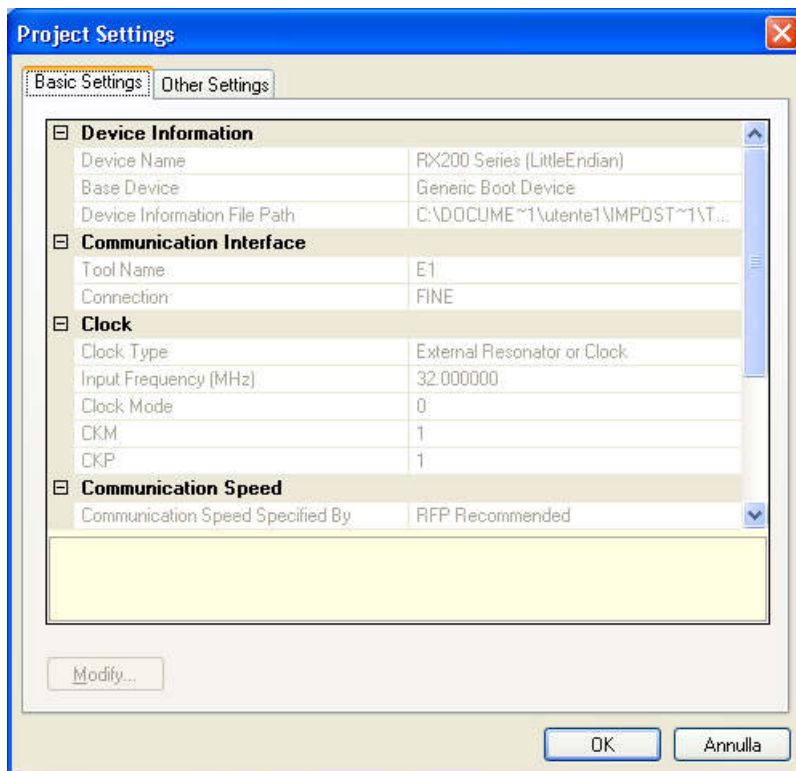
Step9: After all the test will be passed press "OK" button



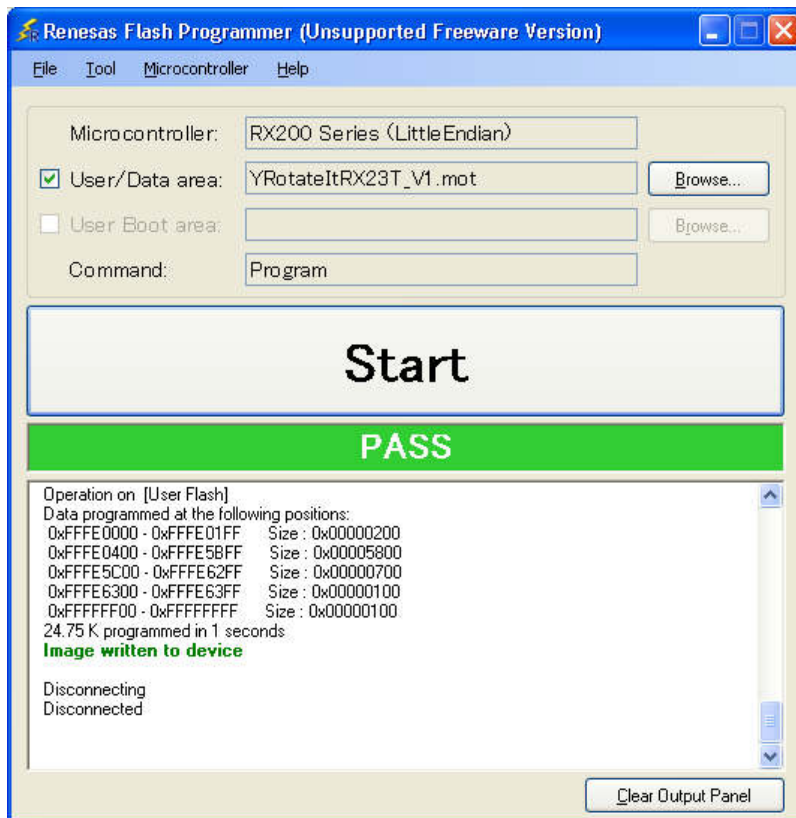
Step10: Input Frequency will be 32MHz, so press "Avanti" button



Step11: click "Fine" button



Step12: Check the summary and then click “OK” button



Step13: Browse the file with the “.mot” extension to be downloaded into the flash memory and then click “START” button.

