RENESAS

16

RL78/F14 BLDC Starter Kit

User's Manual: Hardware

RENESAS MCU
RL78/F14 Series

Y-BLDC-SK-RL78F14

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

For updates of the Starter Kit software and documentation please check:

http://www.renesas.eu/update?oc=Y-BLDC-SK-RL78F14

Renesas Electronics
www.renesas.com

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.

6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

    "Standard":   Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

    "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti- crime systems; safety equipment; and medical equipment not specifically designed for life support.

    "Specific":   Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority- owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Table of Contents

1. Introduction

The Renesas Motor Control starter kit called Y-BLDC-SK-RL78F14, is based on the RL78/F14 device from the powerful 16-bit RL78 microcontroller family with a maximum operating frequency of 32MHz and delivering up to 52DMIPs.

The kit enables engineers to easily test and evaluate the performance of the RL78/F14 in a laboratory environment when driving any 3-phase Permanent Magnet Synchronous Motor (e.g. Brushless Motor) 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 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.

The hardware is designed for easy access to key system test points and for the ability to hook up to an RL78/F14 debugger known as E1.

The Y-BLDC-SK-RL78F14 Starter Kit 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.
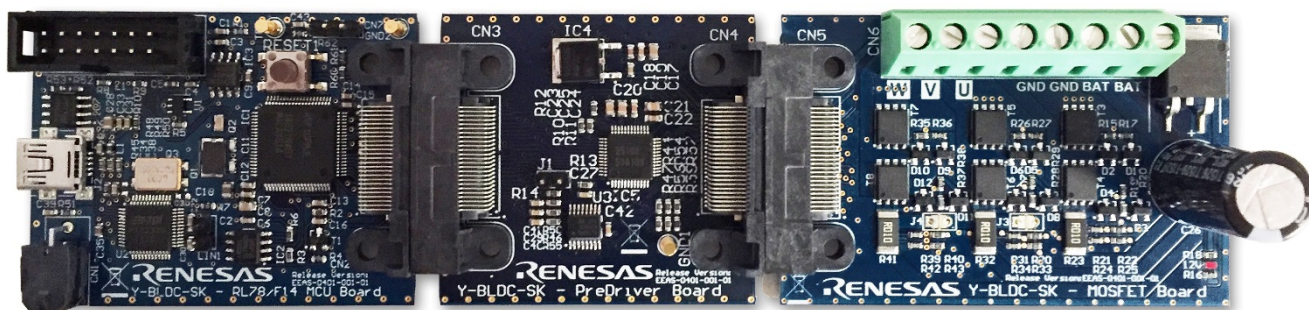

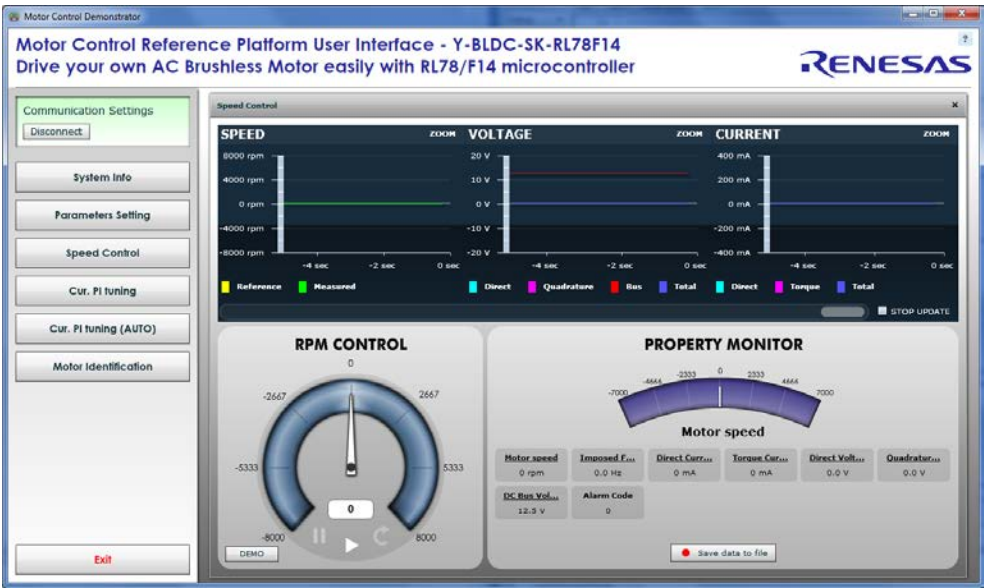**Figure 1.** **RL78/F14 BLDC Starter Kit Top View in Three Shunt Configuration**

Figure 2.   *Overview of PC Graphical User Interface*

CAUTION

1. **Do not connect Power Source with more than 3A before loading your Software to the device!!!**

   Otherwise there may be a short circuit trough the FETs.

   The board can measure up to 17A trough each phase shunt but however as heat dissipation is untested for more than 36W power supply it is not recommended to use a bigger power supply. This is not covered by any warranty.

2. **When power supply of E1 On-Chip debugger is used please note that the maximum current provided by the debugger is limited to 200mA. Thus an external power supply is required in case all functions on the Starter Kit are used to full extend.**

2. 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	Fulling Motor FL28BL26-15V-8006AF, $15V_{DC}$, 8000 RPM
KIT MAX INPUT RANGE	External power supply: 5.3V to $18V_{DC,}$ $3A_{nom}$, $17A_{peak}$, 24V<60s & 40V<500ms
TRANSISTOR USED	Renesas MOSFETs: NP75N04YUG, 40V, 75A
TRANSISTOR DRIVER	Renesas MOSFET driver Intelligent Power Device: R2A25108KFP, Operating voltage: 5.3V to 18V(VBAT), 24V<60s
POWER SUPPLY OPTION	External supply: up to $18V_{DC}$
CURRENT DETECTION	One or three shunts configuration ($10m\Omega$)
USB IC USED ON THE BOARD	FT2232D – Dual USB UART IC from FDTI, 9.6KBd communication speed
MICROCONTROLLER	RL78/F14 (R5F10PMJ), 80-pin QFP, 32MHz, 256KB Flash, 20KB RAM
MCU PERFORMANCE	32MHz, 52DMIPs, 32 CoreMark
KIT CLOCK SOURCE OPTION	32MHz or 24MHz (fIH)
KEY FEATURES (MCU SECTION)	3-phase inverter Timer RD, Event Link controller (ELC), 10-bit A/D Converter, Data transfer controller (DTC), LIN/UART module, CAN interface
MCU EMBEDDED FIRMWARE	Sensor less vector control algorithm (Field Oriented Control)
SWITCHING FREQUENCY	16KHz to 24KHz, 24KHz@32MHz by default (PWM frequency)
CONTROL LOOP FREQUENCY (SAMPLING FREQUENCY)	4KHz to 8KHz, 8KHz@24KHz PWM by default
CONTROL LOOP TIMING	87.5μs under IAR and 78.33μs under CS+ including the FOC loop and current and speed PI controllers
CODE SIZE IN FLASH / RAM	25.87KB / 2.23KB under IAR and 24.79KB / 2.02KB under CS+, including auto/manual PI tuning and Motor identification
TOOL USED, VERSION	IAR Embedded Workbench for Renesas RL78 1.40.X CS+ for CC V4.00.X
COMPILER OPTIMIZATION LEVEL	Maximum optimization for speed
ENVIRONMENT STANDARDS	RoHS compliant including China regulations WEEE, RoHS

Table 1.   Specification and Hardware Overview

The RL78/F14 starter kit consists of three pieces of separated boards, which are congregated by two pairs of connectors, including the system control microcontroller and communications

management board (Y-BLDC-SK-RL78F14 MCU board), the gate pre-driver and current measurement amplifier board (Y-BLDC-SK-RL78F14 pre-driver board) and the integrated low voltage power inverter board (Y-BLDC-SK-MOSFET board).

The PCB is a two layers board and ensures the management of BLDC up to $18V_{DC}$ (24V<60s & 40V < 500ms) and up to $17A_{max.}$
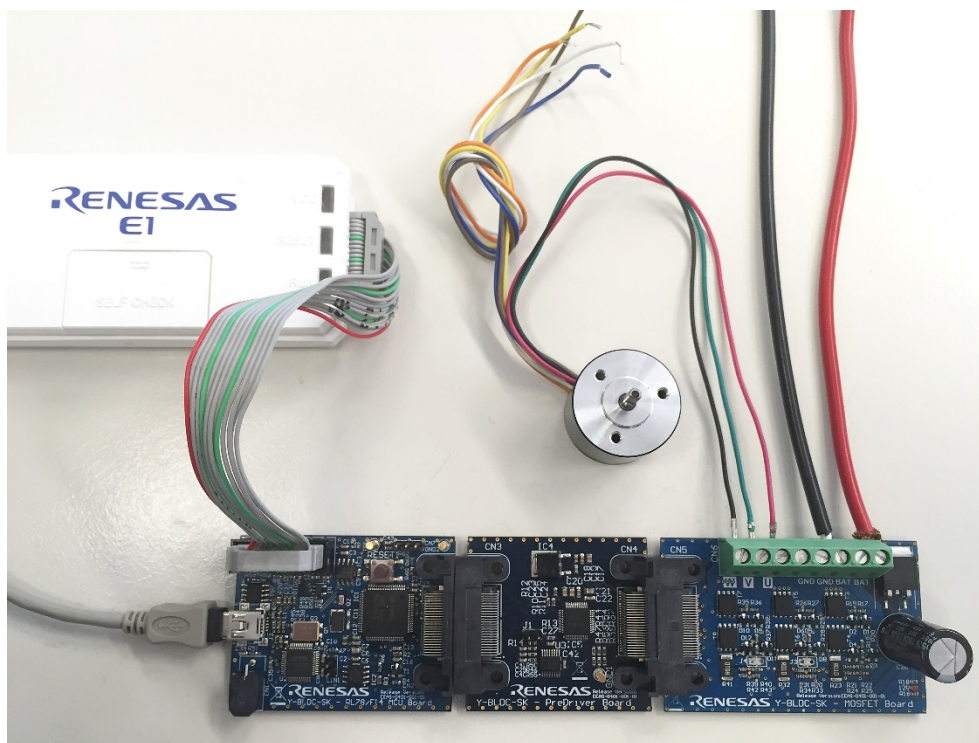
Please find below the Y-BLDC-SK-RL78F14 kit:

Figure 3. *Overview of RL78/F14 BLDC Starter Kit with Motor*

To obtain the maximum flexibility, the YBLDCSKRL78F14 kit includes:

- A complete on-board 3-phase inverter with a low voltage motor (15V), so it becomes easy to test the powerful sensor less algorithm running on the Renesas 16-bit RL78/F14 microcontroller

- DC bus voltage provided via 15V external power supply

- Renesas low voltage MOSFET power devices. Power rating up to 3000W

- One- /Three- shunt current measurement resistors

- Renesas intelligent power device R2A25108KFP to pre-drive the power MOSFET inverter bridge, sense and amplifier the shunt current
  - o Wide range operating: 5.3V to 18V(VBAT), 24V<60s

- o On-chip three phase pre-driver circuit

  - PWM control

  - Totem pole type MOSFET gate drive circuit, high drive capability:

    Ciss = 10000pF

  - Dead time control (adjustable)

- o On-chip current sense amplifier with reference bias buffer

- USB communication using FT2232D USB to Serial UART/FIFO IC. The interface can be configured for

  - o PC GUI control

    - Motor operation, modify motor and control parameters

    - Motor tuning

  - o Offering 9.6KBd communication speed with the PC GUI

In the Y-BLDC-SK-RL78F14 Starter Kit, the RL78/F14 in an 80-pin package was selected to ensure the management of inverter, external communications, three or single shunts, the E1 debugger, three voltages phases, the over-current detection, the Bus voltage monitoring, etc.

In Appendix A: Schematic shows the detailed I/O pins assignment of the RL78/F14 to manage the complete kit.

3. Connectors description

As shown in the following figure, you can find the positions and the descriptions of the connectors present on the board. Please refer to the board schematics for the full description of the connectors in Appendix A: Schematic.

The E1 connector is used for the programming and the debugging of the software running on the RL78/F14. It can be connected to the IAR integrated development environments or CS+ for CC.
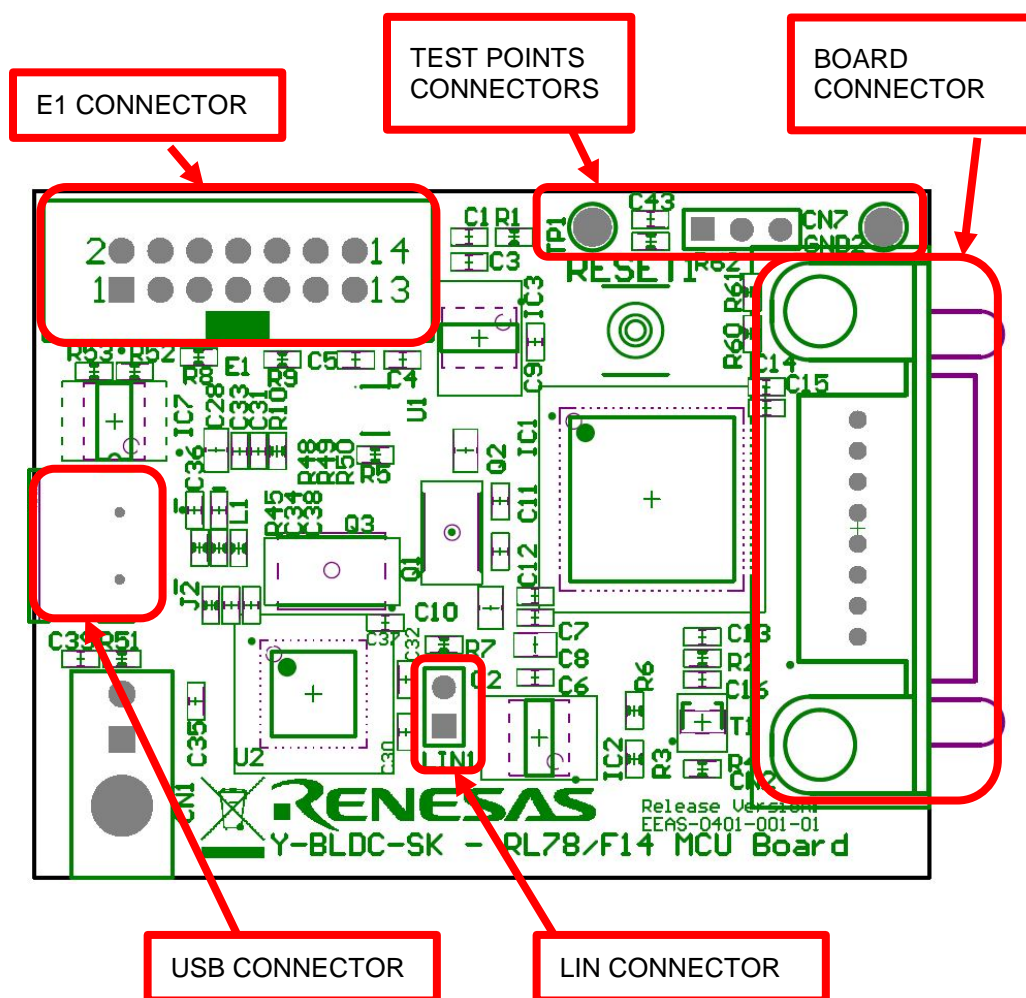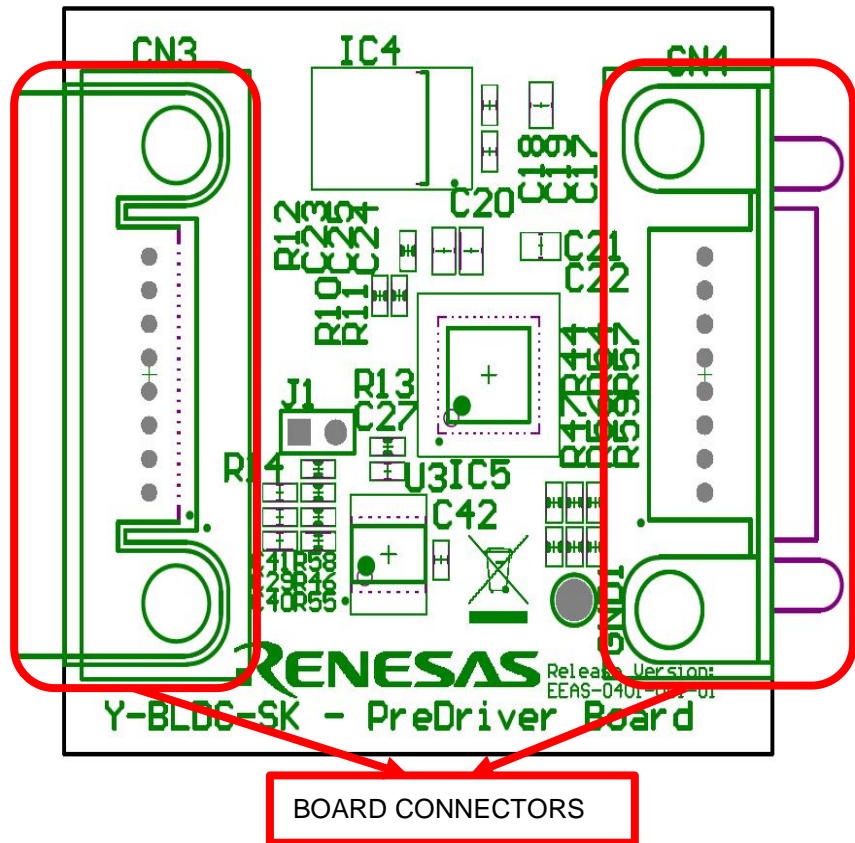

Figure 4.   Connectors of RL78/F14 MCU Board

BOARD CONNECTORS

Figure 5.   Connectors of RL78/F14 Pre-Driver Board

MOTOR CONNECTOR

POWER SUPPLY CONNECTOR

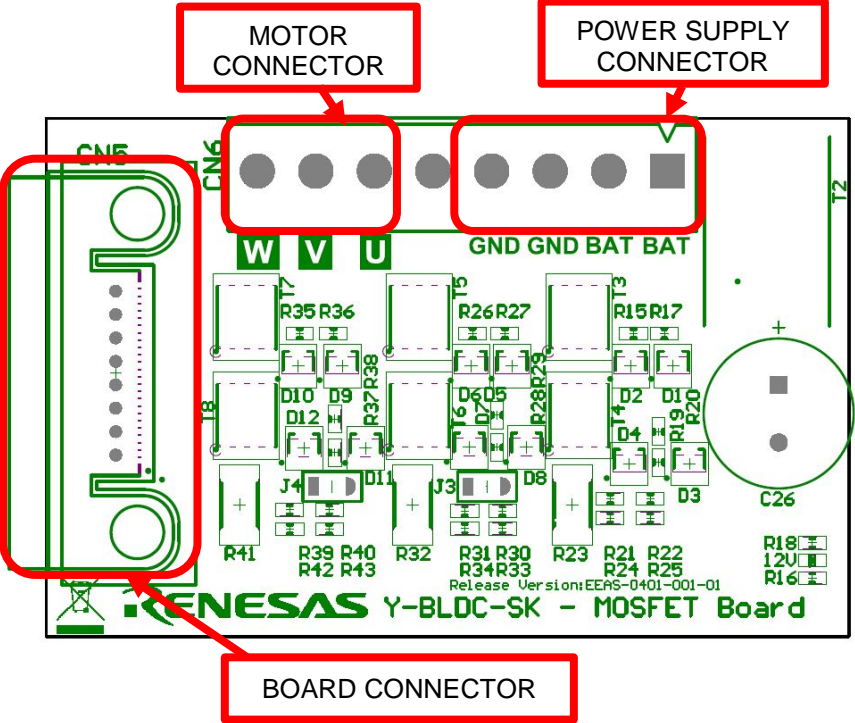BOARD CONNECTOR

Figure 6.   Connectors of RL78/F14 MOSFET Board

4. LED function description

The LED available on the board is directly connected to the three phase bus voltage (15V) and allow the user to understand the status of the board, which indicates the presence of the switches drive supply.
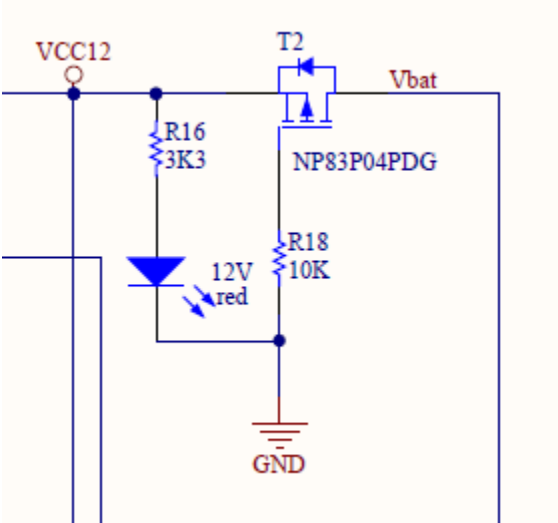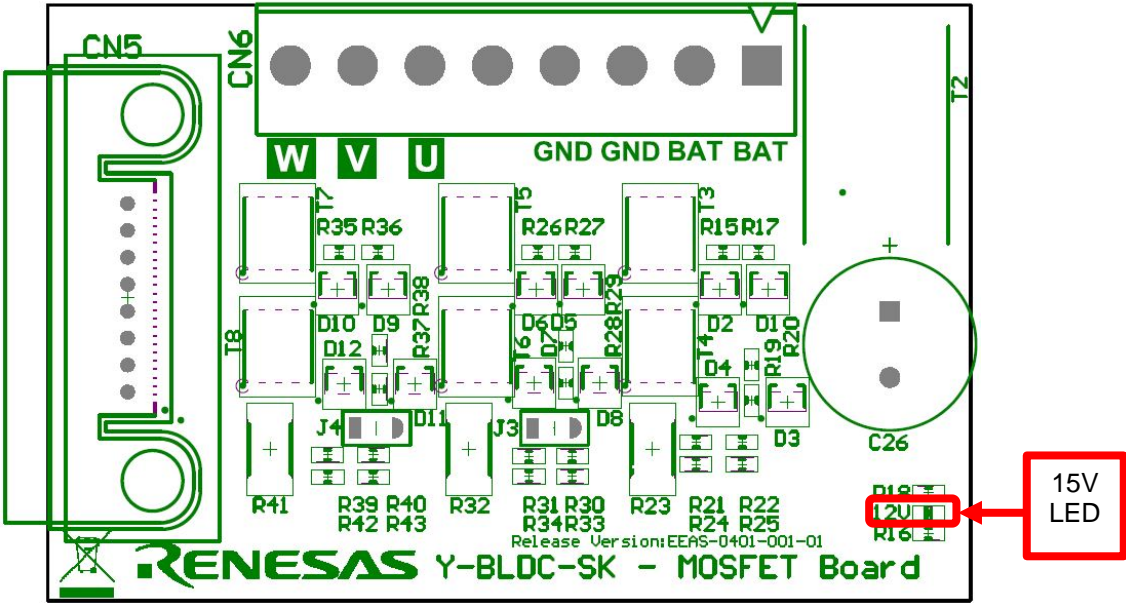

Figure 7.   LED in RL78/F14 MOSFET Board

Figure 8.   LED of RL78/F14 MOSFET Board

5. Test points for debugging

Several specific test points, which connect to some microcontroller pins, are available on the board to visualize with the oscilloscope the behavior of some internal analog signals. For more details about the test points please refer to Appendix A: Schematic.

The figure below is showing the connector TP1 to be used for the tuning of the current PI gains. In this way an oscilloscope is needed to see the response of the system to the stimulation. It is not available in software but another particular procedure to help the calibration --- Cur. PI tuning is offered, in detail see Chapter 20 Motor Auto-calibration using the PC GUI. Tuning can be obtained in the PC GUI, there is no need of an oscilloscope.
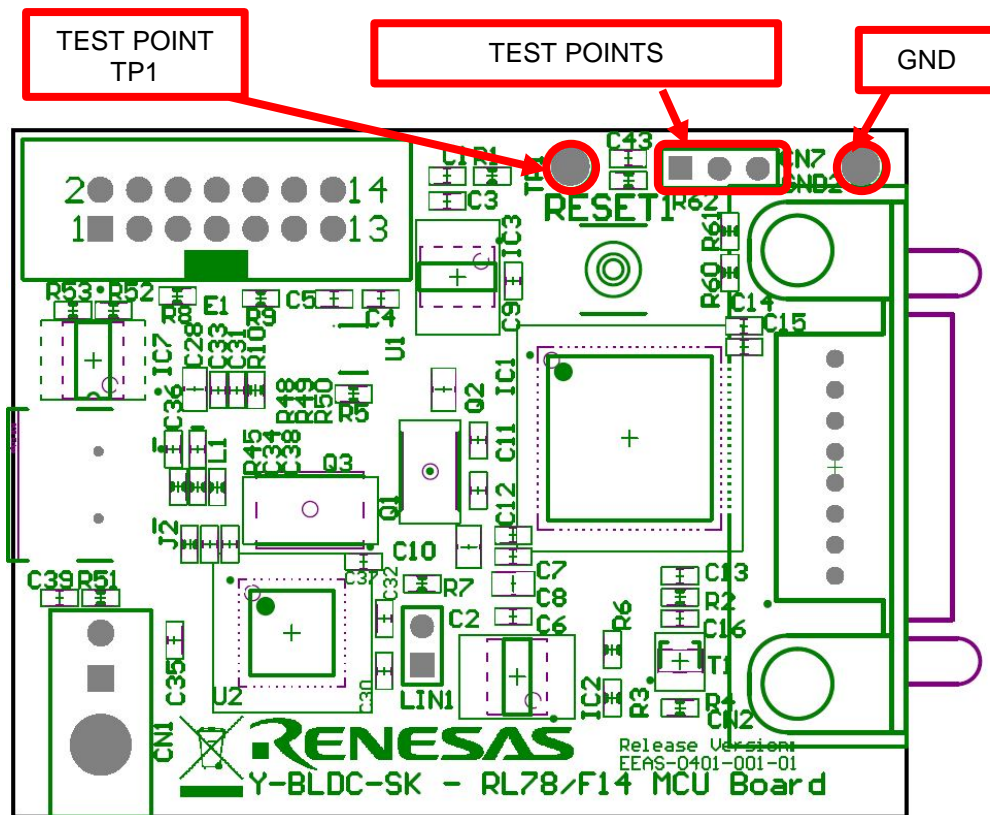

Figure 9.   Test Points of RL78/F14 MCU Board
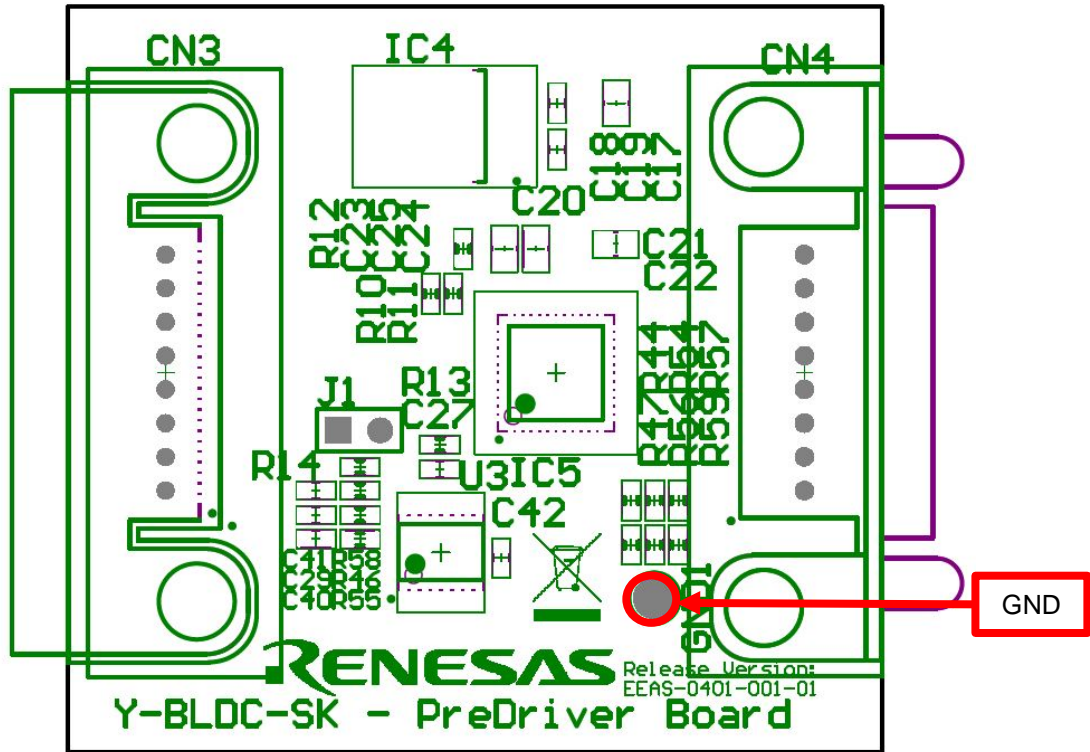
Figure 10.   Test Points of RL78/F14 Pre Driver Board

