Renesas ASSP EASY Motor Control Solution

Based on RISC-V

Introduction

The Renesas ASSP EASY Motor Control Solution is developed and available for the evaluation kit based on the R9A02G0204 RISC-V ASSP.

The kit enables engineers to easily test and evaluate the performance of the ASSP in a laboratory environment when driving any 3-phase Permanent Magnet Synchronous Motor (e.g., AC Brushless Motor) using an advanced sensor-less Field Oriented Control (FOC) algorithm. Typical applications for this type of solution are compressors, air conditioners, 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.

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

The board can be powered directly from the USB port of a Host PC for demo purpose (with power supply limitations), but connectors are provided to utilise an external power supply to evaluate and test the solution for higher power requirements.

The kit can also be extended to connect to additional power stage expansion boards (supporting low voltage-high current and high voltage configurations).

The evaluation kit is an ideal tool to validate all the key performance parameters of the target motor in preparation to the end application system design.

Target Device: R9A02G0204

Contents

1. Overview

The “ASSP EASY for motor control” evaluation kit is composed by the R9A02G0204 ASSP board complete with connectors, USB TYPE C cable, 24Vdc power supply and PMS Motor, as shown in Figure 1 and Figure 2.

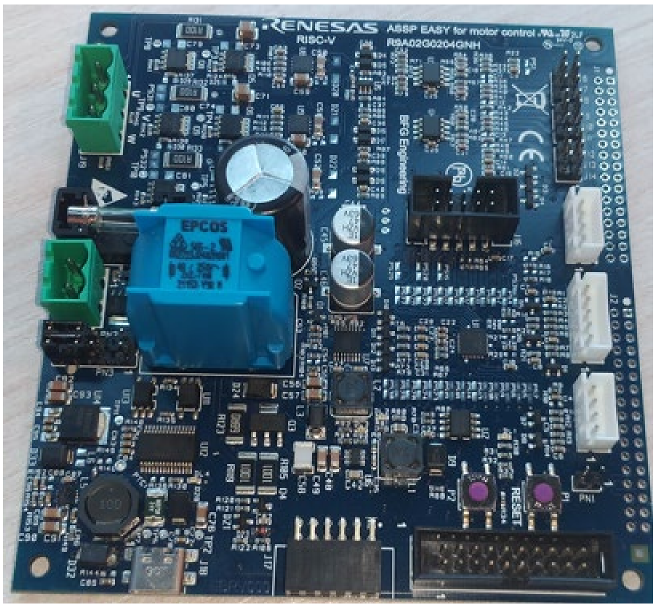


Figure 1: ASSP inverter board.

Figure 2: ASSP solution components

2. Specification

Type of motors supported	3-phase Permanent Magnet Synchronous (PMSM, PMAC, BLAC) 3-phase Brushless DC (BLDC)
KIT Motor part names	NANOTEC DB42S03 or DB42M03, 24V_{DC}, 4000 RPM or Speeder Motion MB57GA240 or Fulling Motor FL28BL38-HS
Kit Max input range	External power supply from: 20V_{DC} to 48V_{DC}, A_{peak} TBD
Transistor used	Renesas Mosfets: 25A 80V 13mOhm RJK0854DPB LFPAK
Power Supply Option	Either USB connection or external supply
Current detection	One or three shunts configuration (100mΩ)
USB IC used on the board	FT232R - USB UART IC from FDTI, 76.6KBd communication speed
Microcontroller	ASSP EASY for motor control: R9A02G0204GNH [48K flash, 16K ram, 32-pin HWQFN, 32MHz, Ta: -40 ÷ +125°C]
MCU Performance	32MHz
Key features	12-bit A/D Converter, fast on-chip PGA
MCU embedded Firmware	Sensor-less vector control algorithm (Field Oriented Control)
Switching frequency	4KHz to 64KHz, 16KHz by default (PWM frequency)
Control Loop Frequency (sampling frequency)	4KHz to 16KHz, 8KHz by default
Control loop timing	28µs with motor not driven 92µs with motor driven
Code size in FLASH / RAM	28KB flash / 2.9KB parity SRAM / 1KB ECC SRAM
Tool used, version	SEGGER Embedded Studio for RISC-V V6.30
Compiler optimization level	OS3
Environment standards	RoHS compliant including China regulations WEEE, RoHS

3. Hardware introduction

The ASSP demo-board is the ideal tool in order to test the motor control algorithms, thanks to the powerful and optimized ASSP hardware features and peripherals, and the flexibility of the kit designed to be adapted to different algorithm implementations and system configurations by simply re-configuring a variety of jumpers or solder points.

4. Block diagram

The next figure shows the block-diagram of the kit.

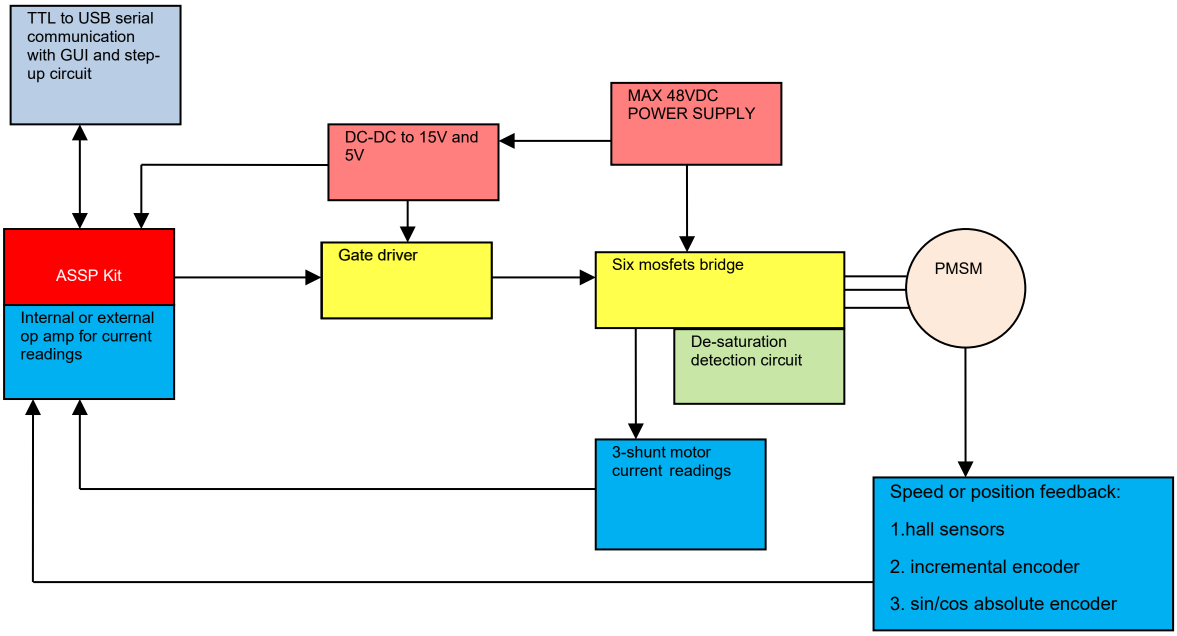


Figure 3 shows where the individual components identified in the block diagram are located on the board.

