

YROTATE-IT-RX66T

UM-YROTATE-IT-RX66T

Rev. 1.1

Feb 6, 2020

Rotate it! – Motor Control RX66T

Introduction

The Renesas Motor Control Kit YROTATE-IT-RX66T is based on the RX66T device from the powerful 32-bit RX microcontrollers family running at 160MHz and delivering 5.8 CoreMark/MHz.

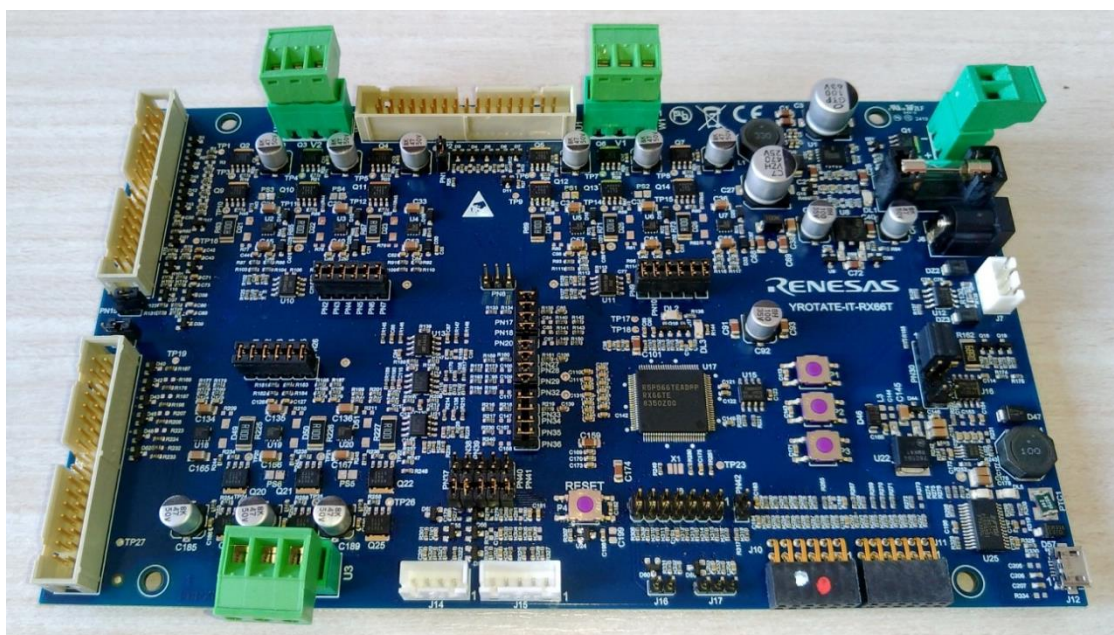
The kit enables engineers to easily test and evaluate the performance of the RX66T in a laboratory environment when driving up to three 3-phase Permanent Magnet Synchronous Motors (e.g. AC Brushless Motors) using an advanced sensor-less Field Oriented Control algorithm. Typical applications for this 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 expensive current sensors.

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 to adapt the control dynamics to the application requirements.

The hardware is designed for easy access to key system test points and for the ability to hook up to the E1 RX66T debugger. 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-RX66T is an ideal tool to check out all the key performance parameters of your selected motors, before embarking on a final end application system design.



Contents

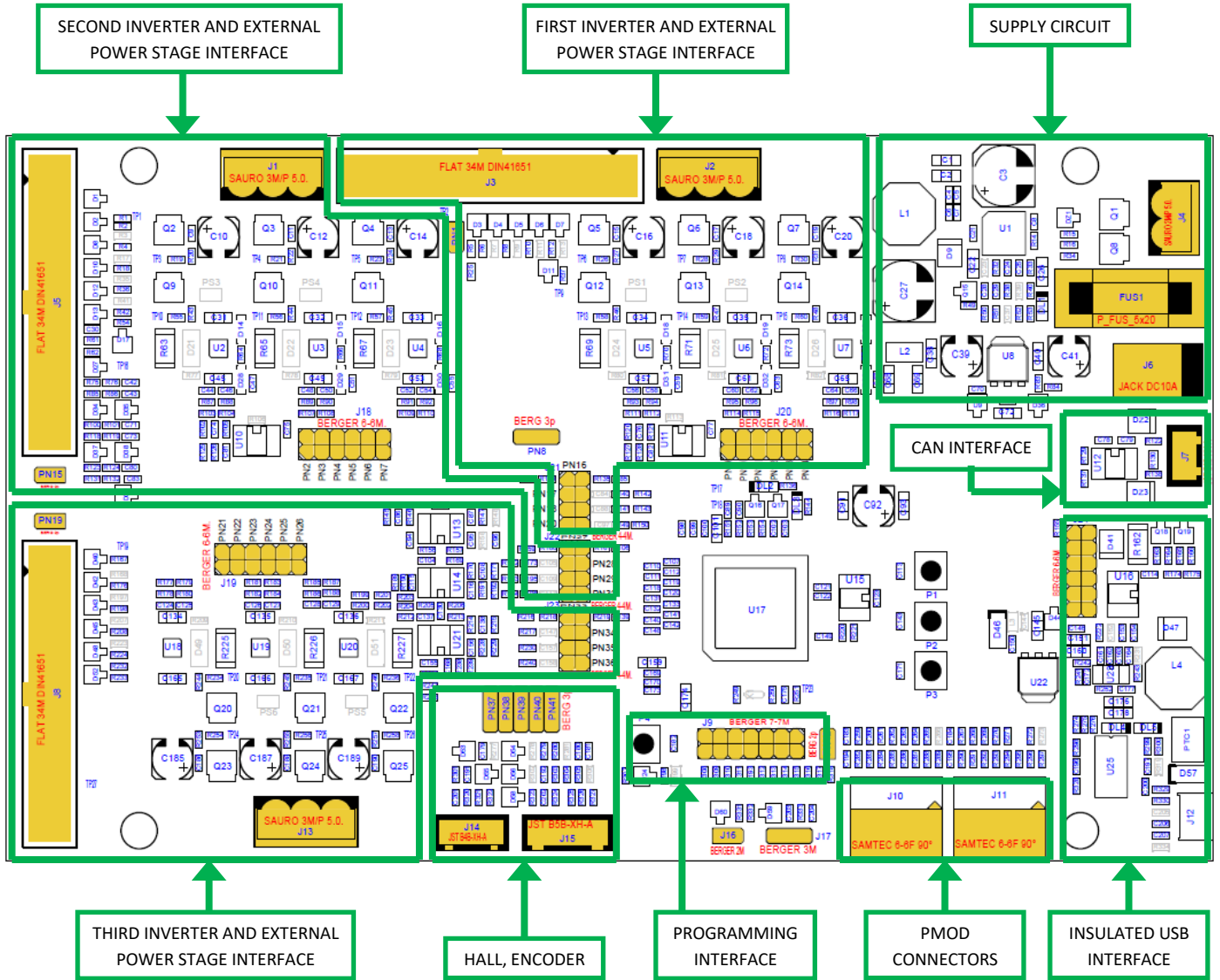
1. Specifications & hardware overview	3
2. Connectors description.....	5
3. Power supply selection.....	6
4. LEDs functions description.....	8
5. Test points for debugging	9
6. Internal power stages description	11
7. Current reading methods	13
8. Interface to an external power stage.....	14
11. Website material: www.renesas.eu/motorcontrol.....	15
12. Microcontroller RX66T short overview.....	16
13. Permanent magnets brushless motor model.....	26
14. Sensor-less Field Oriented Control algorithm	31
15. Software Tools used.....	32
16. Software description and Resources used	36
17. Start-up procedure.....	43
18. Reference system transformations in details	45
19. Rotor position estimation.....	46
20. PC Graphical User Interface in details.....	48
21. Eeprom parameters: detailed description	52
22. Motor Auto-calibration using the PC GUI.....	53
25. Communication Protocol between the MCU and the PC GUI	57

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} , Nominal Speed: 4000RPM
KIT MAX INPUT RANGE	External power supply from: 20V _{DC} to 48V _{DC} , 6A _{peak}
TRANSISTOR USED	Renesas MOSFET: RJK0645DBP (60V, 30A, 8.3mΩ max)
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	RX66T (R5F566TEADFP), 100-pin LQFP, 160MHz, 512KB Flash, 64KB RAM, 32KB Data Flash
MCU PERFORMANCE	160 MHz, 5.8 CoreMark/MHz
KEY FEATURES	Floating Point Unit, 3-phase inverter Timer 12-bit A/D Converter, Programmable Gain Amplifiers, Port Output Enable
MCU EMBEDDED FIRMWARE	Sensor-less 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	42μs including debug features and auto-tuning/self-identification + 3 Axis control
CODE SIZE IN FLASH / RAM	48.3KB / 4.9KB
TOOL USED, VERSION	e ² studio 7.4.0, RXC toolchain, CC RX version v3.00.00
COMPILER OPTIMIZATION LEVEL	Level 2
ENVIRONMENT STANDARDS	RoHS compliant including China regulations WEEE, RoHS

The inverter kit YROTATE-IT-RX66T is a single board triple inverter, based on the 32-bit RX series microcontroller RX66T and includes three 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 PCB overview.



The full schematics and CAD design files (e.g. Gerber) of the inverter kit are available on the website: <https://www.renesas.com/eu/en/solutions/proposal/motor-control.html> , in the section related to the YROTATE-IT-RX66T development kit.

In the YROTATE-IT-RX66T kit, a single RX66T in a 100-pin package was selected to ensure the management of inverters, external communications, EEPROM communication, E1 debugger, Bus voltage monitoring, etc.

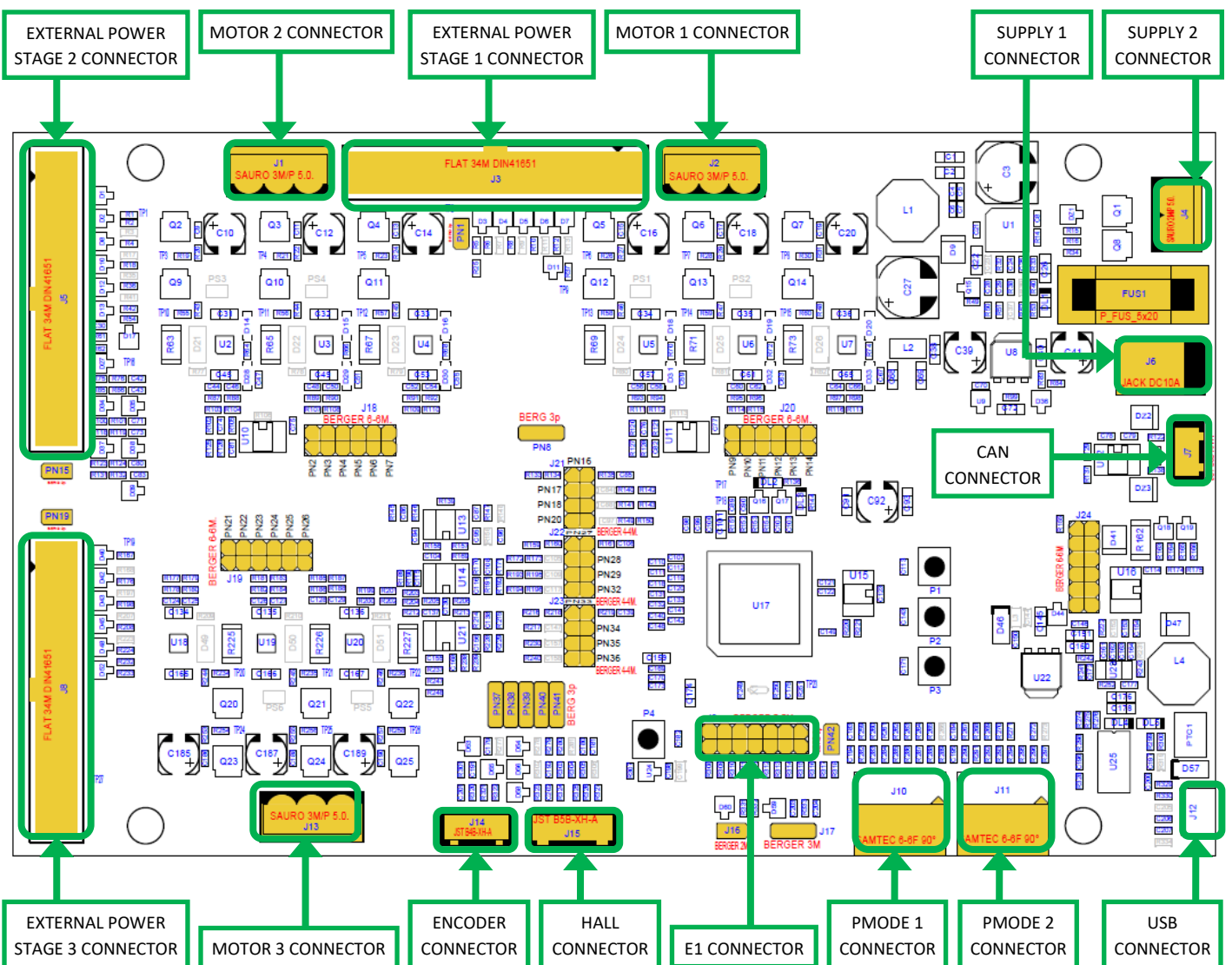
2. Connectors description

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 RX66T. It can be connected either to the E2studio and the CS+ development environments.

The external power stage connectors are 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:

<https://www.renesas.com/eu/en/solutions/proposal/motor-control.html>



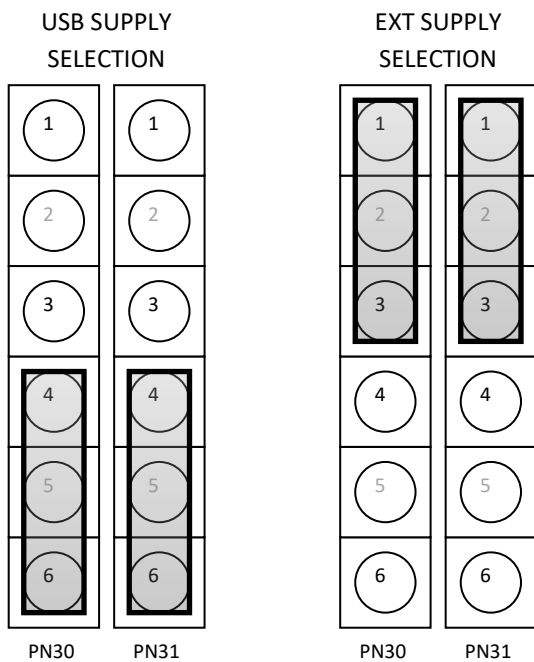
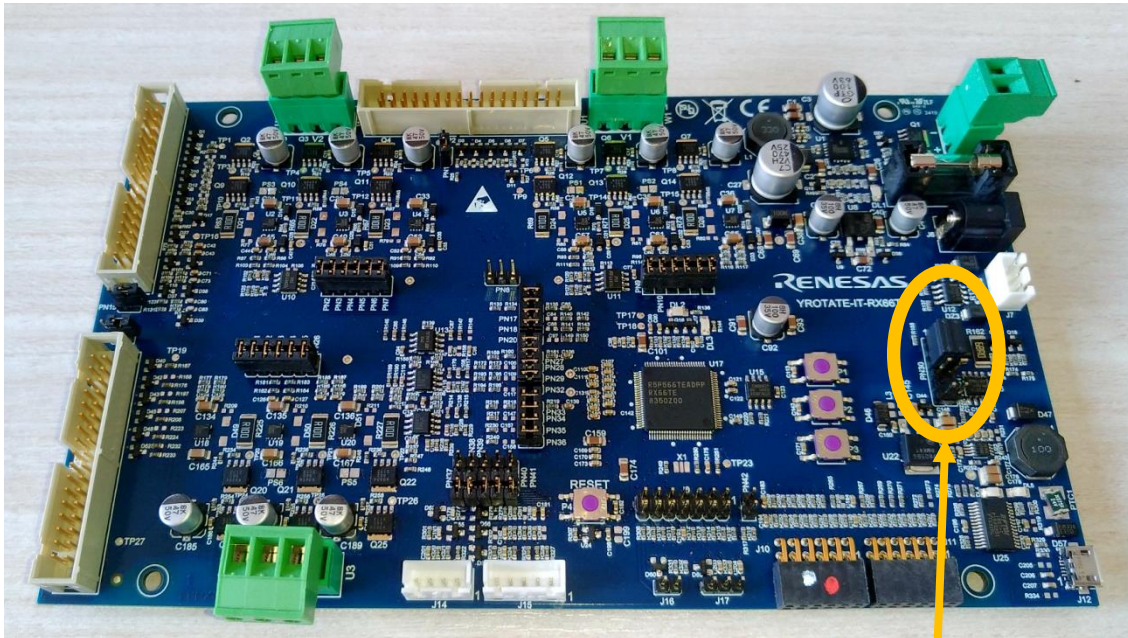
The motor connectors are used to connect directly the low voltage motors to the internal inverters. Hall and encoder connectors are used to connect hall sensors and encoder (since the standard software is sensor-less, they are not normally needed).