6. Internal power board description

The power board is a complete 3-phase bridge composed with discrete low voltage and high current MOSFETs. The MOSFETs are the Renesas **NP75N04YUG** n-channel power MOSFETs. Please refer to the data-sheet available on the Renesas website: www.renesas.eu for the switches characteristics and to the board schematics for the details on the driving circuit. The maximum current is **75A**, and the maximum voltage is **40V$_{DC}$**.
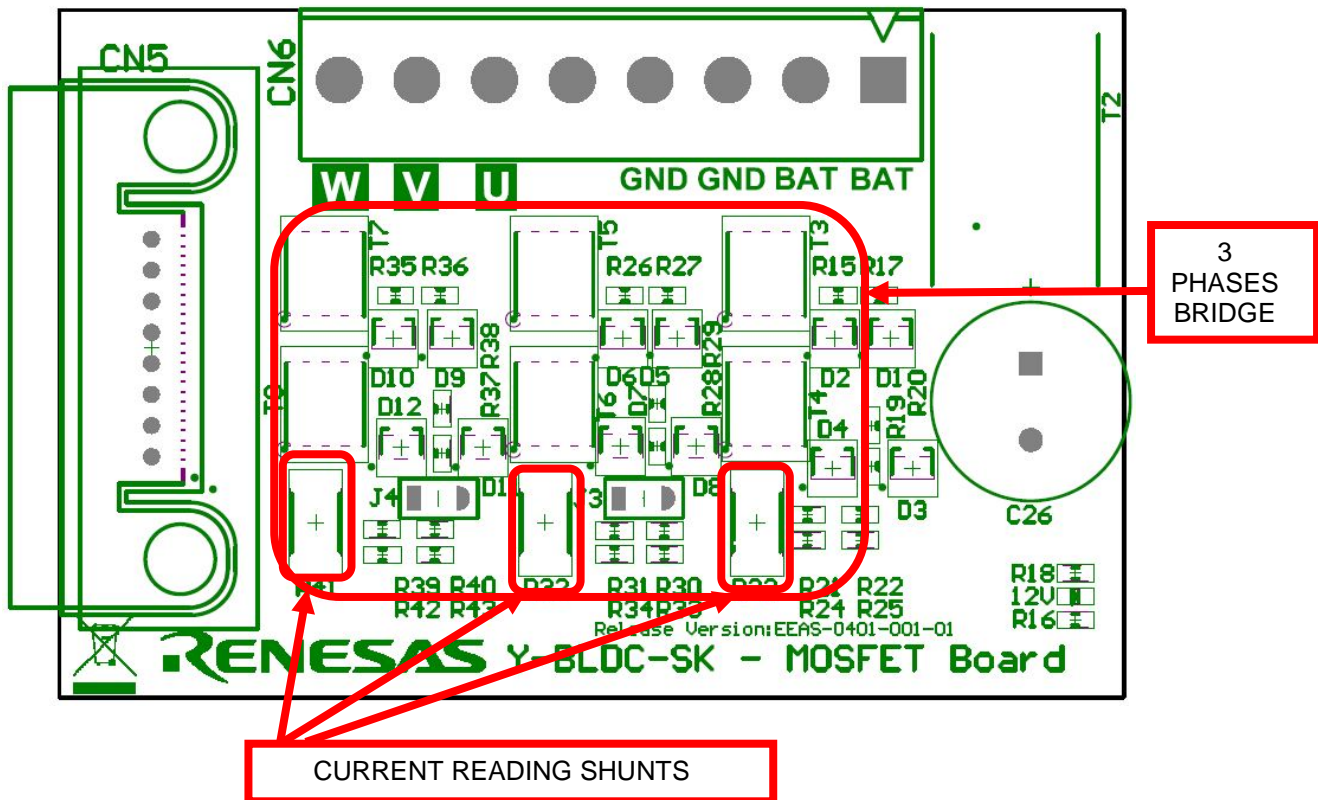
Figure 11. *Three Shunts Current Reading Configuration of RL78/F14 MOSFET Board*

The inverter has the classical schema as showing in the following figure with the three shunts on the lower arms, with the possibility to use a single shunt by removing two of them.
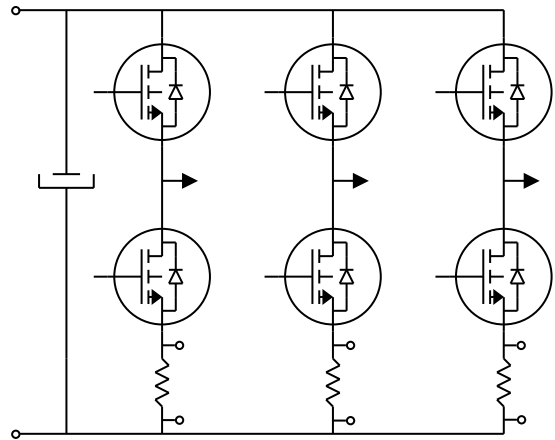
Figure 12. Classical Schema Three Shunts Bridges

Furthermore, the intelligent power device R2A25108KFP is adopted to pre-drive the six low voltage MOSFETS NP75N04YUG.
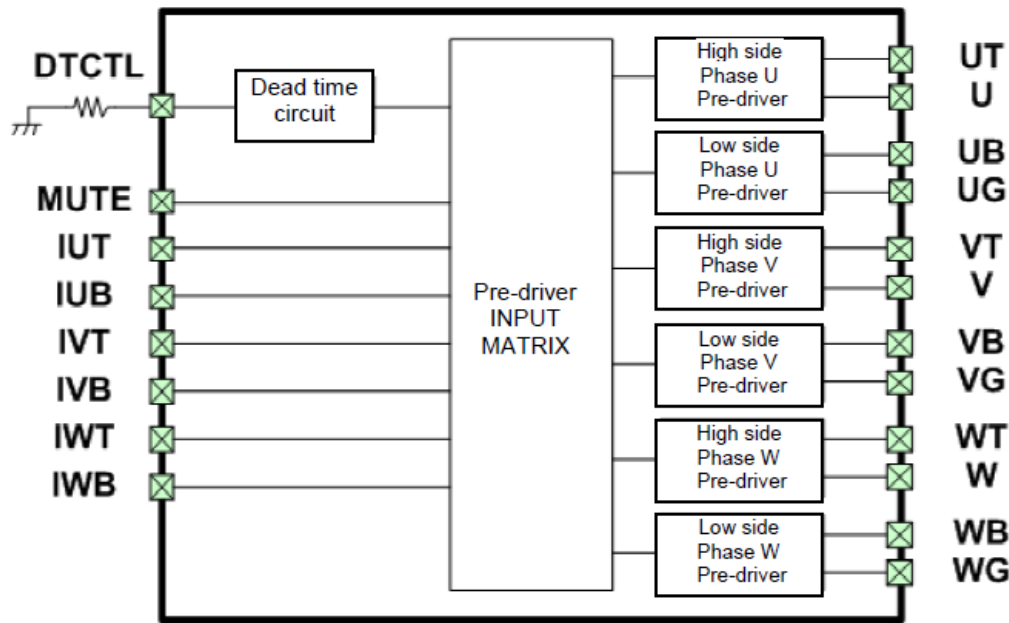
Figure 13. R2A25108KFP for Gate-Driver

Please find in Appendix A: Schematic the drawing in more details.

7. 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 programed into the RL78/F14 flash memory.

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

- Set the J3 and J4 closed, then R41 and R23 are shorted. Only R32 is adopted to measure current.

- Resistors R41 and R23 have to be removed.

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

Single shunt current reading:
Set J3 and J4 closed, remove R41 and R23

Figure 14. *Single Shunt Current Reading Configuration of RL78/F14 MOSFET Board*

8. 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.
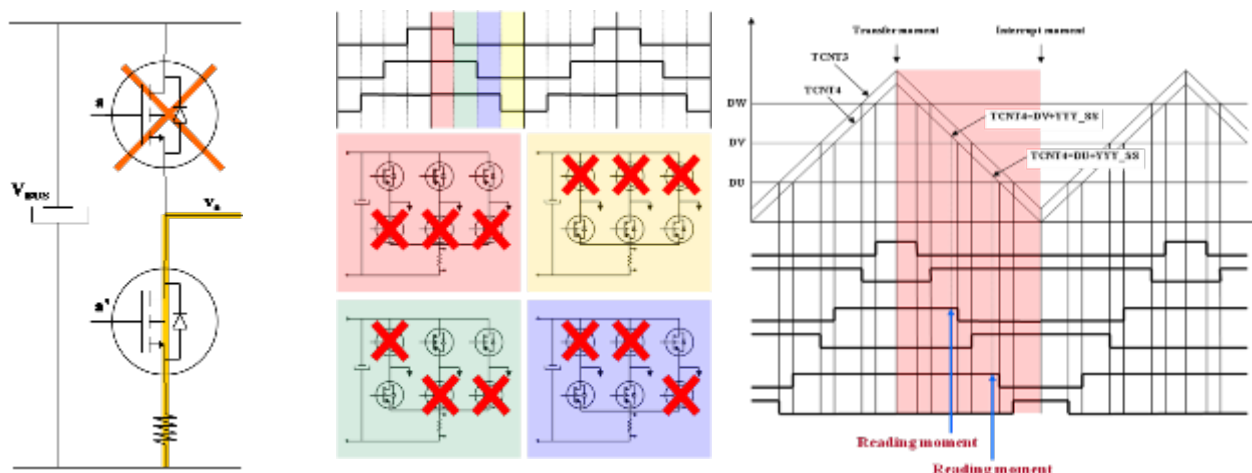
Figure 15. *Three Shunt and Single Shunt Current Reading*
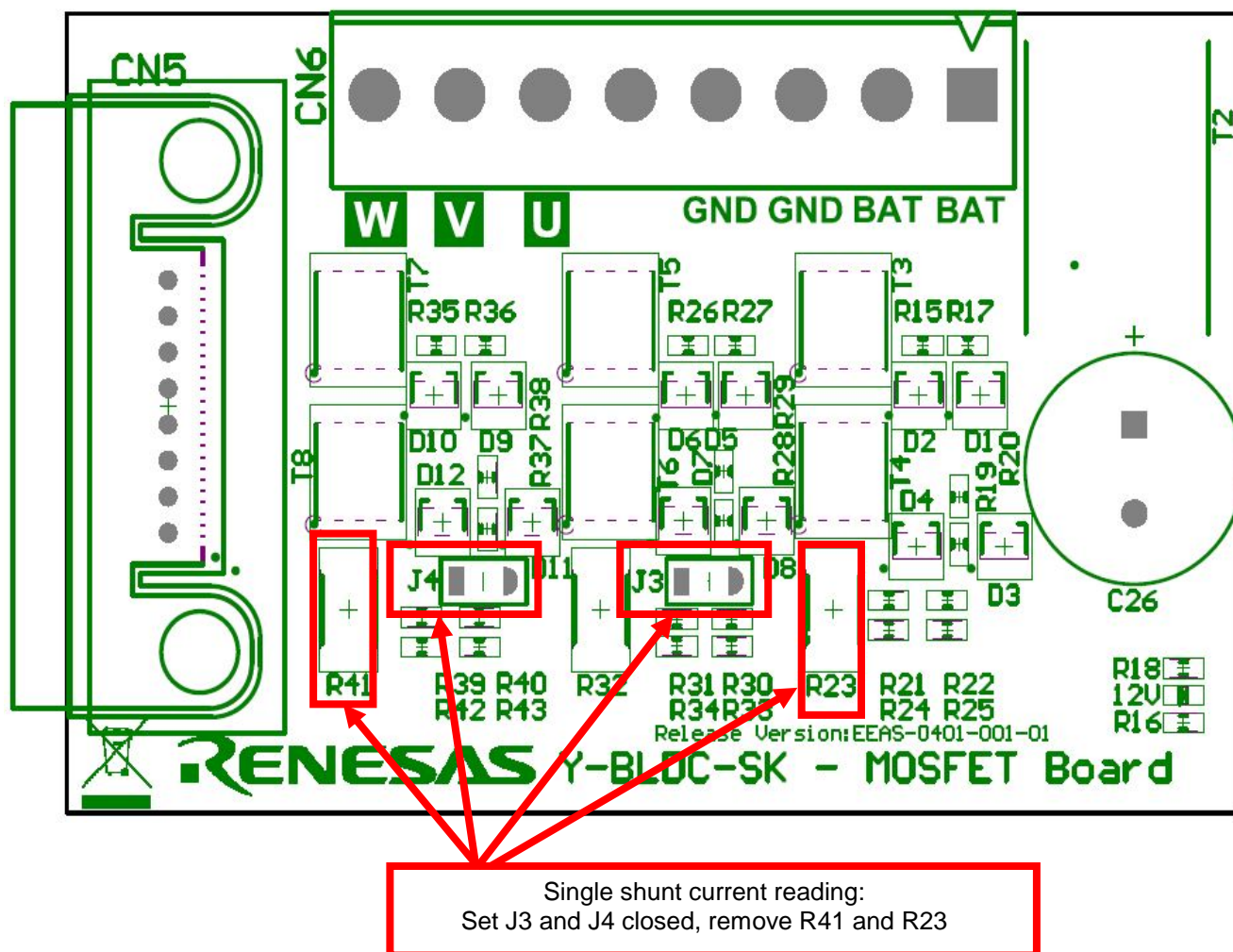
Three shunts configuration (J3 and J4 are open)

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 Y-BLDC-SK-RL78F14 Starter Kit is delivered in the three shunts configuration.

Single shunt configuration (J3 and J4 are closed)

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 projects available on the website: http://www.renesas.eu/update?oc=Y-BLDC-SK-RL78F14 are designed under IAR and CS+ for CC environment for three and single shunts configurations.

The intelligent device **R2A25108KFP** is used to measure in real-time the motor currents flowing through the shunts resistors, where the shunts are called R23, R32, R41.

The default value: **Shunt R_{23} = R_{32} = R_{41} = 10mΩ ; R_{FB} = 15KΩ; R_S = 1KΩ.**

For further information please refer to Appendix A: Schematic.

The amplifier circuit shown below is made to manage current up to 14.7A (17A) flowing through the shunts and the output of the amplifier is connected directly to the Microcontroller RL78/F14.
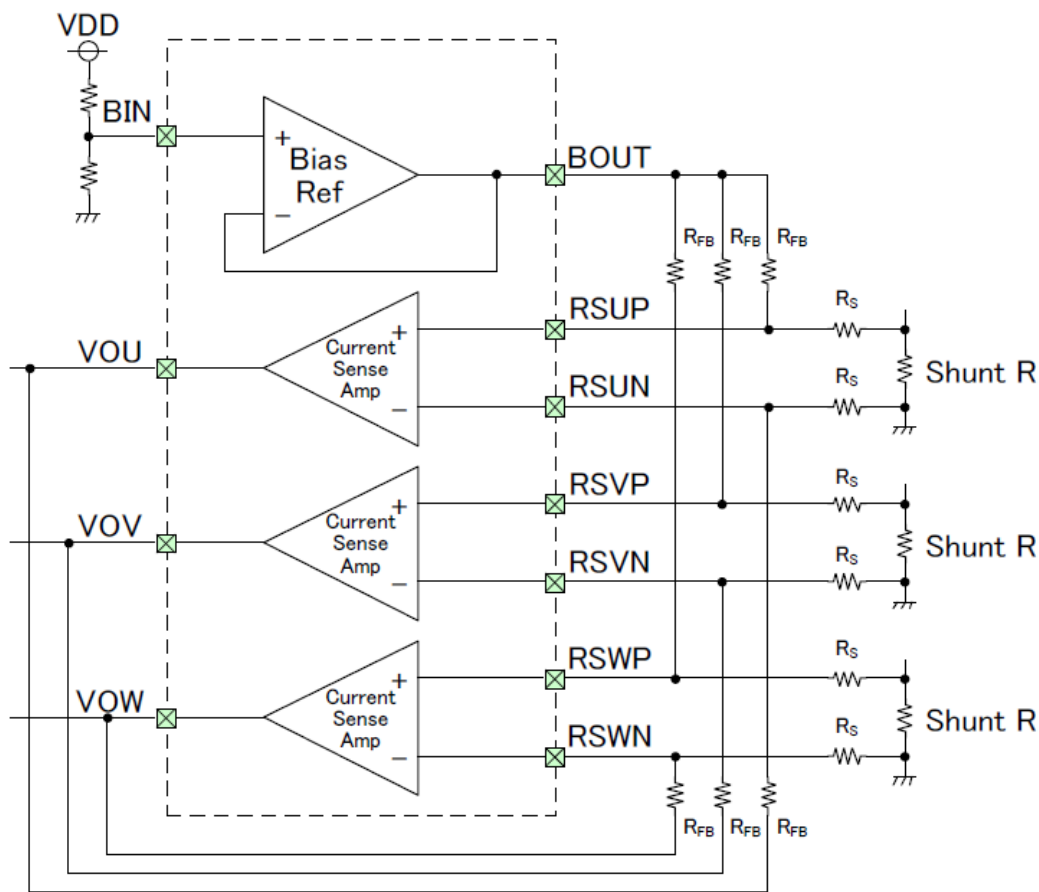
Figure 16. R2A25108KFP for Current Measurement

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 R_{FB}= R_{22} = R_{25} and R_S = R_{21} = R_{24}, then the amplification of the circuit is R_{25}/R_{24} = 15.

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

V_{out} = 2.5V – V_{shunt} x R_{25}/R_{24}, and V_{shunt} = R_{shunt} x I_{shunt}

So

$$V_{out} = 2.5V - (R_{shunt} \text{ x } I_{shunt}) \text{ x } R_{25}/R_{24}$$

And the term

$$Delta\_V = (R_{shunt} \text{ x } I_{shunt}) \text{ x } R_{25}/R_{24}$$

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

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

So the maximum current will be:

$$I_{shunt\ max\ dc} = 2.2V / (R_{shunt} \times (R_{25}/R_{24})$$

The default values are the following:

$R_{shunt} = 0.01\Omega$, $R_{25} = 15K$, $R_{24} = 1K$, so $I_{shunt\ max\ dc}$ = **14.7Adc**

If there exists no margin then $I_{shunt\ max\ dc} = 2.5V / (R_{shunt} \times (R_{25}/R_{24})$ =**17Adc**

To manage different values it is enough to change Rshunt, R25 and R24 **but it is important to keep the following conditions**:

$R_{22} = R_{25}$ and $R_{21} = R_{24}$

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

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

Figure 17. Shunt Resistor Value in Embedded Source Code

9. Microcontroller RL78/F14 short overview

The RL78/F14 is a high-performance 16-bit microcontroller with a maximum operating frequency of 32MHz and delivers up to 52DMIPS. The RL78/F14 includes hardware support for math calculations, multifunction timers for three phase PWM generation, encoder decoding and general support functions (Timers RD, RG and RJ), event link controller (ELC) for reducing software overhead, a data transfer controller (DTC) for automatic data transfer, 10-bit A/D converter and 8-bit D/A converter, LIN/UART module and CAN interface for network communication.

The main specifications of the F14 are as follows:

Item	RL78 Group	
CPU core	RL78 CPU running at 32MHz, delivering 52DMIPs, peripherals running up to 64MHz General registers: 8-bit x 32 Multiply/divide and multiply - accumulate: 16 bits ×16 bits = 32 bits (Unsigned or signed) 32 bits ÷ 32bits = 32 bits (Unsigned) 16 bits × 16 bits + 32 bits = 32 bits (Unsigned or signed)	
Power supply voltage	2.7 to 5.5V	
Flash / RAM memory	Max.128 KB / 8KB	
On-chip peripheral functions	Transfer	Data transfer controller (DTC): max. 52 sources
	Timers	16-bit timer array unit: 8 to 16 channels 16-bit timer RD: 2 channels 16-bit timer RJ: 1 channel Watchdog timer: 1 channel Real-time clock: 1 channel
	Communications	CSI, UART/UART(LIN-bus supported)/SPI LIN module (master/slave supported) I2C/simplified I2C, CAN interface(RS-CAN lite)
	Analog	8/10-bit resolution A/D converter: 4 to 31 channels 8-bit D/A converter: 1 channel On-chip comparator: 1 channel (input pin: 4 channels)
	Safety	WWDT, Illegal instruction execution detection function Flash memory CRC operation function, RAM1 bit error correction function, RAM2 bit error detection function, Invalid memory access detection function, Frequency detection function, Clock monitor function, Stack pointer monitor function, I/O port output signal level detection function, A/D test function
	Clock generation circuit	Main system clock oscillator: 1 to 20 MHz High-speed on-chip oscillator (HOCO): 32 MHZ (typ.)

		Low-speed on-chip oscillator (LOCO): 15KHz (typ.) Subsystem clock oscillator PLL
	Other	Event link controller (ELC) Option byte

Table 2.   Main Specifications Overview of Microcontroller RL78/F14

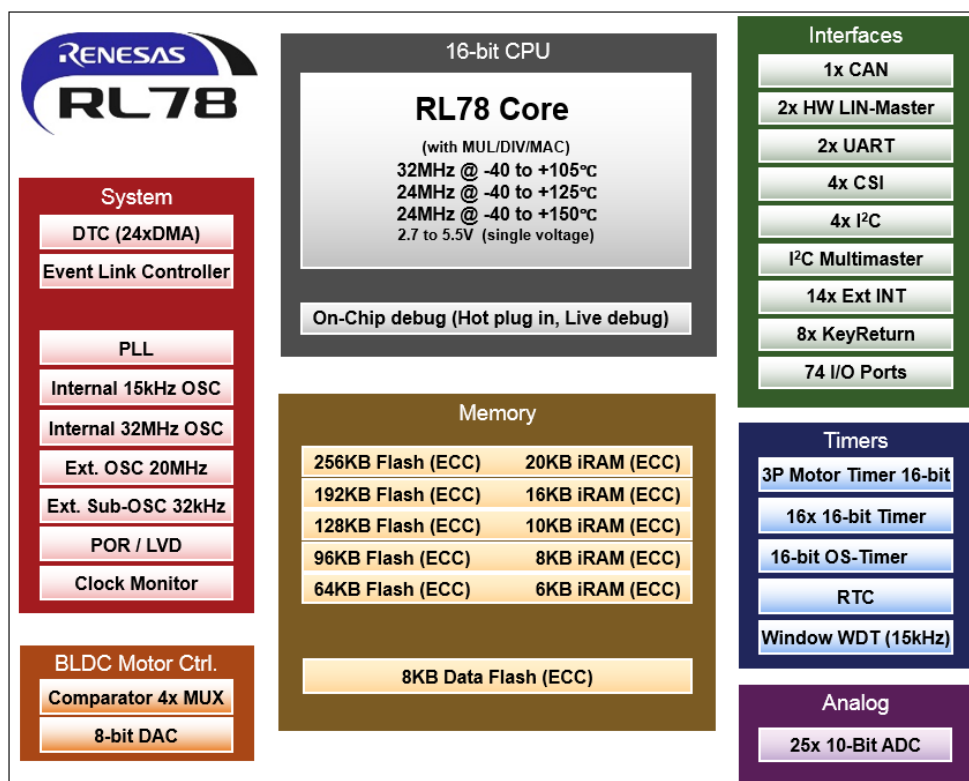Please find below the RL78/F14 package block diagram.

Figure 18. RL78/F14 Package Block Diagram

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

RL78/F14xx - Basic Material Codes

RL78/F14

Flash ROM	Data flash	RAM	20 pins SSOP	30 pins SSOP	32 pins WQFN	48 pins LQFP	48 pins QFN	64 pins LQFP	80 pins LQFP	100 pins LQFP
256	4	20	-	-	-	R5F10PGJxFB	R5F10PGJxNA	R5F10PLJxFB	R5F10PMJxFB	R5F10PPJxFB
196	4	16	-	-	-	R5F10PGHxFB	R5F10PGHxNA	R5F10PLHxFB	R5F10PMHxFB	R5F10PPHxFB
128	4	10	-	-	-	R5F10PGGxFB	R5F10PGGxNA	R5F10PLGxFB	R5F10PMGxFB	R5F10PPGxFB
96	4	8	-	-	-	R5F10PGFxFB	R5F10PGFxNA	R5F10PLFxFB	R5F10PMFxFB	R5F10PPFxFB
64	4	6	-	R5F10PAExSP	R5F10PBExNA	R5F10PGExFB	R5F10PGExNA	R5F10PLExFB	R5F10PMExFB	R5F10PPExFB
48	4	4	-	R5F10PADxSP	R5F10PBDxNA	R5F10PGDxFB	R5F10PGDxNA	-	-	-

x defines temperature range: **L = -40°C to +105°C   K = -40°C to +125°C   Y = -40°C to +150°C**

Figure 19. RL78/F14 memory line-up and temperature range

Note:

To receive a copy of the *Hardware Users' Manual*, please contact your local Renesas sales representative.

10. Renesas intelligent power device R2A25108KFP short overview

The R2A25108KFP device is an Intelligent Power Device to drive the power MOSFET inverter of the three phase brushless motor.
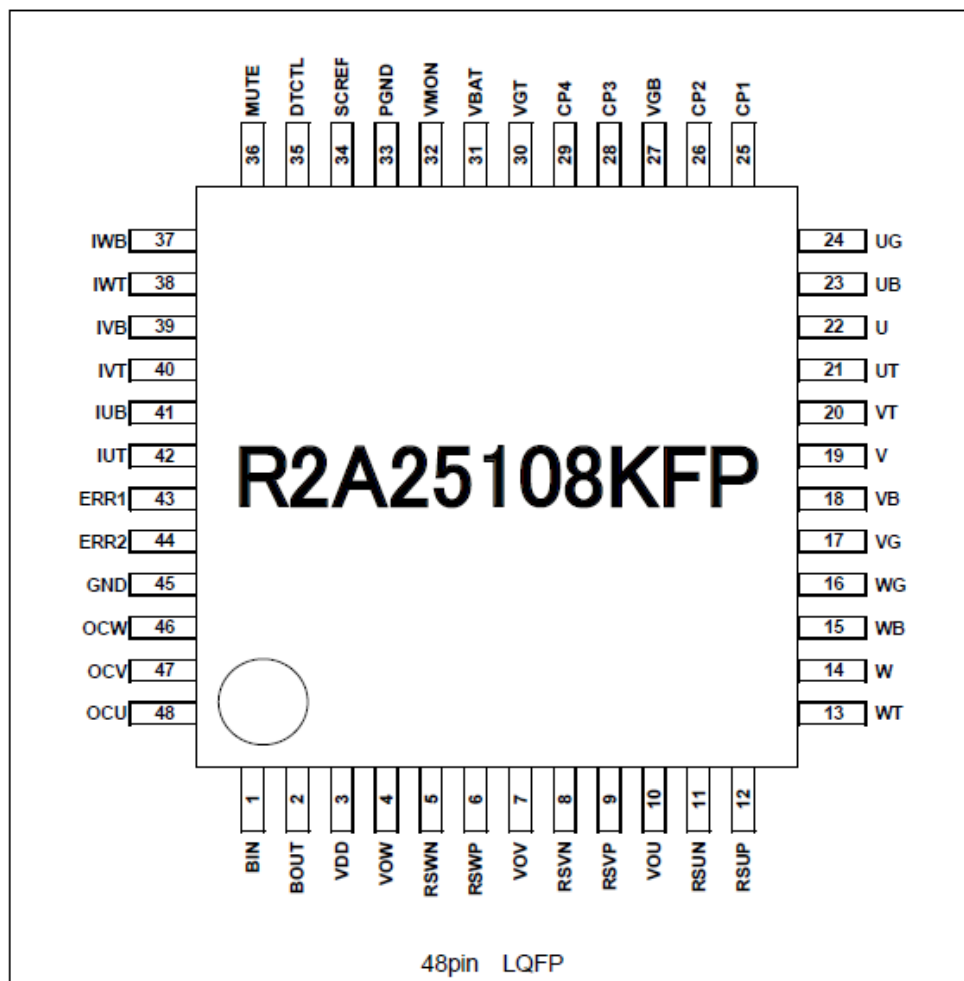
Figure 20. Top View of R2A25108KFP

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

The main features of the R2A25108KFP are as follows:

- Wide range operating voltage: 5.3V to 18V (VBAT)

- On-chip three phase pre-driver circuit

  o PWM control

- o Totem pole type MOSFET gate drive circuit, high drive capability: Ciss = 10000pF

- o Dead time control (adjustable)

- On-chip charge pump circuit (for power supply of gate drive)

- On-chip zero-crossing detection circuit

- On-chip current sense amplifier with reference bias buffer

- On-chip safety functions

  - o Low voltage detection circuit (LVD)

  - o Over voltage detection circuit (OVD)

  - o Thermal shut down circuit (TSD)

  - o Short circuit protection; adjustable detection level

    - Function for the short to battery protection

    - Function for the short to GND protection

- Internal oscillation circuit; 175 KHz typ.