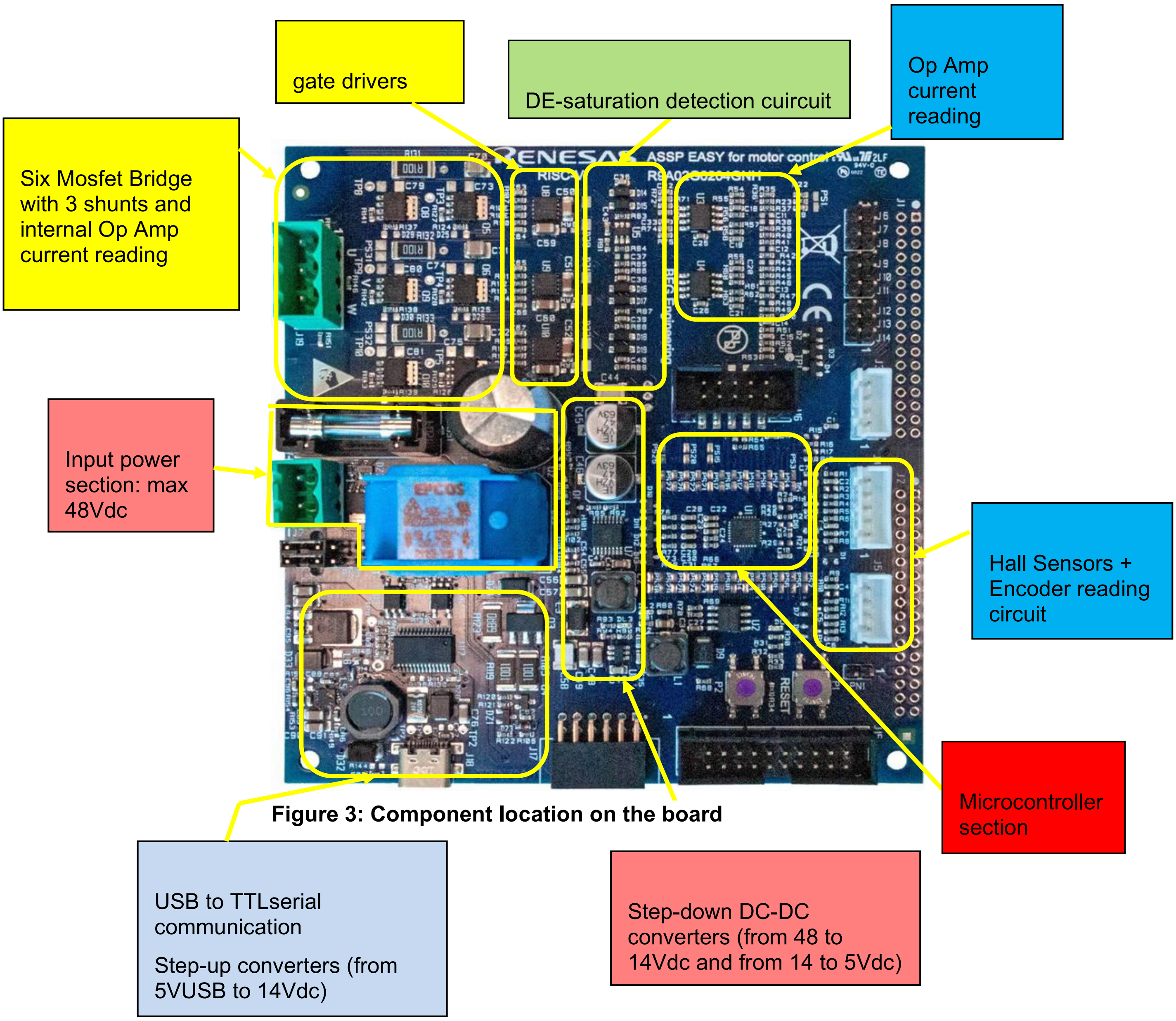
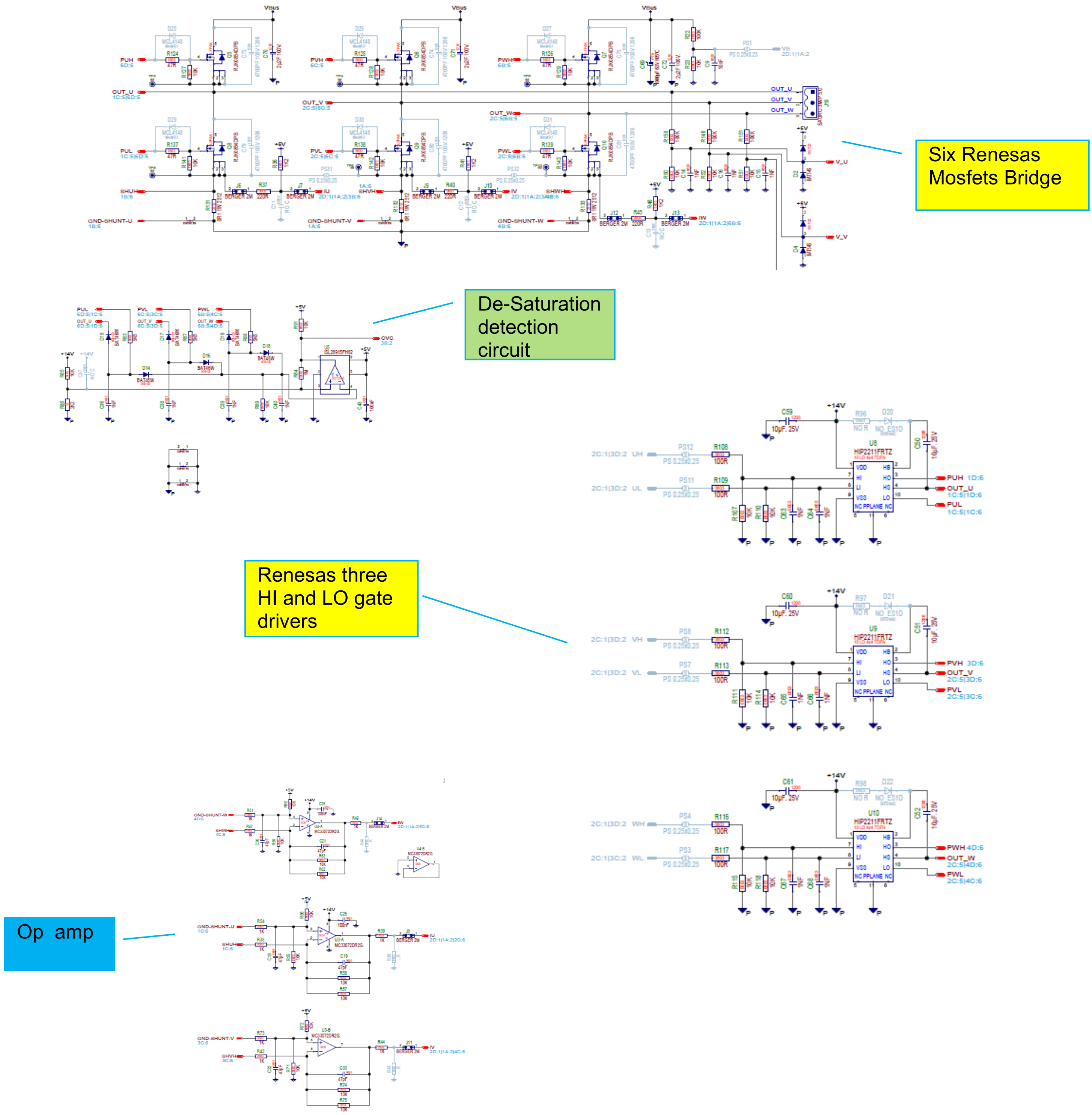
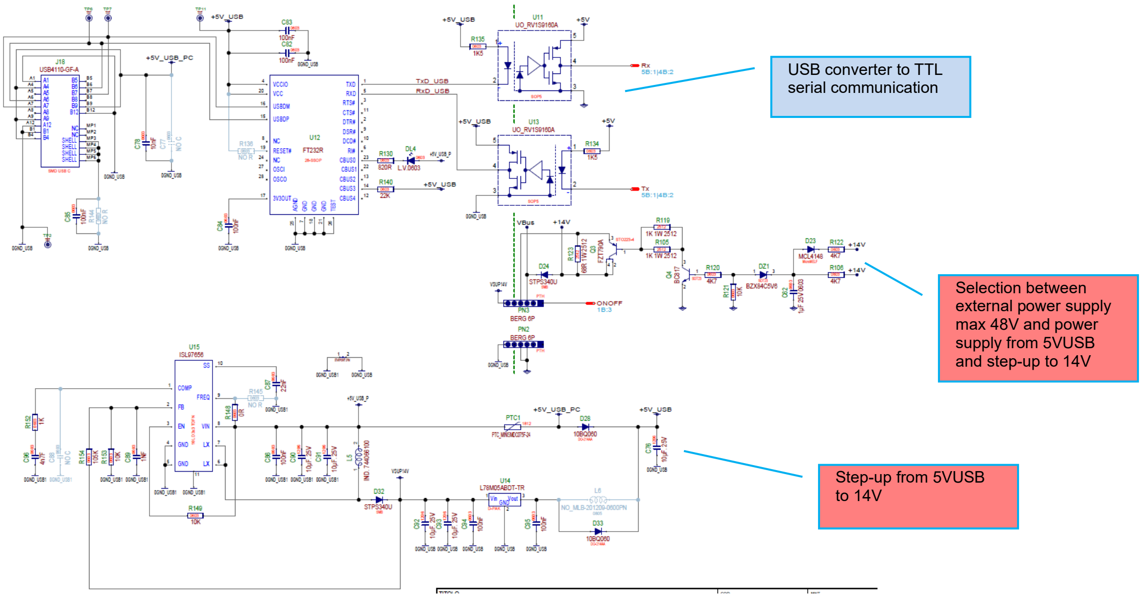
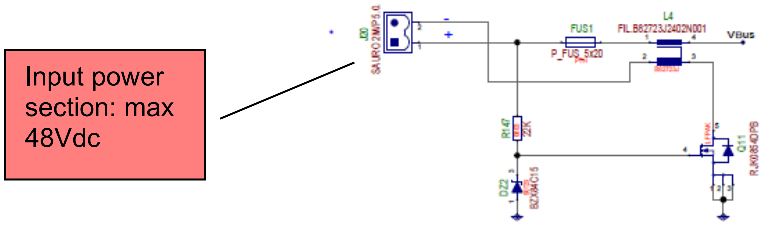


Figure 3: Component location on the board

Following an excerpt of the schematic diagram highlighting the individual sections related to the above mentioned components.