3. Power supply selection

As stated before, there are different 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: **PN30** and **PN31**. Please find below the description:



Supply selection jumpers position

- 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.

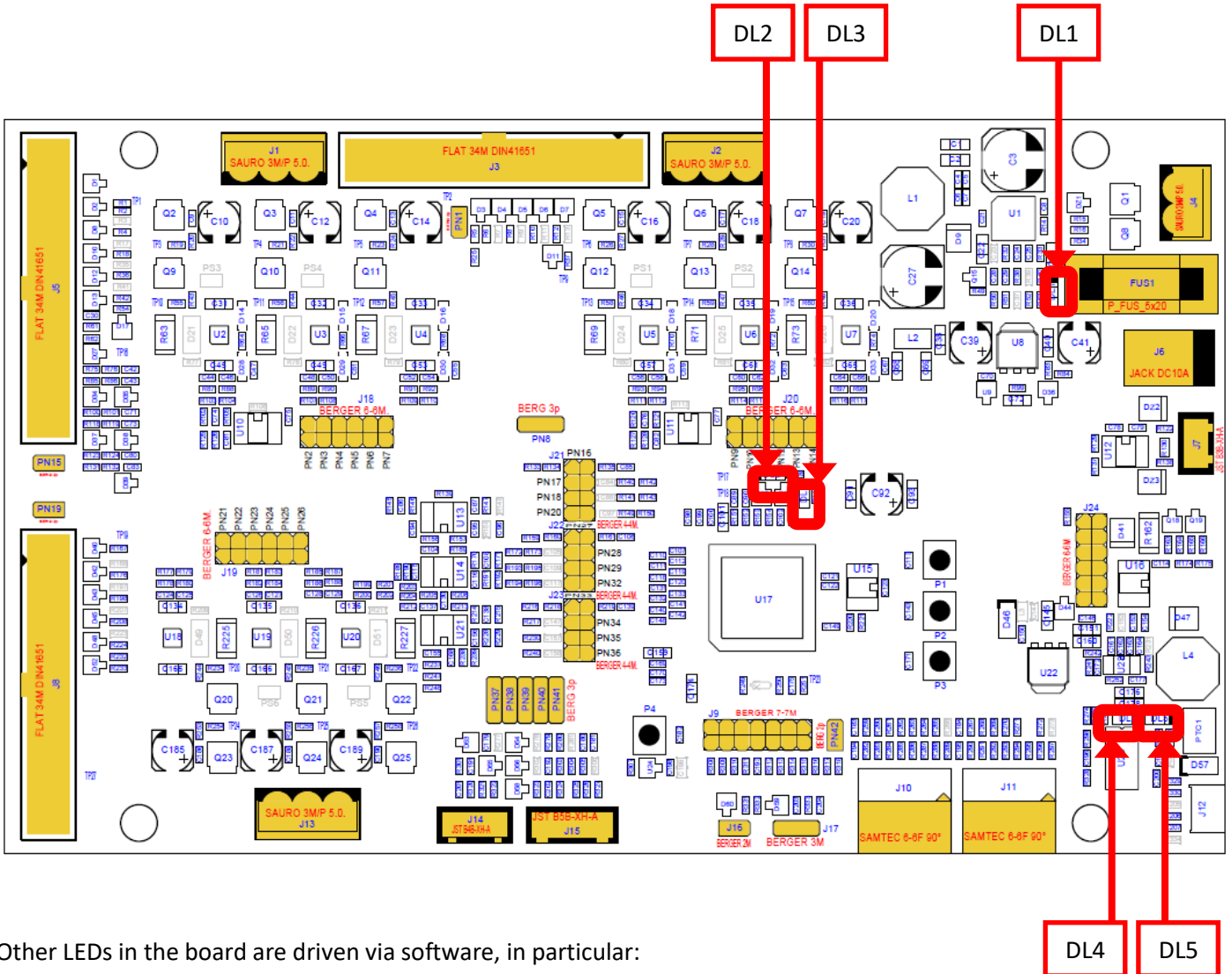
- 2) The second jumper configuration connects the external power supply ground to the inverter ground and the external +V_{DC} to the inverter DC link.

It is also possible to supply the micro board from (one of) the external power stage(s). In such case please set the jumpers for external power stage, but don't connect any supply source to the board connector. Instead close the specific jumper (PN1, PN15, PN19) related to the specific external power stage connector. Please select only one power stage to supply the micro board, and close only one jumper. In other cases, please leave the jumpers (PN1, PN15, PN19) open.

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:

- DL1 is connected to the output of the 15V step-down DC-DC converter and indicates the presence of the switches drive supply;
- DL4, DL5 are dedicated to the USB communication.



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

- DL2 is blinking slowly if the control section of the RX66T microcontroller is running normally; in case of hardware or software alarms, DL2 is blinking quickly (this is valid with the standard software).
- 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 signals. Below you can find their description.

- TP1: analogical temperature input from external power stage 2 connector.
- TP2: analogical supply voltage from external power stage 1 connector.
- TP3: motor 2 output U.
- TP4: motor 2 output V.
- TP5: motor 2 output W.
- TP6: motor 1 output U.
- TP7: motor 1 output V.
- TP8: motor 1 output W.
- TP9: analogical temperature input from external power stage 1 connector.
- TP10: motor 2 U phase low arm MOSFET source.
- TP11: motor 2 V phase low arm MOSFET source.
- TP12: motor 2 W phase low arm MOSFET source.
- TP13: motor 1 U phase low arm MOSFET source.
- TP14: motor 1 V phase low arm MOSFET source.
- TP15: motor 1 W phase low arm MOSFET source.
- TP16: analogical supply voltage from external power stage 2 connector.
- TP17: microcontroller DA1 output test point.
- TP18: microcontroller DA0 output test point.
- TP19: analogical temperature input from external power stage 3 connector.
- TP20: motor 3 W phase low arm MOSFET source.
- TP21: motor 3 V phase low arm MOSFET source.
- TP22: motor 3 U phase low arm MOSFET source.
- TP23: microcontroller PE2 (NMI) test point.
- TP24: motor 3 output W.
- TP25: motor 3 output V.
- TP26: motor 3 output U.
- TP27: analogical supply voltage from external power stage 3 connector.

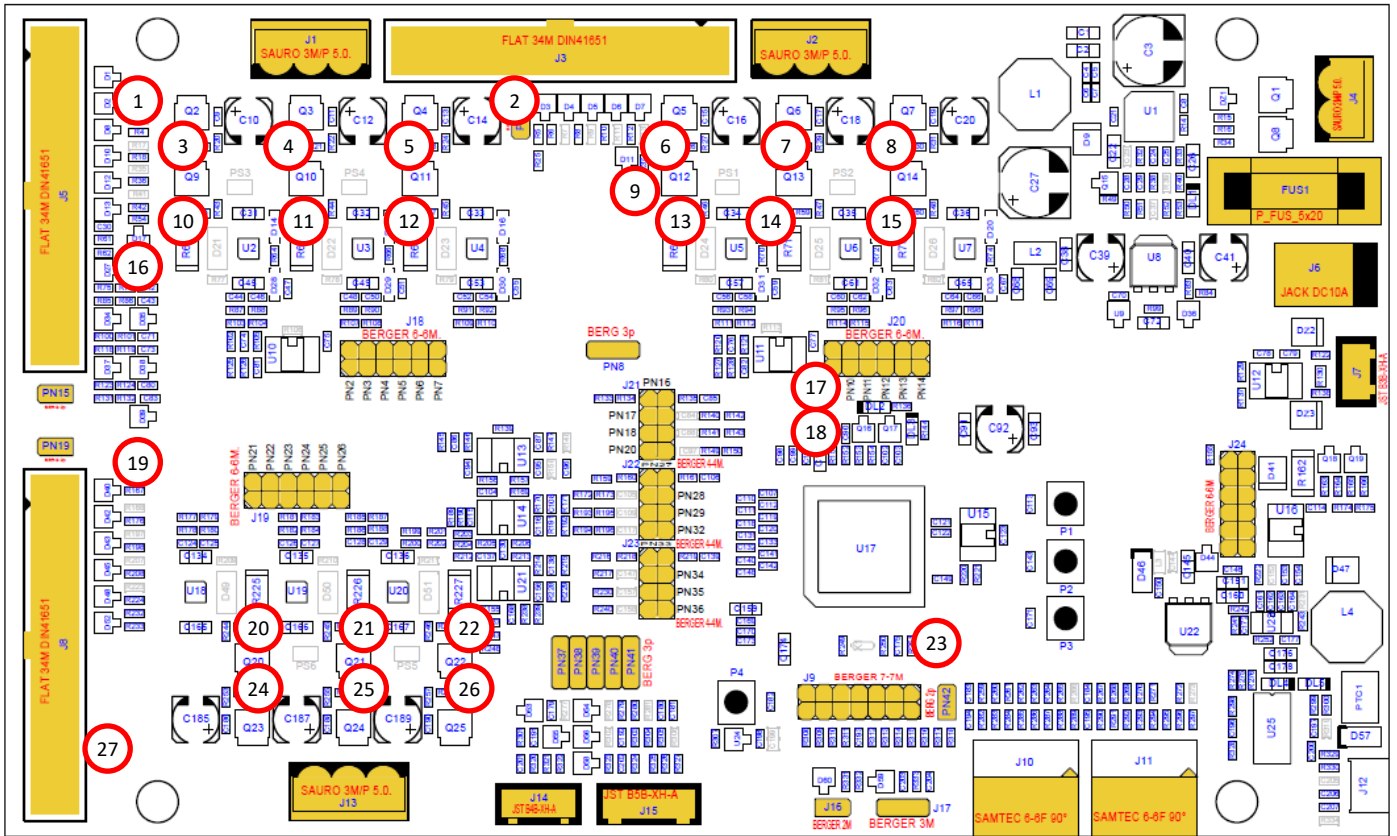


Figure 1 Test points available on the board to visualize with the oscilloscope the behaviour of some internal signals

6. Internal power stages description

The board includes three power stages which are complete 3-phase bridges composed with discrete low voltage and high current MOSFET. The MOSFET are the Renesas RJK0645DBP n-channel power MOSFET. 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**.

Two power stages are identical each other, the third one differs from the previous due to the current reading circuit, which is based on external op-amps, since in the micro controller only two A/D converters have programmable gain amplifiers.

The three power internal power stages share the driving signals and the analogical readings with the external optional power stages, so selection switches are provided in each case. The driving signals are connected directly to the external power stage connector, and through jumpers to the internal power stage. So, if the external power stage is connected, then the jumpers which connect the internal one must be removed, otherwise they have to be connected. The same criteria is applied for the analogical inputs from the external power stages (currents and bus voltages readings).

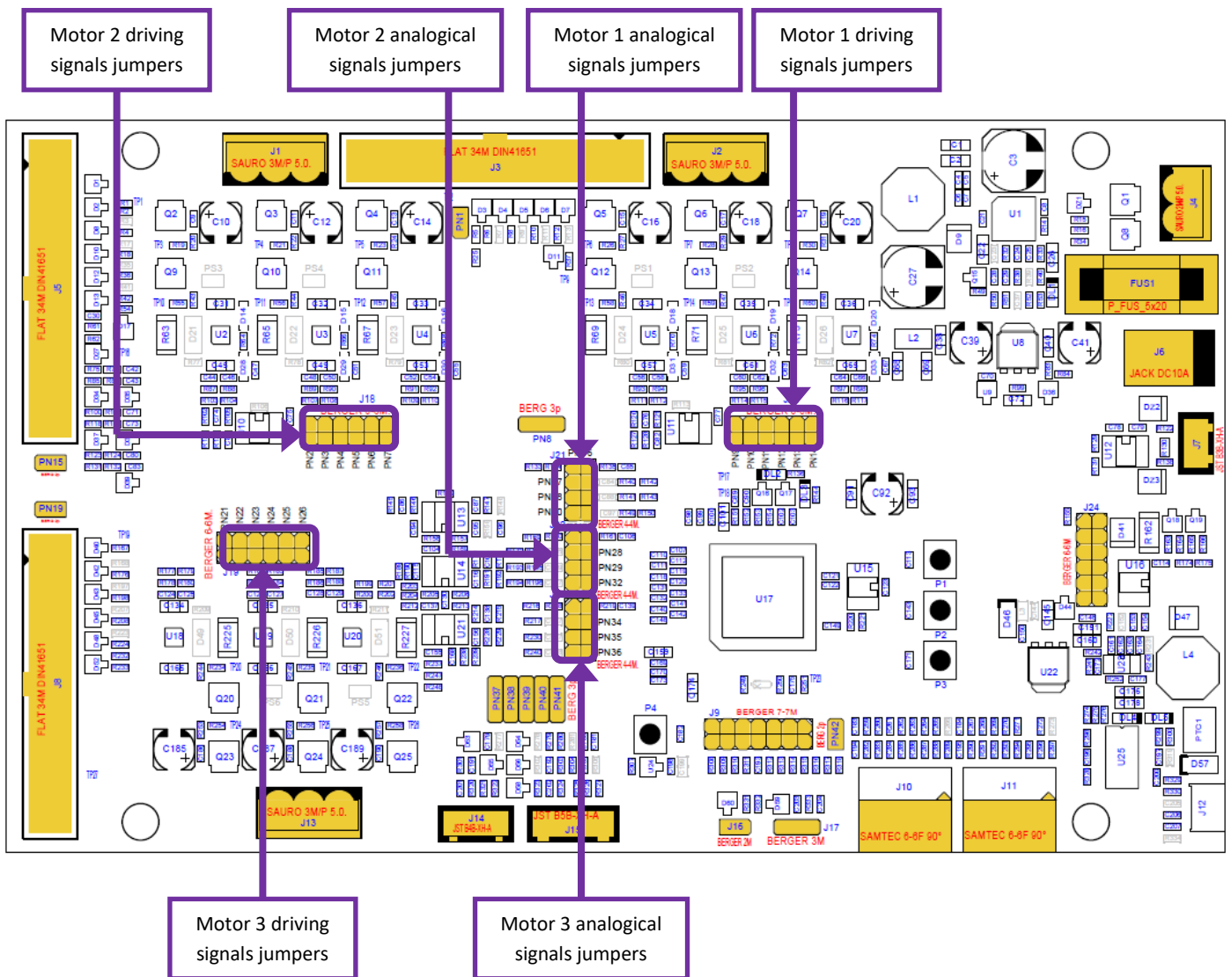


Figure 2 Motor Axis Signal Jumpers

A further difference between the first two power stages and the third one is related to the over-current detection: while in the first two a circuit which detect the de-saturation of the MOSFET is provided, in the third one a direct measurement of the DC current flowing in the circuit is made, through the sum of the three phase currents. The over-current signals are in wired OR with the ones eventually coming from external power stages connectors.