- 48 pin LQFP package

- Comply with AEC-Q100

For further details please refer to the data-sheet available on the Renesas website:
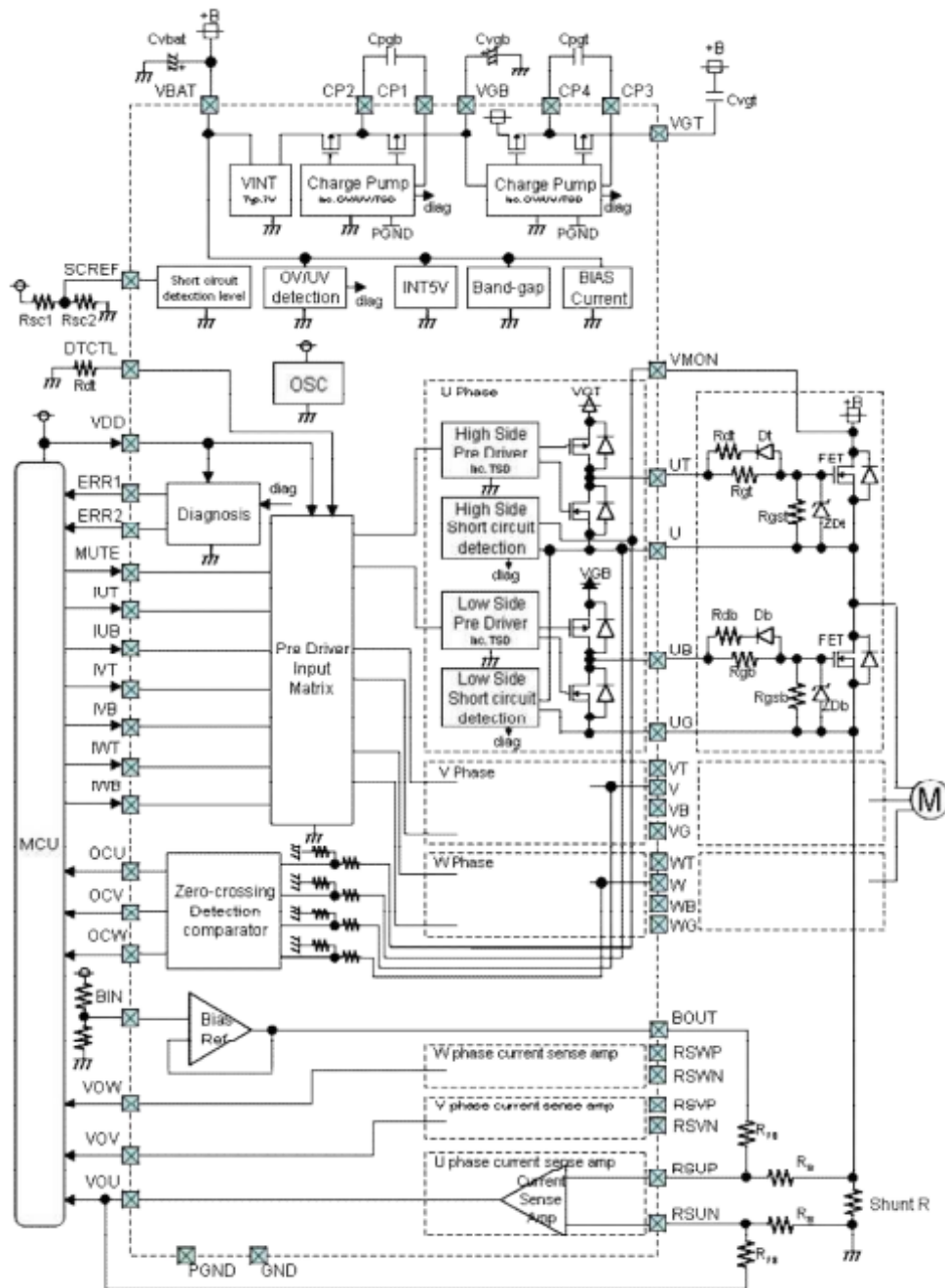www.renesas.eu.

An application example is shown below.

Figure 21. R2A25108KFP Application Example

11. 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:

- A **stator** formed by stacking sheared metal plates where internally the copper wiring is wound, constructing the stator winding

- A **rotor** in which permanent magnets are fixed

- Two covers with ball bearings that keep together the stator and the rotor; the rotor is free to rotate inside the stator
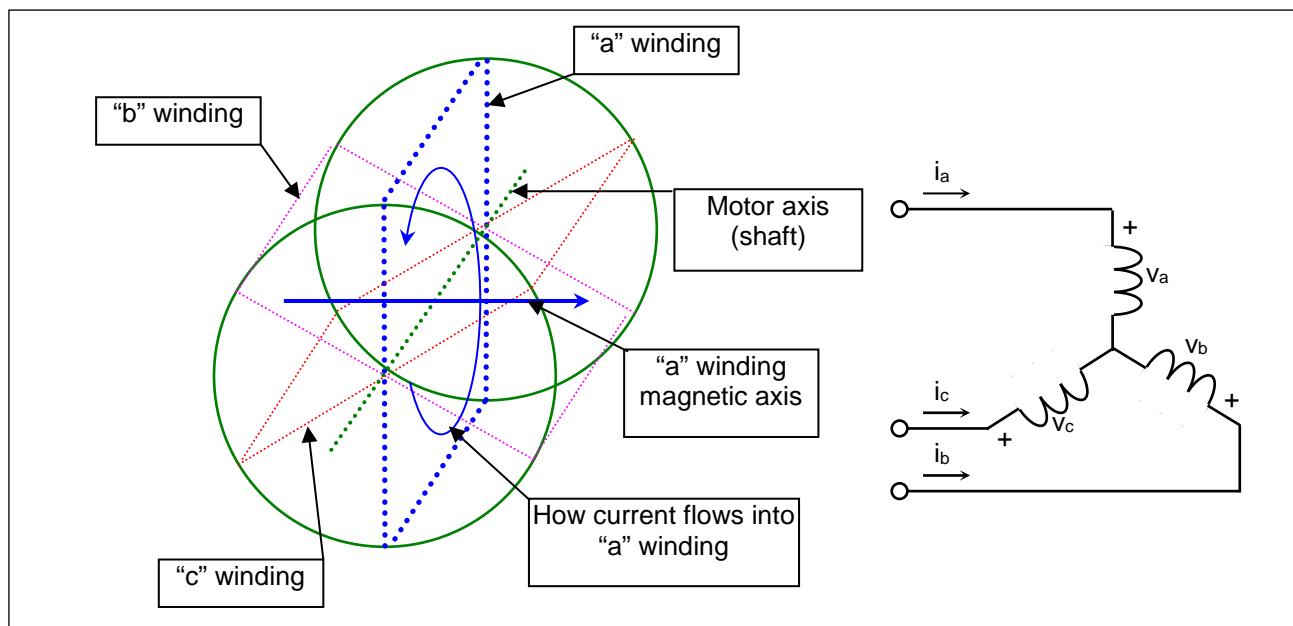
Figure 22. Stator Windings of BLDC Model

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

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

The main challenge in driving this type of motor is to know the rotor position in real-time, so 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 analyze the motor from a mathematic point of view.

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

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

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

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

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

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

The magnetic flux linkages λ_i are composed by two items, one due to the stator currents, one to the permanent magnets.
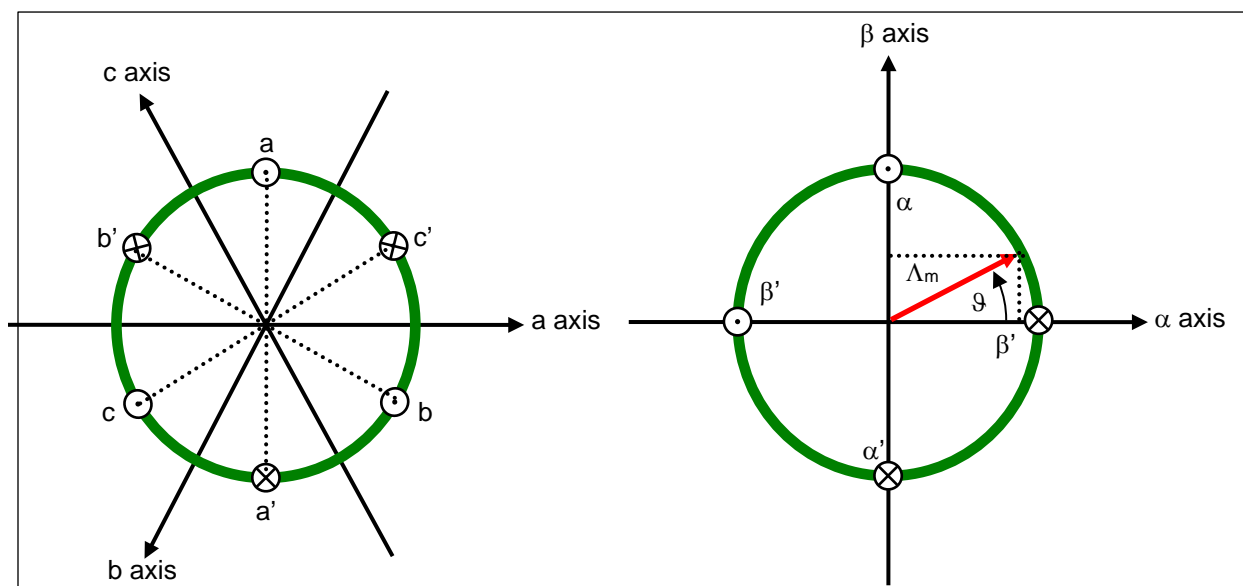
Real axes (a, b, c) and equivalent ones (α, β); a fixed amplitude vector can be completely determined by its position respect the (α, β) system (angle ϑ)

Figure 23. Reference System Transformation from abc to dq axis

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

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

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

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

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

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

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

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

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

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

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

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

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

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

For the magnetic field equations, we got:

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

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

After performing the derivation:

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

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

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

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

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

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

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

We can see

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

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

and

$$\lambda_{dm} = L i_d + \Lambda_m$$

$$\lambda_{qm} = L i_q$$

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

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

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

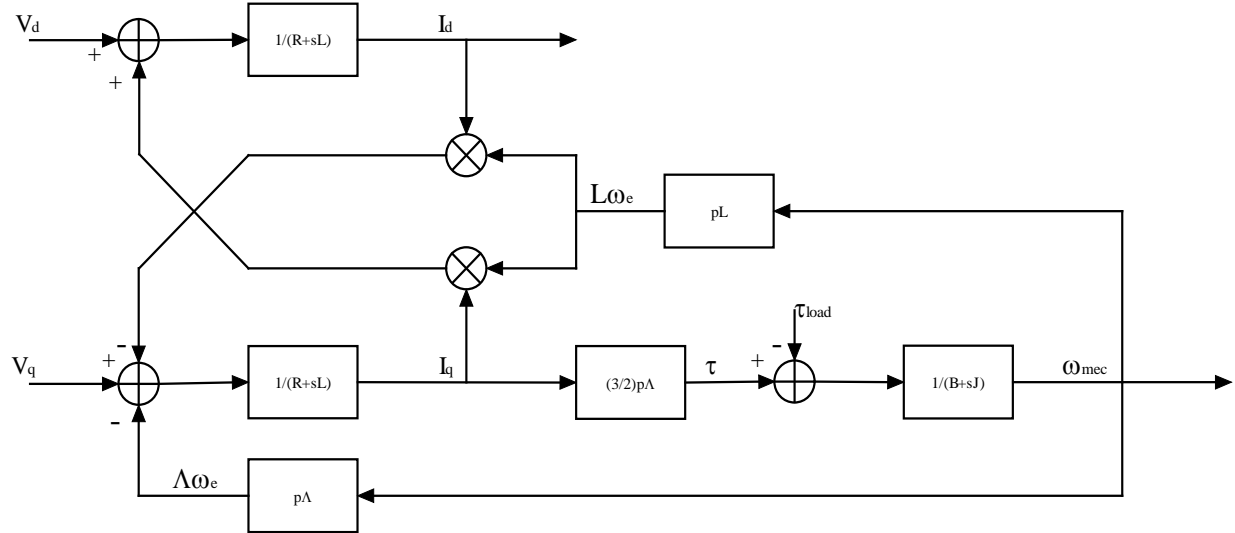These two equations represent the mathematical motor model as shown in the following figure.

Figure 24. *the Mathematical Motor Model*

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

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

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

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

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

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

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

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

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

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

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

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

Furthermore:

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

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

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

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

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

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

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

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

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

12. Sensor less Field Oriented Control algorithm

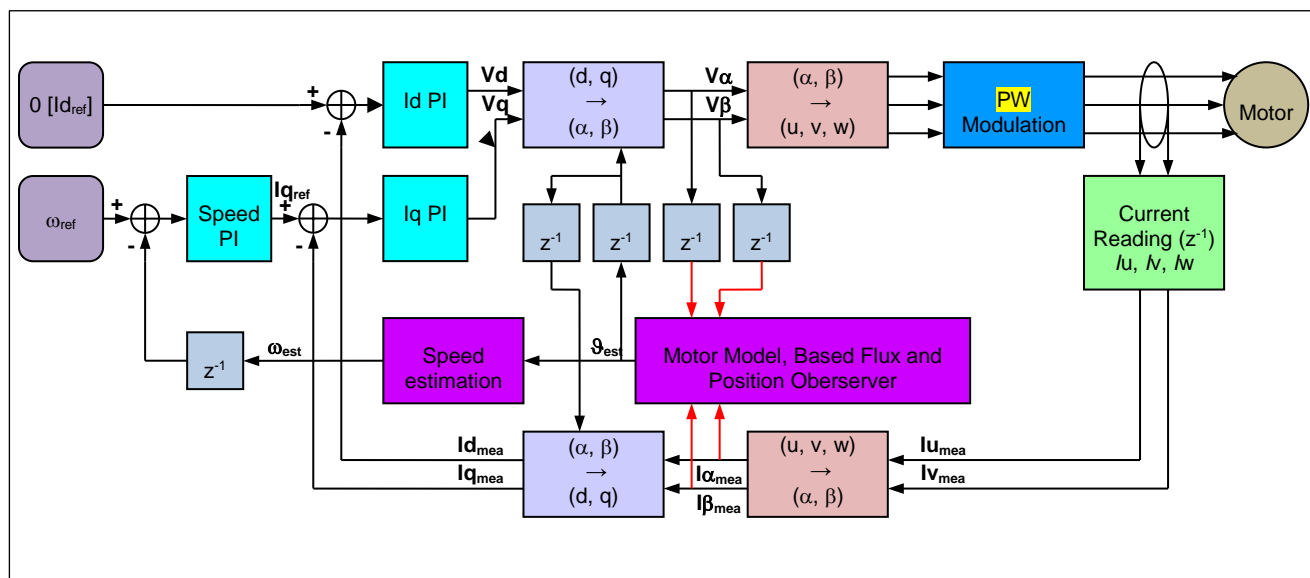Please, find below the sensor less vector control algorithm block diagram.

Figure 25. FOC 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.

13. Software Tools used

The embedded software delivered in the Y-BLDC-SK-RL78F14 Starter Kit is developed under IAR integrated development tool and CS+ for CC IDE. Please find below the details of the IDE used.

13.1 IAR Embedded Workbench IDE

13.1.1 IAR Embedded Workbench Usage

The software was designed under IAR Embedded Workbench for Renesas RL78 Version: **1.40.6**.
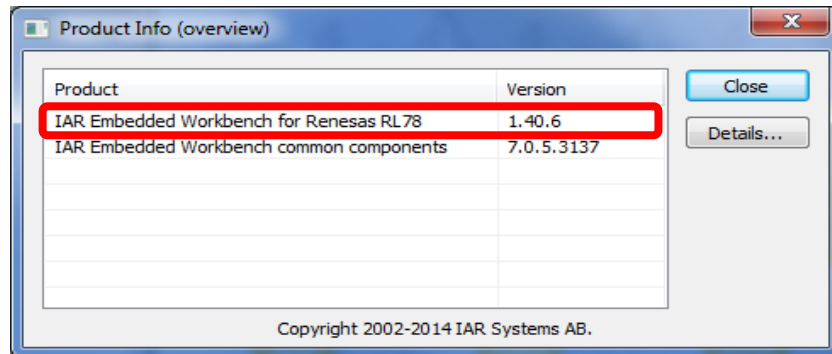

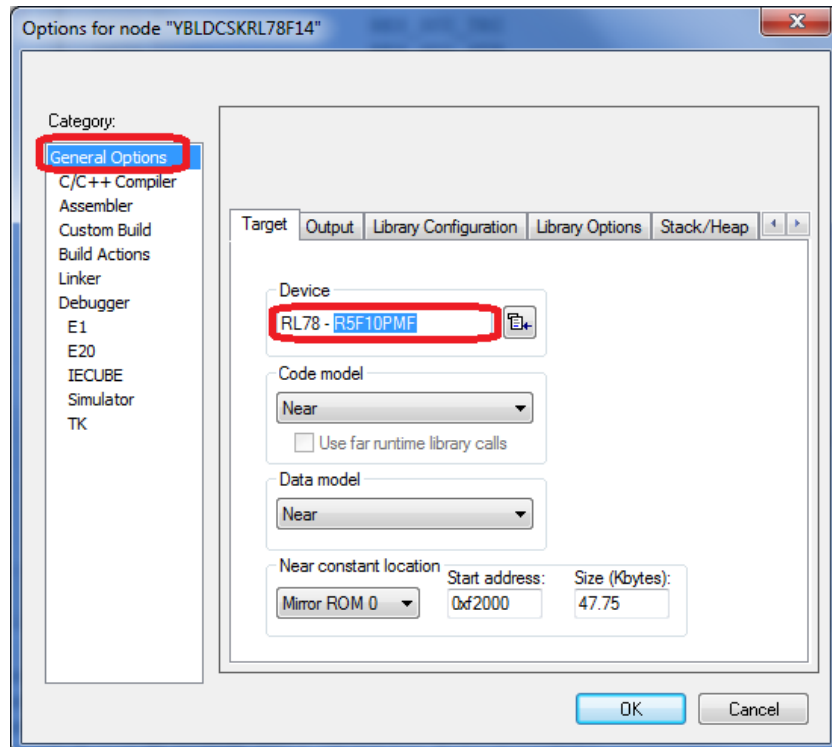*Figure 26. Product Version of IAR Embedded Workspace for Renesas RL78*

Figure 27. Product Version of IAR Embedded Workspace for Renesas RL78

The device selected in the program is: RL78/F14-R5F10PMJ, it means the RL78/F14 with 256KB flash, 20KB RAM in an 80-pin package.

13.1.2 Project importation into IAR Embedded Workbench

The IAR Embedded Workbench will have been installed in the default or user location.

The default location is as follows:

Start Menu =>

All programs =>

IAR Systems =>

IAR Embedded Workbench for Renesas RL78 1.40 =>

IAR Embedded Workbench.exe

Click on the "IAR Embedded Workbench.exe" to open, (Note that Windows Vista and 7 users may have to use "Run as administrator") and the opening screen should open as below.
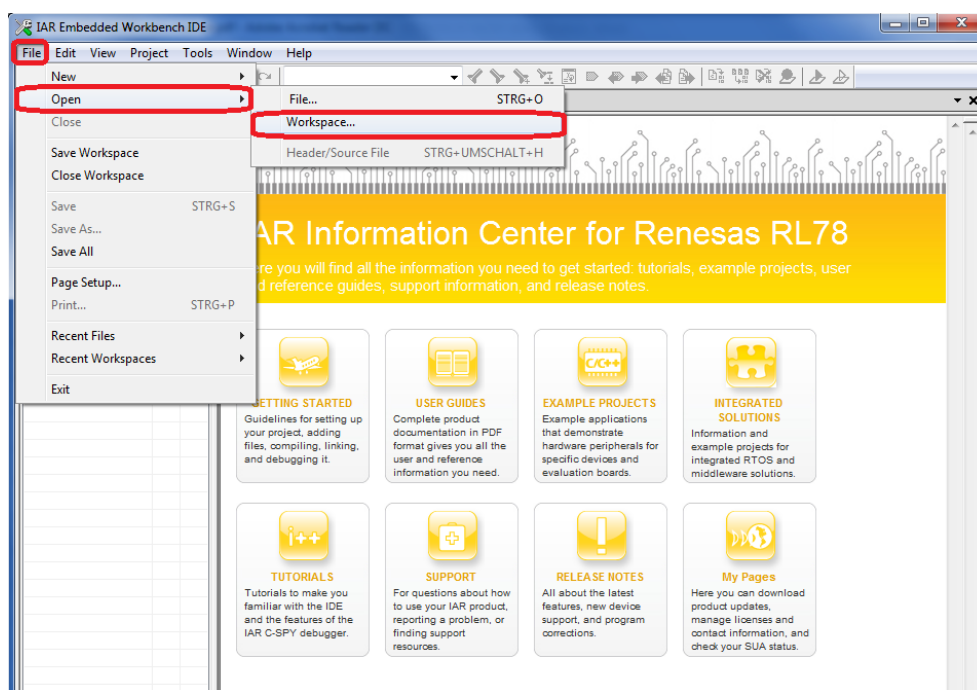

Figure 28.   *Opening Screen of IAR Embedded Workspace for RL78*

To open the YBLDCSKRL78F14 IAR motor control workspace and project follow the sequence shown below (see also figure above):

File =>

Open =>

Workspace =>

Sample Application destination folder\YBLDCSKRL78F14\IAR\Three Shunt =>

Select the "YBLDCSKRL78F14.eww" IAR Workspace file => Press Open
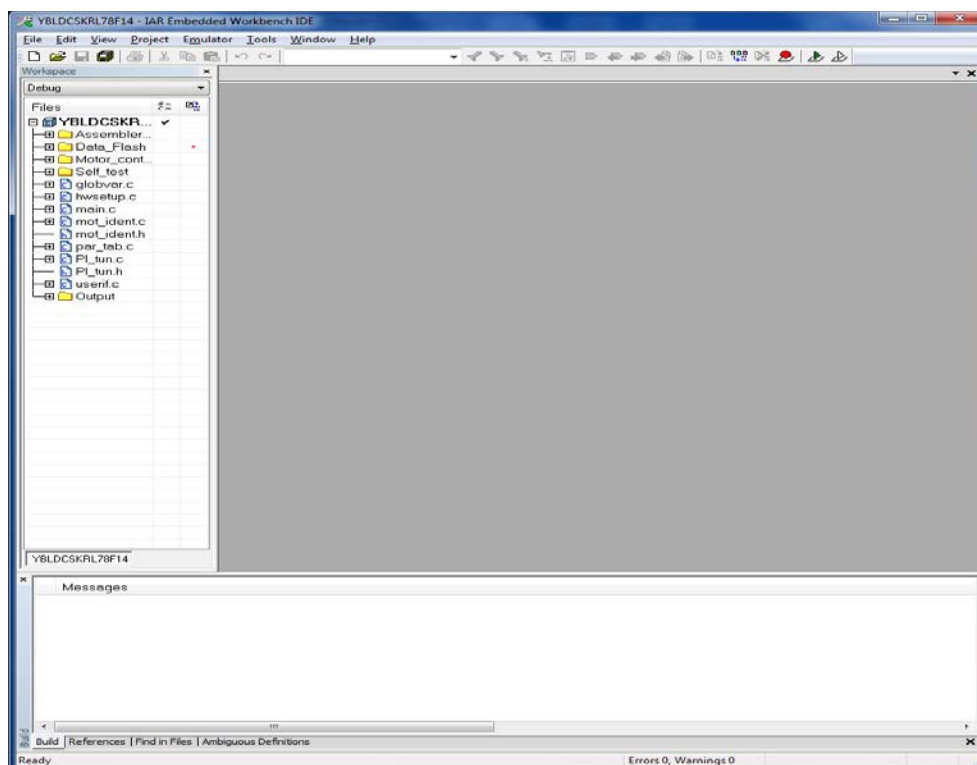
Figure 29.   Opening Screen of "YBLDCSKRL78F14.eww" IAR Workspace

The project should then open in the IAR IDE and should look something like the window above. Here you can see two projects, YBLDCSKRL78F14_32MHz and YBLDCSKRL78F14_24MHz, which means clock source running at 24MHz and 32MHz are both offered. **We take the 32MHz one as a sample to introduce.**

Please note that depending on settings used previously then the IAR workspace and project windows can look slightly different. All the settings have been pre-set so that the workspace appearance is as constant as possible. For full details of IAR Embedded Workbench, please refer to the documentation included as part of the IAR installation.

To open any of the source files listed in the project (on the Left Hand Side of the project window), just double click on the relevant file.

Next the debugging interface E1 should be selected following the 5 steps as shown in the figure below.

Right click on the project name "YBLDCSKRL78F14_32MHz" and select "Options", click "Debugger" and select "E1" option in the driver drop down menu, then press "OK" button.
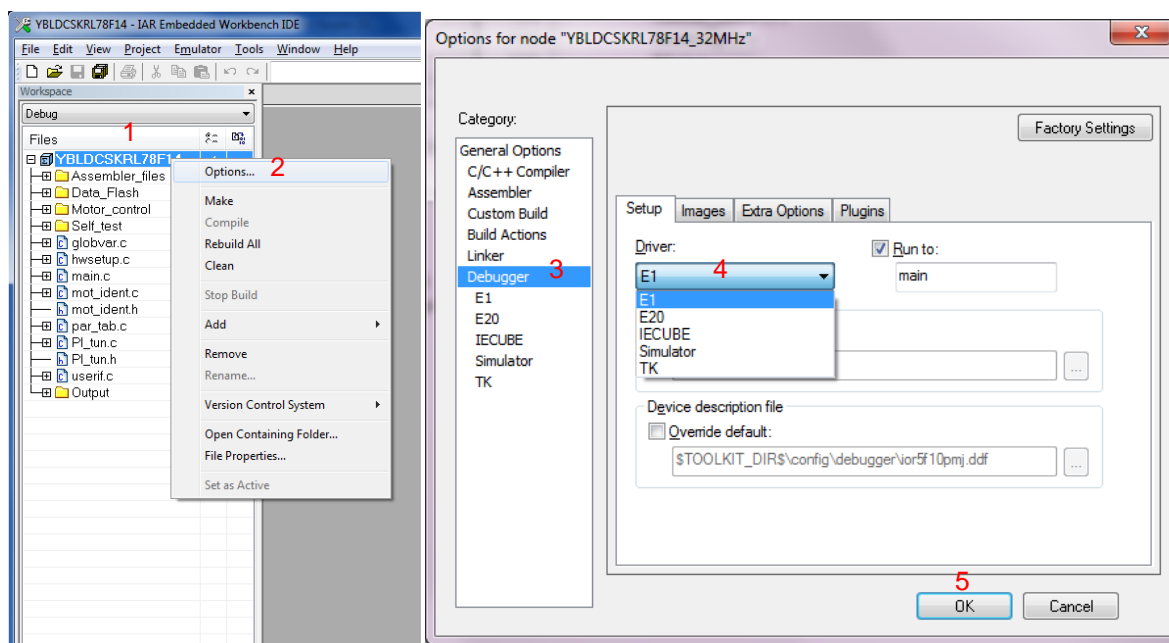
Figure 30. Select Debugging Interface E1

The next step is to build the project.

The necessary settings have been set in the IDE so that it is not necessary to configure or make changes to any of the build options. These can obviously be viewed for reference, just select the "Option" menu as described above and click on any of the relevant button.

Note:

It is recommended that no changes are made to any of the build settings as the resulting build results could not be guaranteed.

The project can be built from the build ICON in the workspace or from the "Rebuild All" option in the "Project" drop down menu.