Six Renesas
Mosfets Bridge
De-Saturation
detection
circuit
Renesas three
HI and LO gate
drivers
Op amp


USB converter to TTL serial communication
Selection between external power supply max 48V and power supply from 5VUSB and step-up to 14V
Step-up from 5VUSB to 14V


Input power section: max 48Vdc
FUS1
P_FUS_5x20
L4
VBus
R147
DZ2
Q11

5. Renesas ASSP, analog and power components cross-references

Description	Renesas part number
RISC-V ASSP	R9A02G0204GNH
4.5V to 72V, 2A, DC/DC Synchronous Step-Down Regulator with Internal Compensation and Programmable Frequency	RAA211820GSP
45V 1A 630kHz DC/DC Step-Down Regulator	RAA2114124GP3
100V, 3A Source, 4A Sink, High Frequency Half-Bridge Gate Drivers	HIP2211FRTZ
Comparator	ISL28915FH62
Mosfet 80V, 25A	RJK0854DPB
Optocoupler	RV1S9160A
Step-Up Regulator with 4A Integrated Switch	ISL97656

6. Jumper configuration descriptions

6.1 Power supply selection

The board can be supplied in two ways. The selection between the two modalities is made with the two jumpers PN2 and PN3 (refer to below Figure 4).

i. The first way is by using a USB C type cable (included in the kit). Thanks to the step-up circuit, a 14V VBUS voltage will be generated. Those conditions are enough to run the motor but are not recommended for motor tuning (use the included external power supply for tuning).

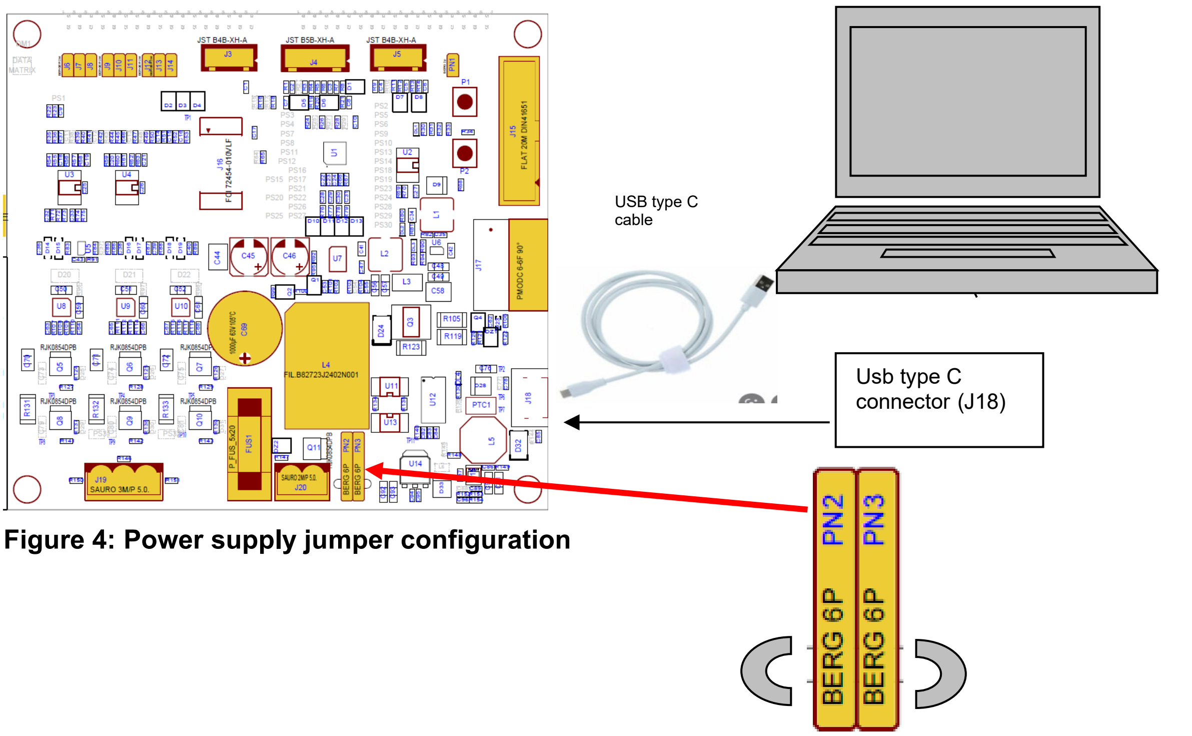
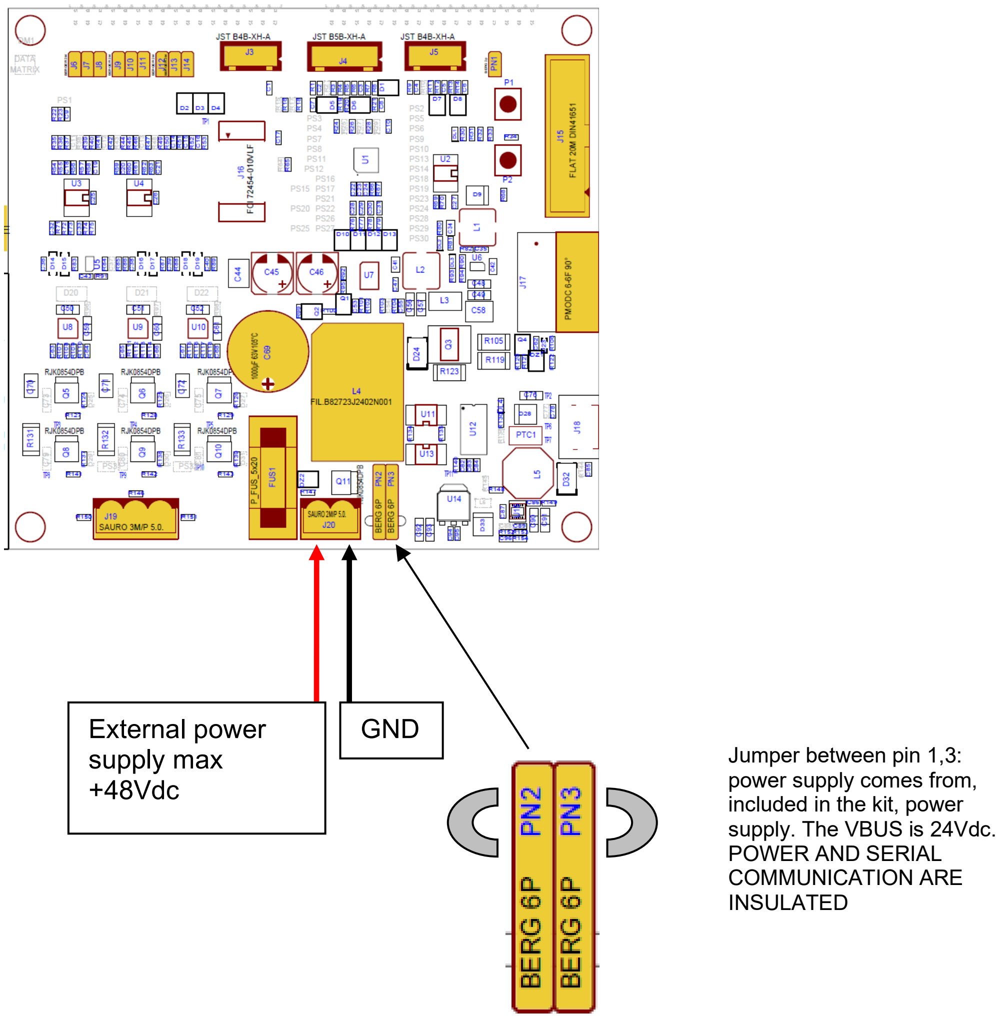


Figure 4: Power supply jumper configuration

Jumper between pin 4,6: power supply comes from USB step-up. The VBUS is 14Vdc. NO ISOLATION BETWEEN POWER AND SERIAL COMMUNICATION

ii. The second way is by using an external power supply (included in the kit).