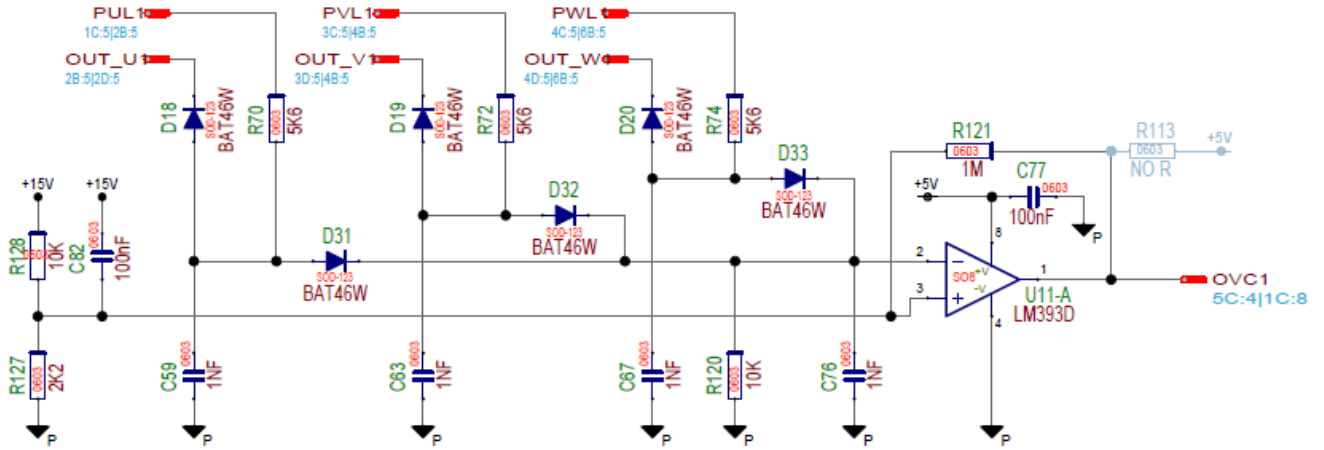


Figure 3: De-saturation detection circuit for motor 1

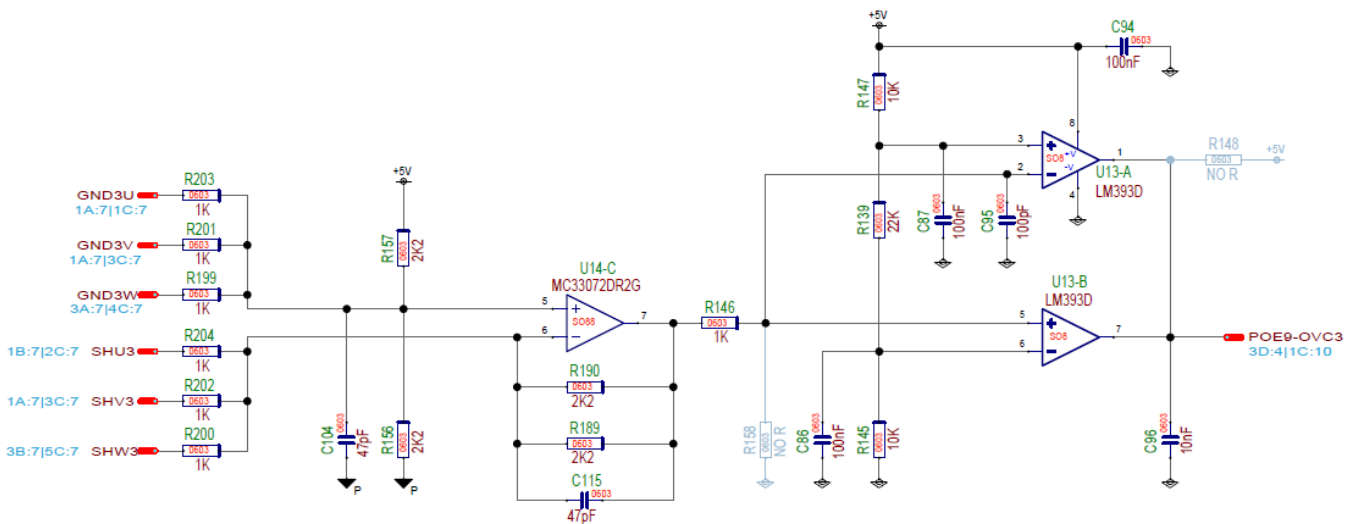
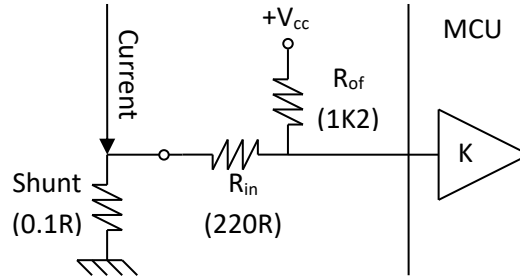


Figure 4 Over-current detection circuit for motor 3

7. Current reading methods

As said, in the RX66T microcontroller are present two A/D converter modules with 3 Programmable Gain Amplifiers for each module. These amplifiers can be used to amplify the analogical signal coming from the current reading shunts. Please find below the circuit used in the board.



The third A/D module has not PGA, so conventional circuit has been chosen to read the current for the third motor:

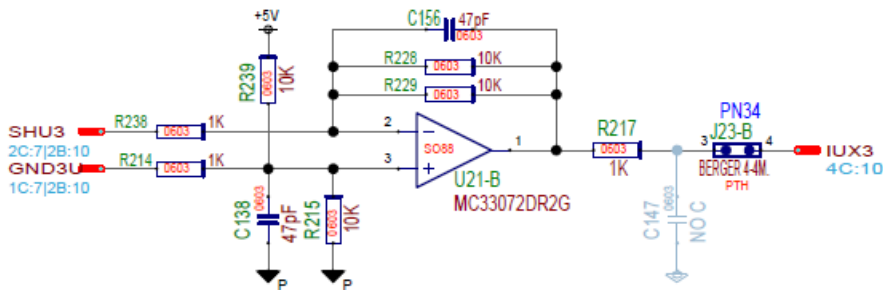


Figure 5 Amplifier Circuit for the third Motor Axis

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

The complete kit material is available on-line at the address:

<https://www.renesas.com/eu/en/solutions/proposal/motor-control.html>

The latest updates of the downloadable material for the YROTATE-IT-RX66T 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-RX66T Drivers	Drivers and setup files for the PC Graphical User Interface
Embedded Software e ² Studio Source Code	Source files for code flashed by default into the Renesas microcontroller
Manuals Kit Motor Specifications Renesas Datasheets Tuned Motors Specifications	Relevant documentation for the kit, the motor and the microcontroller, the MOSFET. 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 RX66T short overview

The RX66T Group is 32-bit microcontroller and suited for up to three inverters 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. RX66T 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 RX66T microcontrollers are reported in table 1.1, directly taken from the hardware manual.

Table 1.1 Outline of Specifications (1/9)

Classification	Module/Function	Description
CPU	CPU	<ul style="list-style-type: none"> • Maximum operating frequency: 160 MHz • 32-bit RX CPU (RXv3) • Minimum instruction execution time: One instruction per state (cycle of the system clock) • Address space: 4-Gbyte linear • Register set of the CPU <ul style="list-style-type: none"> General purpose: Sixteen 32-bit registers Control: Ten 32-bit registers Accumulator: Two 72-bit registers • 111 instructions <ul style="list-style-type: none"> Standard provided instructions: 111 <ul style="list-style-type: none"> Basic instructions: 77 Single precision floating point instructions: 11 DSP instructions: 23 • Addressing modes: 11 • Data arrangement <ul style="list-style-type: none"> Instructions: Little endian Data: Selectable as little endian or big endian • On-chip 32-bit multiplier: $32 \times 32 \rightarrow 64$ bits • On-chip divider: $32/32 \rightarrow 32$ bits • Barrel shifter: 32 bits
	FPU	<ul style="list-style-type: none"> • Single-precision (32-bit) floating-point number • Data types and floating-point exceptions in conformance with the IEEE754 standard
Memory	Code flash memory	<ul style="list-style-type: none"> • Capacity: 1 Mbyte, 512 Kbytes, 256 Kbytes • ROM cache: Operation of an 8-Kbyte instruction fetching cache can be enabled or disabled (this is disabled by default). <ul style="list-style-type: none"> While ROM cache operation is enabled: <ul style="list-style-type: none"> - when the cache is hit, one-cycle access up to 160 MHz - when the cache is missed: <ul style="list-style-type: none"> one to two cycles if ICLK \leq 120 MHz (bus wait: 0 cycles), two to three cycles if ICLK $>$ 120 MHz (bus wait: 1 cycle). While ROM cache operation is disabled: <ul style="list-style-type: none"> one cycle if ICLK \leq 120 MHz (bus wait: 0 cycles), two cycles if ICLK $>$ 120 MHz (bus wait: 1 cycle). • On-board programming: Five types • Off-board programming (parallel programmer mode) (This is not available for 80/64-pin products) • The trusted memory (TM) function protects against the reading of programs from blocks 8 and 9.
	Data flash memory	<ul style="list-style-type: none"> • Capacity: 32 Kbytes • Programming/erasing: 100,000 times
	Unique ID	<ul style="list-style-type: none"> • 12-byte unique ID for the device
	RAM	<ul style="list-style-type: none"> • Capacity: 128 Kbytes, 64 Kbytes • 160 MHz No-wait access • SED (single error detection)
	RAM with ECC	<ul style="list-style-type: none"> • Capacity: 16 Kbytes • 00FF C000h to 00FF FFFFh (16 Kbytes) • SEC-DED (single error correction/double error detection)

Table 1.1 Outline of Specifications (2/9)

Classification	Module/Function	Description
Operating modes		<ul style="list-style-type: none"> Operating modes by the mode-setting pins at the time of release from the reset state <ul style="list-style-type: none"> Single-chip mode Boot mode (SCI interface) Boot mode (USB interface) Boot mode (FINE interface) User boot mode Selection of operating mode by register setting <ul style="list-style-type: none"> Single-chip mode, user boot mode, On-chip ROM disabled extended mode, On-chip ROM enabled extended mode Endian selectable
Clock	Clock generation circuit	<ul style="list-style-type: none"> Main clock oscillator, low-speed/high-speed on-chip oscillator, PLL frequency synthesizer, and IWDI-dedicated on-chip oscillator The peripheral module clocks can be set to frequencies above that of the system clock. Main-clock oscillation stoppage detection Separate frequency-division and multiplication settings for the system clock (ICLK), peripheral module clocks (PCLKA, PCLKB, PCLKC, PCLKD), flash-IF clock (FCLK) and external bus clock (BCLK) The CPU and other bus masters run in synchronization with the system clock (ICLK): Up to 160 MHz Peripheral modules of MTU3 (Internal peripheral bus), GPTW (Internal peripheral bus), HRPWM (Internal peripheral bus), RSPI, and SCI11 run in synchronization with PCLKA, which operates at up to 120 MHz. Other peripheral modules run in synchronization with PCLKB: Up to 60 MHz MTU3 (counter reference clocks), GPTW (counter reference clocks), and HRPWM (reference clocks) are synchronized with PCLKC: Up to 160 MHz ADCLK in the S12AD runs in synchronization with PCLKD: Up to 60 MHz Flash IF run in synchronization with the flash-IF clock (FCLK): Up to 60 MHz Devices connected to the external bus run in synchronization with the external bus clock (BCLK): Up to 40 MHz Multiplication is possible with using the high-speed on-chip oscillator (HOCO) as a reference clock of the PLL circuit
Reset		<p>Nine types of reset</p> <ul style="list-style-type: none"> RES# pin reset: Generated when the RES# pin is driven low. Power-on reset: Generated when the RES# pin is driven high and VCC rises. Voltage-monitoring 0 reset: Generated when VCC falls. Voltage-monitoring 1 reset: Generated when VCC falls. Voltage-monitoring 2 reset: Generated when VCC falls. Deep software standby reset: Generated in response to an interrupt to trigger release from deep software standby. Independent watchdog timer reset: Generated when the independent watchdog timer underflows, or a refresh error occurs. Watchdog timer reset: Generated when the watchdog timer underflows, or a refresh error occurs. Software reset: Generated by register setting.
Power-on reset		<p>If the RES# pin is at the high level when power is supplied, an internal reset is generated. After VCC has exceeded the voltage detection level and the specified period has elapsed, the reset is cancelled.</p>
Voltage detection circuit (LVDA)		<p>Monitors the voltage being input to the VCC pin and generates an internal reset or internal interrupt.</p> <ul style="list-style-type: none"> Voltage detection circuit 0 <ul style="list-style-type: none"> Capable of generating an internal reset The option-setting memory can be used to select enabling or disabling of the reset. Voltage detection level: Selectable from two different levels Voltage detection circuits 1 and 2 <ul style="list-style-type: none"> Voltage detection level: Selectable from five different levels Digital filtering (1/2, 1/4, 1/8, and 1/16 LOCO frequency) Capable of generating an internal reset Two types of timing are selectable for release from reset <ul style="list-style-type: none"> An internal interrupt can be requested. Detection of voltage rising above and falling below thresholds is selectable. Maskable or non-maskable interrupt is selectable Voltage detection monitoring Event linking

Table 1.1 Outline of Specifications (3/9)

Classification	Module/Function	Description
Low power consumption	Low power consumption facilities	<ul style="list-style-type: none"> • Module stop function • Four low power consumption modes Sleep mode, all-module clock stop mode, software standby mode, and deep software standby mode
Interrupt	Interrupt controller (ICUC)	<ul style="list-style-type: none"> • Interrupt vectors : 256 • External interrupts: 16 (pins IRQ0 to IRQ15) • Software interrupts: 2 sources • Non-maskable interrupts: 7 sources • Sixteen levels specifiable for the order of priority • Method of interrupt source selection: The interrupt vectors consist of 256 vectors (208 sources are fixed. The remaining 135 vectors are selected from among the other 48 sources.)
External bus extension		<ul style="list-style-type: none"> • The external address space can be divided into four areas (CS0 to CS3), each with independent control of access settings. Capacity of each area: 2 Mbytes (CS0 to CS3) A chip-select signal (CS0# to CS3#) can be output for each area. Each area is specifiable as an 8- or 16-bit bus space. The data arrangement in each area is selectable as little or big endian (only for data). • Bus format: Separate bus, multiplex bus • Wait control • Write buffer facility
DMA	DMA controller (DMACa)	<ul style="list-style-type: none"> • 8 channels • Three transfer modes: Normal transfer, repeat transfer, and block transfer • Request sources: Software trigger, external interrupts, and interrupt requests from peripheral functions
	Data transfer controller (DTCa)	<ul style="list-style-type: none"> • Three transfer modes: Normal transfer, repeat transfer, and block transfer • Request sources: External interrupts and interrupt requests from peripheral functions
I/O ports	Programmable I/O ports	<ul style="list-style-type: none"> • I/O ports for the 144-pin LFQFP I/O pins: 110 Input pin: 9 Pull-up resistors: 110 Open-drain outputs: 110 5-V tolerance: 4 Large current output: 15 • I/O ports for the 112-pin LQFP I/O pins: 84 Input pin: 9 Pull-up resistors: 84 Open-drain outputs: 84 5-V tolerance: 2 Large current output: 15 • I/O ports for the 100-pin LFQFP (with PGA pseudo-differential input, and with USB) I/O pins: 69 Input pin: 9 Pull-up resistors: 69 Open-drain outputs: 69 5-V tolerance: 3 Large current output: 15 • I/O ports for the 100-pin LFQFP (with PGA pseudo-differential input, and without USB) I/O pins: 72 Input pin: 9 Pull-up resistors: 72 Open-drain outputs: 72 5-V tolerance: 2 (products with 64 Kbytes of RAM), 3 (products with 128 Kbytes of RAM) Large current output: 15 • I/O ports for the 100-pin LFQFP (without PGA pseudo-differential input, and without USB) I/O pins: 73 Input pin: 7 Pull-up resistors: 73 Open-drain outputs: 73 5-V tolerance: 2 (products with 64 Kbytes of RAM), 3 (products with 128 Kbytes of RAM) Large current output: 15

Table 1.1 Outline of Specifications (4/9)

Classification	Module/Function	Description
I/O ports	Programmable I/O ports	<ul style="list-style-type: none"> I/O ports for the 80-pin LQFP, 80-pin LQFP I/O pins: 52 Input pin: 9 Pull-up resistors: 52 Open-drain outputs: 52 5-V tolerance: 2 Large current output: 14 I/O ports for the 64-pin LQFP I/O pins: 39 Input pin: 7 Pull-up resistors: 39 Open-drain outputs: 39 5-V tolerance: 2 Large current output: 14
Event link controller (ELC)		<ul style="list-style-type: none"> Event signals such as interrupt request signals can be interlinked with the operation of functions such as timer counting, eliminating the need for intervention by the CPU to control the functions. 188 internal event signals can be freely combined for interlinked operation with connected functions. Event signals from peripheral modules can be used to change the states of output pins (of ports B and E). Changes in the states of pins (of ports B and E) being used as inputs can be interlinked with the operation of peripheral modules.
Timers	8-bit timers (TMR)	<ul style="list-style-type: none"> (8 bits × 2 channels) × 4 units Select from among seven internal clock signals (PCLKB/1, PCLKB/2, PCLKB/8, PCLKB/32, PCLKB/64, PCLKB/1024, PCLKB/8192) and one external clock signal Capable of output of pulse trains with desired duty cycles or of PWM signals The 2 channels of each unit can be cascaded to create a 16-bit timer Generation of triggers for A/D converter conversion Capable of generating baud-rate clocks for SCI5, SCI6, and SCI12 Event linking by the ELC
	Compare match timer (CMT)	<ul style="list-style-type: none"> (16 bits × 2 channels) × 2 units Select from among four internal clock signals (PCLKB/8, PCLKB/32, PCLKB/128, PCLKB/512) Event linking by the ELC
	Watchdog timer (WDTA)	<ul style="list-style-type: none"> 14 bits × 1 channel Select from among 6 counter-input clock signals (PCLKB/4, PCLKB/64, PCLKB/128, PCLKB/512, PCLKB/2048, PCLKB/8192)
	Independent watchdog timer (IWDtA)	<ul style="list-style-type: none"> 14 bits × 1 channel Counter-input clock: IWDt-dedicated on-chip oscillator Dedicated clock/1, dedicated clock/16, dedicated clock/32, dedicated clock/64, dedicated clock/128, dedicated clock/256 Window function: The positions where the window starts and ends are specifiable (the window defines the timing with which refreshing is enabled and disabled). Event linking by the ELC