 The project should be built without errors as shown in the following figure.
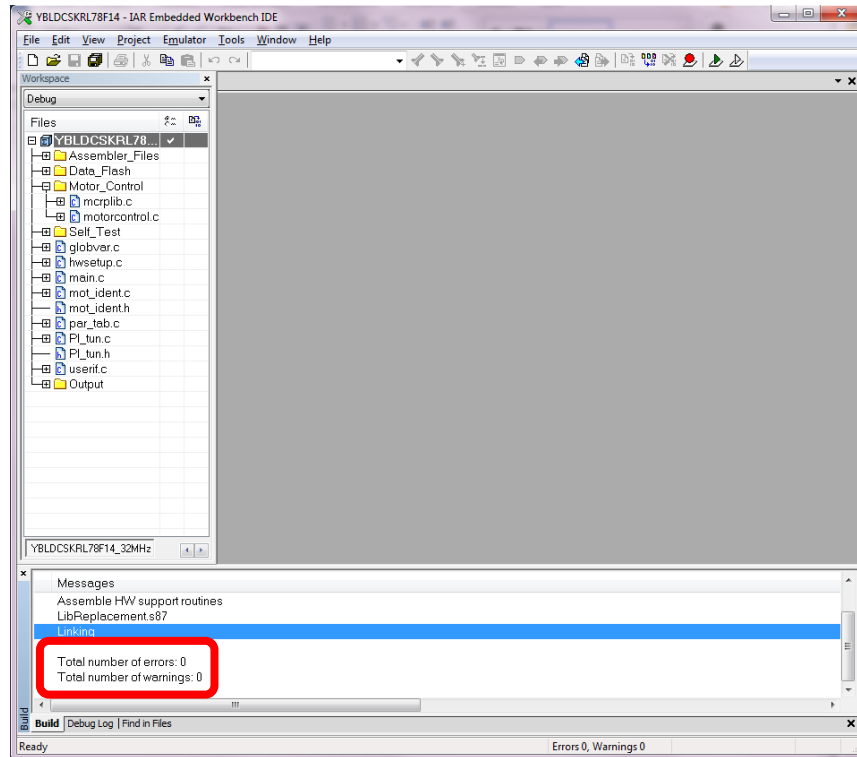
Figure 31. Project Build without Errors

Launch the IAR debugger by clicking on the green button on the embedded workbench IDE window.

Figure 32. Project Download and Debug Button

If the debugger is being used for the first time the emulator needs to be configured before connecting and downloading the code to the board and RL78/F14. This applies to any hardware emulation setting and (i.e. E1, TK and IECUBE).

The user will be prompted by the following pop up window.
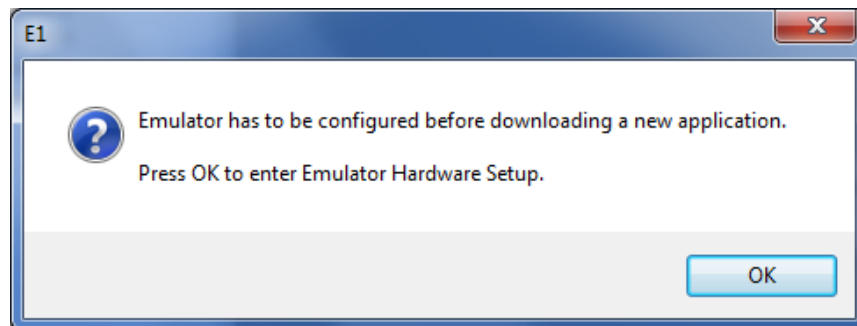
Figure 33. Hardware Emulator Setting Popup Window

The configuration window will open as shown below. Please check that the settings are as shown in the window below, changing any setting as necessary and then press the "OK" button.
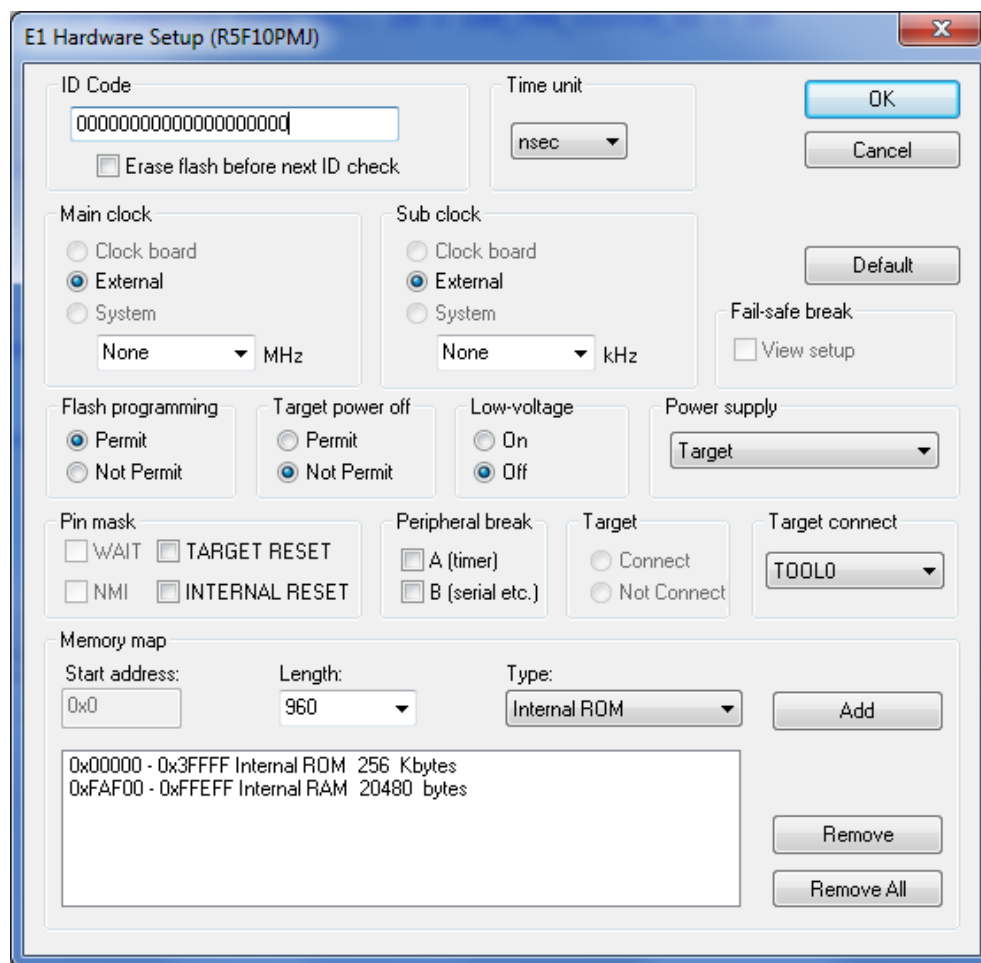
Figure 34. Hardware Emulator Setup Window

After connection the E1 debugger to the target board powered by USB or external power supply, please click on "Download and Debug" to start the debugging and load the program into the microcontroller flash memory.

The IAR debugger will open a progress window which will initially connect to the board and then program the RL78/F14. Once this stage is complete the debugger window will open as shown below.
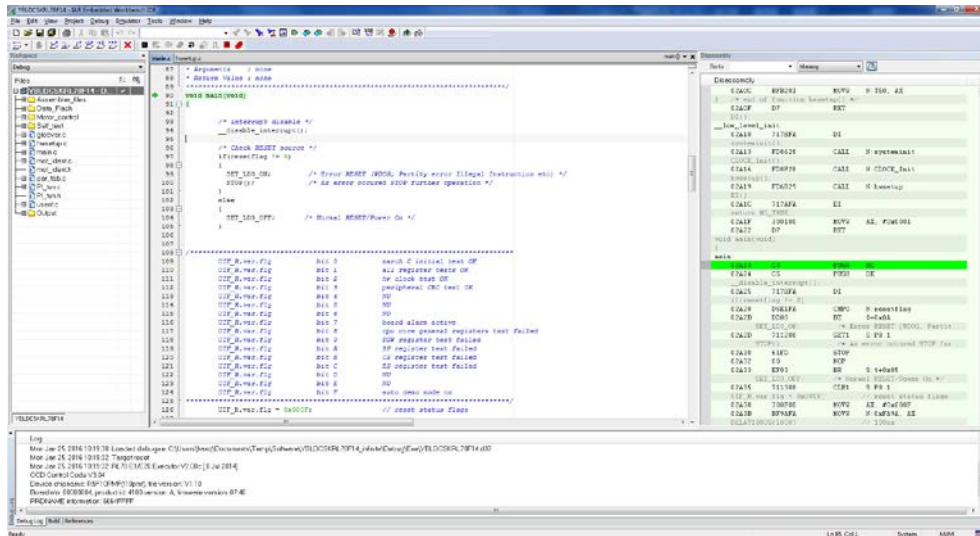
Figure 35. Debugging Window of Project

Other debugging windows can be opened to "watch variables, monitor registers, view the stack, memory etc." These can be selected by using the "View" menu tab at the top of the workbench and then selecting the required debugging function. Please note that there are some other debugging function such as "Events" that are located under "Emulator" tab.

Data is held for all debugging options whether displayed or not, so that windows can be opened or closed as required to make the management of the workspace and the data viewed clearer.

Software breakpoints can be set in the C source or assembler window by simply double clicking on the source code line or the line in the appropriate window. Or by right clicking the mouse button are also available. Click the "Emulator" tab then select "Breakpoint Toggle during Run", this can help setting breakpoint during debugging running, see figure below.
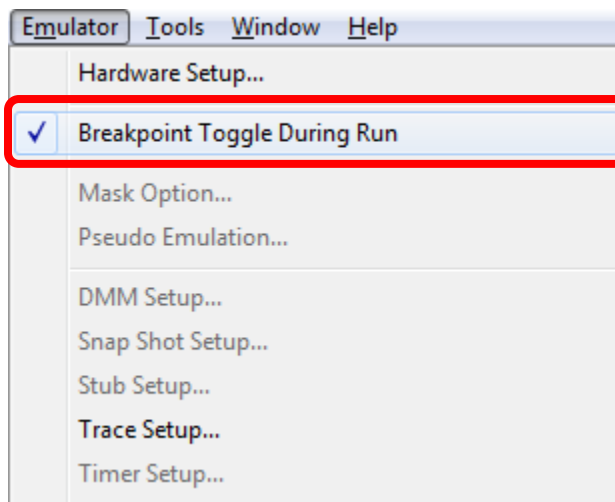
Figure 36. Enable Breakpoints Toggle during Run

The main debugging control functions are shown below.
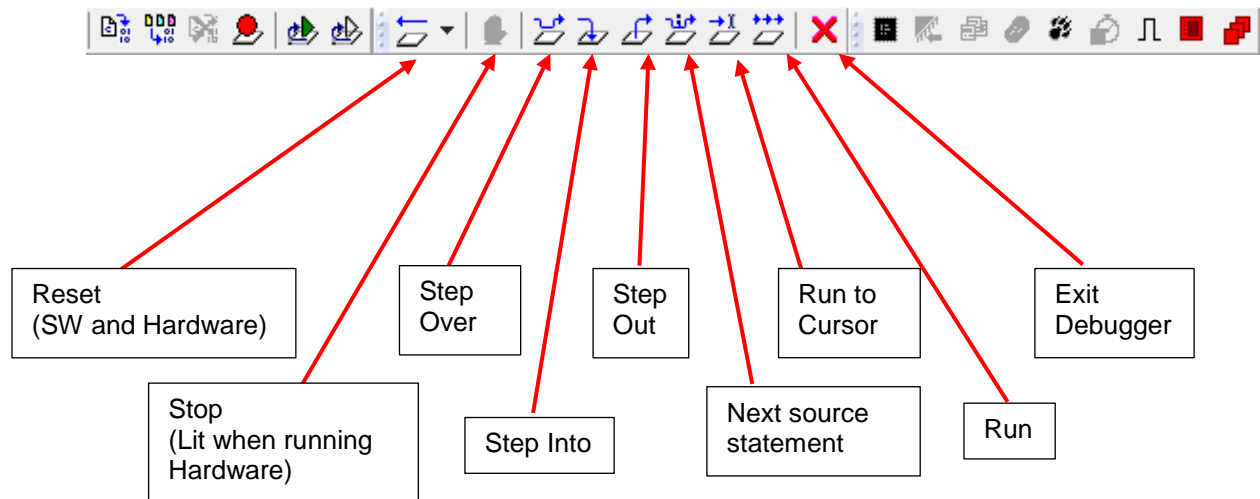
Figure 37. Buttons for Debugging Project under IAR

For a full explanation of all debugging options, please use the full documentation included in the IAR installation. These can be accessed via the "Help" tab in the embedded workbench IDE.

The program is loaded after clicking the "Run" button. The E1 debugger can be removed to run the software on its own. Necessary Parameters can be tuned in the GUI.

13.2 CS+ for CC

An additional software project version for the Y-BLDC-SK-RL78F14 Starter Kit is available under CS+ for CC IDE. Please find below the details of the IDE used.

13.2.1 CS+ for CC Usage

The software was designed under CS+ for CC Version: **V4.00.00**.
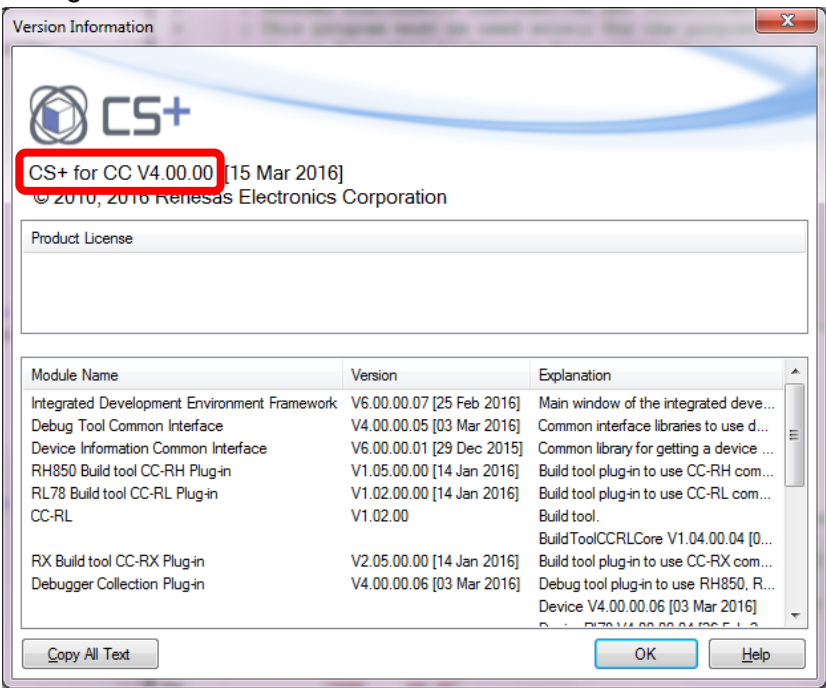
Figure 38. Product Version of CS+ for CC

The device selected in the program is R5F10PMJ. The compiler used is "Renesas CC-RL".

The debug tool adopted is RL78 E1.
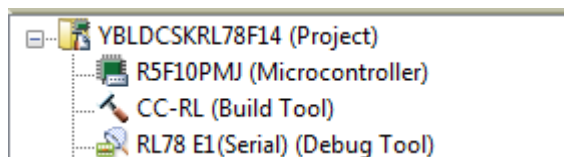
Figure 39. *Device, Compiler and Debugger selected*

13.2.2 Project importation into CS+ for CC

The CS+ for CC will have been installed in the default or user location.

The default location is as follows:

Start Menu =>

All programs =>

Renesas Electronics CS+=>

CS+ for CC (RL78, RX, RH850).exe

Click on the file "CS+ for CC (RL78, RX, RH850).exe" to open. (Note that Windows Vista and 7 users may have to use "Run as administrator") and the opening screen should open as below.
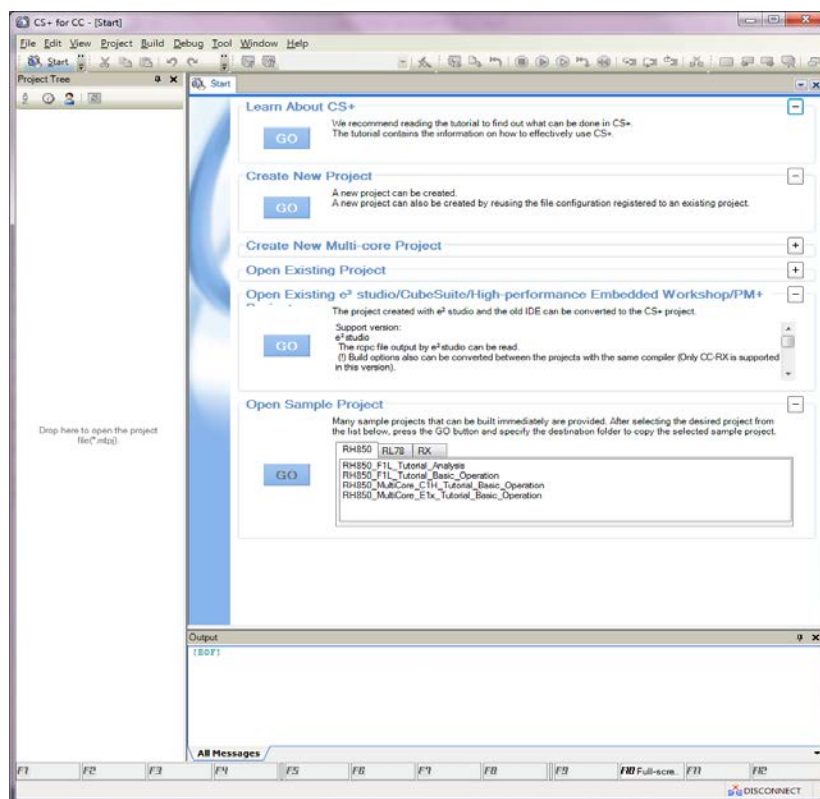
Figure 40. *Opening Screen of CS+ for CC*

To open the YBLDCSKRL78F14 CS+ for CC motor control project follow the sequence shown below:

File =>

  Open =>

      Sample Application destination folder\YBLDCSKRL78F14\CS+\Three Shunt =>

        Select the file "YBLDCSKRL78F14.mtpj" =>

          Press Open
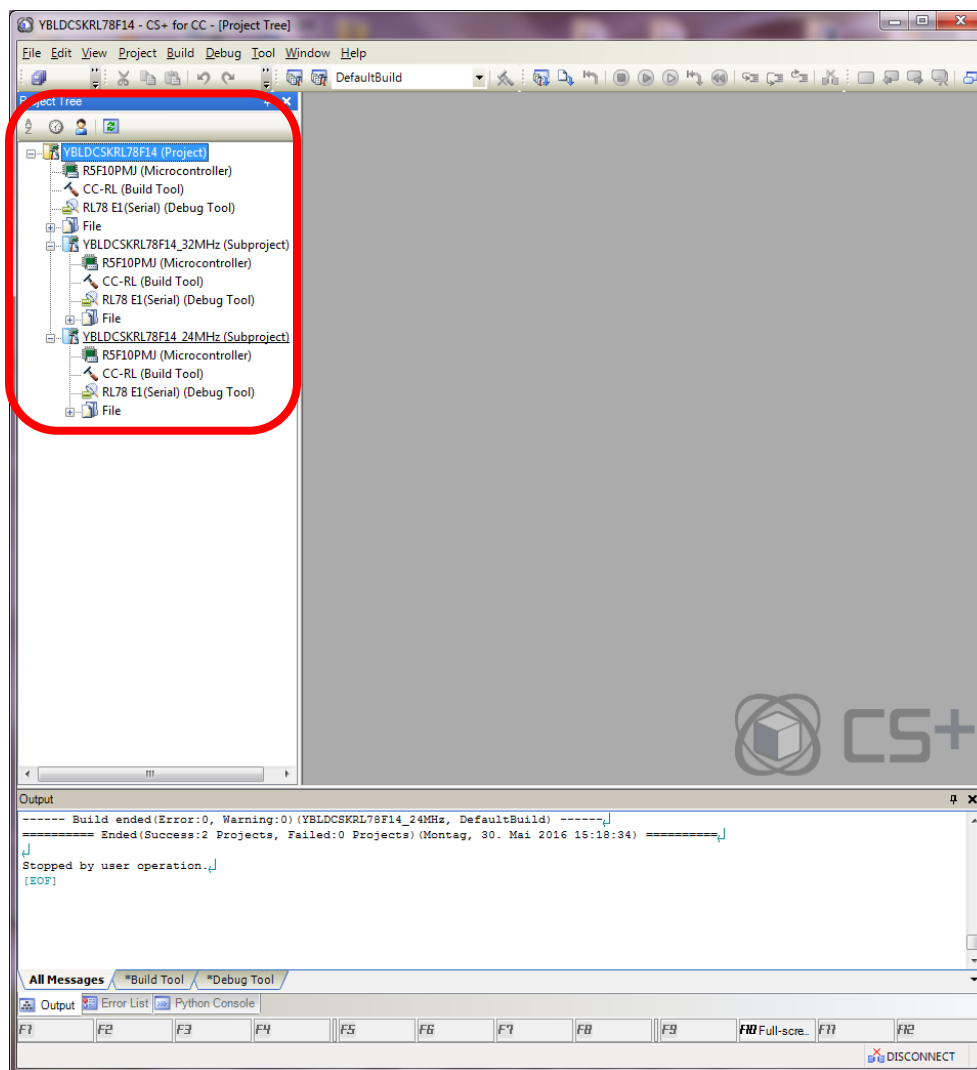
Figure 41. *Opening Screen of YBLDCSKRL78F14 under CS+ for CC*

The project should then open in the CS+ for CC IDE and should look something like the window above. Here you can see project YBLDCSKRL78F14 and two subprojects YBLDCSKRL78F14_32MHz and YBLDCSKRL78F14_24MHz, which means clock source running at 32MHz and 24MHz are both offered. **We take the 32MHz one as a sample to introduce by right click on the project name "YBLDCSKRL78F14_32MHz" and click on the file "Set**

YBLDCSKRL78F14_32MHz as Active Project" if necessary as shown in the following figure.
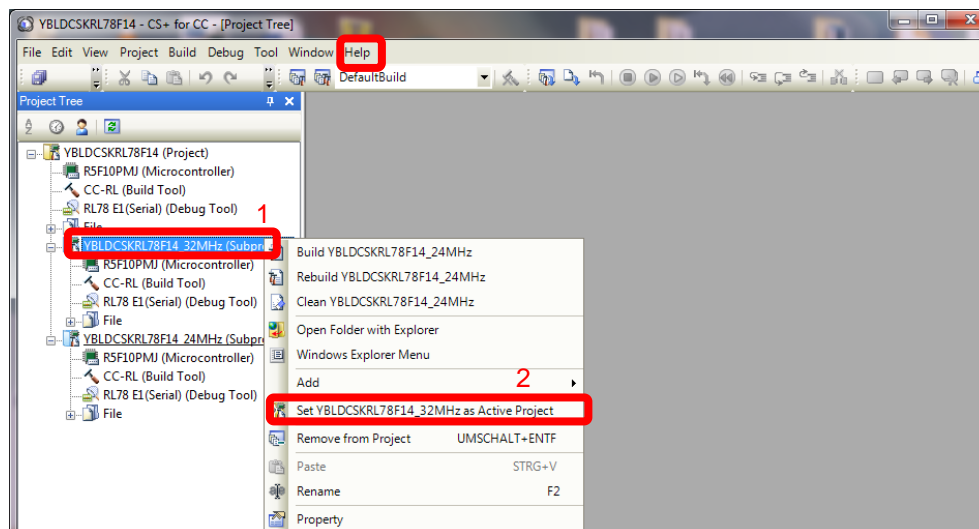
Figure 42. Set YBLDCSKRL78F14_32MHz as Active Project

Please note that depending on settings used previously then the CS+ for CC project windows can look slightly different. All the settings have been pre-set so that the IDE appearance is as constant as possible. For full details of CS+ for CC, please refer to the documentation included as part of the CS+ for CC installation or click on the file "Help" in the CS+ for CC IDE as shown in the figure above.

To open any of the source files listed in the project (on the left hand side of the project window), just double click on the relevant file.

The next step is to build the project.

The necessary settings have been set in the IDE so that it is not necessary to configure or make changes to any of the build options. These can obviously be viewed for reference, just right click on the "CC-RL" menu and select the "Property" option then click on any of the relevant button as shown in the following 3 steps in the figure below.

Note:

It is recommended that no changes are made to any of the build settings as the resulting build results could not be guaranteed.
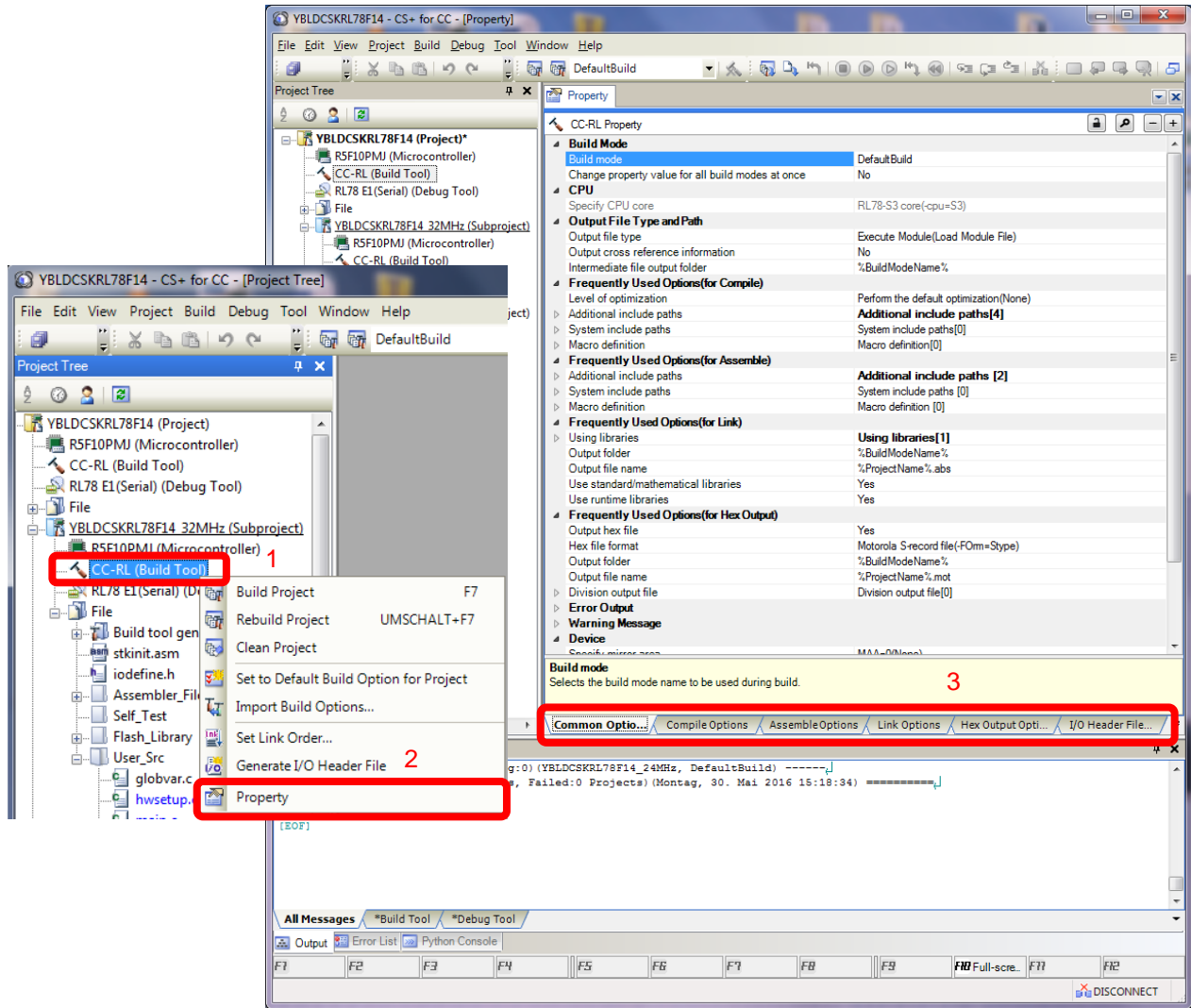
Figure 43. Build Settings in CS+ for CC

The project can be built from the build ICON or the rebuild ICON .