At the end of the process “Image written to device” will appear.

The new program is now into the flash of the RX23T.

Congratulations! The new firmware is now programmed into the RX23T flash memory.

24. Integration of user code to the motor control algorithm

There are several places in the embedded source code to add user code depending on the type of routines. Basically, either the user code can be placed within the main routines in the module called “main.c” or in the interrupt service routine offering the highest priority, used to execute the field oriented control algorithm in the module called: “motorcontrol.c”.

The example below shows how to add specific code in the module “main.c” to add a specific speed profile on the YROTATE-IT-RX23T. The 1st step is to add a main variable “COUNTER_CYCLE” to generate a timer base.

```
#define _MAIN_C // insert define BEFORE including header files

#include "globdef.h"
#include "ges_eqp.h"
#include "motorcontrol.h"
#include "userif.h"

#define LOW 0
#define HIGH 1
#define TIMING_CYCLE 120
unsigned int COUNTER_CYCLE = 0;
unsigned char MOTOR_CYCLE_STATUS = LOW;

void main(void)
{
```

The 2nd step is to add at the beginning of the main loop the speed ramps and the profile wished.

In the example, the motor speed will reach **30000 RPM** with the first cycle period and slow-down to **9000 RPM** in the second phase and it will work periodically.

The routine is generating a
The main loop timing is 10ms, so 120 times, it means 1.2 seconds is the period of the cycle.

```
while(1)
{
    // time synchronization
    while(MC_WaitSync())
    {
        // this time can be used for asynchronous functions
    }

    COUNTER_CYCLE++;
    if (COUNTER_CYCLE > TIMING_CYCLE)
    {
        COUNTER_CYCLE = 0;
        if (MOTOR_CYCLE_STATUS == LOW)
        {
            MOTOR_CYCLE_STATUS = HIGH;
            UIF_W.var.rif = 30000;
        }
        else
        {
            MOTOR_CYCLE_STATUS = LOW;
            UIF_W.var.rif = 9000;
        }
    }
}
```

25. 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"). **In the versions with auto-tuning the answer code is ("d").**

The maximum serial communication speed tested is 76.6 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 ? s c k
(l=5)
word read.: l ? s w a n k
(l=7)
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:      |   #   s   o   k
          (l=5)
check:    |   !   s   C   k
          (l=5)
word read.: |   !   s   w   a   n   D11   D10   ..   Dn1   Dn0   k
           (l=7+2*n)
word writ.: |   !   s   W   k
           (l=5)

```

ADDRESSES:

If the address "a" specified in the question is a < NUM_PAR_EQP (number of EEPROM parameters), then an EEPROM parameter 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_EQP.

Operation example:

PC request of reading 16 words from the structure UIF_R, starting from the second one (UIF_R.ram_tab[1], .., IF_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 1 st word of data (UIF_R.ram_tab[1]=UIF_R.var.rpm, speed)
7	00	LSB of the 1 st word of data
8	00	MSB of the 2 nd word of data (UIF_R.ram_tab[2]=UIF_R.var.fre, imposed frequency)
9	00	LSB of the 2 nd word of data
10	00	MSB of the 3 rd word of data (UIF_R.ram_tab[3]=UIF_R.var.id, d axis current)
11	00	LSB of the 3 rd word of data
12	00	MSB of the 4 th word of data (UIF_R.ram_tab[4]=UIF_R.var.iq, q axis current)
13	00	LSB of the 4 th word of data
14	00	MSB of the 5 th word of data (UIF_R.ram_tab[5])
15	00	LSB of the 5 th word of data
16	00	MSB of the 6 th word of data (UIF_R.ram_tab[6])
17	00	LSB of the 6 th word of data
18	00	MSB of the 7 th word of data (UIF_R.ram_tab[7]=UIF_R.var.vb, bus voltage)
19	18	LSB of the 7 th word of data
20	00	MSB of the 8 th word of data (UIF_R.ram_tab[8])
21	00	LSB of the 8 th word of data
22	00	MSB of the 9 th word of data (UIF_R.ram_tab[9]=UIF_R.var.all, alarm)
23	01	LSB of the 9 th word of data
24	00	MSB of the 10 th word of data (UIF_R.ram_tab[10])
25	00	LSB of the 10 th word of data
26	00	MSB of the 11 th word of data (UIF_R.ram_tab[11])
27	00	LSB of the 11 th word of data

28	00	MSB of the 12 th word of data (UIF_R.ram_tab[12])
29	00	LSB of the 12 th word of data
30	00	MSB of the 13 th word of data (UIF_R.ram_tab[13])
31	00	LSB of the 13 th word of data
32	00	MSB of the 14 th word of data (UIF_R.ram_tab[14])
33	00	LSB of the 14 th word of data
34	00	MSB of the 15 th word of data (UIF_R.ram_tab[15])
35	00	LSB of the 15 th word of data
36	00	MSB of the 16 th word of data (UIF_R.ram_tab[16])
37	00	LSB of the 16 th word of data
38	69	checksum

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 1 st 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: Two new operation codes have been added:

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

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

In these cases the address is a < NUM_PAR_EQP.

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