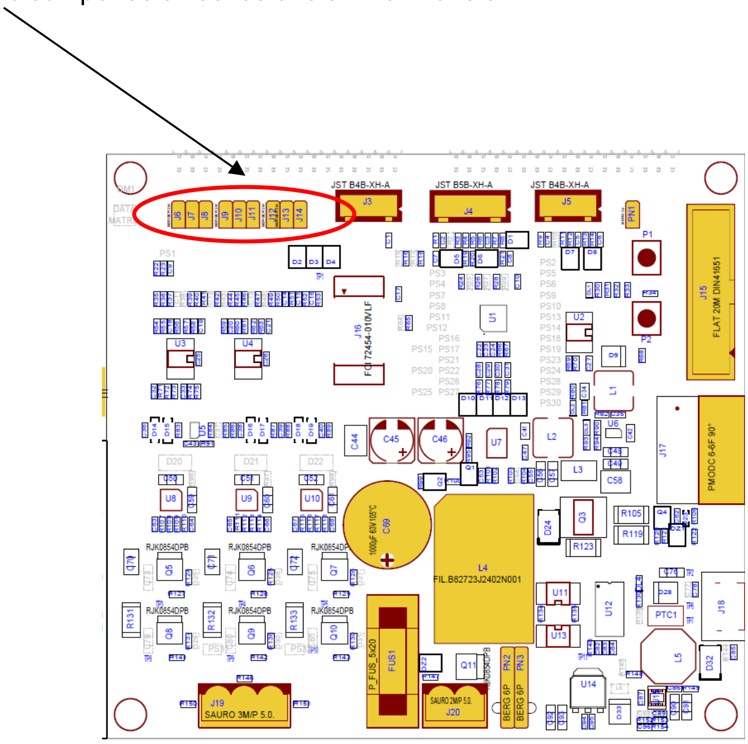
Important Note: caution on Isolation Voltage

The second Jumper configuration is intended to isolate a small and electrically safe GND gap between the user system and user-accessible circuitry. In any case isolation voltage must be maintained within SELV limits i.e. less than 40VAC, or 60VDC.

The insulated part cannot be treated as safety isolation system. The part could be expected to function correctly at higher voltage across the isolation barrier; but then the circuitry on both sides of the barrier must be regarded as operating at an unsafe voltage and further isolation/insulation systems must form a barrier between these circuits and any user-accessible circuitry according to safety standard requirements.

6.2 Operational amplifier / PGA selection (external or on-board in the ASSP)

Refer to Jumper J6 J7 J8 J9 J10 J11 J12 J13 J14.



i. ASSP on-board PGA:

- close jumpers J6, J7, J9, J10, J12, J13
- open jumpers J8, J11, J14

ii. External operational amplifier:

- open jumpers J6, J7, J9, J10, J12, J13
- close jumpers J8, J11, J14

7. Solder joints description (see kit schematics)

Selection between different configurations is made with the help of some solder joints.

Figure 5 details their location on the kit and below a list the configuration options to implement the desired connections.

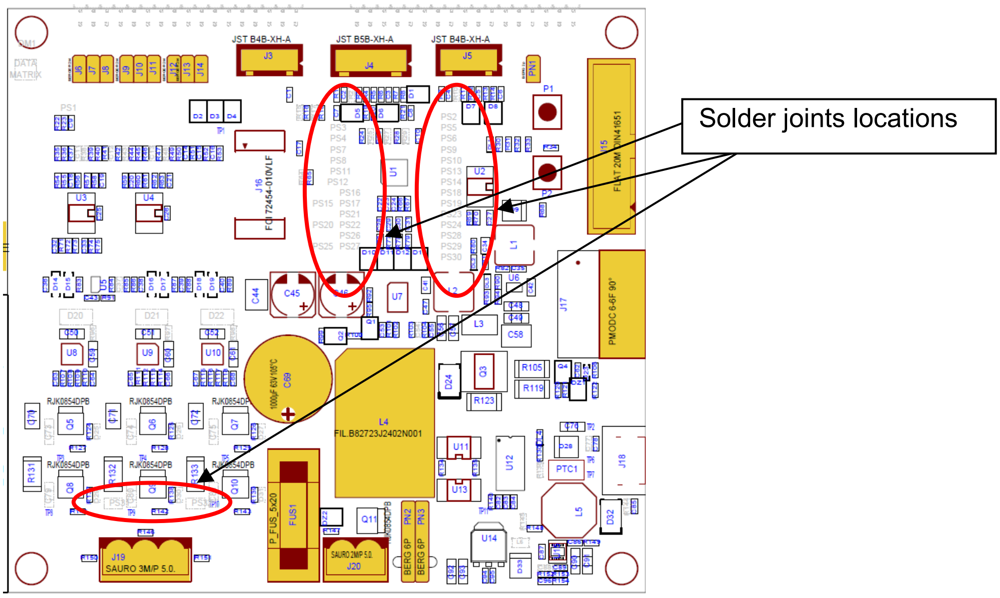


Figure 5: solder joints configuration

Connection A: sin/cos absolute encoder

- close PS16, PS21, PS26
- open PS17, PS22, PS27

Connection B: motor voltages

- open PS16, PS21, PS26
- close PS17, PS22, PS27

Connection C: PMOD

- close PS2, PS6, PS13, PS18, PS24, PS30
- open PS5, PS25, PS29, PS15, PS10, PS14, PS20, PS19, PS23, PS28, PS29

Connection D: jlink SEGGER emulator

- close PS10, PS14, PS23, PS29
- open PS30, PS28, PS19, PS24, PS18, PS20, PS13

Connection E: internal inverter

- close PS1,3,4,7,8,11,12
- select MCU internal opamp or external opamp for current reading

8. Jumper, Solder Joints and 0ohm resistors Default configuration

Following the description of the default configuration for the demo kit as shipped (jumpers, 0-Ohm resistors and solder joints).

8.1 Resistors

R15,17,64,25,27,29 = 0ohm

R65,16,18,24,26,28 = DO NOT MOUNT

This configuration to get the correct pwm commands

8.2 Solder joints

PS10,14,23,29,15,25 = shorted

PS16,17,21,22,26,27,20,2,5,6,9,13,18,19,24,28,30,31,32 = open

This configuration to get:

- SEGGER emulator connections
- Three shunt
- EEPROM connections

8.3 Jumpers

J6,7,9,10,12,13 shorted

J8,11,14 = open

This configuration to get:

- MCU internal opamp
- On board inverter

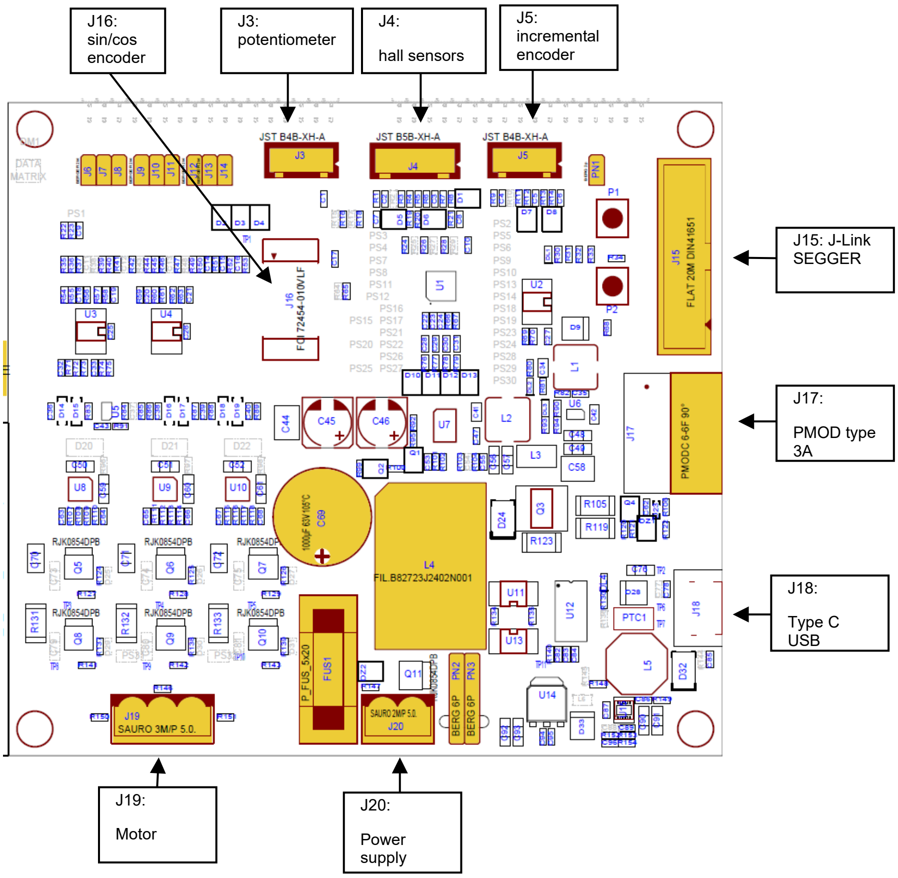
8.4 Solder joints

PS1,3,4,7,8,11,12 shorted

PN1 = open

For details please see chapter 6 and the schematics.

9. Connectors description



Pinout description

J3: potentiometer

- J3-1 +5vDC
- J3-2 brush
- J3-3 gnd
- J3-4 n.c.

J4: hall sensor

- J4-1 gnd
- J4-2 W hall signal
- J4-3 V hall signal
- J4-4 U hall signal
- J4-5 5Vdc

J5: incremental encoder

- J5-1 gnd
- J5-2 B encoder signal
- J5-3 A encoder signal
- J5-5 5Vdc

J15: 20pin J-Link SEGGER

J16: sin/cos encoder

J17: PMOD TYPE 3A

J18: USB TYPE C

J19: motor

- J9-1 U motor
- J9-2 V motor
- J9-3 W motor

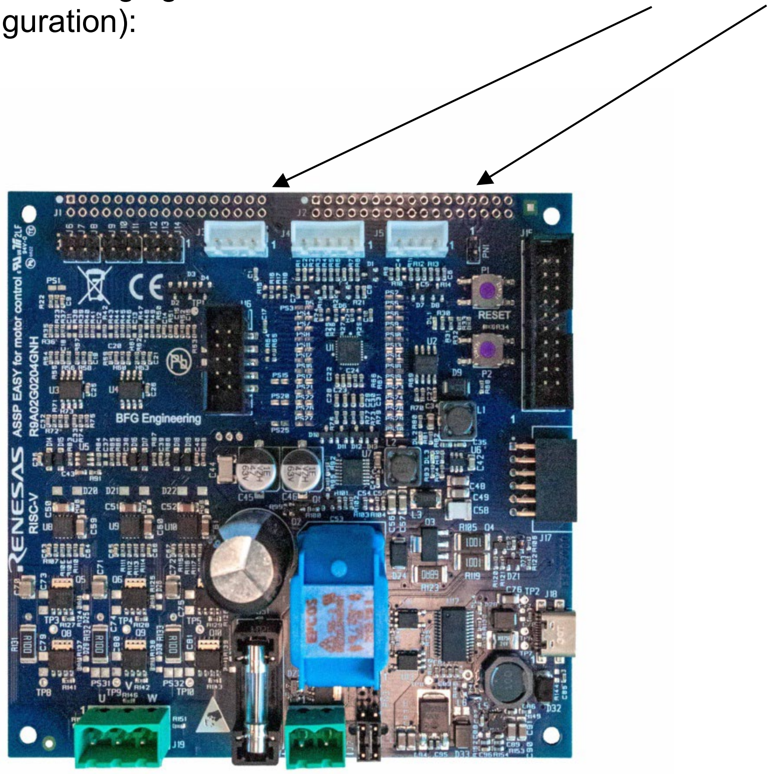
J20: power supply

- J20-1 max +48Vdc power supply
- J20-2 gnd

10. External power stages connections

The ASSP demo board can be connected to some external power stages with different capability.

As shown in the following figure, the interface connectors are J1 and J2 (not mounted in the default configuration):



POWER STAGES LIST:

- **MCI-LV-3**: max 48Vdc power input, equipped with H7N1002LS, 100V, 75A mosfets, MCU or external opamp for current readings, three or single-shunt selection
- **MCI-LV-2**: max 48Vdc power input, equipped with H7N1002LS, 100V, 75A mosfets and RAA227063 Smart Pre-Driver.
- **E6140:** max 350Vdc power input, equipped with 600V, 15A intelligent power module

11. Application software

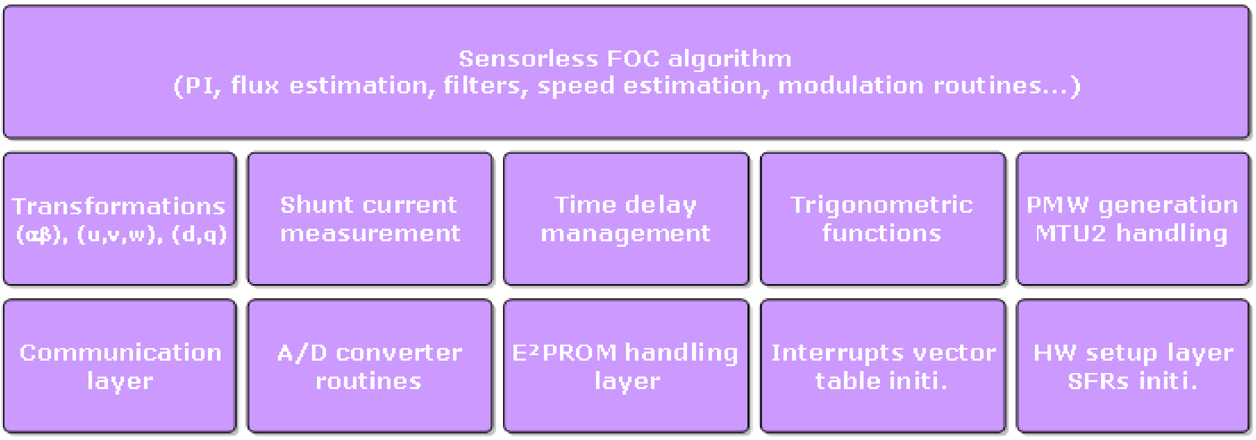
11.1 Tool Chain

The embedded software delivered pre-programmed in the ASSP has been developed using SEGGER Embedded Studio for RISC-V v. 6.30 and GCC compiler.

11.2 Software description and resources used

- The software delivered in the kit, previously described, is working on the RISC-V based ASSP clocked at 32MHz, and its operating voltage is 5V which guarantees a high noise immunity.
- Using the interrupt skipping function it is possible to regulate separately the PWM frequency (Pulse Width Modulation) and the sampling frequency (also called control loop frequency). For instance, when the PWM frequency is set to **16KHz** and the control loop is set to **8KHz**, such ratio of 2 means that the full vector control algorithm is processed every two PWM cycles.
- Finally, the main interrupt is called at the control loop rate which leaves enough time to perform the sensor-less vector control algorithm and the system control if and as needed.

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



The complete software requires the resources below in the three shunts configuration. It includes the serial communication interface, the board management, the LED management, the E^{2}prom/Data-flash parameter management, the auto-tuning algorithm and self-identification, and the complete sensor-less vector control algorithm. 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 phase of evaluation and configuration of the software.