The project should be built without errors as shown in the following figure.
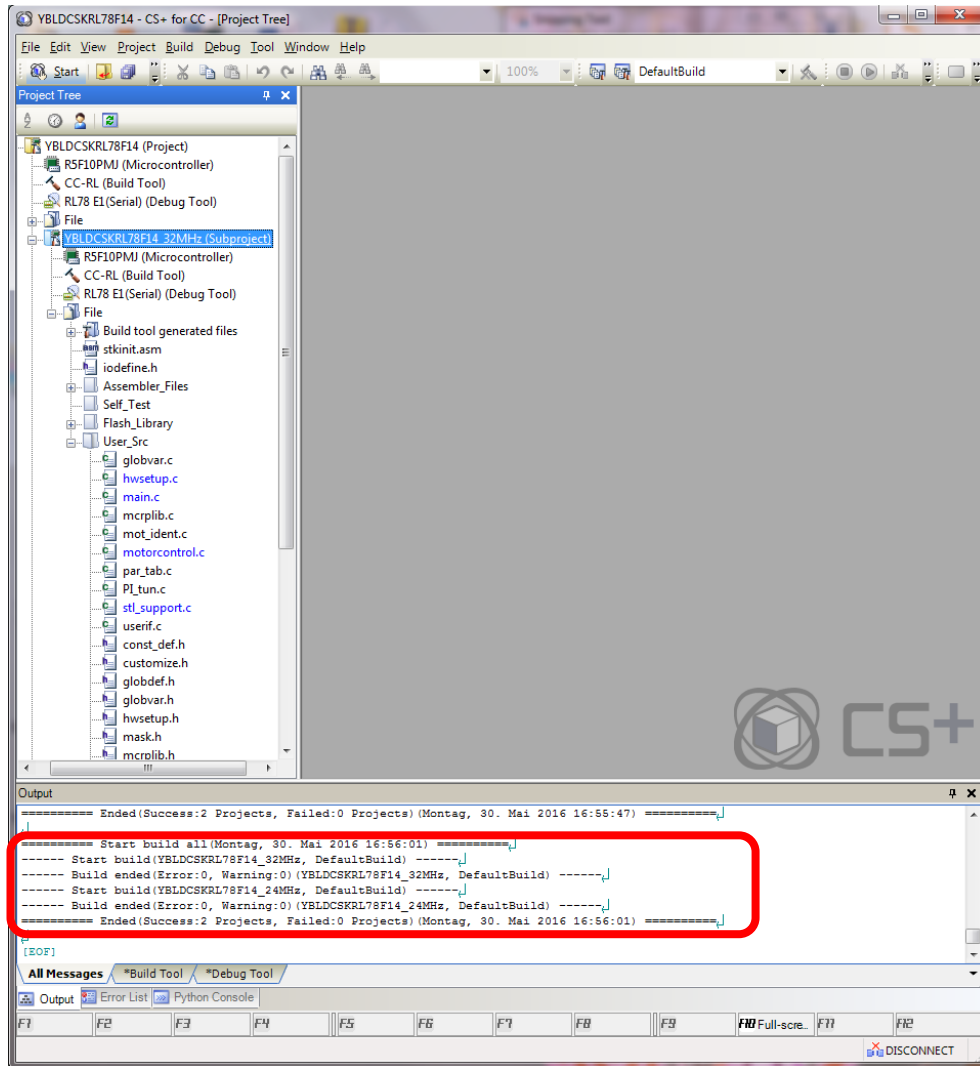
Figure 44. Project Built without Errors

Launch the CS+ for CC debugger by clicking on the white button .

The CS+ for CC debugger will open a progress window which will initially connect to the board and then program the RL78/F14. Once this stage is complete the debugger window will open as shown below.
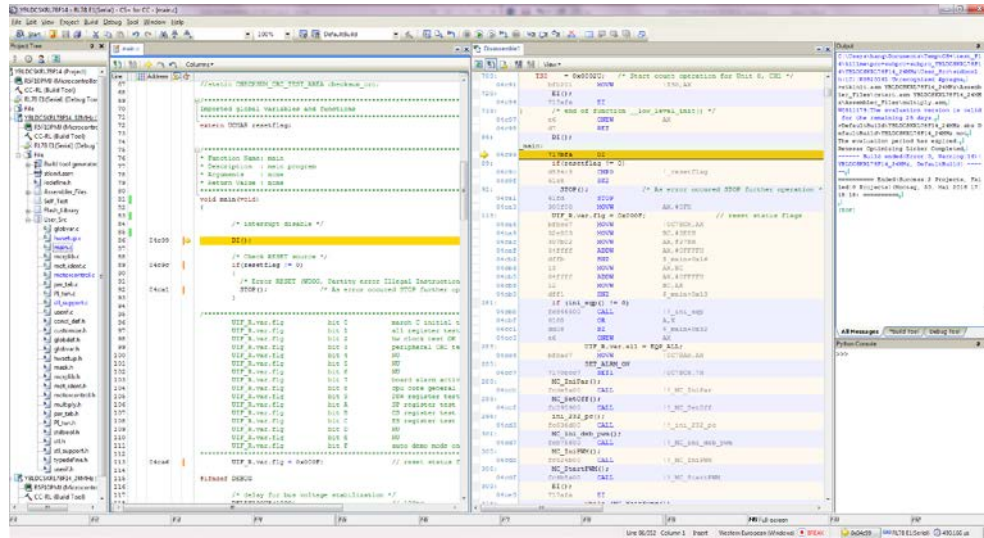
Figure 45. *Debugging Window of Project*

Other debugging windows can be opened to "watch variables, CPU registers, trace, memory etc." These can be selected by using the "View" menu tab at the top of the IDE and then selecting the required debugging function.

Data is held for all debugging options whether displayed or not, so that windows can be opened or closed as required to make the management of the IDE and the data viewed clearer.

The main debugging control functions are shown below.
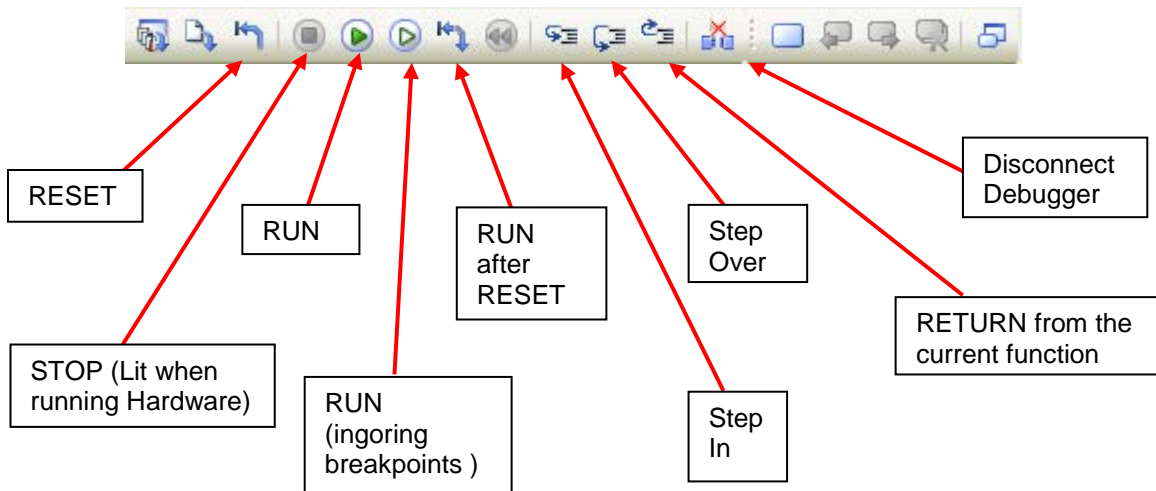
Figure 46. *Buttons for Debugging Project under CS+ for CC*

For a full explanation of all debugging options, please use the full documentation included in the CS+ for CC installation. These can be accessed via the "Help" tab in the CS+ for CC IDE.

The program is loaded after clicking the Run button. The E1 debugger can be removed to run the software on its own. Necessary Parameters can be tuned in the GUI.

14. Software description and Resources used

The software delivered in the Y-BLDC-SK-RL78F14 Starter Kit, previously described, is working on the RL78/F14 microcontroller clocked at 32MHz and its operating voltage is 5V which guarantee a high noise immunity.

Note: Clock source running at 24MHz and 32MHz are both offered.

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 **24 KHz** and the control loop is set to **8 KHz**, so the ratio is 3 which means that the full vector control algorithm is processed every three PWM cycles.

Please find below detailed information related to the software blocks of the motor control embedded software:
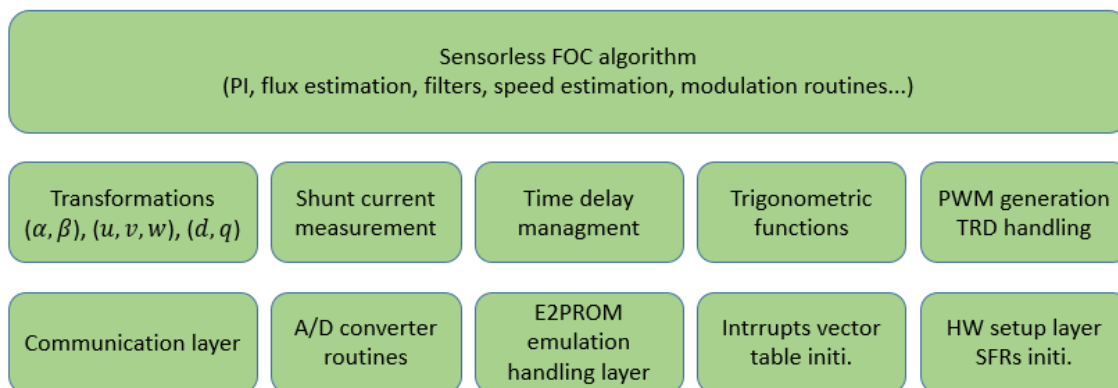
Figure 47. *Software Blocks of FOC Motor Control Embedded Software*

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: 25.87KB   and   RAM memory usage: 2.23KB under IAR**

- **FLASH memory usage: 24.79KB   and   RAM memory usage: 2.02KB under CS+ for CC**

The embedded software package is called **"YBLDCSKRL78F14_iar.zip" and "YBLDCSKRL78F14_cs+.zip"** including three shunt and single shunt current measurement configurations with clock source at 32MHz and 24 MHz, respectively, running under IAR environment and CS+ for CC.

The control loop of the field oriented control algorithm (e.g. sampling frequency, "**SAM_FRE_CUSTOM_HZ**") is set to **8 KHz** by default. The Pulse Width Modulation (PWM,

"**PWM_FRE_CUSTOM_HZ**") frequency is set by default to **24 KHz**. The parameters are visible in the module name "customize.h".

As the sampling period is **125µs** (8KHz), and the control loop for the inverter control takes maximum of **87.5µs under IAR** and **78.33µs under CS+,** the CPU load of the motor control algorithm with all the option enable is: 87.5/125 = **70% under IAR** and 78.33/125 = **62.66% under CS+,** where the CPU clock is running at 32MHz.It leaves 30% under IAR and 37.34% under CS+ of the CPU to perform additional tasks, like board management, system control, display, etc.

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

Please find below the flowchart for the main loop, the interrupt service routines and the Automatic Tuning.
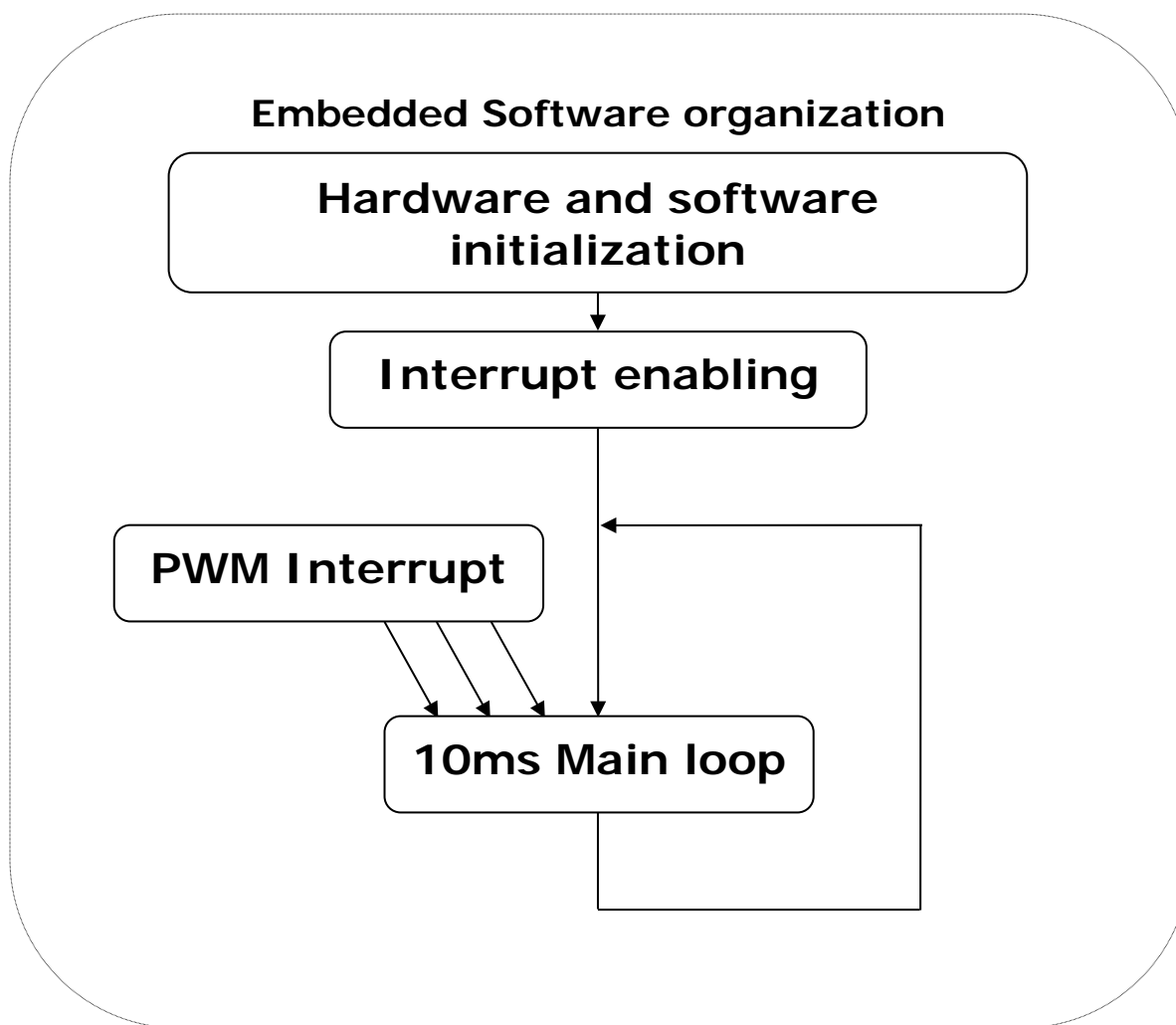
Figure 48. Flow Chart for the Main Loop
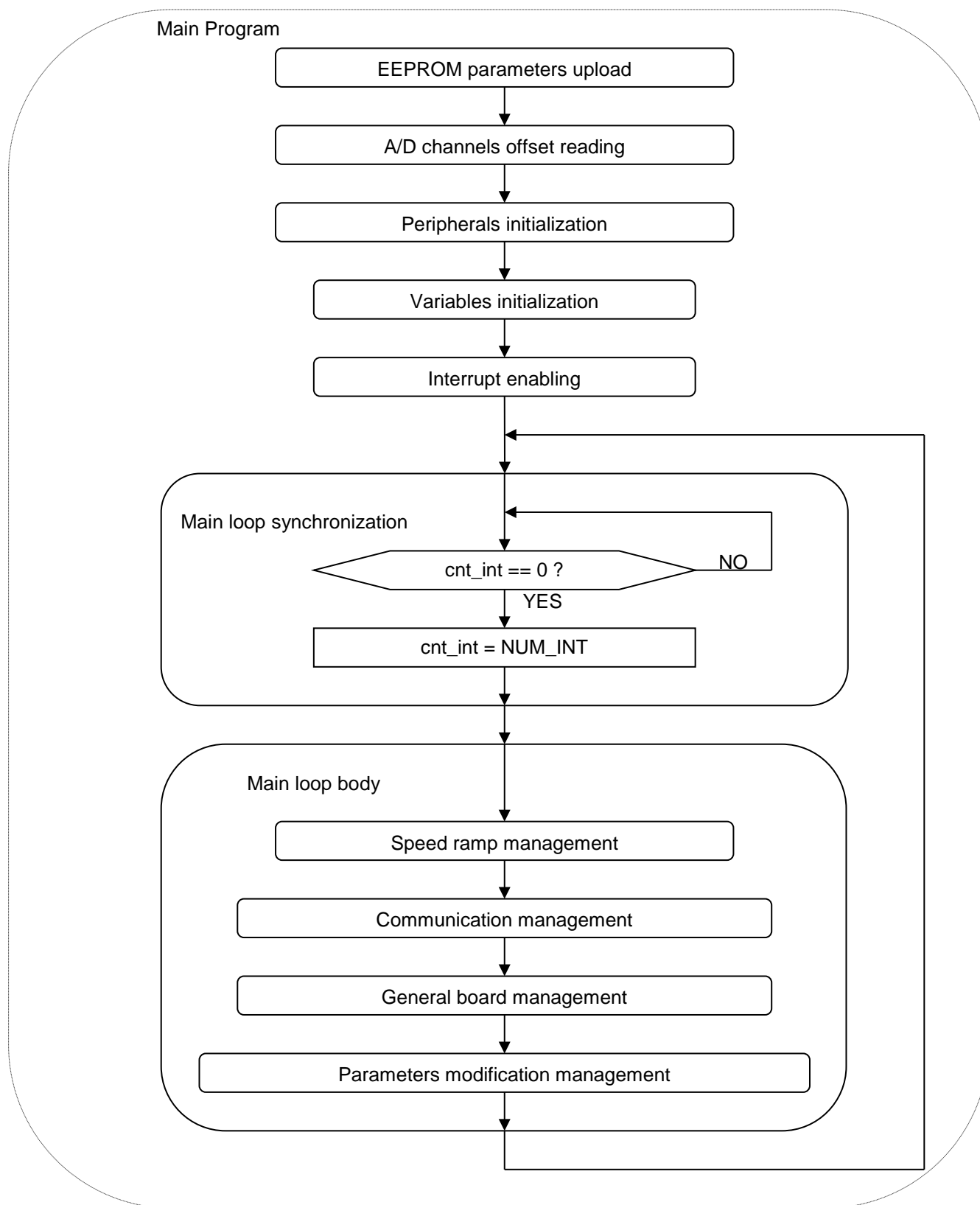
Figure 49. Flow Chart for the Main Loop in detail

Control Interrupt in three shunts current reading mode

Phase currents (iu_{mea}, iv_{mea}) reading

Transformations (using the phase angle ϑ): (iu_{mea}, iv_{mea}) $\rightarrow$ (ia_{mea}, ib_{mea}) $\rightarrow$ (id_{mea}, iq_{mea})

Read DC Link voltage v_{bus}

Phase angle update: $\vartheta = \vartheta_{new}$

Current PI controls use (id_{ref}, iq_{ref}), (id_{mea}, iq_{mea}) to produce (vd_{out}, vq_{out})

Transformations (using the phase angle ϑ): (vd_{out}, vq_{out}) $\rightarrow$ (va_{out}, vb_{out}) $\rightarrow$ (vu_{out}, vv_{out})

PWM output commands generation (using vu_{out}, vv_{out})

v_{bus} is used to calculate maximum phase voltage (used in current PI controls)

Phase estimation based on old_va_{out}, old_vb_{out}, ia_{mea}, ib_{mea}, produces new estimated phase angle ϑ_{est}

Voltage memories update: $old\_va_{out} = va_{out}$, $old\_vb_{out} = vb_{out}$

Speed estimation produces ω_{est}

Estimation errors detection (if errors: alarm is set)

Start-up in progress?

YES — Start-up procedure produces id_{ref}, iq_{ref}, ϑ_{stup}

$\vartheta_{new} = \vartheta_{stup}$

NO — $id_{ref} = 0$

Speed PI control uses (ω_{ref}, ω_{est}) to get iq_{ref}

$\vartheta_{new} = \vartheta_{est}$

Main loop synchronization

cnt_int > 0 ? — NO

YES

--cnt_int

Figure 50. *Control Interrupt in Three Shunt Current Reading Model*

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 starter and configuration of the software.
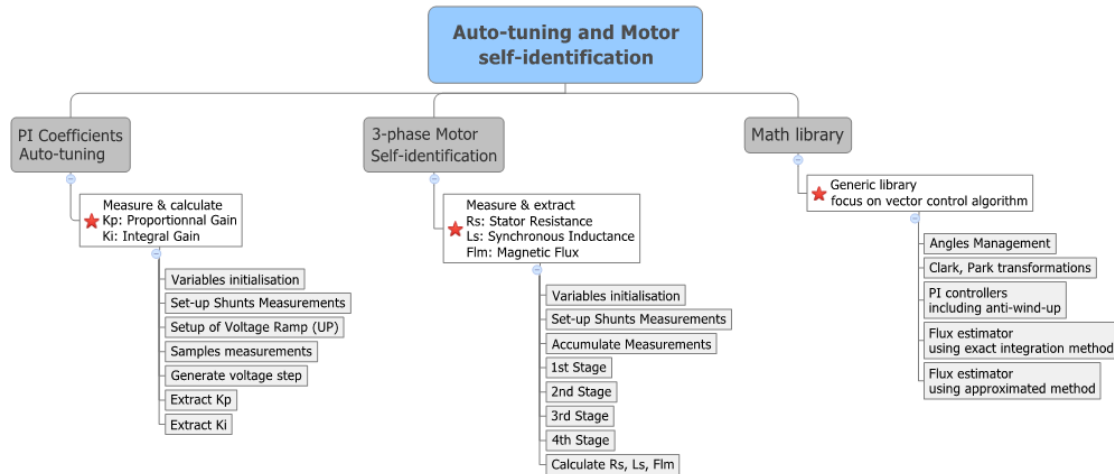
Figure 51. *Features of Auto-Tuning and Motor Self-Identification*

The three blocks mentioned above and the completed FOC algorithm are located in the library called: "BLDC-SK-Lib_threeShunt" or "BLDC-SK-Lib_singleShunt" located in the folder called: "MC_Library" as shown below:

MC_Library	Function Description
mcrplib.h	Header file for the Math Library block
mod_ident.h	Header file for the Motor Self-identification
PI_tun.h	Header file for the PI Coefficients auto-tuning
motorcontrol.h	Header file for complete FOC algorithm in interrupt service routines
BLDC-SK-Lib_threeShunt	Library file containing the four blocks above

Table 3. *Contents of MC_Library*

The complete project source code under IAR embedded environment and CS+ for CC is described below. For each C module, a specific header file is associated.

Source Files	Functions Descriptions
BLDC-SK-Lib_threeShunt	Contains PI auto-tuning, Motor identification and estimators routines
hwsetup.c	Basic hardware initialization
stl_support.c	Support routines for the self-test functions
main.c	The main program loop
userif.c	Communication routines (i.e. GUI)
par_tab.c	The Parameter management definitions and tables
globalvar.c	Global variable definitions

Table 4. *C Modules of Project Source Code*

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

Header Files	Functions Descriptions
customise.h	Basic motor parameters (Not modifiable through the GUI)
const_def.h	Definition of the basic numerical constants
par_tab.h	Parameter definitions, function prototypes and references
hwsetup.h	Hardware definitions, references and function prototypes
globalvars.h	Global variable definitions and references
multiply.h	Assembler Basic math function references
mask.h	General support definitions and references
userif.h	General support definitions, references and function prototypes

Table 5.   Head Files of Project Source Code

The following table shows the assembler modules in the project.

Assembler Modules	Functions Descriptions
Self-Test	IEC assembler Self-test routines (RAM, FLASH, Registers and Clock)
multiply	Combined math's functions
cstartup	Customized start up file (Includes March C RAM test)
delay	1us and 100us delay

Table 6.   Assembler Modules of Project Source Code

The following table shows the data flash library modules in the project.

Data Flash Library	Functions Descriptions	Environment
fdl*	RL78 FDL T01 (flash data library) files	IAR V1.xx
		RENESAS and IAR V2.xx

Table 7.   Data Flash Library Files of Project Source Code

Note:

For updates of the data flash library (FDL) please check on the following site:

http://www.renesas.eu/update?oc=EEPROM_EMULATION_RL78

15. 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*).
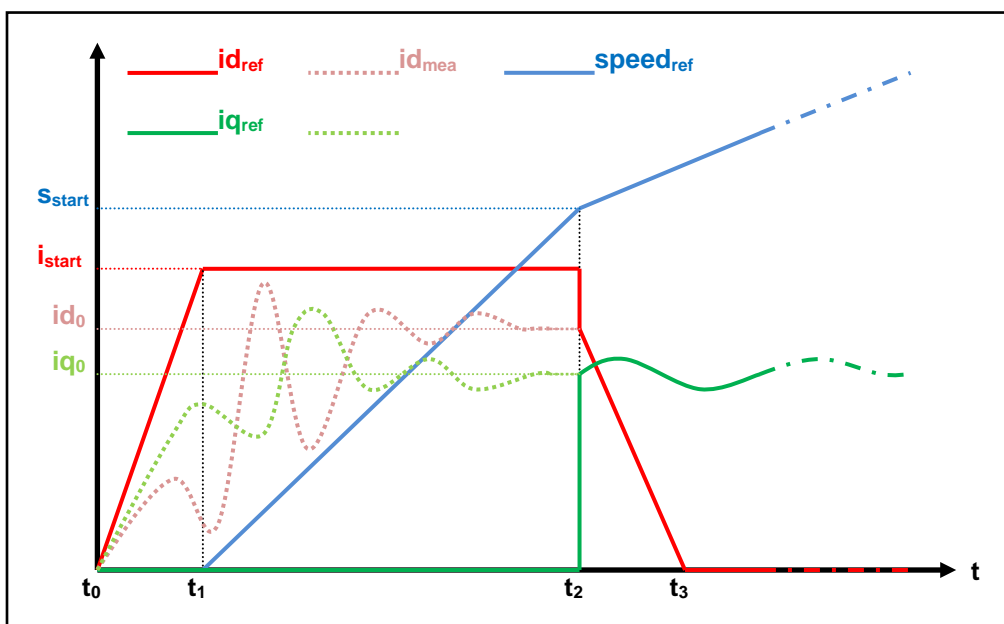
Figure 52. Startup Procedure in Embedded Software

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

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

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

The references are referred to an arbitrary (d_a, q_a) system based on the arbitrary phase ϑ_a. From this moment, the phase estimation algorithm begins to be performed, and the estimated phase ϑ_{est} is used to calculate the components of the measured current, referred to the (d, q) system based on the estimated phase, id_{mea} and iq_{mea}. The components of the current referred

to the arbitrary (d_a, q_a) system are controlled to follow the references by the current PI controllers. On the other hand, since the phase ϑ_{est} is still not correctly estimated, id_{mea} and iq_{mea} have no physical meaning. Even if they are not shown in the graph, the applied voltages are subjected to the same treatment (vd_{mea} and vq_{mea} are calculated in the algorithm).

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

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

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

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

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

16. Reference system transformations in details

Find below the detailed equations used for the coordinates transformations in the embedded software for the RL78/F14 microcontroller.

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

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

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

$$g_u = g_\alpha$$

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

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

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

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

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

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

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

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

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

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

17. 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 RL78/F14 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 second low-pass filter, as in the following schemes in Figure 53 and Figure 54:

First order low-pass $1/(s+\omega_0)$	Derivative: s	First order low-pass: $1/(s+\omega_0)$
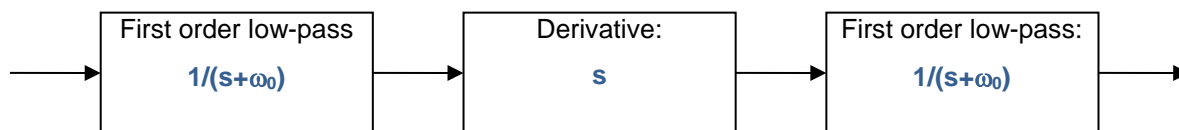
Figure 53. *Filter Diagram of approximated Integration in Rotor Position Estimation*

$$Y(s) = \frac{s}{(s+\omega_0)^2}$$
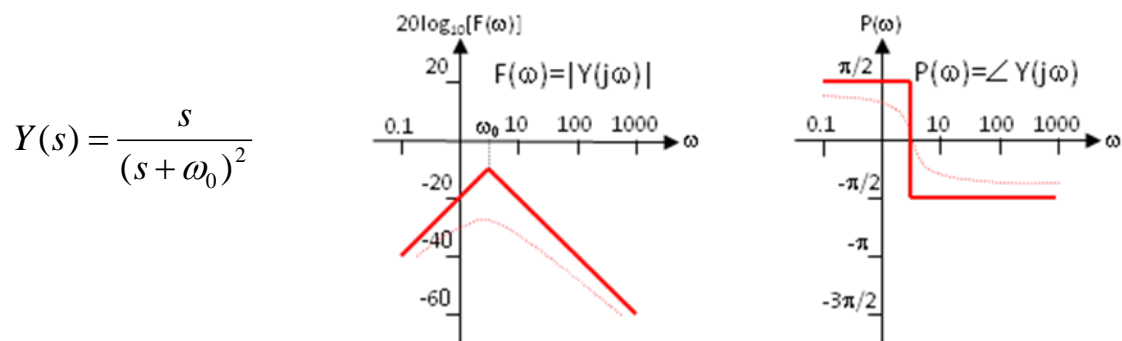
Figure 54. Corresponding S-domain Functions for Filters in Figure 53

It is evident the relationship between Y(s) and the integral I(s)=1/s for s=jω, when ω>>ω_0 .