Table 1.1 Outline of Specifications (5/9)

Classification	Module/Function	Description
Timers	Multifunction timer pulse unit 3 (MTU3d)	<ul style="list-style-type: none"> • 9 channels (16 bits × 9 channels) • Maximum of 28 pulse-input/output and 3 pulse-input possible • Select from among 14 counter-input clock signals for each channel (PCLKC/1, PCLKC/2, PCLKC/4, PCLKC/8, PCLKC/16, PCLKC/32, PCLKC/64, PCLKC/256, PCLKC/1024, MTCLKA, MTCLKB, MTCLKC, MTCLKD, MTIOC1A) 11 of the signals are available for channels 1, 3, 4, 12 are available for channel 2, and 10 are available for channel 5. • 43 output compare/input capture registers • Counter clear operation (synchronous clearing by compare match/input capture) • Simultaneous writing to multiple timer counters (TCNT) • Simultaneous register input/output by synchronous counter operation • Buffered operation • Support for cascade-connected operation • 45 interrupt sources • Automatic transfer of register data • Pulse output mode Toggle/PWM/complementary PWM/reset-synchronized PWM • Complementary PWM output mode Outputs non-overlapping waveforms for controlling 3-phase inverters Automatic specification of dead times PWM duty cycle: Selectable as any value from 0% to 100% Delay can be applied to requests for A/D conversion. Non-generation of interrupt requests at peak or trough values of counters can be selected. Double buffer configuration • Reset synchronous PWM mode Three phases of positive and negative PWM waveforms can be output with desired duty cycles. • Phase-counting mode: 16-bit mode (channels 1 and 2); 32-bit mode (channels 1 and 2) • Counter functionality for dead-time compensation • Generation of triggers for A/D converter conversion The timing of the generation of requests to start A/D conversion can be monitored by an external pin. • A/D converter start triggers can be skipped • Digital filter function for signals on the input capture and external counter clock pins • Event linking by the ELC • Internal peripheral bus clock: PCLKA • Counter reference clock: PCLKC • Frequency ratio: PCLKA to PCLKC = 1: N (N = 1 or 2)
	Port output enable 3 (POE3B)	<ul style="list-style-type: none"> • Control of the high-impedance state of the MTU3/GPTW's waveform output pins, and control of switching to the general I/O port pin • 9 pins for input from signal sources: POE0, POE4, POE8, POE9, POE10, POE11, POE12, POE13, POE14 • Initiation by detection of short-circuited outputs (detection of PWM outputs that have become an active level simultaneously) • Initiation by comparator detection/oscillation stop detection/software • Additional programming of output control target pins is enabled

Table 1.1 Outline of Specifications (6/9)

Classification	Module/Function	Description
Timers	General PWM timer (GPTW)	<ul style="list-style-type: none"> • 32 bits × 10 channels • Counting up or down (sawtooth-wave), counting up and down (triangle-wave) selectable for all channels • Clock sources independently selectable for each channel • 2 input/output pins per channel • 2 output compare/input capture registers per channel • For the 2 output compare/input capture registers of each channel, 4 registers are provided as buffer registers and are capable of operating as comparison registers when buffering is not in use. • In output compare operation, buffer switching can be at peaks or troughs, enabling the generation of laterally asymmetrically PWM waveforms. • Registers for setting up frame intervals on each channel (with capability for generating interrupts on overflow or underflow) • Generation of dead times in PWM operation • Capable of synchronous start, stop, or clearing of counter for any channel • Capable of a start, stop, clearing, or up-/down-counting of the counter supporting maximum of 8 ELC events • Capable of a start, stop, clearing, or up-/down-counting of the counter supporting input level comparison • Capable of a start, stop, clearing, or up-/down-counting of the counter supporting maximum of 4 external triggers • Output pin disabling function by a dead time error or a short circuit detection among output pins • Capable of generating conversion start triggers for the A/D converters as well as monitoring external pins for a start timing of conversion. • Capable of outputting events, such as compare-match from A to F and overflow/underflow, to ELC • Capable of using noise filter of input capture • Internal peripheral bus clock: PCLKA • Counter reference clock: PCLKC • Frequency ratio: PCLKA to PCLKC = 1: N (N = 1 or 2)
	High resolution PWM (HRPWM)	<ul style="list-style-type: none"> • Capable of generating the PWM waveform that is generated by GPTW0 through GPTW3 with resolution of minimum of 195 ps.
	Port output enable for GPTW (POEG)	<ul style="list-style-type: none"> • Controlling the output disable for GPTW waveform output • Initiation by input level detection of GTETRG pins • Initiation by output disable request from GPTW • Initiation by detection of comparator interrupt request • Initiation by detection of oscillation stop or by software
Communication function	USB 2.0 FS host/function module (USBb)	<ul style="list-style-type: none"> • Includes a UDC (USB Device Controller) and transceiver for USB 2.0 FS • One port • Compliance with the USB 2.0 specification • Transfer rate: Full speed (12 Mbps), low speed (1.5 Mbps) (host only) • Self-power mode and bus power are selectable • OTG (On the Go) operation is possible (low-speed is not supported) • Incorporates 2 Kbytes of RAM as a transfer buffer • External pull-up and pull-down resistors are not required

Table 1.1 Outline of Specifications (7/9)

Classification	Module/Function	Description
Communication function	Serial communications interfaces (SCIj, SCli, SCih)	<ul style="list-style-type: none"> • 7 channels SCIj: SCI1, SCI5, SCI6, SCI8, SCI9 SCli: SCI11 SCih: SCI12 • SCIj, SCli, SCih Serial communications modes: Asynchronous, clock synchronous, and smart-card interface Multi-processor function On-chip baud rate generator allows selection of the desired bit rate Choice of LSB-first or MSB-first transfer Average transfer rate clock can be input from TMR timers for SCI5, SCI6, and SCI12 Start-bit detection: Level or edge detection is selectable. Simple I²C Simple SPI 7, 8, 9-bit transfer mode Bit rate modulation Double-speed mode Data match detection (SCI12 is not supported) Event linking by the ELC (supported by SCI5 only) • SCli Only Capable of serial sending and receiving with 16-byte FIFO-buffered structure both at transmission and reception sections • SCih Only Supports the serial communications protocol, which contains the start frame and information frame Supports the LIN format
	I ² C bus interface (RIICa)	<ul style="list-style-type: none"> • 1 channel Communication formats I²C bus format/SMBus format Supports the multi-master Max. transfer rate: 400 kbps • Event linking by the ELC
	CAN module (CAN)	<ul style="list-style-type: none"> • 1 channel • Compliance with the ISO11898-1 specification (standard frame and extended frame) • 32 mailboxes per channel
	Serial peripheral interface (RSPIC)	<ul style="list-style-type: none"> • 1 channel • RSPI transfer facility Using the MOSI (master out, slave in), MISO (master in, slave out), SSL (slave select), and RSPCK (RSPI clock) signals enables serial transfer through SPI operation (four lines) or clock-synchronous operation (three lines) Capable of handling serial transfer as a master or slave • Data formats Switching between MSB first and LSB first The number of bits in each transfer can be changed to any number of bits from 8 to 16, 20, 24, or 32 bits. 128-bit buffers for transmission and reception Up to four frames can be transmitted or received in a single transfer operation (with each frame having up to 32 bits) • Buffered structure Double buffers for both transmission and reception • RSPCK can be stopped with the receive buffer full for master reception. • Event linking by the ELC

Table 1.1 Outline of Specifications (8/9)

Classification	Module/Function	Description
12-bit A/D converter (S12ADH)		<ul style="list-style-type: none"> • 12 bits (8 channels × 2 units, 14 channels × 1 unit) • 12-bit resolution • Minimum conversion time 0.9 μs per channel (when ADCLK operates at 60 MHz) • Operating mode Scan mode (single scan mode, continuous scan mode, or 3 group scan mode) Group A priority control (only for 3 group scan mode) • Sample-and-hold function channel-dedicated sample-and-hold function (unit 0 × 3 channels, unit 1 × 3 channels) included • Sampling variable Sampling time can be set up for each channel. • Conversion function in order of arbitrarily selected channels (Serial conversion of the same channel cannot be allowed) • Double trigger mode (A/D conversion data duplicated) • Three ways to start A/D conversion Software trigger, synchronous trigger (MTU, TMR, ELC), external trigger • Prioritization in group scanning can be controlled among group A, B, and C. • Digital comparison Method: Comparison to detect voltages above or below thresholds and window comparison Measurement: Comparison of two results of conversion or comparison of a value in the comparison register and a result of conversion • Self-diagnostic function • Detection of analog input disconnection • Event linking by the ELC • Input signal amplification function by the programmable gain amplifier (unit 0 × 3 channels, unit 1 × 3 channels) Capable of supporting single end/pseudo-differential input
12-bit D/A converter (R12DAb)		<ul style="list-style-type: none"> • 2 channels • 12-bit resolution • Output voltage: 0 V to AVCC2 • Capable of providing as a reference voltage for comparator • Event linking by the ELC
Comparator C (CMPC)		<ul style="list-style-type: none"> • 6 channels • Function to compare the reference voltage and the analog input voltage • Reference voltage is selectable from 4 inputs • Analog input voltage is selectable from 4 inputs • Digital filtering
Temperature sensor		<ul style="list-style-type: none"> • 1 channel • Relative precision: ±1.0°C • The voltage of the temperature is converted into a digital value by the 12-bit A/D converter (unit 2).
Safety	Memory protection unit (MPU)	<ul style="list-style-type: none"> • Protection area: Eight areas (max.) can be specified in the range from 0000 0000h to FFFF FFFFh. • Minimum protection unit: 16 bytes • Reading from, writing to, and enabling the execution access can be specified for each area. • An access exception occurs when the detected access is not in the permitted area.
	Trusted Memory (TM) Function	<ul style="list-style-type: none"> • Protects against the reading of programs from blocks 8 and 9 of the code flash memory • Instruction fetching by the CPU is the only form of access to these areas when the TM function is enabled.
	Register write protection function	<ul style="list-style-type: none"> • Protects important registers from being overwritten for in case a program runs out of control.

Table 1.1 Outline of Specifications (9/9)

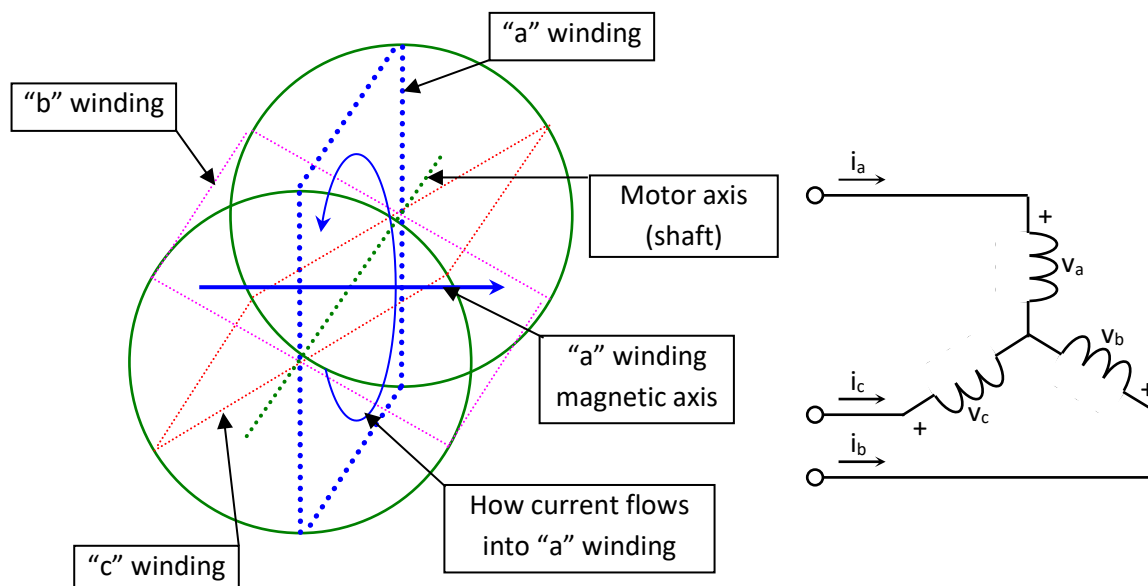
Classification	Module/Function	Description
Safety	CRC calculator (CRCA)	<ul style="list-style-type: none"> • Generation of CRC codes for 8-/32-bit data 8-bit data Selectable from the following three polynomials $X^8 + X^2 + X + 1$, $X^{16} + X^{15} + X^2 + 1$, $X^{16} + X^{12} + X^5 + 1$ 32-bit data Selectable from the following two polynomials $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$, $X^{32} + X^{28} + X^{27} + X^{26} + X^{25} + X^{23} + X^{22} + X^{20} + X^{19} + X^{18} + X^{14} + X^{13} + X^{11} + X^{10} + X^9 + X^8 + X^6 + 1$ • Generation of CRC codes for use with LSB-first or MSB-first communications is selectable
	Main clock oscillation stop detection function	<ul style="list-style-type: none"> • Main clock oscillation stop detection: Available
	Clock frequency accuracy measurement circuit (CAC)	<ul style="list-style-type: none"> • Monitors the clock output from the main clock oscillator, low- and high-speed on-chip oscillators, the PLL frequency synthesizer, IWDT-dedicated on-chip oscillator, and PCLKB.
	Data operation circuit (DOC)	<ul style="list-style-type: none"> • The function to compare, add, or subtract 16-bit data
Encryption functions	Trusted Secure IP (TSIP-Lite)	<ul style="list-style-type: none"> • Access management circuit • Encryption engine 128- or 256-bit key sizes of AES Block cipher mode of operation: GCM, ECB, CBC, CMAC, XTS, CTR, GCTR • Hash function • True random number generator • Prevention from illicit copying of a key
Operating frequency		Up to 160 MHz
Power supply voltage		VCC = 2.7 to 5.5V AVCC0 = AVCC1 = AVCC2 = 3.0 to 5.5V (VCC ≤ AVCC0 = AVCC1 = AVCC2) With USB in use: VCC_USB = 3.0 to 3.6V (VCC ≥ VCC_USB) With USB not in use: VCC_USB = VCC VSS = AVSS0 = AVSS1 = AVSS2 = VSS_USB = 0V
Operating temperature		D-version: -40 to +85°C G-version: -40 to +105°C (in planning)
Package		144-pin LQFP 0.5 mm pitch (in planning) 112-pin LQFP 0.65 mm pitch 100-pin LQFP 0.5 mm pitch 80-pin LQFP 0.5 mm pitch 80-pin LQFP 0.65 mm pitch 64-pin LQFP 0.5 mm pitch
Debugging interfaces		<ul style="list-style-type: none"> • JTAG and One-line FINE interfaces

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:

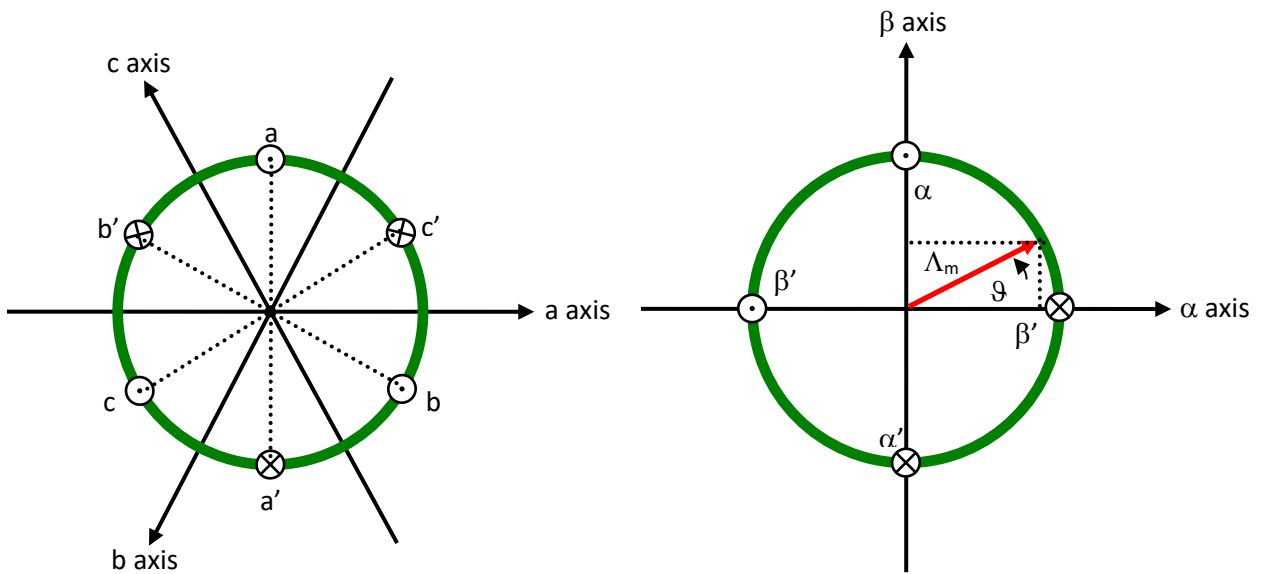
$$v_a = R_s i_a + \frac{d\lambda_a}{dt}$$

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

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

- 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).