```

/*****
Macro definitions
*****/
#define N_RAM_READ      ( 32 )
#define N_RAM_WRITE    (  8 )
#define MAX_SAM_CNT    ( 120 ) // can be asked all at the same time
#define OUTBUF_XDIM    ( 128 )
#define OUTBUF_SDIM    OUTBUF_XDIM
#define OUTBUF_CDIM    (OUTBUF_XDIM * 2)

typedef union
{
  uint16_t      ram_tab[N_RAM_READ];
  struct
  {
    int16_t      rif;          // 0   internal speed reference
    int16_t      rpm;          // 1   measured speed
    int16_t      fre;          // 2   imposed frequency
    int16_t      id;           // 3   direct current
    int16_t      iq;           // 4   quadrature current
    int16_t      vd;           // 5   direct voltage
    int16_t      vq;           // 6   quadrature voltage
    int16_t      vb;           // 7   bus voltage
    int16_t      all;          // 8   alarm
    int16_t      flg;          // 9   flags
    int16_t      toti;         // 10  current vector amplitude
    int16_t      totv;         // 11  voltage vector amplitude
    int16_t      du0;          // 12
    int16_t      du1;          // 13
    int16_t      du2;          // 14
    int16_t      du3;          // 15
    int16_t      mod;          // 16  mode
    int16_t      rs;           // 17  stator resistance
    int16_t      ls;           // 18  synchronous inductance
    int16_t      fl;           // 19  permanent magnet flux
    int16_t      kpi;          // 20  current loop kp
    int16_t      kii;          // 21  current loop ki
    int16_t      pwmf;         // 22  pwm frequency [Hz]
    int16_t      sf;           // 23  sampling frequency [Hz]
    int16_t      ena;          // 24  extra features enable flags
    int16_t      du4;          // 25
    int16_t      du5;          // 26
    int16_t      du6;          // 27
    int16_t      du7;          // 28
    int16_t      du8;          // 29
    int16_t      du9;          // 30
    int16_t      du10;         // 31
  }
} UIF_R_t;

```

```

typedef union
{
    uint16_t      ram_tab[N_RAM_WRITE];
    struct
    {
        int16_t   trg;          // 0   trigger
        int16_t   mod;          // 1   mode
        int16_t   rif;          // 2   speed reference
        int16_t   cra;          // 3   current ratio
        int16_t   sel;          // 4   variable and time scale selection
        int16_t   du0;          // 5
        int16_t   du1;          // 6
        int16_t   du2;          // 7
    } var;
} UIF_W_t;

/*****
Variables definitions/declarations
*****/

#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 // _USERIF_C

```

```

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

```

```

Special working modes are controlled through 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 is 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 behaviour
#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_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 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 &= WCLR8;

#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;

```


Revision History

Rev.	Date	Description	
		Page	Summary
0.50	May 20, 2015		First Edition
1.00	October 6, 2015		Chapter 10 to 24 are reworked or added

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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. Handling of Unused Pins

Handle unused pins in accord 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable.

When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

Before changing from one product to another, i.e. to a product with a different type number, confirm that the change will not lead to problems.

- The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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.