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

The block scheme of the exact BEMF integration method for flux position estimation is the following:
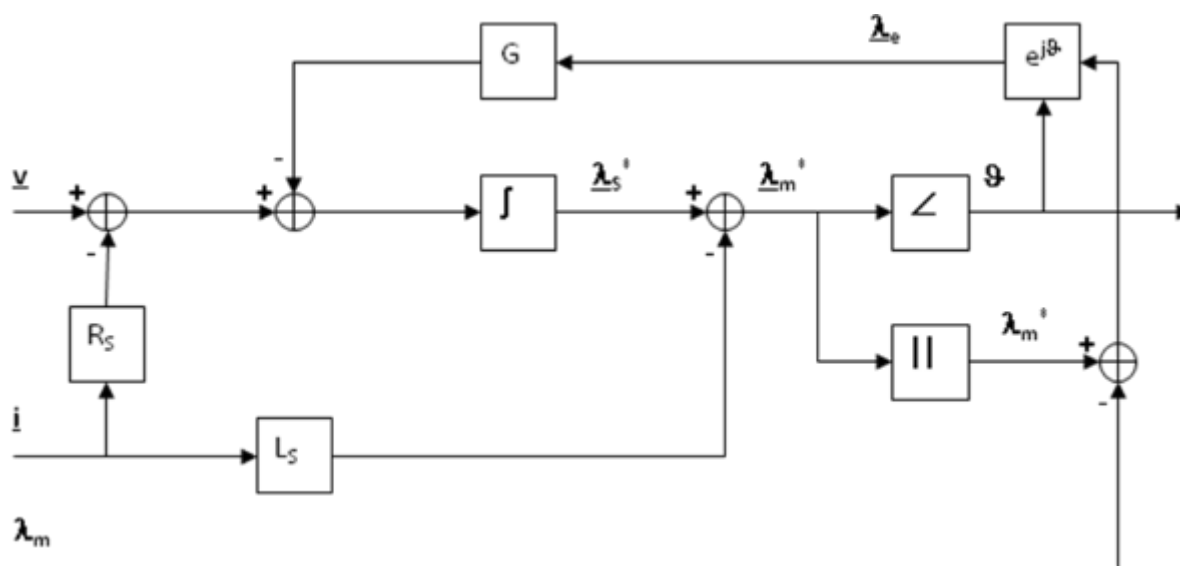
Figure 55. Block Scheme of the Exact BEMF Integration Method for Flux Position Estimation

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

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

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

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

In the **default embedded software** delivered on the Y-BLDC-SK-RL78F14 Starter Kit, this **first** strategy is selected. The choice to test the second 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.

18. 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 Y-BLDC-SK-RL78F14 Starter Kit. After connecting the Fulling Motor (FL28BL26-15V-8006AF, 15V_{DC}, 8000RPM), please connect the board RL78/F14 and select the COM port or use the Auto-detection mechanism. See in Chapter 20 Figure 65.

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 descriptions of the PC GUI tabs and windows.
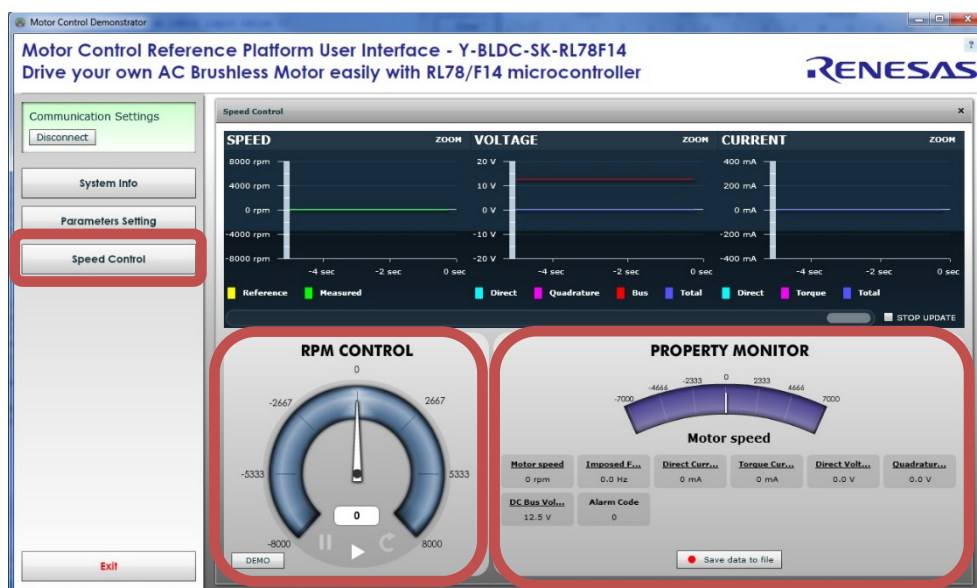
Figure 56. Detail Descriptions of PC GUI

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 described in figure below:

Click to record in real-time All the values and store them In a ".CSV" format file.

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

Figure 57. Save Data to File in Real-Time

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

Alarm 0: Normal operation
Alarm 1: EEPROM fault
Alarm 3: Loss of Phase
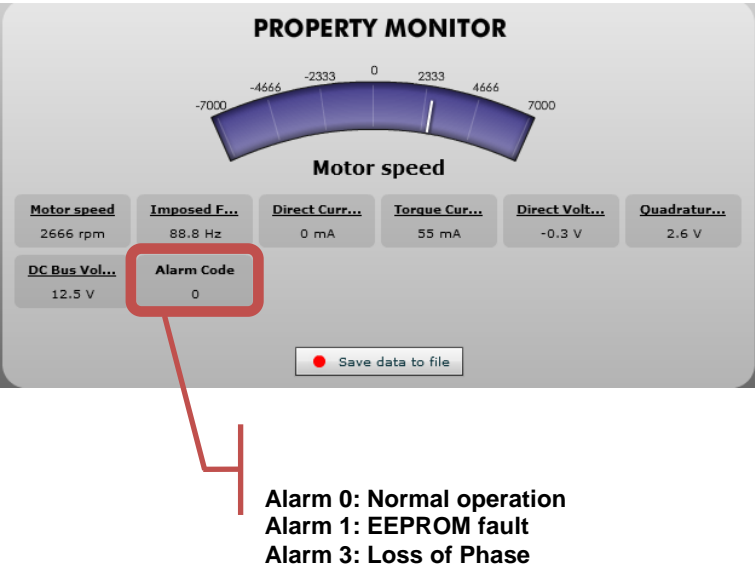
Figure 58. Alarm Codes Status of the Board

Alarm code 1:

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

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

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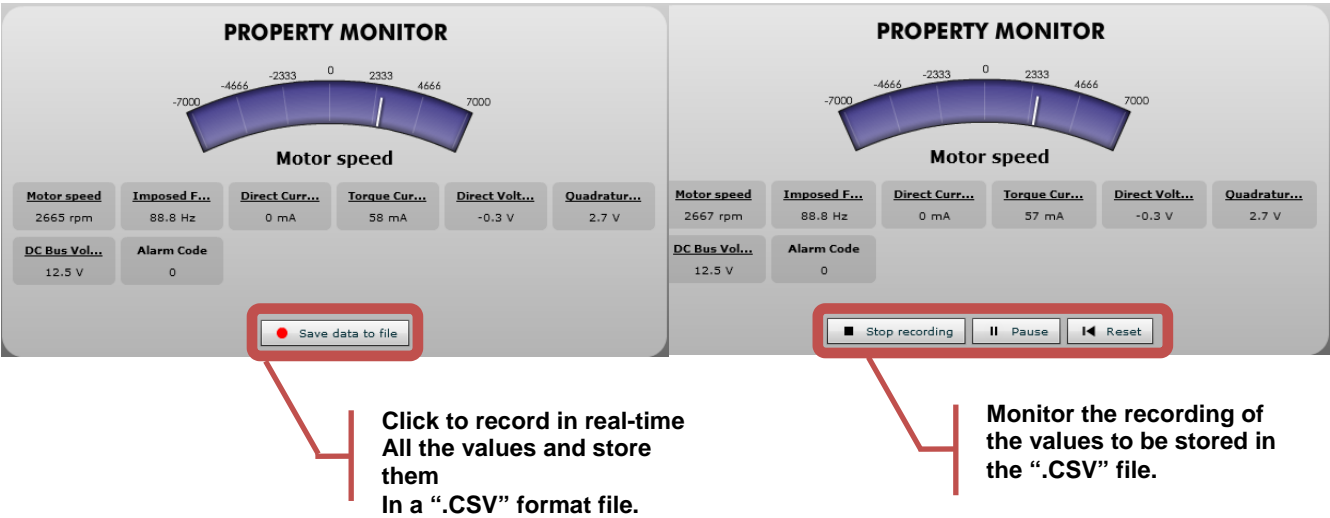econd step is to reset the board by pressing the "reset" button on the PCB, then click the "Reload" button in the parameters setting window._

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

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

Alarm code 3:

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.