Control Loop Frequency (sampling frequency)	4KHz to 16KHz, 8KHz by default
Control loop timing	28µs with motor not driven 92µs with motor driven
Code size in FLASH / RAM	28KB flash / 2.9KB parity SRAM / 1KB ECC SRAM
Tool used, version	SEGGER Embedded Studio for RISC-V V6.30
Compiler optimization level	OS3

11.3 Project organization

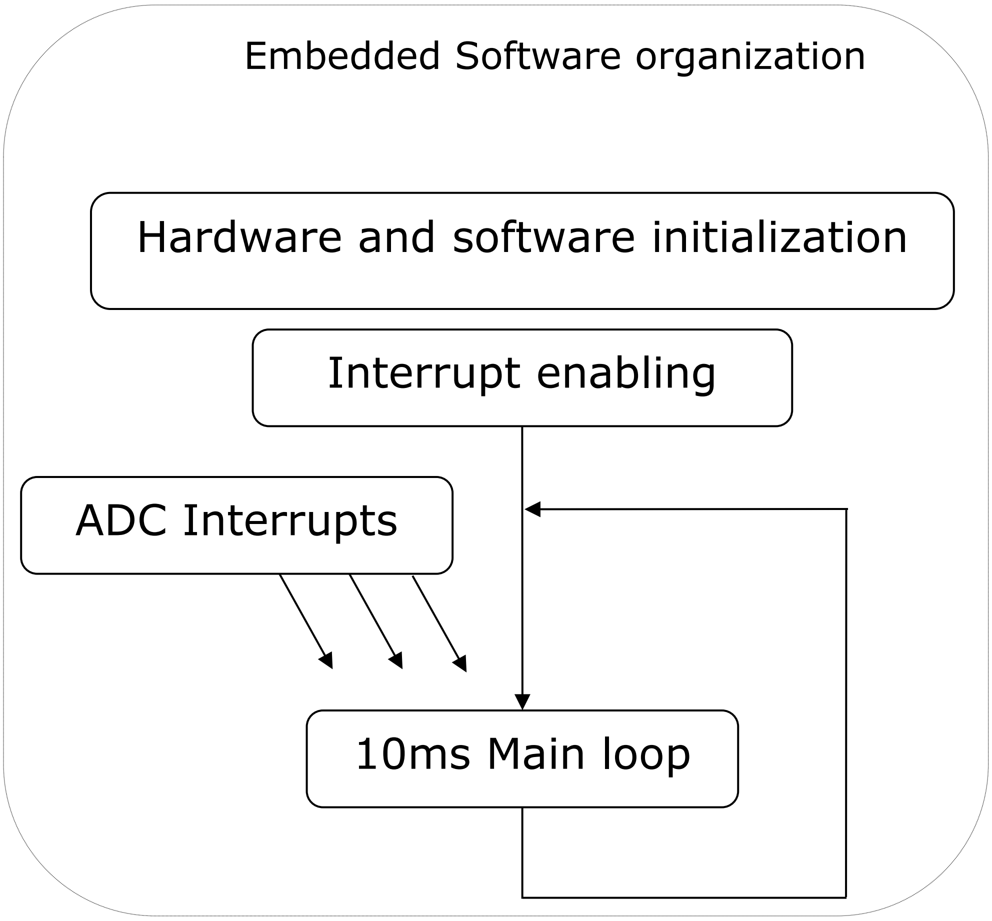
This section is provided as a reference to explain the overall software architecture. This might be helpful to understand the application software organization, although there is no requirement for the end user to perform any specific development. For any necessary customization or inquiry about the software implementation, please contact BFG engineering.

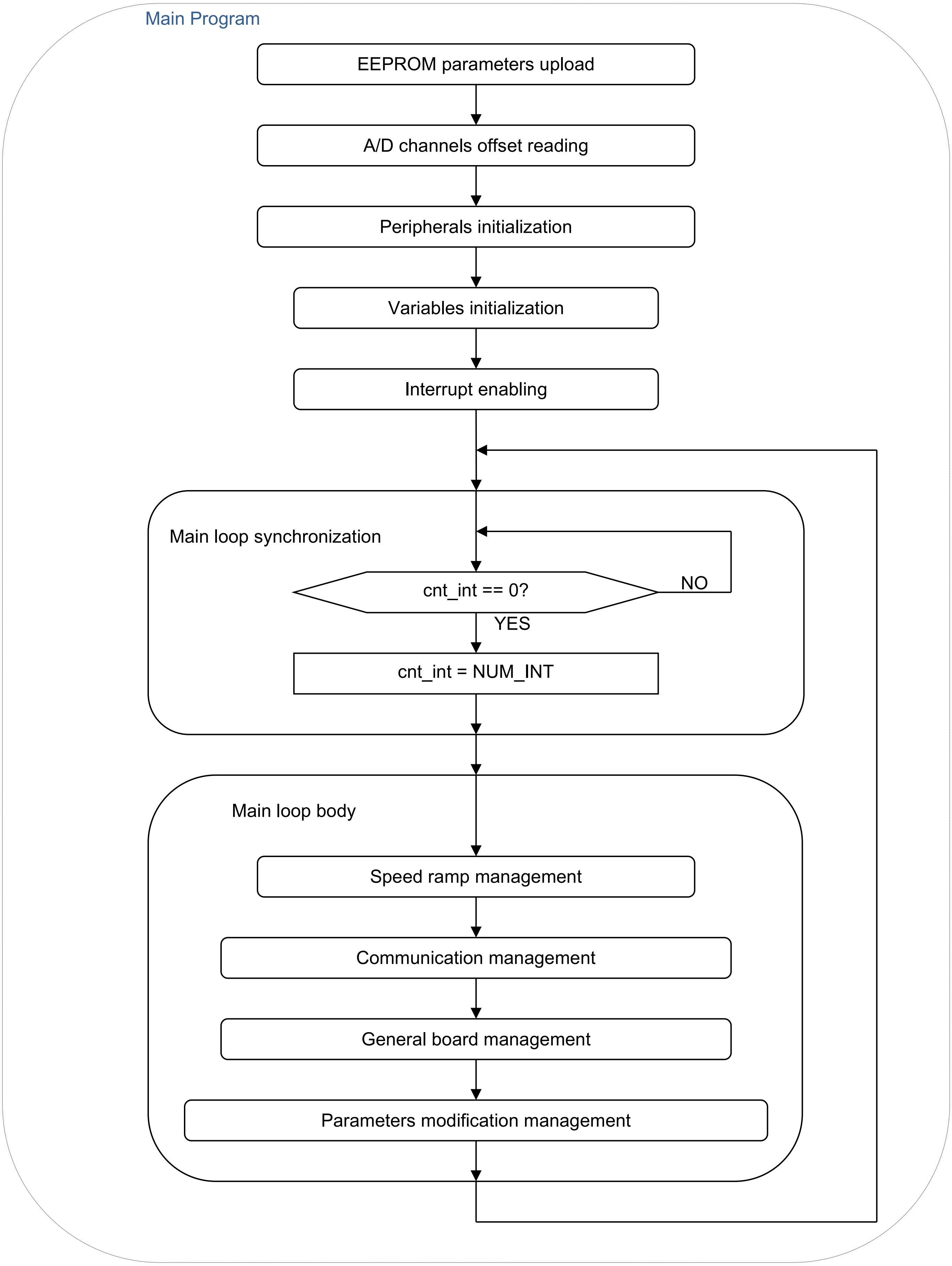
src/
library/: contains sensorless estimator, auto tuning and motor identification routines
smc_gen/ : code including drivers for various IPs