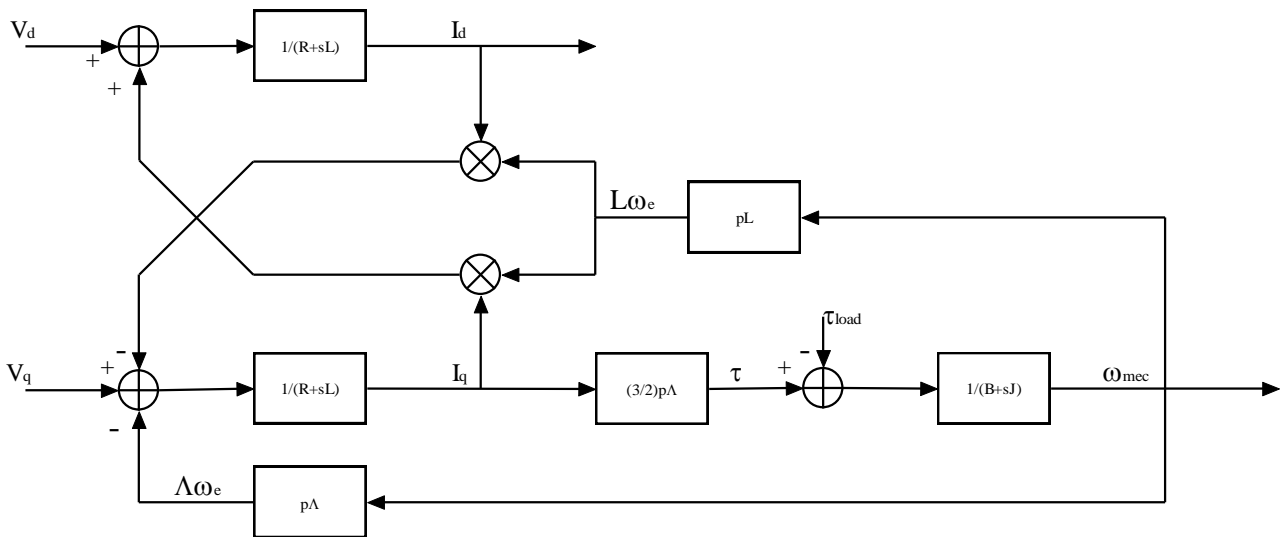
The reference frame transformations from the (α, β) system to the (d, q) system depends on the instantaneous position angle θ .

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:

$$\Lambda_m \cos(\mathcal{G}) = \lambda_\alpha - Li_\alpha$$

$$\Lambda_m \sin(\mathcal{G}) = \lambda_\beta - Li_\beta$$

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

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

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

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

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

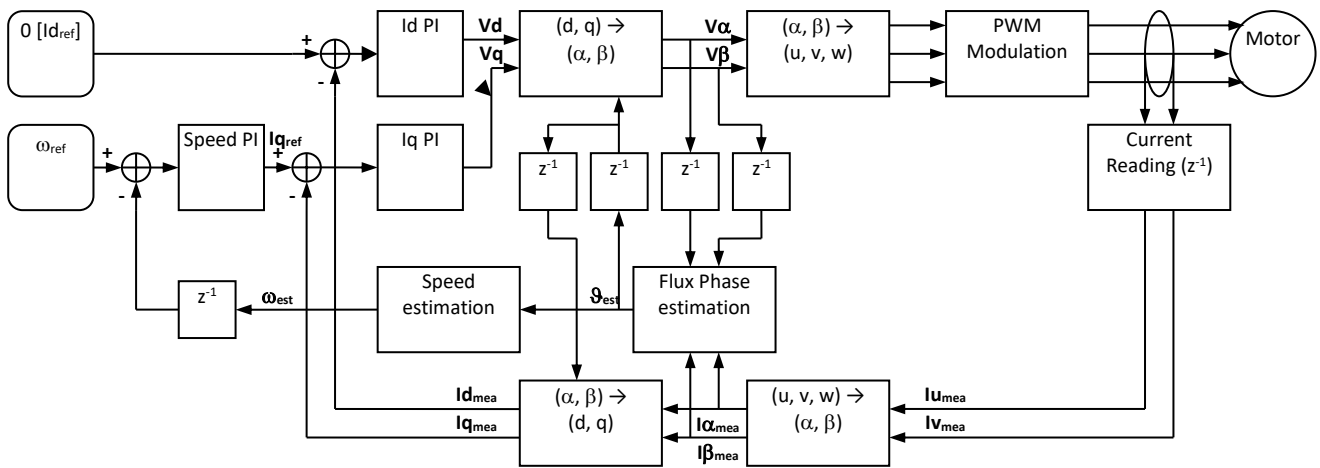
The system speed ω can be obtained as the derivative of the angle \mathcal{G} .

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

Based on this, a sensor-less control algorithm was developed to give the imposed phase voltages, to measure phase currents, to estimate the angular position \mathcal{G} and finally the system speed.

14. Sensor-less Field Oriented Control algorithm

Please, find below the sensor-less 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.

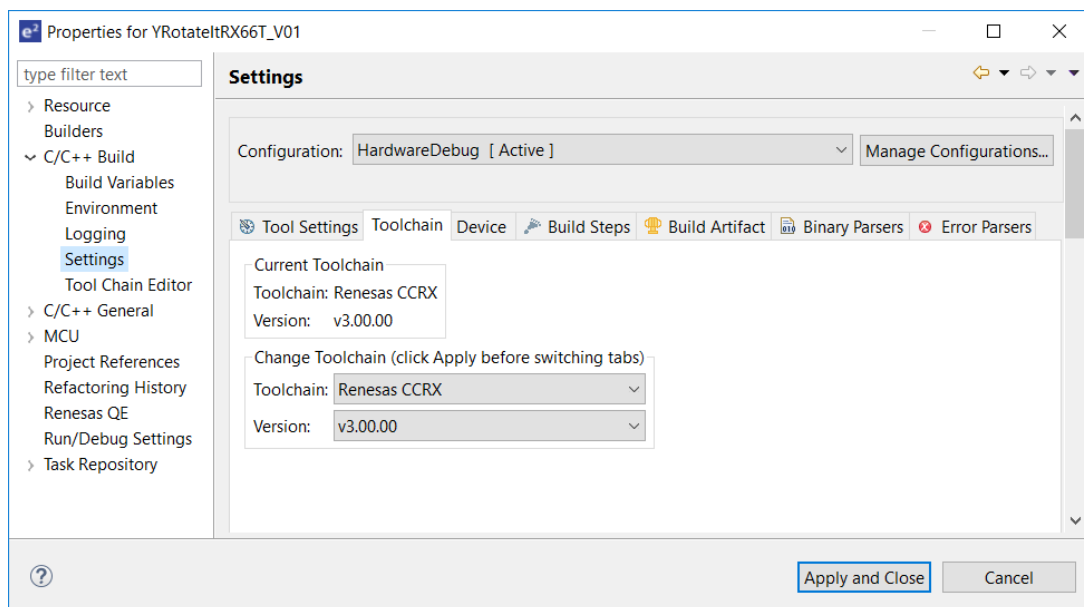
15. Software Tools used

15.1 IDE and e2studio Tool Chain used

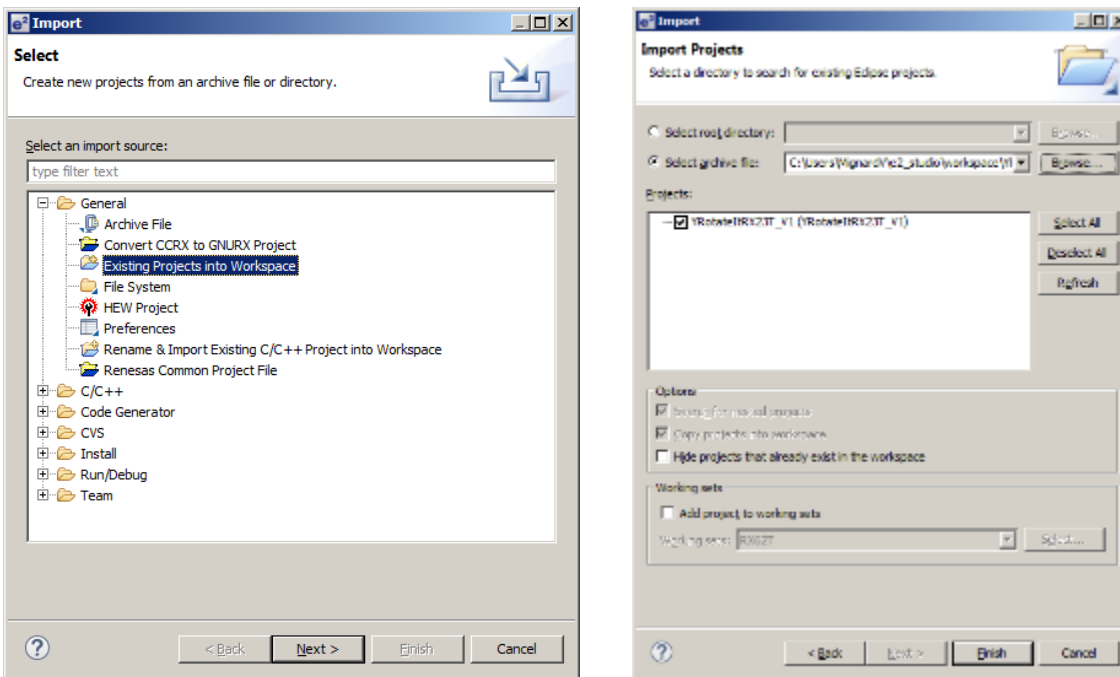
The embedded software delivered in the YROTATE-IT-RX66T kit is developed under e²studio integrated development tool. Please find below the details of the IDE used.



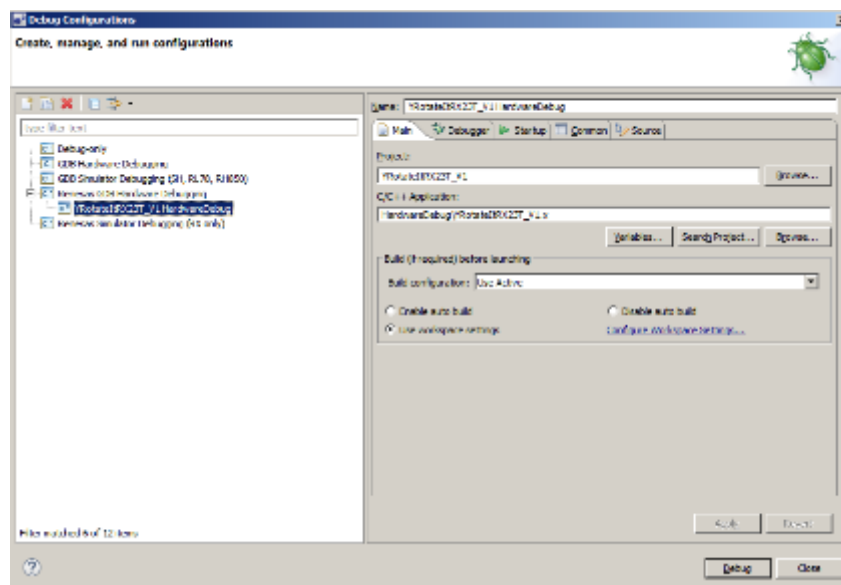
The Renesas RXC Toolchain, version v3.00.00 is used. The compiler used is the “Renesas CCRX”, as visible on the picture below.



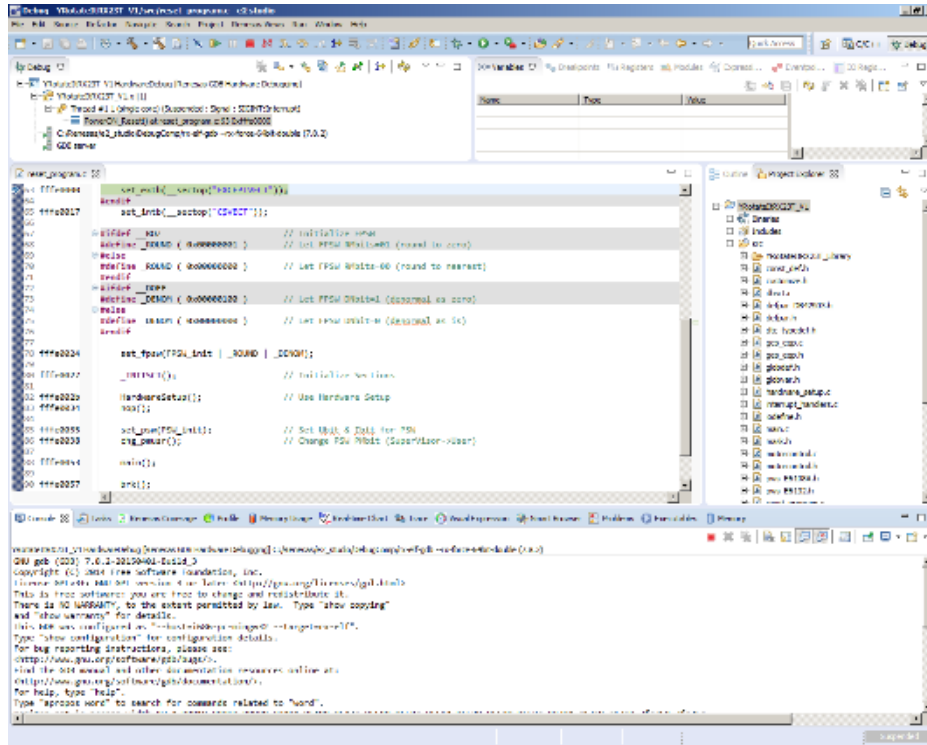
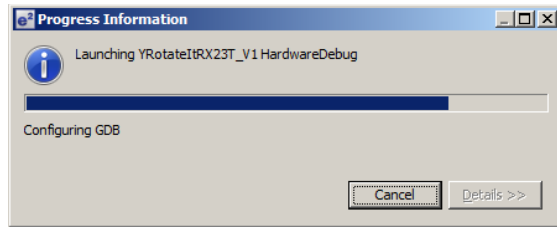
15.2 Project importation into e²studio



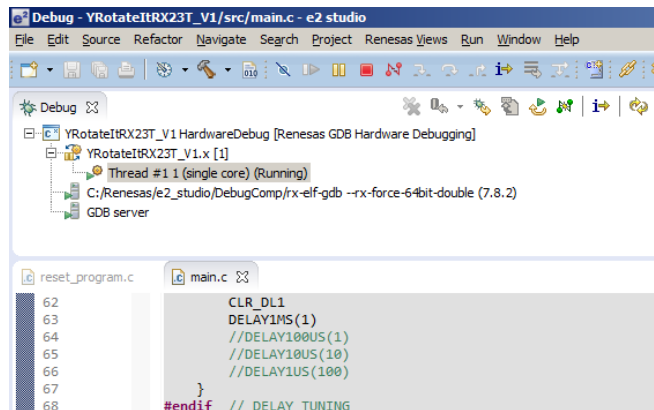
Launch the debugger by clicking on “Debug configuration” and the window below will appear:



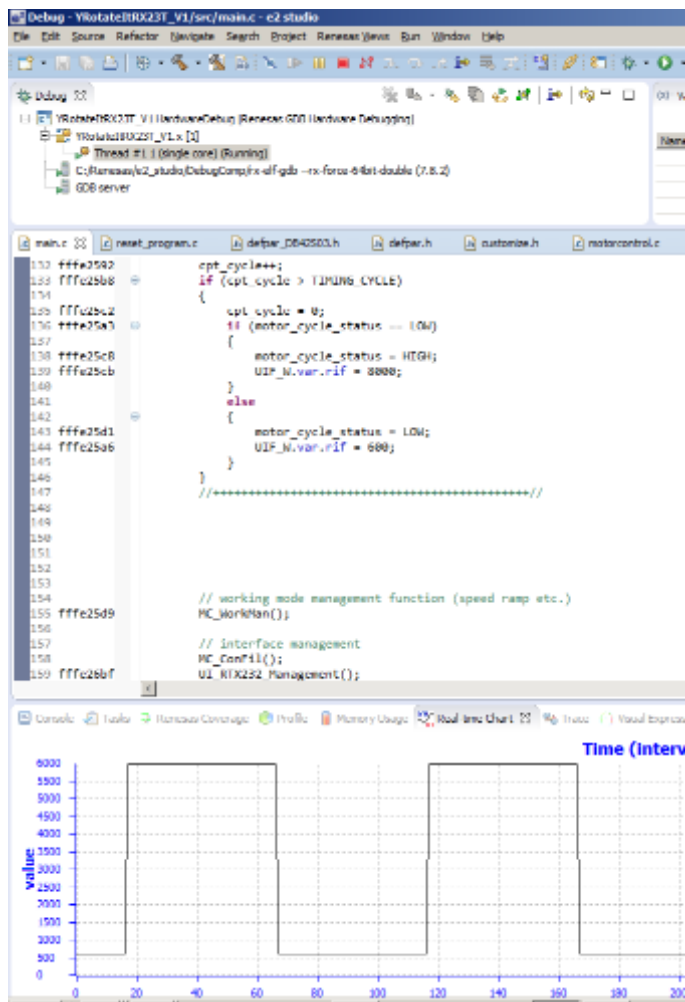
After connection the E1 debugger to the target board powered by USB or external power supply, please click on “Debug” to start the debugging and load the program into the microcontroller flash memory. The windows below appears:



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.