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 )                        // ipm hardware alarm code
#define TRIP_ALL         ( 3 )                          // loss of phase alarm code
```

Figure 59. *Function Descriptions of Alarm Code Statuses in Source Code in Embedded Software*

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.
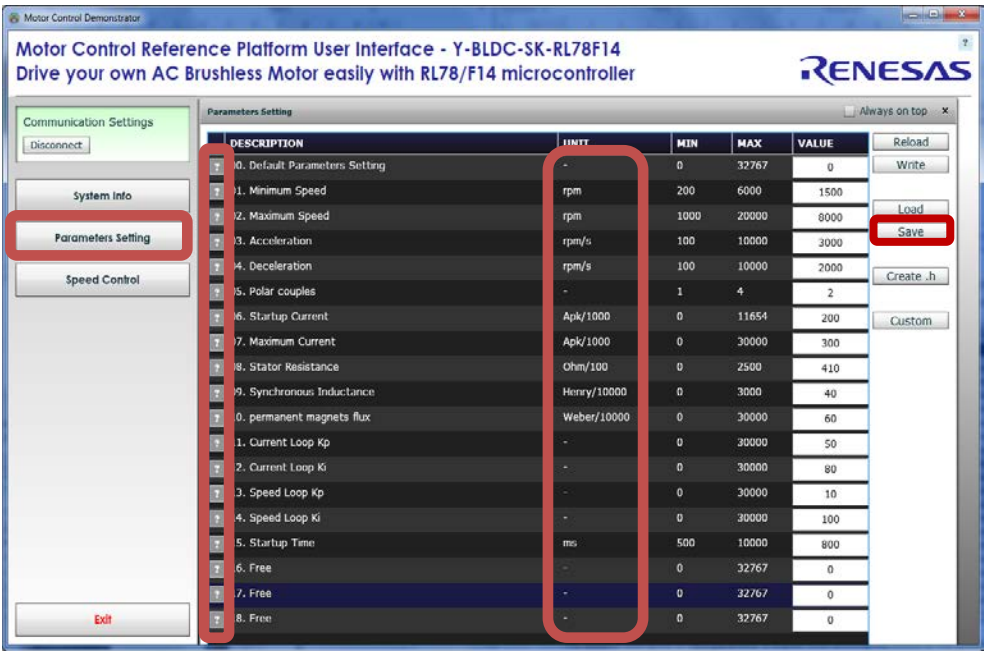
Figure 60. Question Mark Functions in Parameters Window

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

Note: To change one value in real-time, simply enter the new value and click on "Write" to program the new value into the EEPROM.

Reload is used to recall EEPROM default parameters. **Write** is used for programing the new entered value inside the EEPROM

Load is used to reload an existing parameters file saved previously. **Save** is used to save the parameters value into a text file.

Create.h is a button used to generate a full 'customize.h' file that can be imported in any IAR and CS+ for CC project
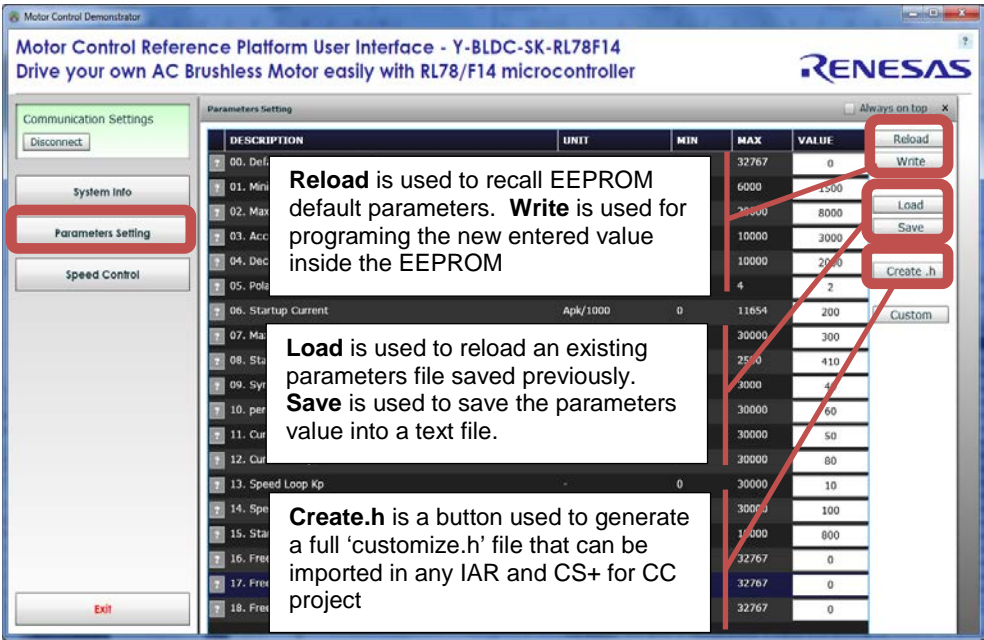
Figure 61. Buttons Function-Descriptions in Parameters Window

All the parameters can be changed on the fly and after pushing the "Write" button, it's automatically set.

Speed range limitations

The Y-BLDC-SK-RL78F14 Starter Kit is driving any BLDC 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 Fulling Motor FL28BL26-15V-8006AF delivered with the kit, the minimum speed is **700RPM**. Below this speed, the current flowing through the three shunts are too low to be detected.

The FL28BL26-15V-8006AF brushless motor is able to reach its maximum speed of **8000RPM** (without load) when the power supply is 15V.

19. 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
1	RPM_MIN	Set the Minimum Speed in RPM
2	RPM_MAX	Set the Maximum Speed in RPM
3	R_ACC	Set the acceleration [RPM/s]
4	R_DEC	Set the deceleration [RPM/s]
5	C_POLI	Set the number of polar couples
6	I_START	Set the start-up current (peak) [Ampere/AMP_DIV]. Used to specify the peak phase current value to be used during the start-up
7	I_MAX	Set the maximum phase current (peak) [Ampere/AMP_DIV]
8	R_STA	Set the stator resistance [Ohm/OHM_DIV]
9	L_SYN	Set the synchronous inductance [Henry/HEN_DIV]
10	PM_FLX	Set the permanent magnets flux [Weber/WEB_RES]. This value is only used when the exact integration flux estimation algorithm is selected. By default, it's not needed as the approximated integration is selected.
11	KP_CUR	Set the Current loop Proportional coefficient: KP
12	KI_CUR	Set the Current loop Integral coefficient: KI
13	KP_VEL	Set the Speed loop Proportional coefficient: KP
14	KI_VEL	Set the Speed loop Integral coefficient: KI
15	STP_TIM	Set the Start-up acceleration time [ms]

Table 8.   Full Descriptions of Software Parameters in EEPROM

20. Motor Auto-calibration using the PC GUI

The full calibration of any 3-phase Brushless DC motor can be performed automatically using the PC Graphical User Interface. Three specific buttons are now available for and shown below:
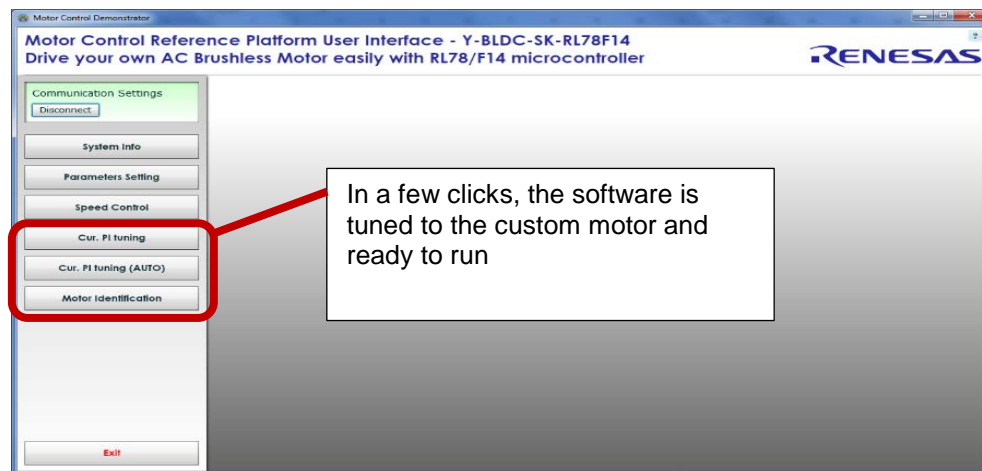
Figure 62. *Buttons of Current PI-tuning, Auto PI-tuning and Motor Identification*

In terms of DC 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 Fulling Motor FL28BL26-15V-8006AF delivered with the Y-BLDC-SK-RL78F14 Starter Kit. The FL28BL26-15V-8006AF motor 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) Please find below the specifications of the Motor delivered by the motor maker:

Motor Manufacturer: **FULLING**   www.motor-fulling.com

Motor type: 3-phase DC Brushless **FL28BL26-15V-8006AF**

Maximum current: **0.3A**

Bus Voltage: **15V**

Speed rated: **8000 RPM**

Number of pole pairs: **2**

NUMBER OF POLES	4
LINE TO LINE RESISTANCE	8.2±10% ohms @20°C
LINE TO LINE INDUCTANCE	2.3±20% mH
NOMINAL VOLTAGE	15VDC
NO LOAD SPEED	9600±10% RPM
NO LOAD CURRENT	< 0.3 A @25°C
RATED SPEED	8000±10% RPM
RATED TORQUE	5 mN.m
PEAK TORQUE	15mNm
TORQUE CONSTANT	13.7 mN.m/A
BACK EMF	1.06±10%Vrms/Krpm
ROTOR INERTIA	2.35g.cm^2
WEIGHT	60g
WIRE DIAGRAM	UL1007 AWG26#

Figure 63. Specifications of three-phase Brushless DC FL28BL26-15V-8006AF

b) Let's connect the 15VDC Power supply to the YBLDCSKRL78F14 kit.

c) Now, connect the USB cable to the PC and the Kit and connect the motor to the kit:
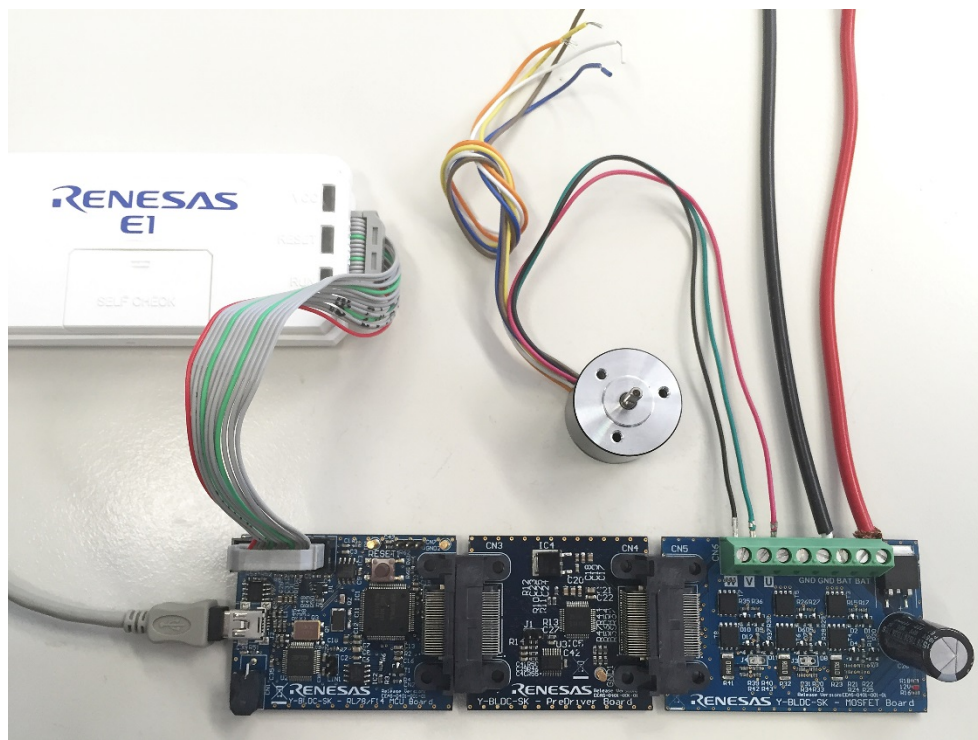
Figure 64. Top Overview of RL78/F14 BLDC Starter Kit with Motor

Figure 65. Connect PC GUI to the RL78/F14 BLDC Starter Kit

d) Launch the PC GUI by clicking on the ""Motor Control Demonstration for RL78 F14" ICON on the desktop, or follow these instructions "Start Menu => All programs => Renesas Electronics Europe GmbH => Motor Control Demonstrator for RL78F14 => Motor Control Demonstrator for RL78 F14.exe".

e) Firstly click on the "setup" button and select "RL78F14_Kit" and select appropriate serial port "COM X" (in my case it is COM 11) and click on "Connect" to ensure the PC GUI is connected to the RL78/F14 kit.

On the left hand side, the new buttons appear: "Cur. PI tuning", "Cu. PI tuning (AUTO)" and "Motor Identification", which are needed for the self-calibration of the motor. See in Figure 62.

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:

1) Click on the "Parameters Setting" button and enter the magic number "33" in the first line called: "00. Default Parameters setting"

2) Click the "Write" button in the parameter setting window

3) Then push the "RESET" button on the board as shown in figure below

4) Click the "Reload" button in the parameter setting window to get the default parameters defined in the "customize.h" header file in the IDE workspace.
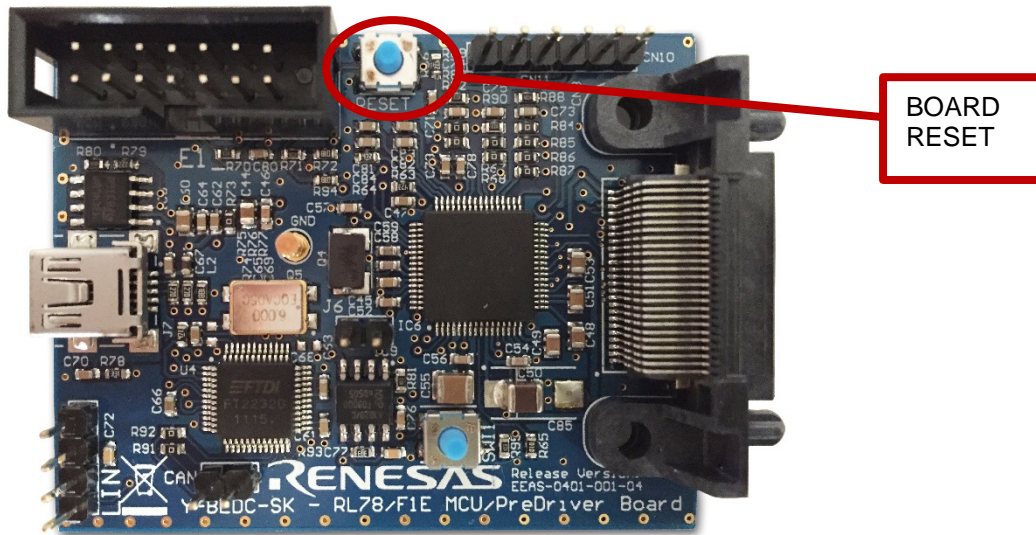
Figure 66. *Board Reset Button in RL78/F14 BLDC Starter Kit*

5) Set the maximum current (parameter n°07) as it will influence all the next steps: Click on "Parameters Setting", Enter the value: 300 (the unit is in mA) and click on "Write" to save the parameter into the EEPROM and close the parameters 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.
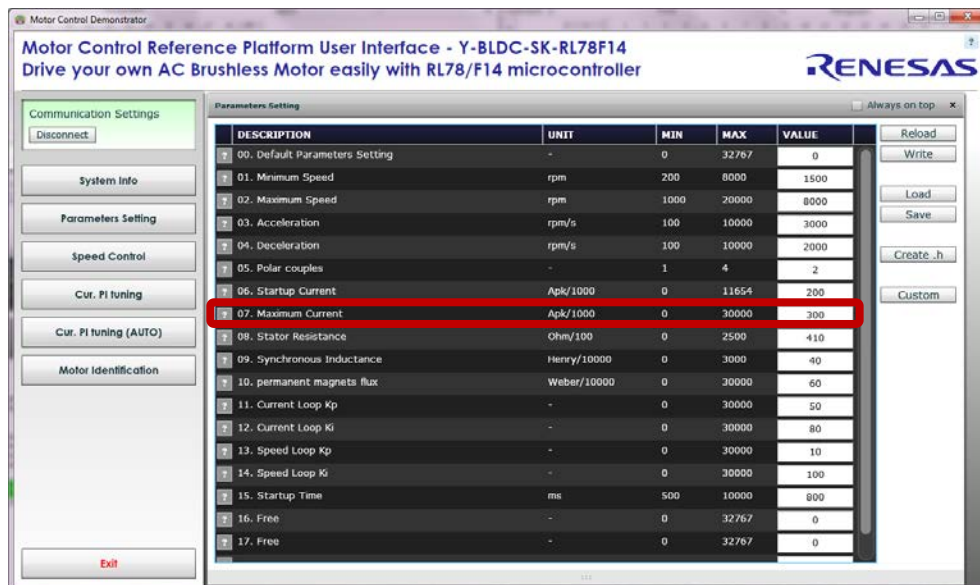
Figure 67. *Maximum Current for Auto-tuning*

g) 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.
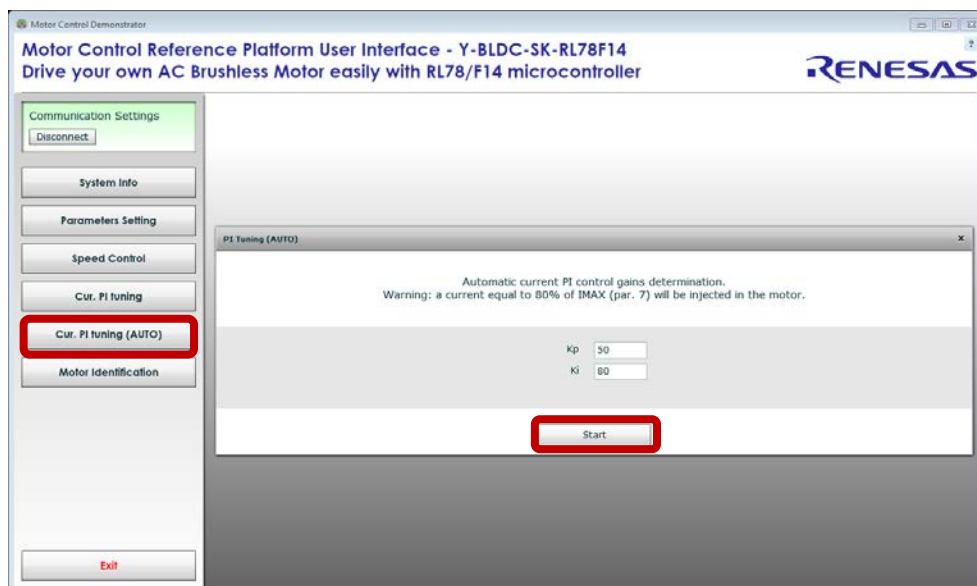
Figure 68. Start Current Auto PI-tuning

And click on "Yes" to accept the results to be programed into the EEPROM as shown below.
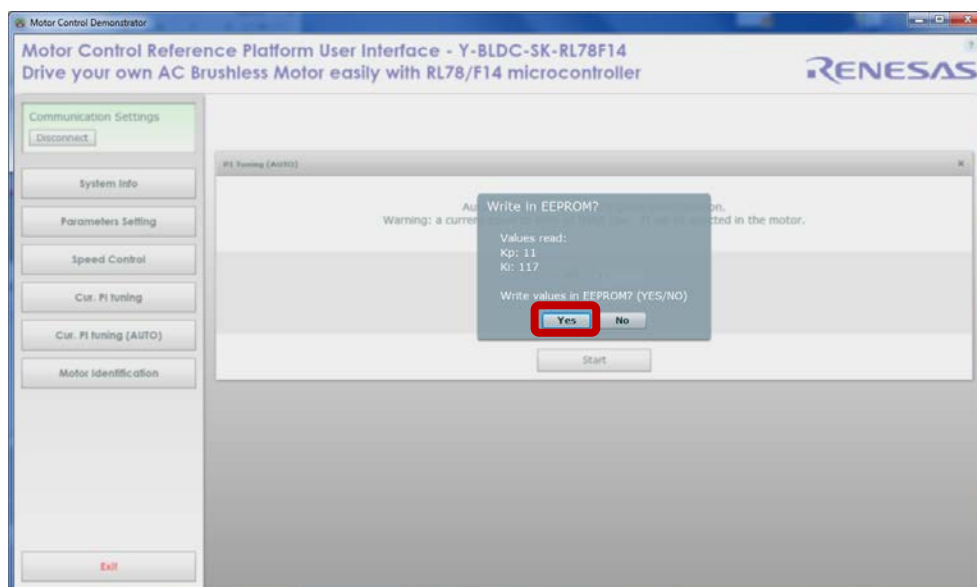
Figure 69. Accept Results of Current Auto PI-tuning

Important note: The proportional and integral coefficients are just **starter values** for the next step "Cu. PI tuning".

h) 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.
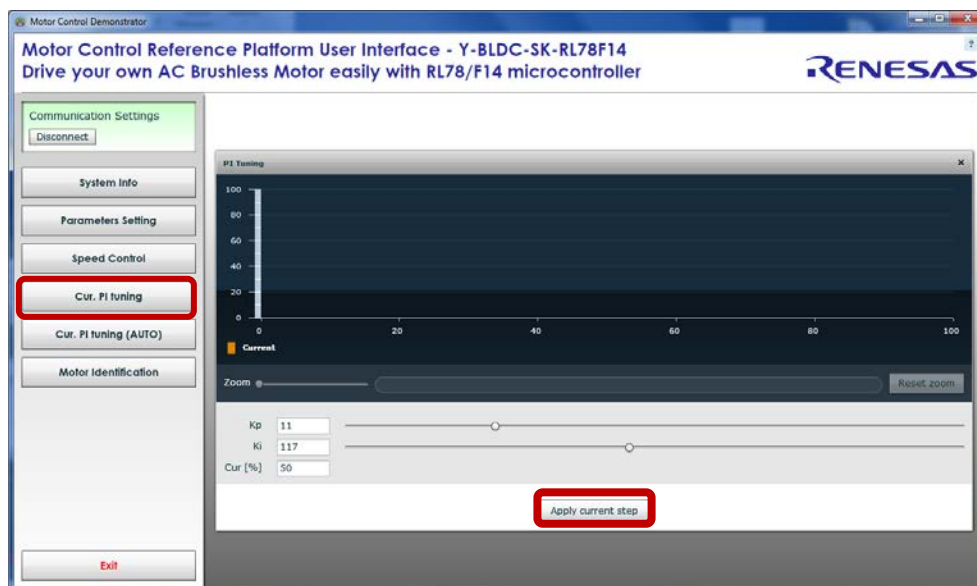
Figure 70. Start Step Response Checking the Results of Current Auto PI-tuning
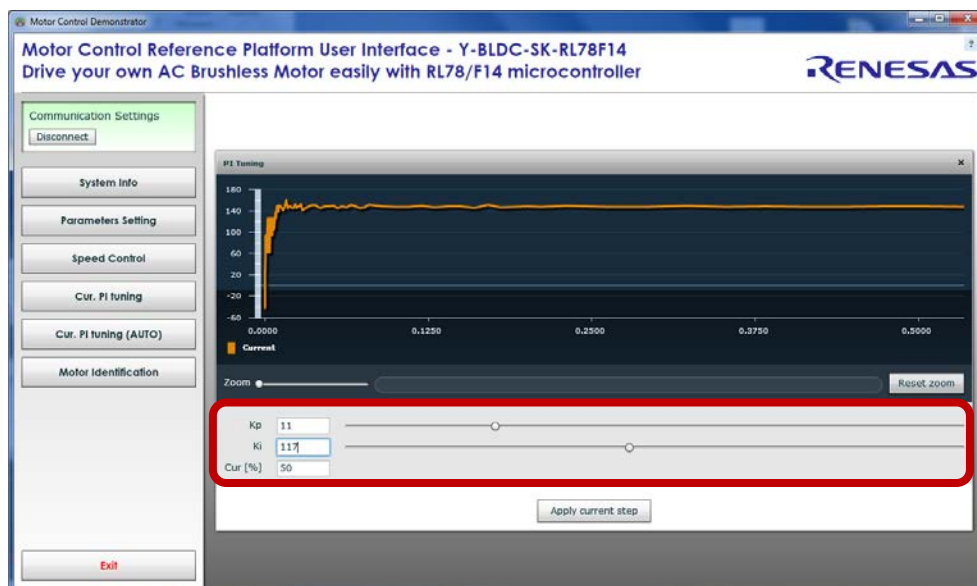
Figure 71. Results of Step Response

Note: You can manually adjust the value of Kp and Ki, or just slide the slider point 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%.

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.

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

i) Perform an auto-identification of the motor parameters by clicking on "Motor Identification" and click "Start":
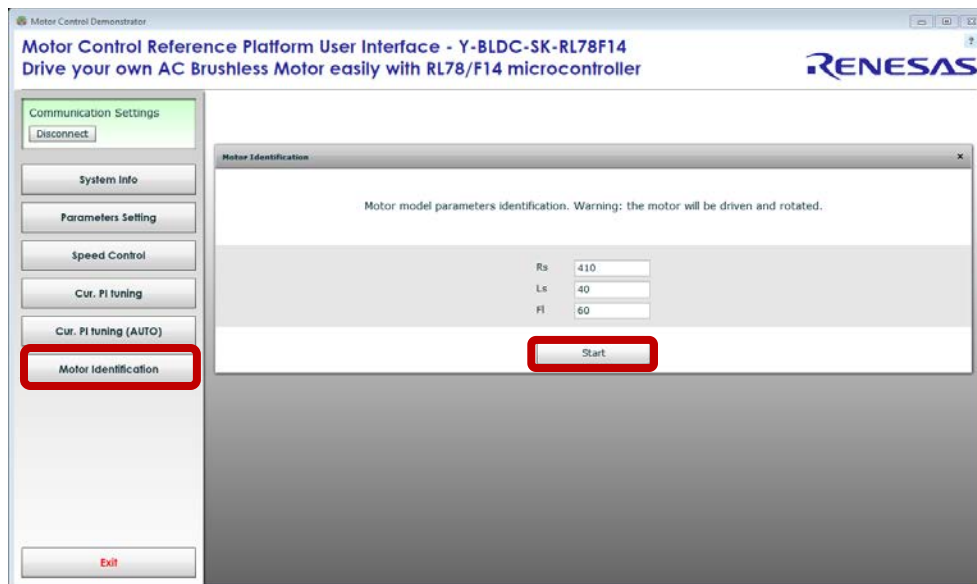
Figure 72. Start Motor Identification

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", as shown in the following figure.
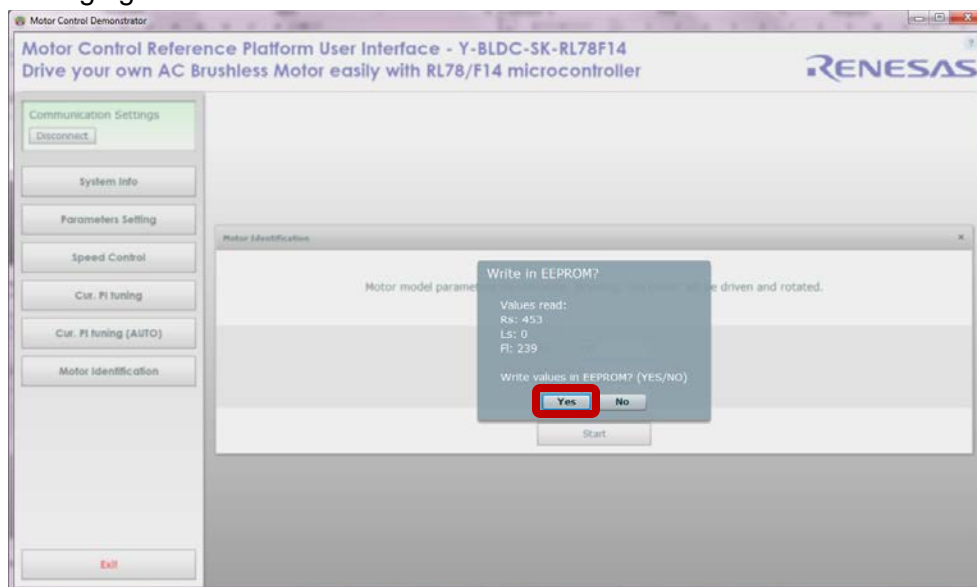
Figure 73. Accept the Result of Motor Identification

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

Important note:

- The value above are just **starter values**, alternately the phase resistance can be found to be sure by measuring the phase-to phase resistance.

- The measurement unit is Ohm/100. The measured value should firstly be divided by 2 then multiplied by 100, please enter this value in the parameter 08, "Stator Resistance".

j) Now please click on "Parameters Setting" and enter the number of pole pairs: **2** (parameter n°5) and enter a minimum speed or 1000 RPM

k) Set a start-up current equal to less than the maximum current. Please enter an average value that will not damage the motor. Here enter 200 in the parameter 06, "Startup Current", which means 0.2 Amperes, as shown in the figure below.

Figure 74. Set Startup Current

Then let's close the window.

l) Now, let's try to run the motor. Please click on the button: "Speed Control":
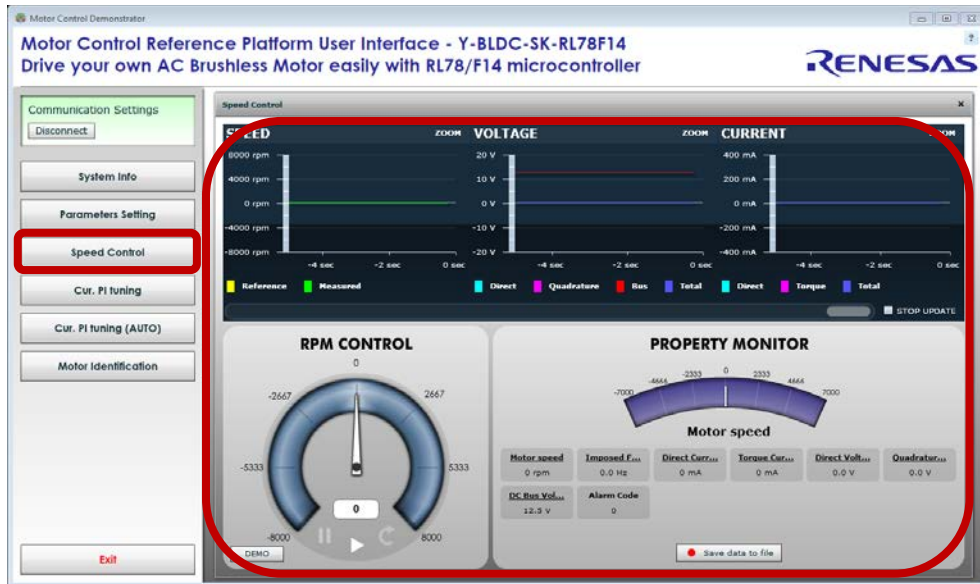
Figure 75. Speed Control Interface Overview

To start the motor, let's enter a speed which is 1.5 times the minimum speed, in this case **1500 RPM**
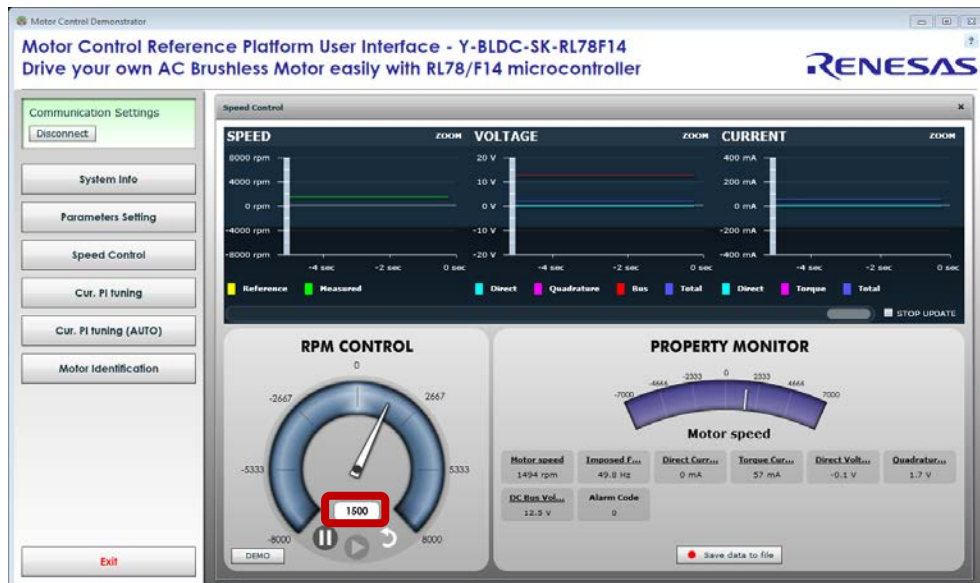
Figure 76. Set the min. Speed in Speed Control Interface

m) 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

Speed PI proportional gain should be tuned in the real application and under load conditions.

As starting values, low values should be chosen, they can be increased at medium working speed until instability arises. High frequency instability is related to the proportional value too high, low frequency instability is related to integral value too high. When instability arises, the value should be halved. Some kind of tuning of speed parameters can be performed using high value of acceleration ramp, and imposing speed reference variations, as done with the current PIs.

The PI calibration procedure should be iterated till the desired system response is reached. The speed reference could be changed depending on the motor/application. You can find below two graphs indicating an example of tuning procedure. This procedure should be made using the real working environment.