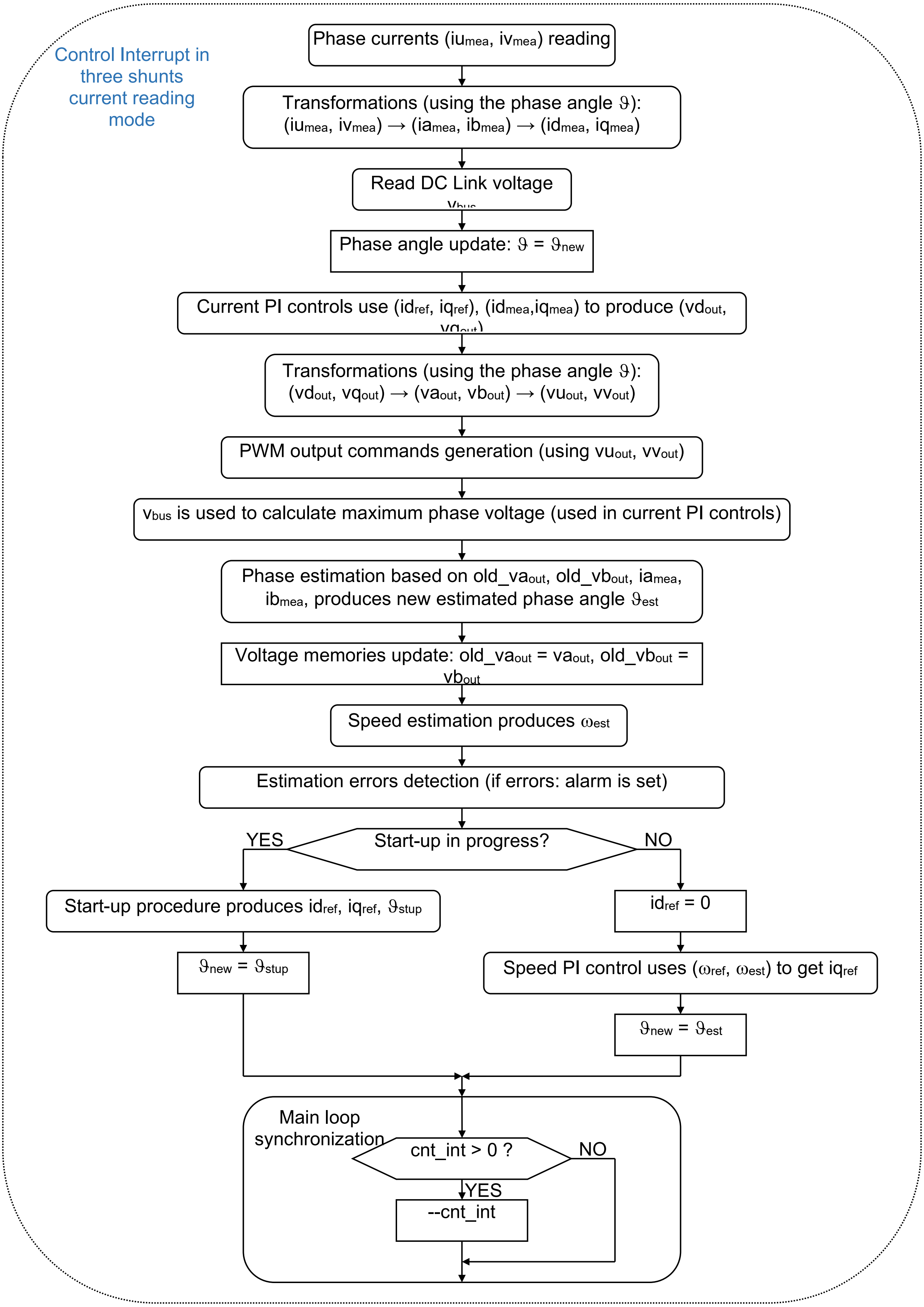
customize.h: defines which can be used by the developer to enable/disable some program features, and/or some constants which have not been included in the EEPROM configuration parameters (any modification requires a re-compilation of the project).
const_def.h: definition of the basic constants used in the project.
ges_eqp.c: eeprom management routines
ges_eqp.h : eeprom management header file
defpar.h: here the macro referred to the default values of the parameters are placed. Here the developer can modify the default values of the parameters; this file can be automatically generated by the GUI.
globdef.h: definition of general utility macro, referred to peripherals, I/O etc.
globvar.h: definition of general gloabal variables.
hardware_setup.c: Hardware_setup file
hardware_setup.h: Header for hardware_setup.c
interrupt.c: Interrupt management file
interrupt.h: Header for interrupt.c
main.c: "high level" startup, the first initializations and the main management loop.
mask.h: definition of general utility bitmasks.
motorcontrol.c: in this file the function directly related to motor-control are included, for example the main control interrupt.
motorcontrol.h: interface definitions for the use of the functions contained in motorcontrol.c.
ofs_id.c: ofs and id code settings
param_man.h: here the data-structure used for the EEPROM parameters management is defined.
param_def.h: macros are used in the previous file (to modify the maximum and minimum values of all the configuration parameters).
typedefine.h: definition of custom types used in the project (included MISRA types).
pws_EBRV000.h: hardware related defines, referred to the on-board low voltage powers stage (for example the gains of the current reading circuit and other hardware-related constants).
userif.c: this file contains the code which manages the communication protocol with the GUI.
userif.h: interface definitions for the use of the functions contained in userif.c.
R9A02G020.h: register interface header file
risc-v.h: architecture specific header
units.h: internal representation of physical variables

11.4 Program Flow

At startup, the ASSP software takes care of all the necessary initializations. The PWM timer starts an ADC conversion on every PWM through; when the conversion is finished, end-of-conversion interrupts are generated.
In the callback function the motor control functions are managed and a synchronization timer to release the main loop is managed. The main thread at every wake-up check the counter and every 10ms performs one complete cycle. In this way the operating system becomes completely transparent to the user. So, we can examine the program flow considering the three following graphs.




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


Control Interrupt in three shunts current reading mode
Phase currents (iumea, ivmea) reading
Transformations (using the phase angle ϑ): (iumea, ivmea) → (iamea, ibmea) → (idmea, iqmea)
Read DC Link voltage Vbus
Phase angle update: ϑ = ϑnew
Current PI controls use (idref, iqref), (idmea,iqmea) to produce (vdout, vqout)
Transformations (using the phase angle ϑ): (vdout, vqout) → (vaout, vbout) → (vuout, vvout)
PWM output commands generation (using vuout, vvout)
vbus is used to calculate maximum phase voltage (used in current PI controls)
Phase estimation based on old_vaout, old_vbout, iamea, ibmea, produces new estimated phase angle ϑest
Voltage memories update: old_vaout = vaout, old_vbout = vbout
Speed estimation produces ωest
Estimation errors detection (if errors: alarm is set)
YES
Start-up in progress?
NO
Start-up procedure produces idref, iqref, ϑstup
idref = 0
ϑnew = ϑstup
Speed PI control uses (ωref, ωest) to get iqref
ϑnew = ϑest
Main loop synchronization
cnt_int > 0 ?
NO
YES
--cnt_int

The interrupt frequency is linked to the PWM frequency (since the interrupt is generated by the end of an A/D conversion launched at PWM trough). It is possible to choose (inside certain limits) the sampling frequency and the ratio between the PWM frequency and the sampling frequency. (In fact, using the interrupt skipping function the user can have a sampling frequency which is a submultiple of the PWM frequency, for example when a particularly high PWM frequency is required).

The embedded software is by default set to 8KHz sampling frequency, i.e. 125µs for the sampling period and the PWM frequency is set to 8KHz. Such parameters can be modified using the PC GUI without recompiling the overall project and changing the parameters #19 and #20 of the “parameters table” here below and resetting the board.

The parameter #19 set the control loop speed. If 8KHz is selected by entering the value “8000”, the PWM frequency can be set to four different values depending on the motor and the applications either 8KHz, 16KHz, 24KHz or 32KHz.
It’s basically done by entering the ratio value in the parameter #20. Please find below the possible values that can be entered.

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

11.5 Dataflash Parameters Table

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

Parameter number	Short name	Description
0	SEL_OP	Default parameters setting, Used to perform special operations, like default parameter set re-loading, or current PI tuning working mode setting
1	RPM_MIN	Set the Minimum Speed in RPM
2	RPM_MAX	Set the Maximum Speed in RPM
3	R_ACC	Set the acceleration [RPM/s]
4	R_DEC	Set the deceleration [RPM/s]
5	C_POLI	Set the number of polar couples
6	I_START	Set the start-up current (peak) [Ampere/AMP_RES]. Used to specify the peak phase current value to be used during the start-up
7	I_MAX	Set the maximum phase current (peak) [Ampere/AMP_RES]
8	R_STA	Set the stator resistance [Ohm/OHM_RES]
9	L_SYN	Set the synchronous inductance [Henry/HEN_RES]
10	PM_FLX	Set the permanent magnets flux [Weber/WEB_RES]. This value is only used when the exact integration flux estimation algorithm is selected. By default, it’s not needed as the approximated integration is selected.
11	KP_CUR	Set the Current loop Proportional coefficient: KP
12	KI_CUR	Set the Current loop Integral coefficient: KI
13	KP_VEL	Set the Speed loop Proportional coefficient: KP
14	KI_VEL	Set the Speed loop Integral coefficient: KI
15	FB_GAIN	Set the flux amplitude feedback gain. This value is only used when the exact integration flux estimation algorithm is selected. By default, it’s not needed as the approximated integration is selected
16	PHA_OFF	Set the phase offset [deg]. It is used to add a phase offset to the phase estimation, to reach better alignment
17	ST_TIM	Set the Start-up acceleration time [sec/SEC_RES]
18	FL_FTAU_DEF	Filter time constant [ms]. Only needed if the approximated integration flux estimation algorithm is chosen as by default. If the exact integration method is selected, this value is not used.
19	SAM_FRE_DEF	Set the sampling frequency [Hz] of the control loop
20	F_RATIO_DEF	Set the ratio between the PWM frequency and sampling frequency, e.g. if 8000 is set in the parameter #19 and 2 in the parameter #20, the PWM frequency is 16KHz.