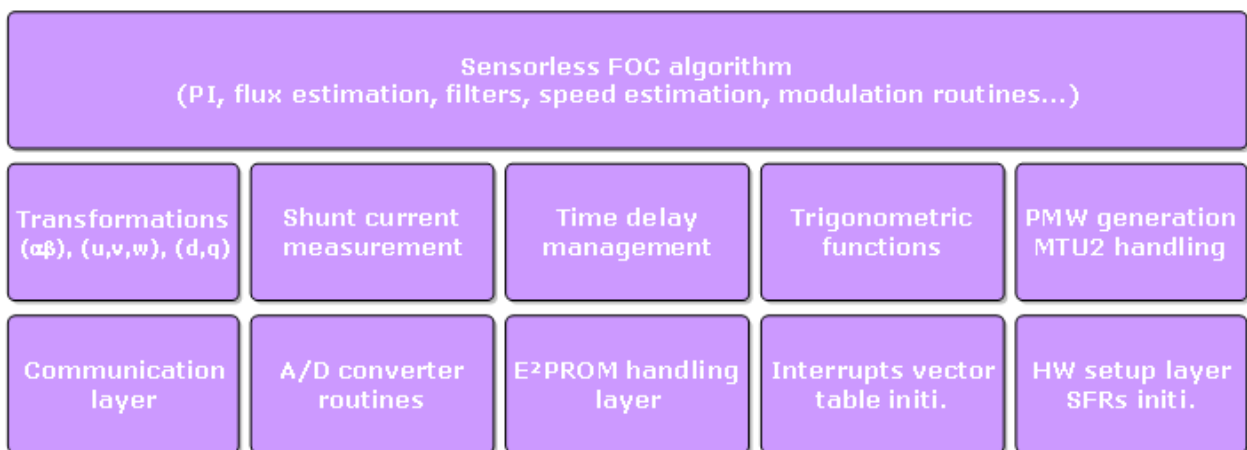
16. Software description and Resources used

The software delivered in the YROTATE-IT-RX66T kit, previously described, is working on the RX66T microcontroller clocked at 160MHz and its operating voltage is 5V which guarantees 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 at the control loop rate which leaves enough time to perform the sensor-less 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 sensor-less vector control algorithm.

- **FLASH memory usage: xxKB and RAM memory usage: yyKB**

The embedded software package is called “**YRotateItRX66T_V01.zip**” running under e2studio environment.

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 modifiable in the GUI.

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

The parameter “**F_RATIO**” is the ratio between the sampling frequency and the PWM and is set to **2**, so it means $2 \times 8000\text{Hz} = 16000\text{Hz}$ (16KHz).

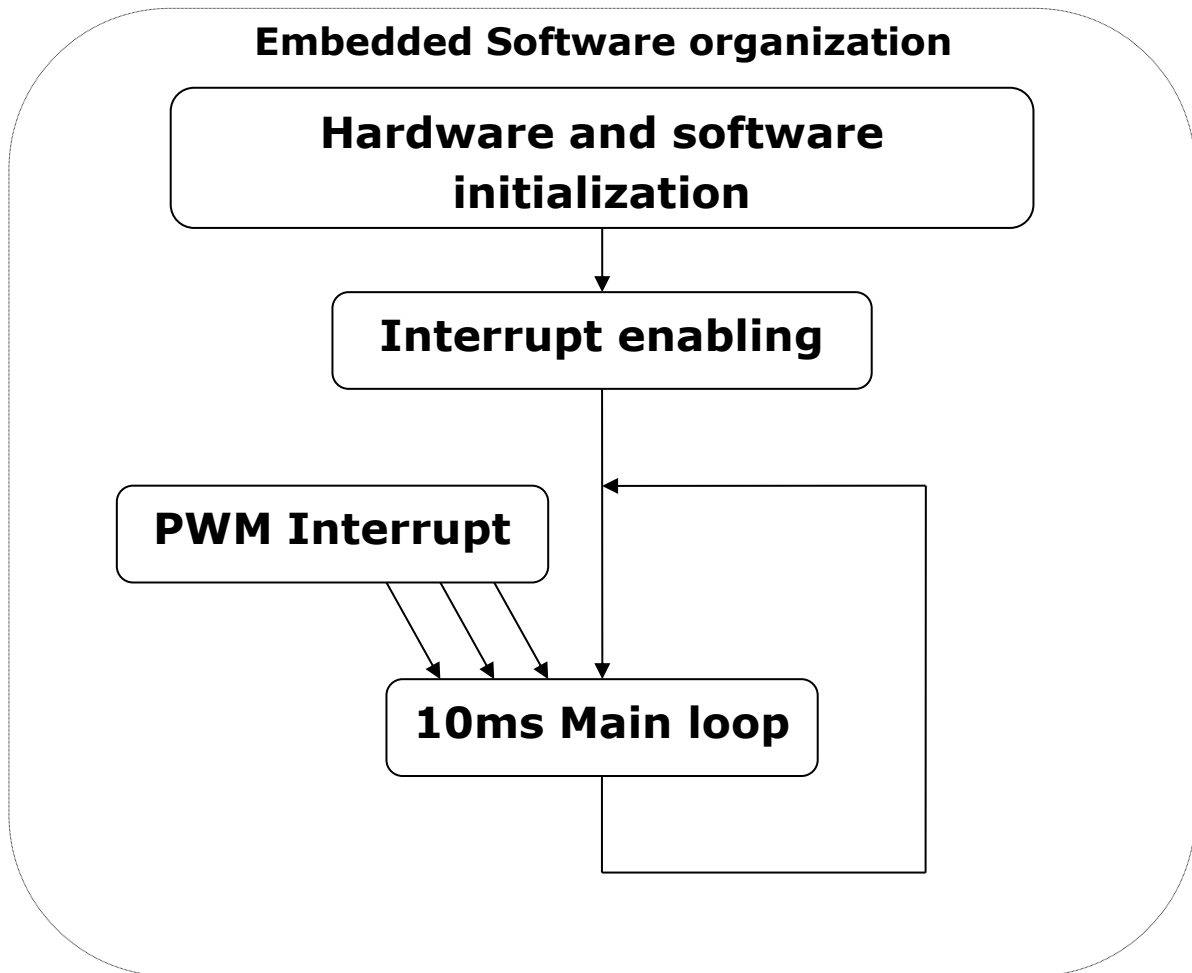
Such parameters can be modify dynamically using the PC GUI without recompiling the overall project changing the parameters below and resetting the board. The PC GUI provide a direct access to the parameters below.

18	SAM_FRE	Set the sampling frequency [Hz] of the control loop
19	F_RATIO	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.

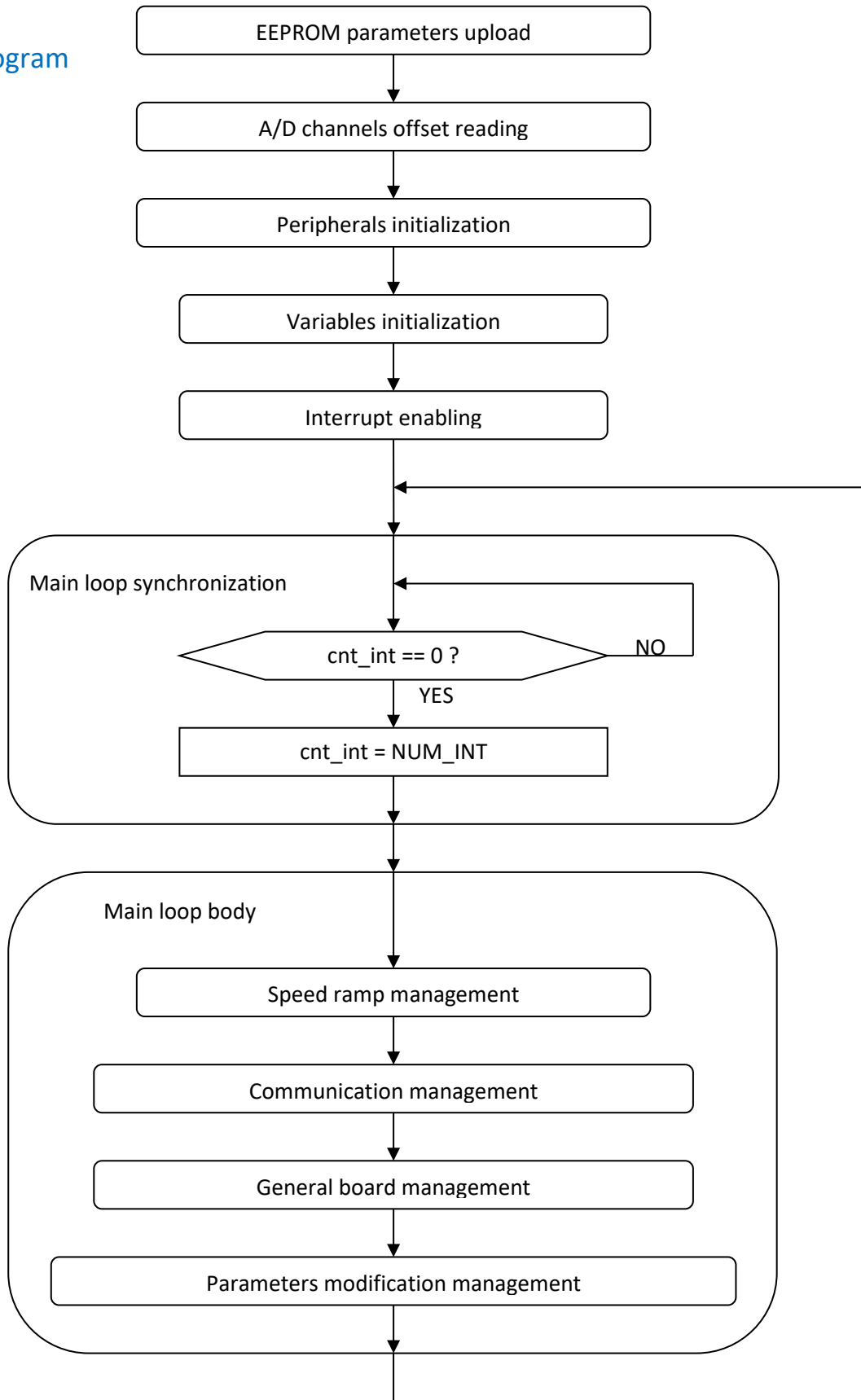
The parameter #18 is setting the control loop speed. By default, 8KHz is selected by entering the value "8000". The PWM frequency can be set to four different values depending on the motor and the applications either **8KHz**, **16KHz**, **24KHz** or **32KHz** by entering either **1, 2, 3, 4** as ratio between the two frequencies.

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

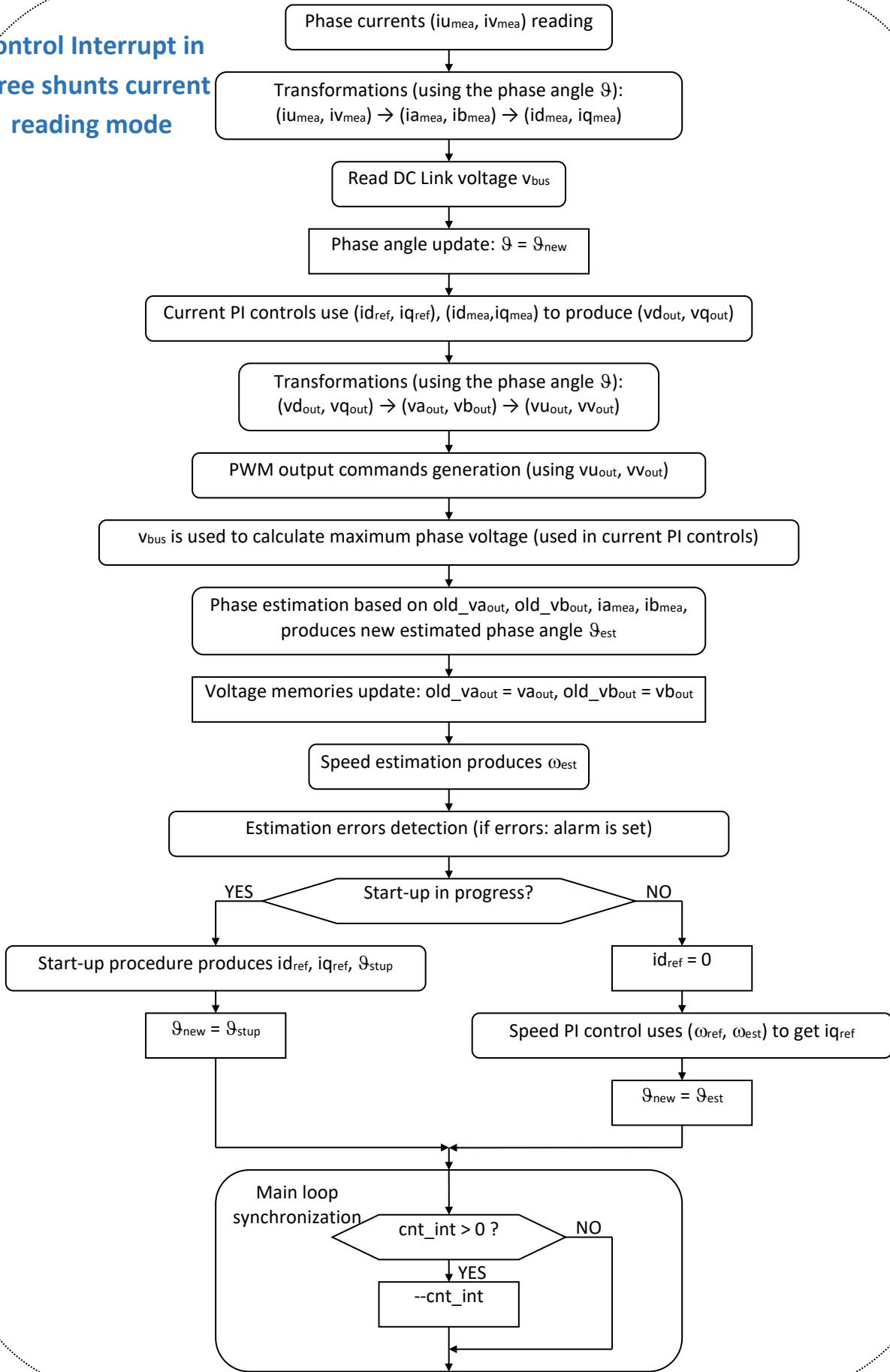
Please find below the flowchart for the main loop, the interrupt service routines and the Automatic Tuning.