Speed parameters can influence the success of the start-up phase: if the algorithm fails in this phase, giving alarm n°3, try modifying the speed proportional gain first, and then the integral gain.
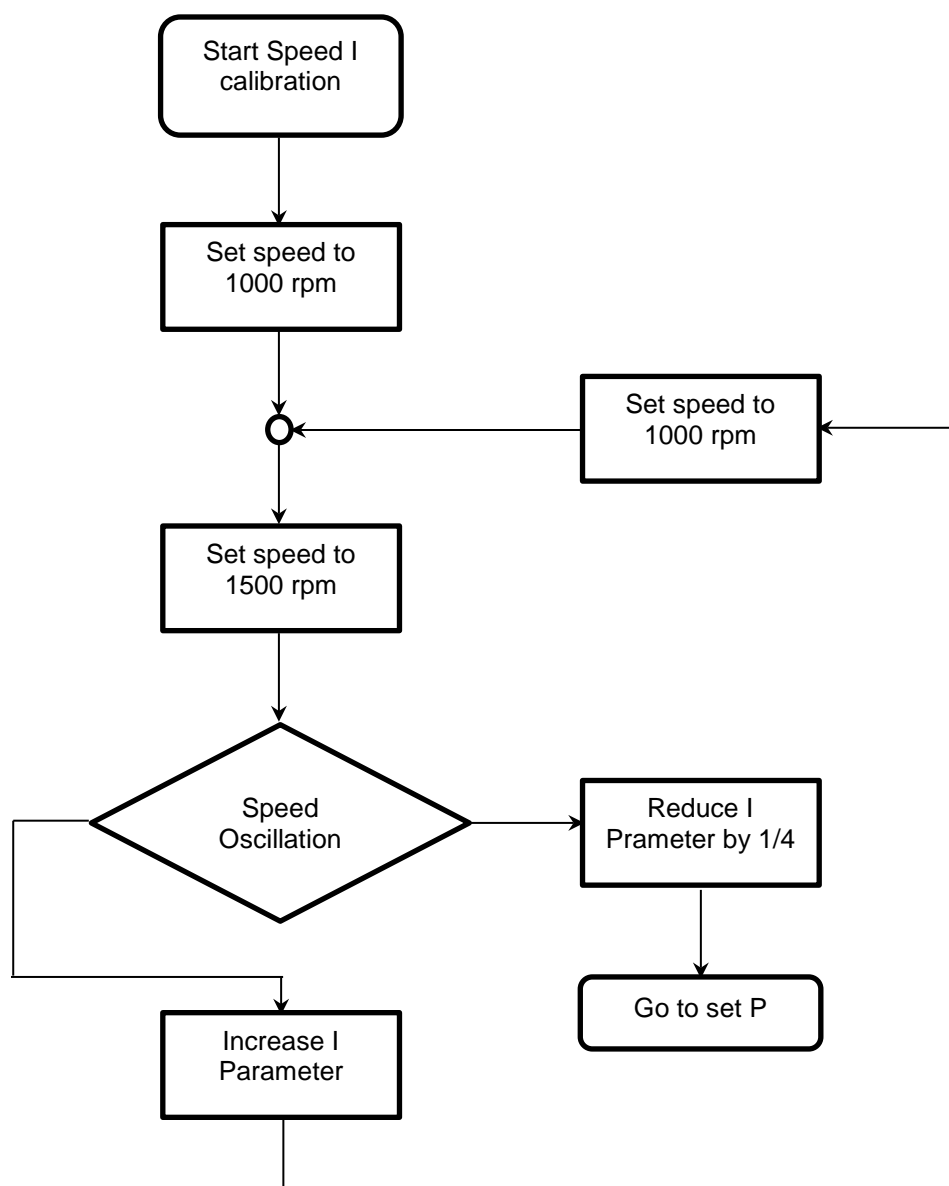
Figure 77. The Integral Term Tuning Procedure
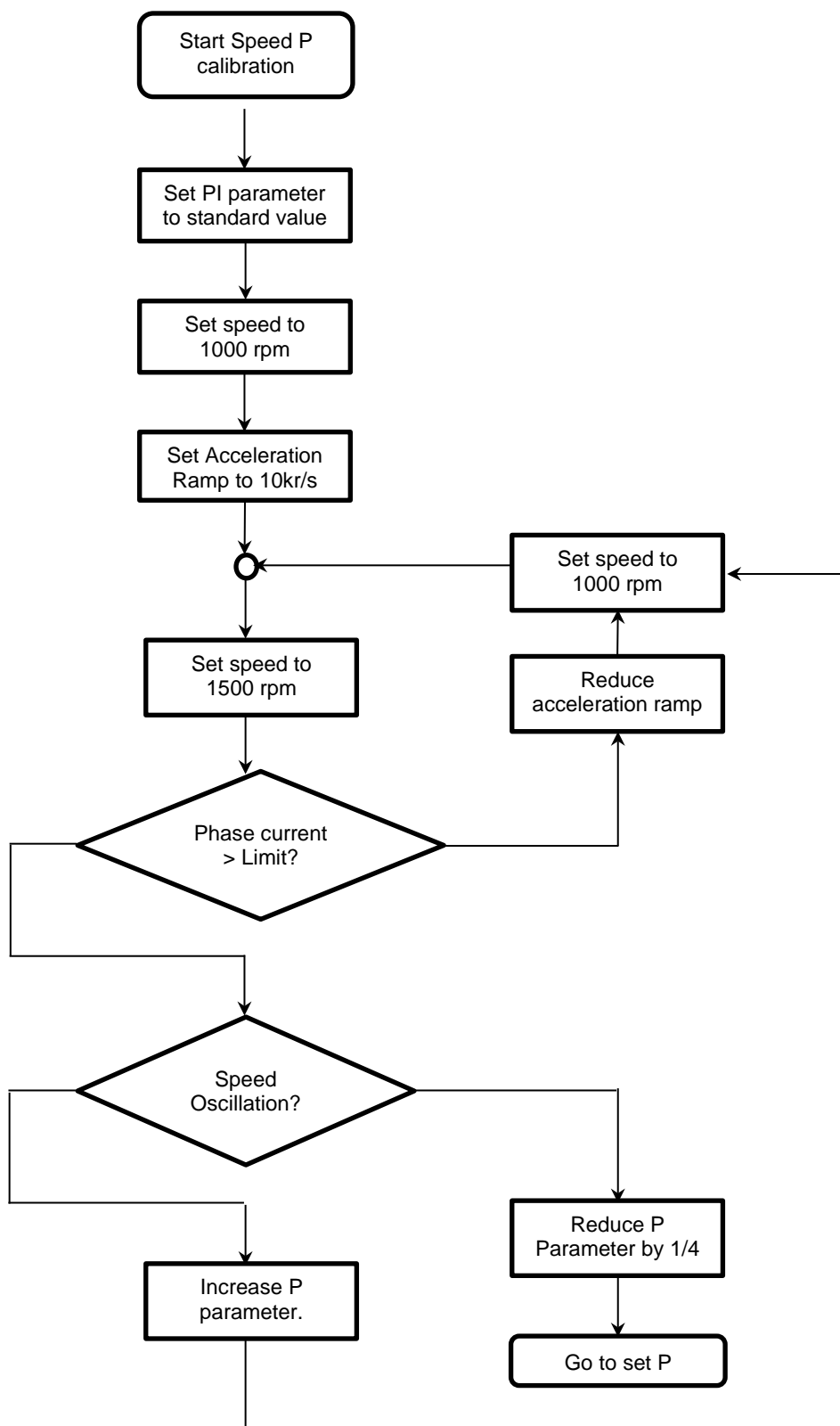
Figure 78. The Proportional Term Tuning Procedure

n) Test the parameters found in all the speed ranges and different rotations.

o) Finally the parameters list can be saved in a file in .CSV ("Save" button) or .h file ("Create.h" button) format for further use and can also be uploaded later on:
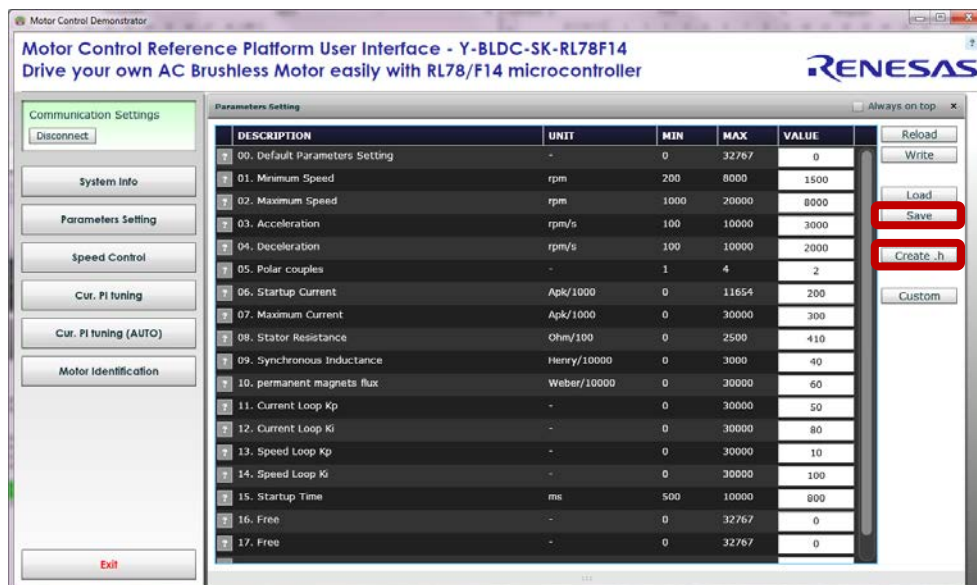
Figure 79. Save PC GUI Parameters

Troubleshooting:

At the stage l) 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. The value of the stator resistor can affect the startup and rotating procedures.

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=freq*60/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 (shunt value is 0.01 Ohm).

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

21. Updating the RL78/F14 Flash memory using the Renesas Programming Flash Tool

The procedure below explains in details how to re-program the RL78/F14 flash memory using the Renesas Flash Programing tool, called the RFP. It could be used to update the RL78/F14 kit with the latest Firmware version downloaded from the website: http://www.renesas.eu/update?oc=Y-BLDC-SK-RL78F14

The RFP must be used with the E1 debugger and there's no need to install the full version of the development environment IAR.

Please have a look at the specific Flash Programming Tool website to us the latest software version for RFP V30100:

http://www.renesas.eu/updates?oc=RFP-EE

Please follow the steps below to update the flash memory of your RL78/F14 MCU using the RFP tool.

Step1: Open **RFP software v3.01.00** select and click on "**Create new Project**", select "**RL78**" in Microcontroller drop down menu, write the project name and browse your project folder.

Step 2: Select the **E1** in Tool drop down menu in the communication box and make sure in tool details pop window select power supply "**None**"
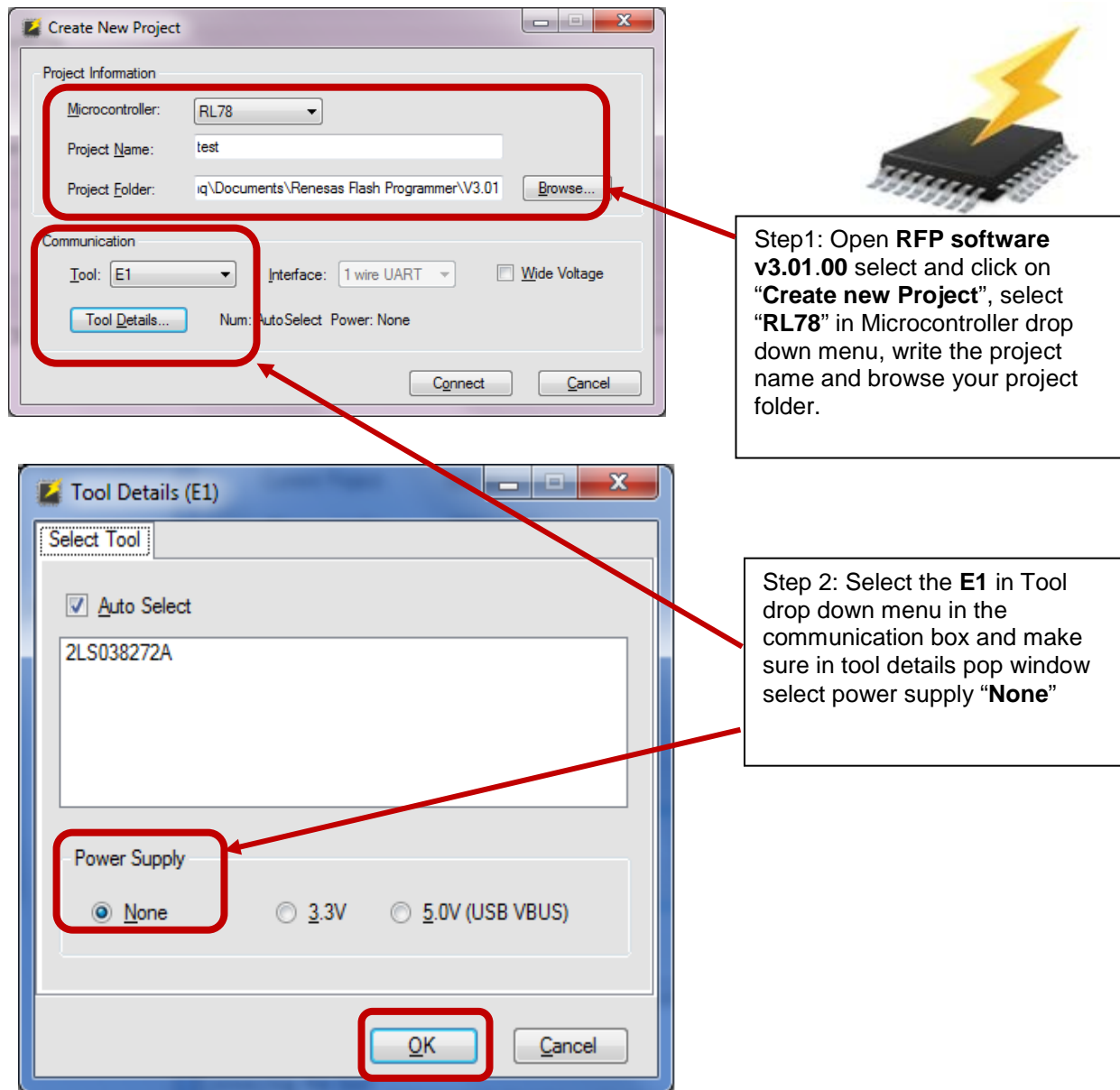
Figure 80. Update the Flash Memory using RFP

Step 3: Browse the file with the ".hex" of your project in the program and then click the large "**Start**" button
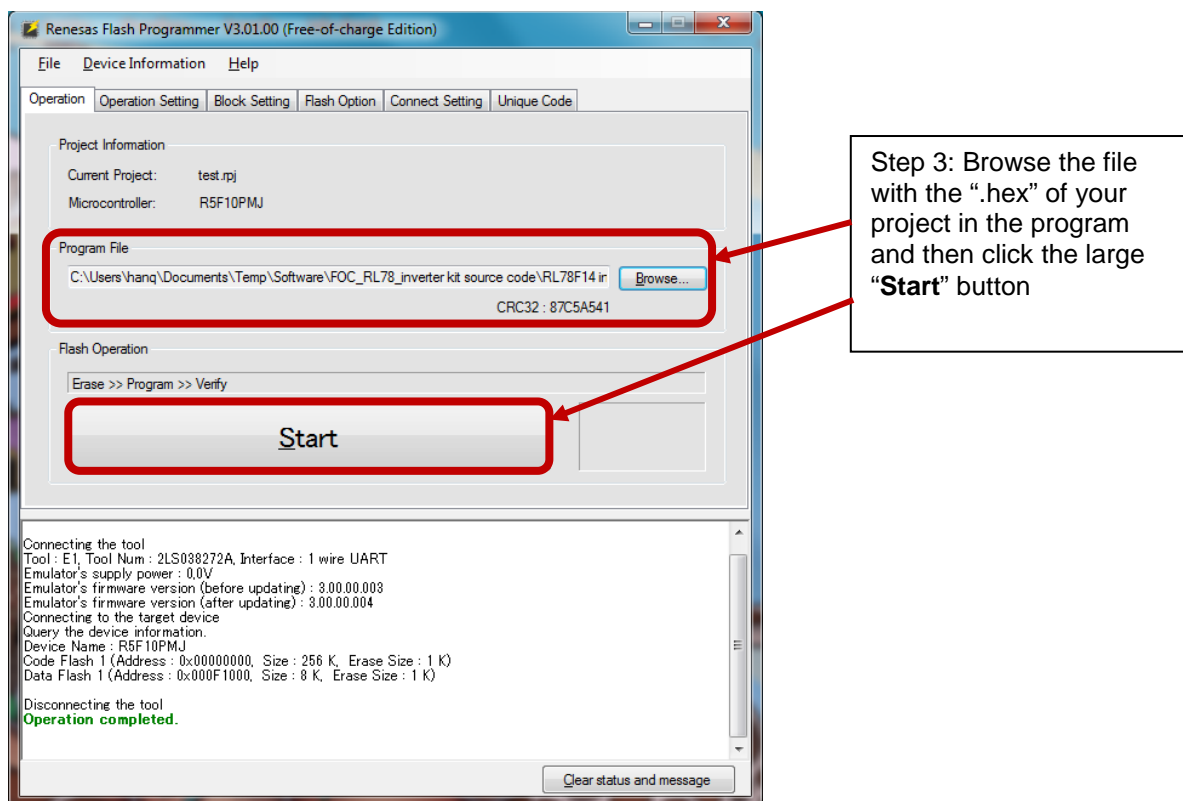
Figure 81. Start Updating the Flash Memory using RFP

The RFP will open a progress bar and connect to the board and device. The results of reading from the device should be as shown below.

Step 4: At the end of the process "**OK**" will appear. The new program is now downloaded into the flash of the RL78/F14
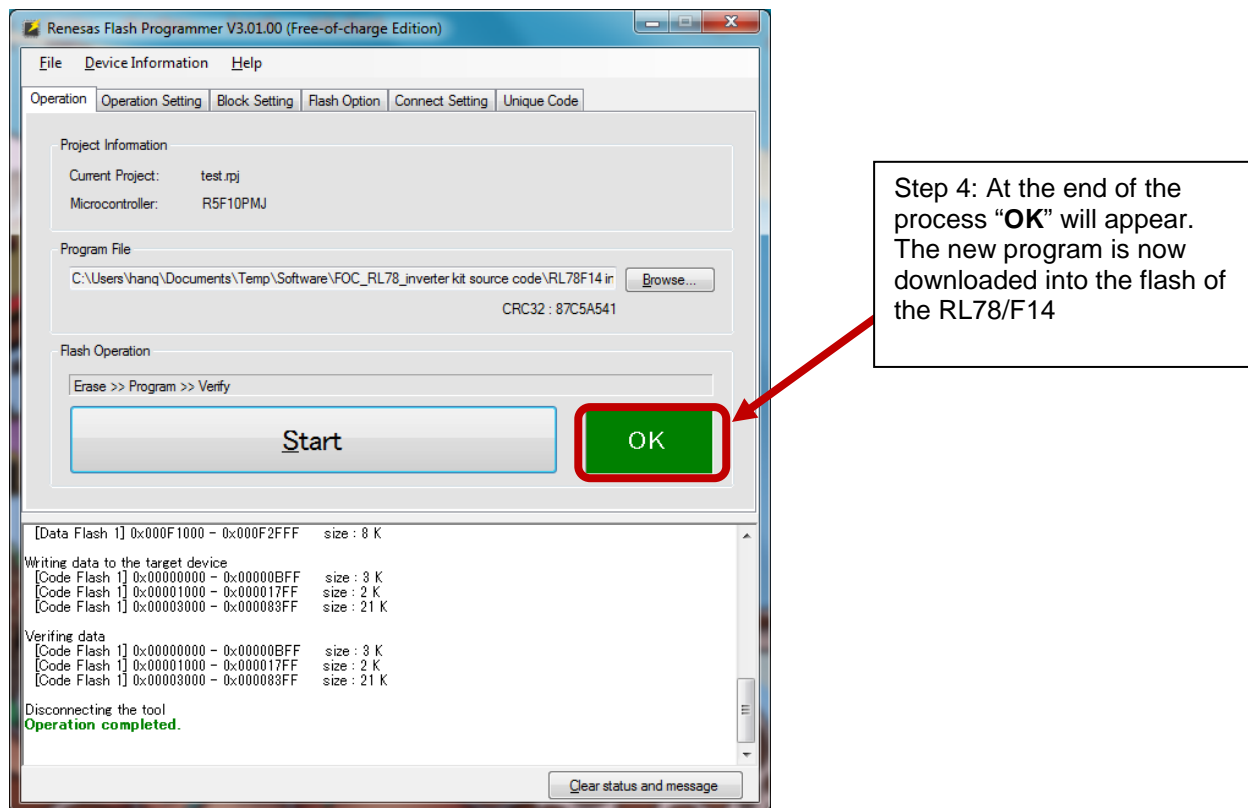
Figure 82. Successful Update the Flash Memory using RFP

Congratulations! The new firmware is now programed into the RL78/F14 flash memory.

22. 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"). I**n the versions with auto-tuning the answer code is ("d").**

The serial communication speed tested is 9.6 KBd.

****

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

****

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

Master String:

l i s o a n D1 .. Dm k

l = frame total length (1 byte)

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

s = station address (1 byte)

o = operation code (1 byte)

a = data address (1 byte)

n = data number (1 byte)

Dx = x-th data byte (1 byte)

k = checksum (1 byte)

Master operation codes:

'c' = check request

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

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

Possible master frames (questions):

check: l ? 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: l   #   s   o   k   (l=5)

check: l   !   s   C   k   (l=5)

word read.:   l   !   s   w   a   n   D11   D10   ..   Dn1   Dn0   k   (l=7+2*n)

word writ.: l   !   s   W   k   (l=5)

ADDRESSES:

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

Operation example:

Example 1

PC request of reading 16 words from the structure UIF_R, starting from the second one (UIF_R.ram_tab[1], .., 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 1st word of data (UIF_R.ram_tab[1]=UIF_R.var.rpm, speed)

7   00 LSB of the 1st word of data

8   00 MSB of the 2nd word of data (UIF_R.ram_tab[2]=UIF_R.var.fre, imposed frequency)

9   00 LSB of the 2nd word of data

10   00 MSB of the 3rd word of data (UIF_R.ram_tab[3]=UIF_R.var.id, d axis current)

11   00 LSB of the 3rd word of data

12   00 MSB of the 4th word of data (UIF_R.ram_tab[4]=UIF_R.var.iq, q axis current)

13   00 LSB of the 4th word of data

14   00 MSB of the 5th word of data (UIF_R.ram_tab[5])

15   00 LSB of the 5th word of data

16   00 MSB of the 6th word of data (UIF_R.ram_tab[6])

17   00 LSB of the 6th word of data

18   00 MSB of the 7th word of data (UIF_R.ram_tab[7]=UIF_R.var.vb, bus voltage)

19   18 LSB of the 7th word of data

20   00 MSB of the 8th word of data (UIF_R.ram_tab[8])

21   00 LSB of the 8th word of data

22   00 MSB of the 9th word of data (UIF_R.ram_tab[9]=UIF_R.var.all, alarm)

23   01 LSB of the 9th word of data

24   00 MSB of the 10th word of data (UIF_R.ram_tab[10])

25   00 LSB of the 10th word of data

26   00 MSB of the 11th word of data (UIF_R.ram_tab[11])

27   00 LSB of the 11th word of data

28   00 MSB of the 12th word of data (UIF_R.ram_tab[12])

29   00 LSB of the 12th word of data

30   00 MSB of the 13th word of data (UIF_R.ram_tab[13])

31   00 LSB of the 13th word of data

32   00 MSB of the 14th word of data (UIF_R.ram_tab[14])

33   00 LSB of the 14th word of data

34   00 MSB of the 15th word of data (UIF_R.ram_tab[15])

35   00 LSB of the 15th word of data

36   00 MSB of the 16th word of data (UIF_R.ram_tab[16])

37   00 LSB of the 16th word of data

38   69 checksum

Example 2

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

0   0F Number of bytes in the frame (0Fh=15dec)

1   3F Master string indicator "?"

2   00 Station address (it is always 0 in our boards)

3   57 word writing operation "W"

4   42 data start address (2(address in UIF_W.ram_tab)+40h(offset to add for ram reading/writing))

5   04 number of data

6   03 MSB of the 1st word of data (value (03E8h=1000dec) to be written in UIF_W.ram_tab[2] = UIF_W.var.rif, speed ref)

7   E8 LSB of the first word of data

8   00 MSB of the second word of data (value to be written in UIF_W.ram_tab[3], not used)

9   00 LSB of the second word of data

10   00 MSB of the third word of data (value to be written in UIF_W.ram_tab[4], not used)

11   00 LSB of the third word of data

12   00 MSB of the fourth word of data (value to be written in UIF_W.ram_tab[5], not used)

13   00 LSB of the fourth word of data

14   E7 checksum

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

0   05 Number of bytes in the frame

1   21 Slave string indicator "!"

2   00 Station address (it is always 0 in our boards)

3   57 word writing operation "W"

4   E6 checksum

Note: Four new operation codes have been added:

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

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

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

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

In these cases the address is a < NUM_PAR.

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

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

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

Figure 83.   Variables in RAM

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

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

Figure 84.   Variable Externs

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

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

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

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

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

Figure 85. Special Working Modes (1)

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


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


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


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


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


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

Figure 86. Special Working Modes(2)

23. Revision History

<table>
<tr><td colspan="4" align="center">RL78/F14 BLDC Starter Kit User Manual: Hardware</td></tr>
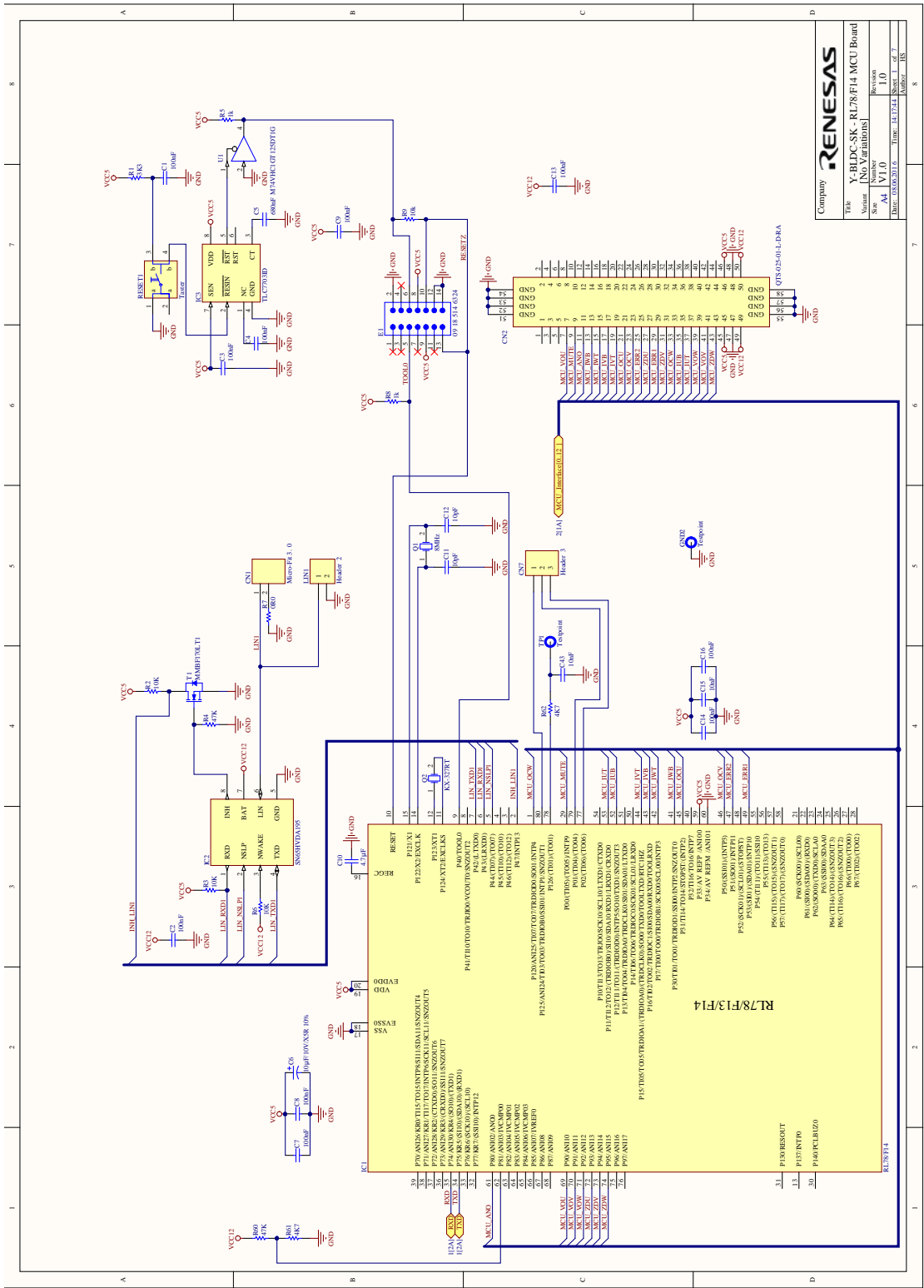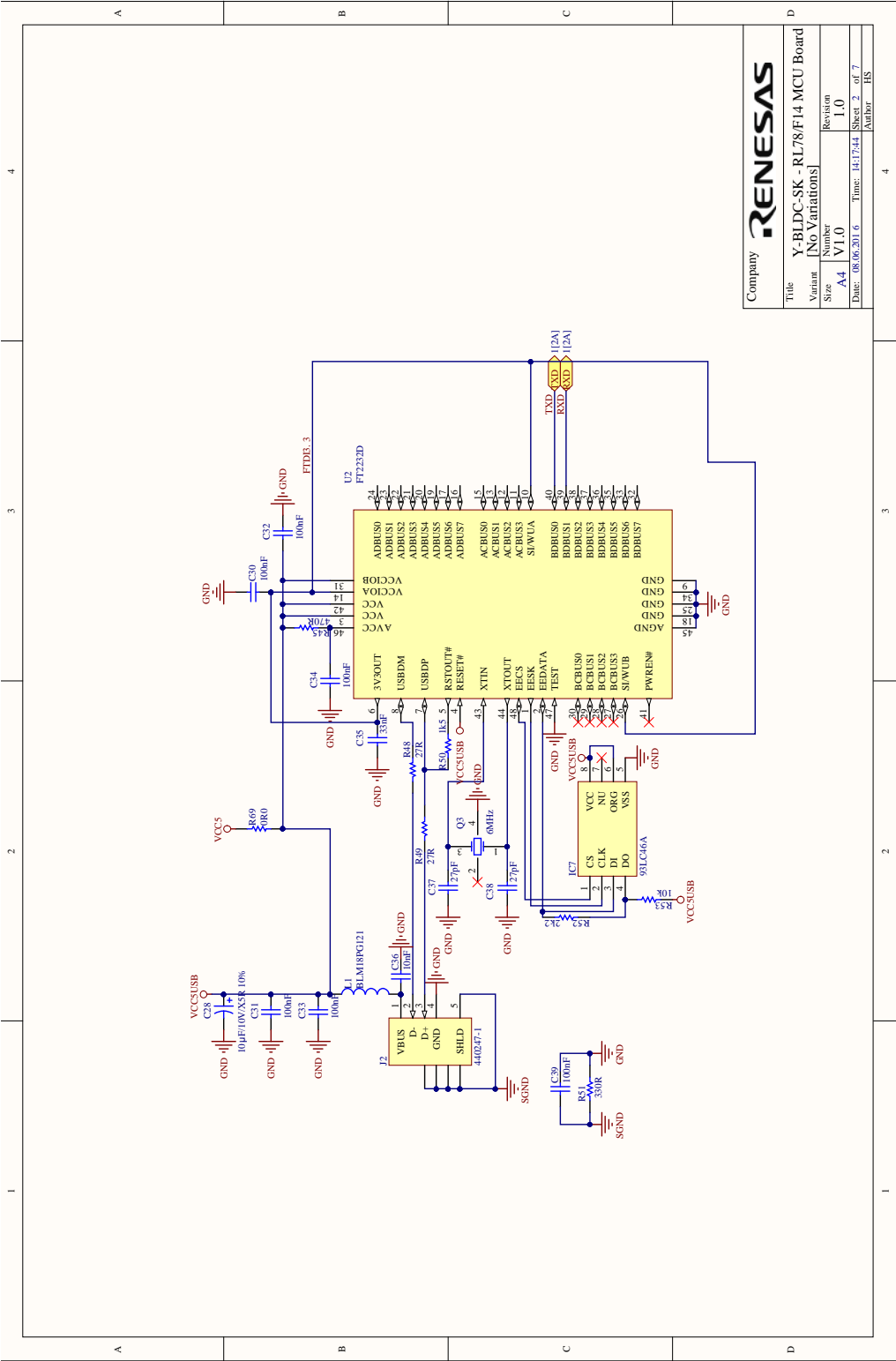<tr><td rowspan="2">Rev.</td><td rowspan="2">Date</td><td colspan="2" align="center">Description</td></tr>
<tr><td align="center">Page</td><td align="center">Summary</td></tr>
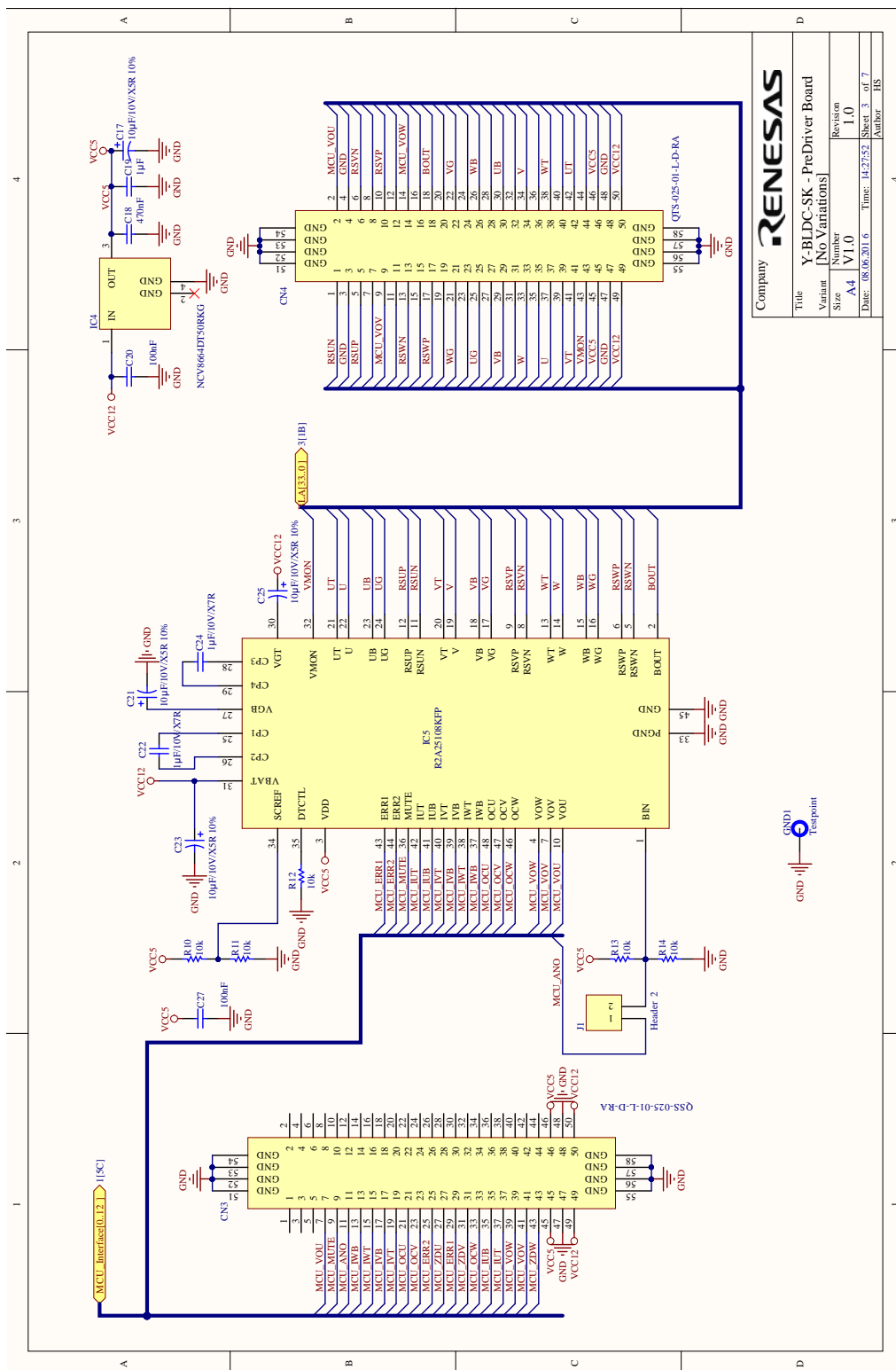<tr><td>1.0</td><td>June 17, 2016</td><td align="center">—</td><td>First edition issued</td></tr>
<tr><td>1.1</td><td>June 19, 2017</td><td align="center">58</td><td>Corrected description of Figure 53, fix of typing errors</td></tr>
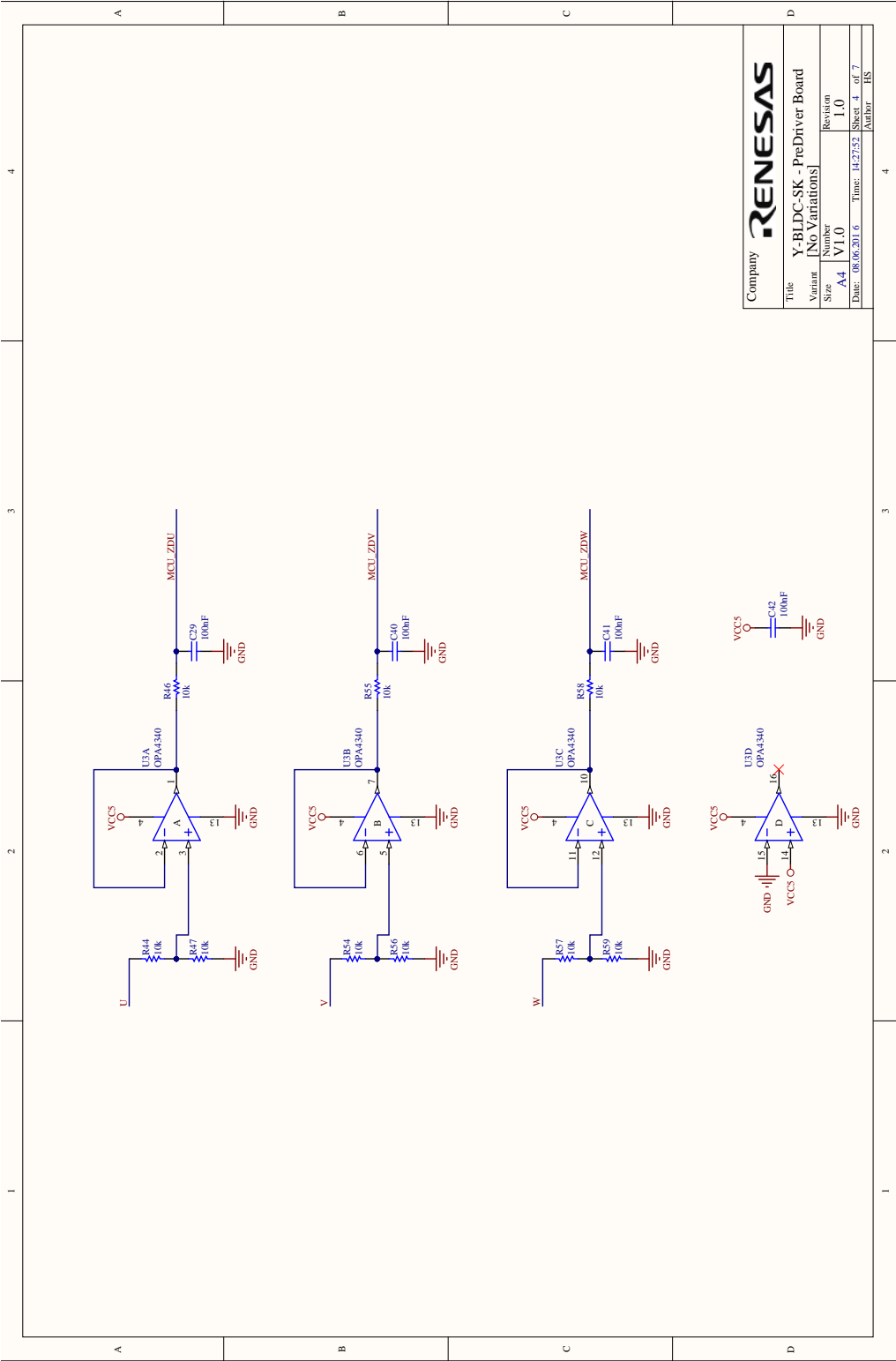<tr><td>1.2</td><td>June 20, 2017</td><td align="center">92</td><td>Revision History completed, fix bug in PDF creation</td></tr>
</table>

24. Appendix A: Schematic

RL78/F14 BLDC Starter Kit User Manual: Hardware

Publication Date: Rev. 1.02   June 20, 2017

Published by: Renesas Electronics Europe

RENESAS

Renesas Electronics Corporation

http://www.renesas.com

SALES OFFICES

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

RL78/F14 BLDC Starter Kit

RENESAS

Renesas Electronics Corporation

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[Renesas Electronics](#):
   [Y-BLDC-SK-RL78F14](#)