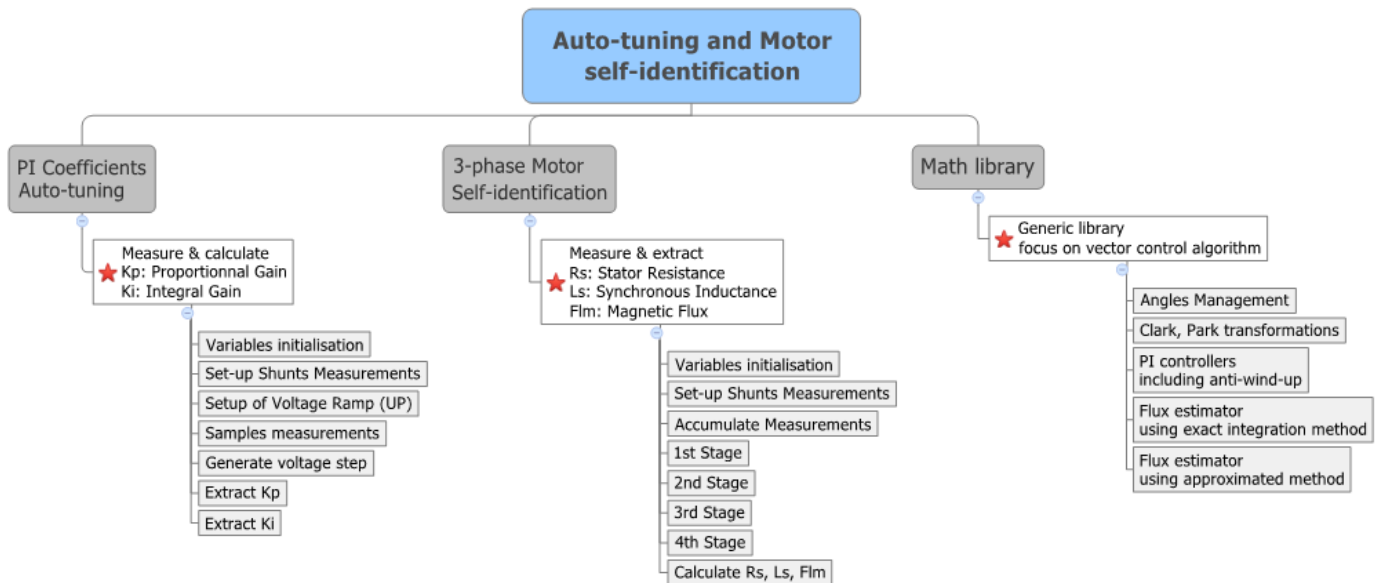
Main Program



Control Interrupt in three shunts current reading mode














The auto-tuning process and the self-identification mechanisms are fully independent from the main sensor-less 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 “YRotateItRX66T_LIB_V1.lib” located in the folder “Library”.

The complete project source code under e2studio is described below.

Subdirectories and C modules:

 Library	Contains sensorless estimator, auto tuning and motor identification routines
 dbstc.c	Renesas project generator: setting of B, R sections
 ges_eqp.c	Eeprom management routines
 globvar.c	Global variables
 hwsetup.c	Hardware setup for MCU SFRs and peripherals (clock etc.)
 intprg.c	Interrupt services routines definition
 main.c	Main program (includes calling of init routines and main loop)
 motorcontrol.c	Peripheral initialization routines, motor control routines and main interrupt
 resetprg.c	Reset procedure
 userif.c	User interface management
 vecttbl.c	Defines address for each interrupt service routine

Header files:

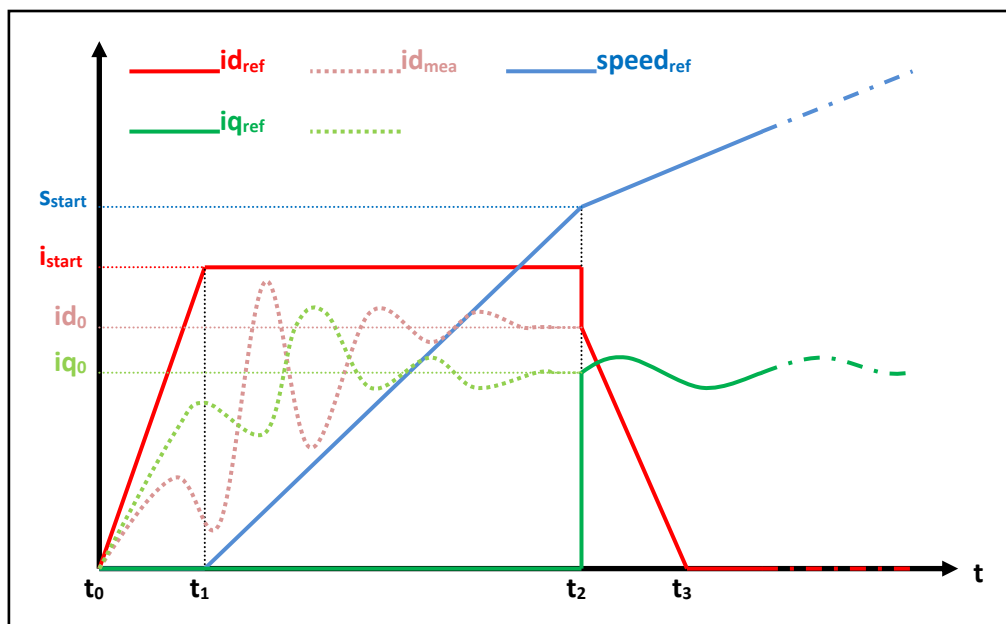
 const_def.h	Constants definition (base timing, A/D converter relations, alarms etc.)
 customize.h	Customizable options (enable debug switches, etc.)
 defpar.h	Default parameter set
 dtc_typedef.h	Type definitions for use of DCT
 ges_eqp.h	Header for ges_eqp.c and parameters definitions
 globdef.h	Global definitions to simplify the use of some peripherals
 globvar.h	Global variables declaration
 hwsetup.h	Header for hwsetup.c
 iodefine.h	Definition of all the SFR
 mask.h	Bitwise masks definition
 motorcontrol.h	Header file for motorcontrol.c
 par_defines.h	Definitions related to parameters
 pws_EBRX66T.h	Definitions related to the specific hardware
 stacksct.h	Stack definition
 typedefine.h	Types definitions
 userif.h	Header file for userif.c
 vect.h	Header file vecttbl.c

17. Start-up procedure

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

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

The following graph illustrates the strategy used (the suffix “*ref*” stands for *reference*, the suffix “*mea*” stands for *measured*).



Referring to the graph, the 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_{d0} and i_{q0} (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_{d0} and i_{q0} , and all the PI controllers are initialized with these new values. The speed PI integral memory is initialized with the value i_{q0} , while the current PI integral memories are initialized with the analogous voltage values v_{d0} and v_{q0} , 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 RX66T microcontroller.

$$g_{\alpha} = \frac{2}{3} \left(g_u - \frac{1}{2} g_v - \frac{1}{2} g_w \right) = g_a$$

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

(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)

$$g_d = g_{\alpha} \cos(\mathcal{G}) + g_{\beta} \sin(\mathcal{G})$$

$$g_q = -g_{\alpha} \sin(\mathcal{G}) + g_{\beta} \cos(\mathcal{G})$$

(α, β) → (d, q)

$$g_{\alpha} = g_d \cos(\mathcal{G}) - g_q \sin(\mathcal{G})$$

$$g_{\beta} = g_d \sin(\mathcal{G}) + g_q \cos(\mathcal{G})$$

(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 enabled by default in the RX66T inverter kit.

Please find below the fundamental equations:

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

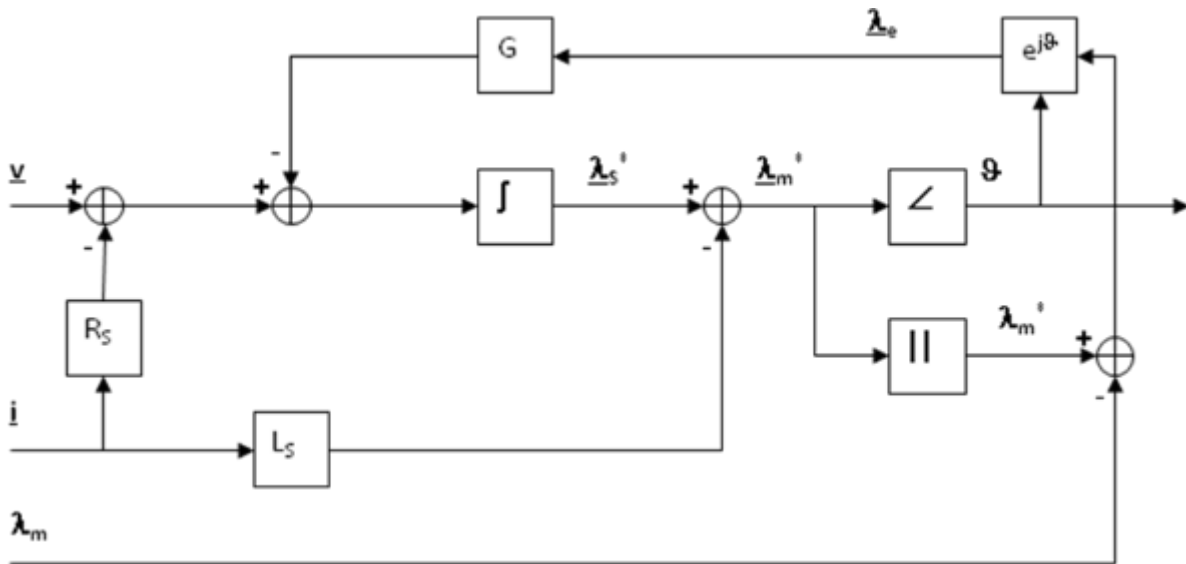
$$y = \Lambda_m \sin(\vartheta) = \lambda_\beta - L_S i_\beta = \lambda_{\beta 0} + \int_0^t (v_\beta - R_S i_\beta) dt - L_S i_\beta$$

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

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

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. These difficulties have been fronted correcting 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.

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.

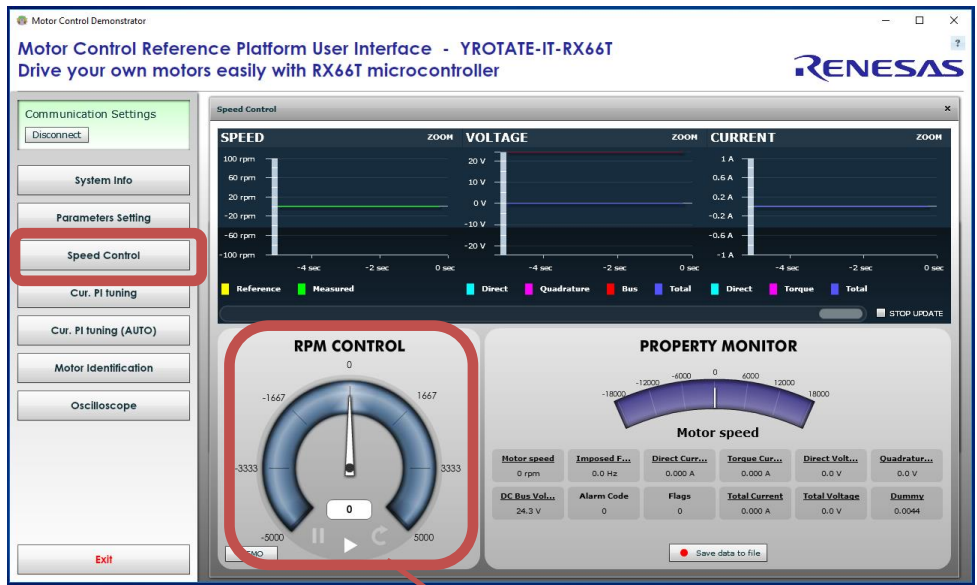
20. PC Graphical User Interface in details

Please install the Motor Control PC GUI on your machine by following the instructions of the Quick Start Guide delivered in the YROTATE-IT-RX66T kit. After connecting the Nanotec Motor (DB42S03, 24V_{DC}, 4000RPM), please connect the board RX66T and select the COM port or use the Auto-detection mechanism.

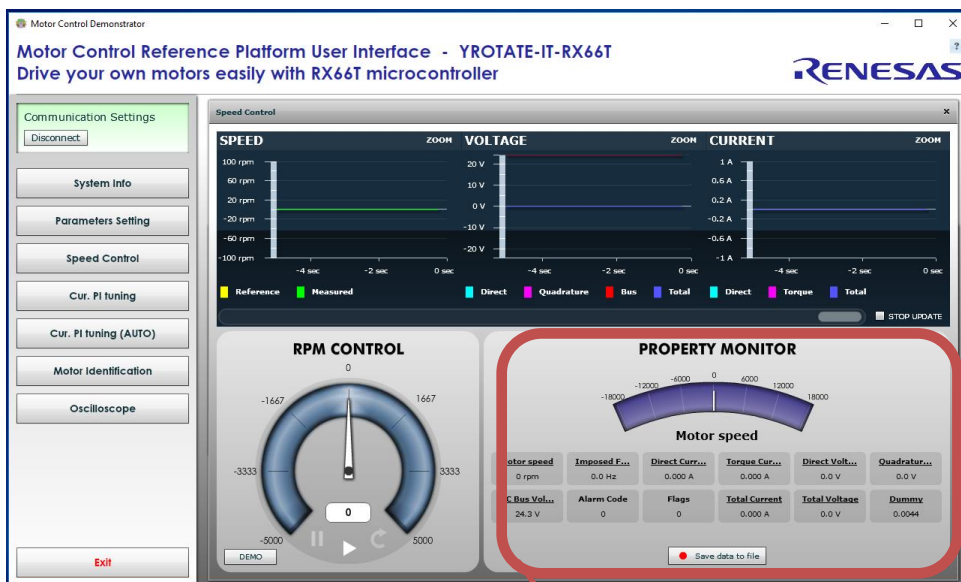
The PC Graphical User interface supports such Operating Environment:

- 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 and change turn direction



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:



Click to record in real-time all the values and store it in a “.CSV” format file.

Monitor the recording of the values to be stored in the “.CSV” file.

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



- Alarm 0: Normal operation
- Alarm 1: EEPROM fault
- Alarm 2: Hardware Overcurrent
- Alarm 3: Loss of phase

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 in "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, 4, 6:

The alarm 2 is called “hardware over-current” 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 over-current signal. The hardware over-current 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.

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

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.

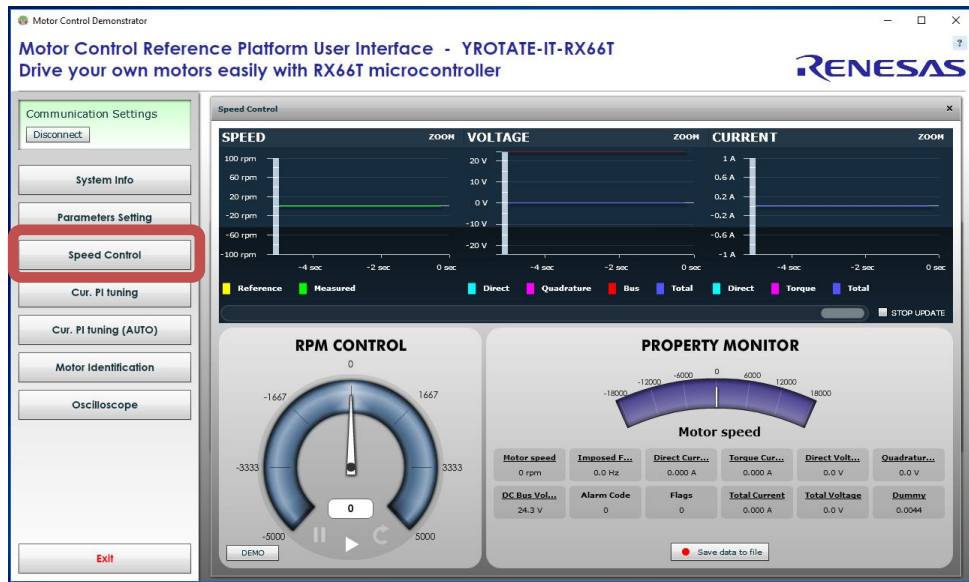
Alarm code 3, 5, 7:

The alarm 3 is called “loss of phase” and described in the software by “TRIP_ALL”. This alarm is produced when the sensor-less 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.

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



The detailed description of each parameter is displayed when pointing the mouse on the question mark. 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 board eeprom.

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

There are three exceptions related to the parameters #5, #19 and #20:

5	POLE PAIRS	Set the number of pole pairs of the motor (only used to calculate the speed)
19	SAM_FRE	Set the sampling frequency [Hz] of the control loop
20	F_RATIO	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), by setting the right value in parameter #20.

Parameter #5 sets the number of pole pairs (only used to visualize the speed).

For all these three parameters the writing is not immediate, but requires to reset the board to take effect. After entering the new values, in this case 8KHz of sampling frequency and a ratio of 2 in parameter #20 to obtain 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-RX66T kit is driving any 3-phase Permanent Magnet Motors using a sensor-less 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, the minimum speed is **600RPM**. 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 reaches easily 6200RPM, its maximum rated speed without load. Of course, in the embedded software you can enable flux weakening techniques and providing more current to the board the motor can reach **8000RPM**.

21. Eeprom parameters: detailed description

To maintain the compatibility with previous versions of the GUI, which were referred to a single motor, a trick has been used: putting in parameter n°0 the value 0, the parameters displayed the settings and the measurement are all referred to the first motor, putting the value 1 all the data become relative to the second motor, and putting 2 one can drive the third motor. In such way it become possible to tune separately all the three motors, without adding more windows. Other values in parameter 0 maintains the old meaning (ex. magic value 33 means reset to default). Please find below the software parameters list including their full description.

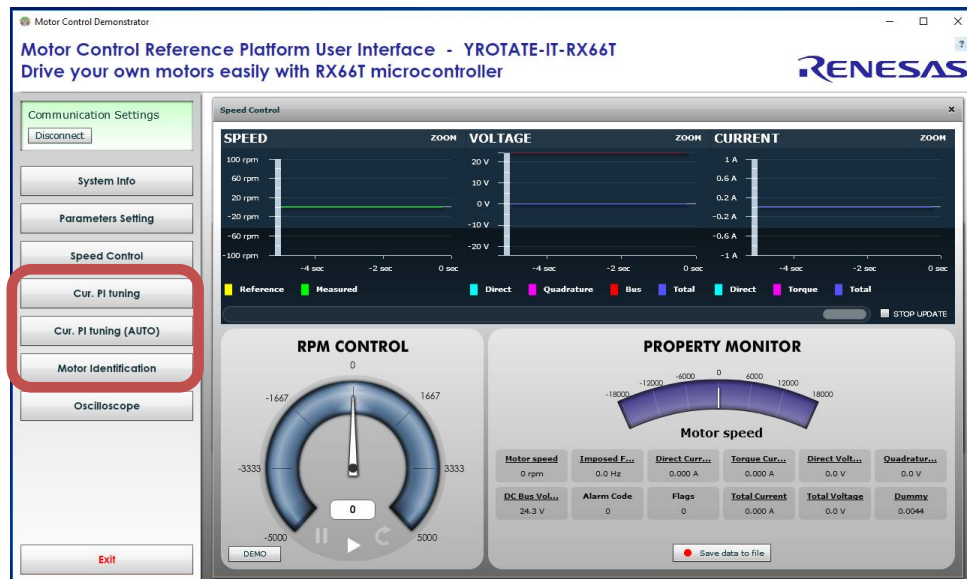
Number	Short name	Description
0	SEL_OP	Operation mode setting
1	RPM_MIN	Minimum speed [rpm]
2	RPM_MAX	Maximum speed [rpm]
3	R_ACC	Acceleration [rpm/s]
4	R_DEC	Deceleration [rpm/s]
5	C_POLI	Number of polar couples
6	I_START	Start-up phase current (peak) [Ampere]
7	I_MAX	Maximum phase current (peak) [Ampere]
8	R_STA	Stator resistance [Ohm]
9	L_SQ	Q axis synchronous inductance [Henry]
10	PM_FLX	Permanent magnets linked flux [Weber]
11	KP_CUR	Current loop proportional coefficient
12	KI_CUR	Current loop integral coefficient
13	KP_VEL	Speed loop proportional coefficient
14	KI_VEL	Speed loop integral coefficient
15	FB_GAIN	Flux amplitude feedback gain
16	PHA_OFF	Phase offset [deg] to be added to the phase estimation
17	ST_TIM	Start-up acceleration time [sec]
18	SAM_FRE	Sampling frequency [Hz] of the control loop
19	F_RATIO	Ratio between the PWM frequency and sampling frequency
20	PAR_20	Not used

The indexes reported in the table are related to the motor connected to the first power stage. The parameters related to other power stages are the repetition of the parameters 1, .., 17, and are saved in eeprom from the place 21 to 37, and 39 to 55. But they are overloaded when 1 or 2 is written on parameter n°0.

To set the speed for one motor set first the parameter 0, then play with the interface. **You can also set the same speed for all the three motors writing 3 in parameter 0 (visualizations will be relative to the first motor).**

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 available for and shown below:



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

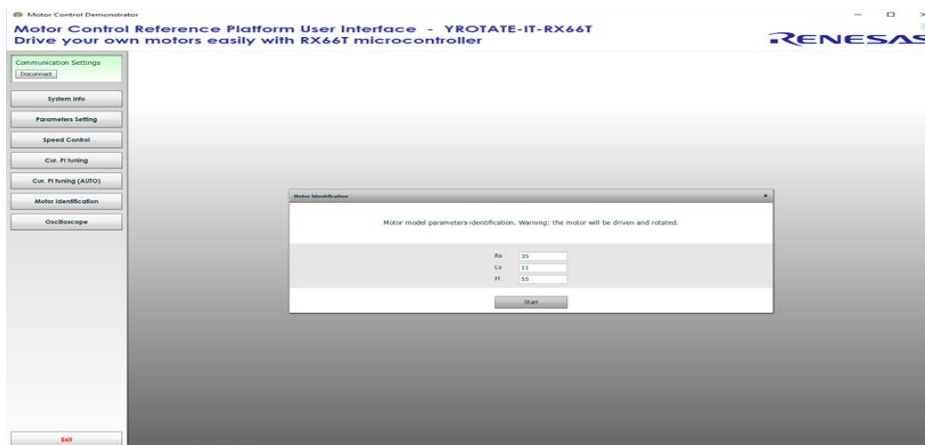
In terms of AC Brushless motor driven in sinusoidal mode and FOC algorithm, the most important parameters to tune are:

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

Please find below the auto-tuning process step by step of the Nanotec Motor DB42S03 which is a low voltage Permanent Magnet Synchronous Motor. The auto-tuning procedure will be performed using the kit running the sensor-less vector control algorithm.

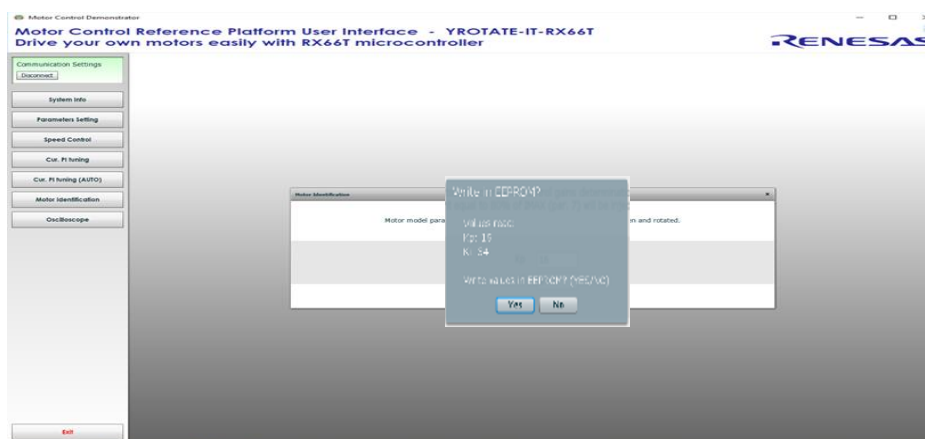
- a) Let's setup the motor control kit for external power supply: the jumper JP1 and JP2 needs to be set to 1-3 position as explained in the "Chapter 3 Power Supply selection".
- b) Let's connect the 24V_{DC} Power supply to the RX66T motor control reference kit.
- c) Now, connect the USB cable to the PC and the Kit and connect the 24V to the kit and the motor to the kit: Launch the PC GUI, select the right .ini file (RX66T Kit) from the first tab and in the second tab select "Auto-detect"; press "Connect"; on the left hand side, you can find the buttons: "Cur. PI tuning", "Cu. PI tuning (AUTO)", "Motor Identification" and "Oscilloscope" which are needed for the self-calibration of the motor.
- d) 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 parameter #0 (Operation Select) and push the RESET button on the board. At the restart, the default values will be loaded into the eeprom.
- e) Set the maximum current (parameter n°07) as it will influence all the next steps: Click on "Parameters settings", enter the value: 3 (the unit is in A_{pk}) 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.

- f) 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.

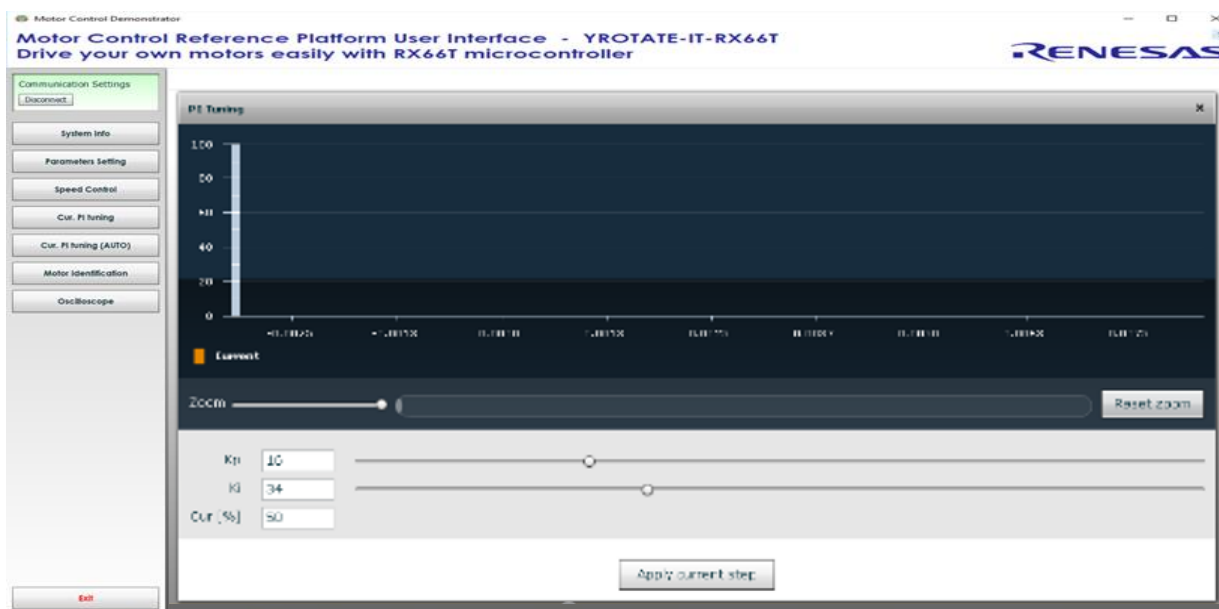


g)

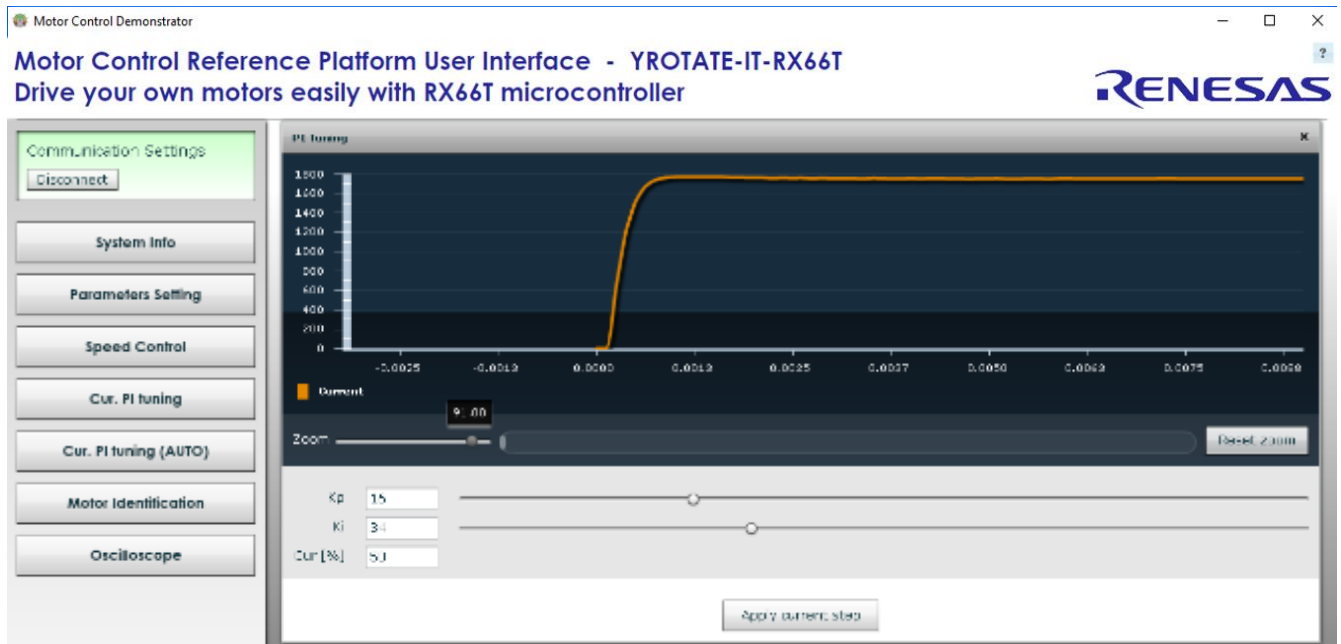
- h) And click on “Yes” to accept the results to be programmed into the EEPROM as shown below.



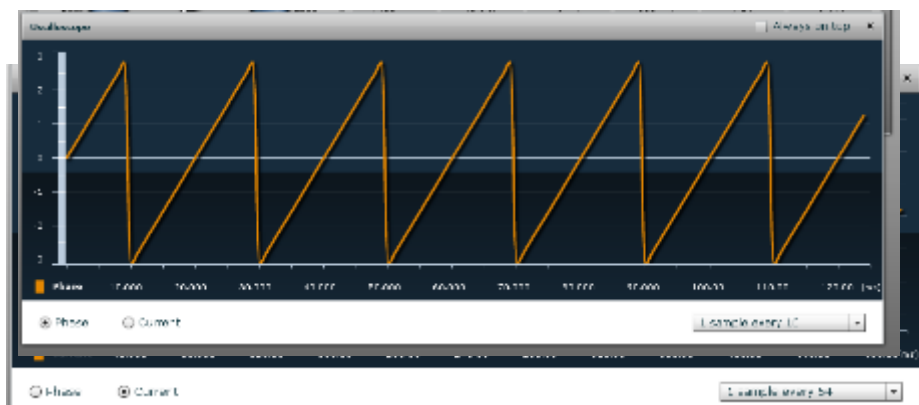
- i) Now click on the button “Cu. PI tuning” to open the manual current PI tuning window and check the step answer by clicking on “Apply current step” button.



- j) 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:



- k) 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.
- l) 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. Finally accept the results to store them into the eeprom by clicking on “yes”. The stator resistance, the synchronous inductance and the permanent magnets linked flux have been measured and tuned.
- m) Now please click on “parameters settings”, enter the number of pole pairs of the motor (parameter n°5) and reset the board; this is needed to give the correct speed reading and to make a sense to min and max speed.
- n) Enter a minimum speed or 1000RPM, set a start-up current equal to 25% of the maximum current. In our case 25% of 3A is 0.75A. Please enter the value 0.75 into the parameter n°6 and click on the “write” button on the right hand side. Then let’s close the window and try to run the motor. Please click on the button: “Speed Control”, and to start the motor, let’s enter a speed which is 1.5 times the minimum speed, in this case 1500RPM.
- o) 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.



- p) Let's start with an intermediate value and adjust it in order to see some periods of the current or the phase. When the motor is running, you can adjust the two speed PI parameters: the proportional and integral terms: #13 and #14. Please open the Oscilloscope window and the Parameters Setting windows together.
- q) To tune the coefficients, start by increasing the Parameter n°13 (Kp) until the instability arises; this can be displayed in the current or phase waveform window. 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". 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).



- r) Test the parameters found in all the speed ranges and different rotations. Finally the parameters list can be saved in a file in .CSV ("Save" button) or .h file ("Create .h" button); the .h file can be included in the project, substituting defpar.h already present.

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.

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.

The maximum serial communication speed tested is 76.6 KBd.

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

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

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
'h'    = reading in measurement samples vector
'l'    = long reading in ram table
'L'    = long writing in ram table
'p'    = long reading in parameters vector
'P'    = long writing in parameters vector
'Y'    = long reading in parameters minimum values
'Z'    = long reading in parameters default values
'J'    = long reading in parameters maximum values
```

Possible master frames (questions):

```
check (l=5):
  l      ?      s      c      k
long reading (l=7):
  l      ?      s      l      a      n      k
long writing (l=7+4*n):
  l      ?      s      L      a      n      D13   D12   D11   D10   ..
  ..      Dn3   Dn2   Dn1   Dn0   k
```

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:

'e' = check answer (programs with floating point parameters)
 'h' = reading in measurement samples vector
 'l' = long reading in ram table
 'L' = long writing in ram table
 'p' = long reading in parameters vector
 'P' = long writing in parameters vector
 'Y' = long reading in parameters minimum values
 'Z' = long reading in parameters default values
 'J' = long reading in parameters maximum values

Possible slave frames (answers):

nok (l=5):

| # s o k

check (l=5):

| ! s e k

long reading (l=7+4*n):

| ! s l a n D13 D12 D11 D10 ..
 .. Dn3 Dn2 Dn1 Dn0 k

long writing (l=5):

| ! s L k

To understand in more details the software implementation, please read the source files `userif.c` and `userif.h`, parts of the embedded software.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	July 23, 2019		First Edition
1.10	Feb 6, 2020		Second Edition

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.