Here below the parameters list for the MB57GA240 or FL28BL38-HS motor included in the KIT

n. par **value**

MB57GA240			FL28BL38-HS	
Num par.	**value**		**Num par.**	**value**
Par. n.0	0			0
Par. n. 1	310			300
Par. n. 2	3000			3000
Par. n. 3	3000			4000
Par. n. 4	3000			2000
Par. n. 5	2			2
Par. n. 6	500			500
Par. n. 7	1500			1500
Par. n. 8	62			228
Par. n. 9	3			13
Par. n. 10	180			48
Par. n. 11	208			30
Par. n. 12	1109			1300
Par. n. 13	2000			500
Par. n. 14	400			200
Par. n. 15	100			10
Par. n. 16	0			0
Par. n. 17	1000			1000
Par. n. 18	64			64
Par. n. 19	8000			5000
Par. n. 20	2			2

12. APPENDIX A

12.1 Internal representation of physical variables

The idea which lies under the internal representation of physical variables is to maximize the resolution, keeping as simple as possible the calculations and keeping reasonably low the memory occupation. So, whenever it had been possible, the physical variables have been represented under a “per unit” criteria.
Please find below the description of the representation for each physical quantity.

Angles
The interval [0, 2pi) is represented with the interval [0, 65536), with the resolution of 2pi/65536 rad.

Angle[internal_angle_unit] = KA * Angle[rad]
KA = 32768 / pi (= 10430.37835)

Note that in this way the angle can be considered unsigned in the range [0, 65536), or signed in the range [-32768, 32768), with identical results. In every case the representation requires a 16bit word.

Trigonometric functions
sin(a), cos(a) are normalized to the value NORMVAL = 16384.

Internal_sin(a[internal_angle_unit]) = NORMVAL * sin(a[rad]), NORMVAL = 16384
-NORMVAL <= Internal_sin() <= NORMVAL (the same for Internal_cos())

Time
The time is expressed as a multiple of the sampling period Ts.

Time[internal_time_unit] = KT * Time[sec]
KT = Fs (Fs = sampling_frequency = 1 / Ts)

Angular velocity
The angular velocity is expressed as a function of angles and time, in order to obtain it as the subtraction of two angles in two sampling moments; for resolution reasons, an amplification is needed, and we choose this amplification equal to NORMVAL=16384.
Omega[internal_angular_velocity_unit1] = KO1 * Omega[rad / sec] =
= (KO1 * KT / KA) * Angle[internal_angle_unit] / Time[internal_time_unit]
Since we want:

Omega[internal_angular_velocity_unit1] =
= NORMVAL * Angle[internal_angle_unit] / Time[internal_time_unit]

(→ Omega[internal_angular_velocity_unit1] = NORMVAL * (Angle(n) - Angle(n - k)) / k**)**

We obtain:

KO1 = NORMVAL * KA / KT = NORMVAL * 65536 / (2 * pi * Fs)

The entire speed range cannot, in general, be represented in a 16bit word, but a long is needed. This high resolution can be useful for some calculations, while when, for example, the speed is used to calculate voltages, lesser resolution is enough. To reduce the overall calculation time, the most effective choice is to have a second representation of the angular speed, coherent with the voltage and current representations, which are "per unit" based. So, the second representation of the angular speed is based on a normalized value:

BASE_SPEED_R_S = MAX_OMEGA_R_S

The so-called MAX_OMEGA_R_S is the maximum angular velocity required by the application, and we will associate this to NORMVAL. This value is linked to the maximum frequency (MAX_OMEGA_R_S = 2pi * MAX_FRE_HZ). The second representation is the following:

Omega[internal_angular_velocity_unit2] = KO2 * Omega[rad / sec]

KO2 = NORMVAL / BASE_SPEED_R_S

To pass from a representation to the other we have the following relationship:

Omega[internal_angular_velocity_unit2] = (KO2 / KO1) * Omega[internal_angular_velocity_unit1]

KO2 / KO1 = (2 * pi * Fs) / (65536 * MAX_OMEGA_R_S)

KO1 / KO2 = 65536 * MAX_FRE_HZ / Fs

Voltage

We can start our considerations from the maximum voltage readable by the A/D converter; this value is the maximum DC bus voltage and it is related to the maximum peak phase voltage by the relation: Vout_pk = (2/3)*Vbus (in case of over-modulation); this would already leave a good margin in voltage representation, but in case of deep flux weakening, the intermediate calculations can lead to higher voltage values, so we choose as the base voltage value the following:

BASE_VOLTAGE_VOLT = (2 ^ K) * MAX_VOLTAGE_VOLT, with K related with the application

MAX_VOLTAGE_VOLT is the maximum voltage readable by the A/D converter. With normal applications, (K = 1) leaves a margin for the maximum phase voltage equal to 3 times Vbus, which is more than enough. The voltage representation becomes:

Voltage[internal_voltage_unit] = KV * Voltage[Vol]

KV = NORMVAL / BASE_VOLTAGE_VOLT

Current

The maximum current readable by the A/D converter is chosen as the base value:

BASE_CURRENT_AMP = MAX_CURRENT_AMP

It is represented with NORMVAL = 16384:

Current[internal_current_unit] = KI * Current[Amp]

KI = NORMVAL / BASE_CURRENT_AMP

Impedance

The base impedance value can be deduced by the base voltage and current values; in fact the extended value chosen as the base voltage keeps into account the flux weakening, and no other trick are required in case of PM motor (in case of induction motor, the current can be much higher than the ratio between voltage and the impedance due to the magnetizing inductance: this would require some modification to the representation). So we keep simply:

BASE_IMPEDANCE_OHM = BASE_VOLTAGE_VOLT / BASE_CURRENT_AMP

The internal representation is:

Impedance[internal_impedance_unit] = KZ * Impedance[Ohm]

KZ = NORMVAL / BASE_IMPEDANCE_OHM =

= NORMVAL * BASE_CURRENT_AMP / BASE_VOLTAGE_VOLT

Resistance
The resistance is expressed in function of the "base" resistance, which is kept equal to the base impedance; this leads usually in a "poor" representation of the resistance in terms of resolution, but the resistance itself is highly variable with many factors, and an higher resolution is usually not required.

BASE_RESISTANCE_OHM = BASE_IMPEDANCE_OHM
Resistance[internal_resistance_unit] = KR*Resistance[Ohm]
KR = KZ

Inductance
The base inductance value is derived from the impedance and the angular velocity:

BASE_INDUCTANCE_HEN = BASE_IMPEDANCE_OHM / BASE_SPEED_R_S

so the internal representation becomes:

Inductance[internal_inductance_unit] = KL * Inductance[Henry]
KL = NORMVAL / BASE_INDUCTANCE_HEN =
= NORMVAL * BASE_SPEED_R_S * BASE_CURRENT_AMP / BASE_VOLTAGE_VOLT

Flux
In a similar way, the "base" flux can be chosen equal to:

BASE_FLUX_WEB = BASE_VOLTAGE_VOL / BASE_SPEED_R_S

Then we can express the flux as:

Flux[internal_flux_unit] = KF * Flux[volt * sec / rad]
KF = NORMVAL / BASE_FLUX_WEB

Calculation relationships
Please find below some useful relations derived from the previous assumptions (we will indicate all the “internal_xxxx_unit” with “int”):

Impedance[int] = (Inductance[int] * Omega[in2]) / NORMVAL
Flux[int] = (Inductance[int] * Current[int]) / NORMVAL
Voltage[int] = (Impedance[int] * Current[int]) / NORMVAL
Voltage[int] = (Flux[int] * Omega[in2]) / NORMVAL

As you can notice, the calculations becomes particularly simple (x/NORMVAL is x>>14).

13. APPENDIX B

13.1 Alarm Description

Alarm code 1:
The alarm 1 is called “EEPROM alarm” and described in the software by “EQP_ALL”. This alarm is set when one or more EEPROM parameters are higher than the maximum allowed value or lower than the minimum allowed value.
The LED DL4 is quickly blinking on the main board to indicate that an alarm is set.
The maximum and minimum values are specified in the two constants tables called: "par_max[]" "par_min[]" in the "ges_eqp.h" header file. Another root cause for the alarm 1 is the EEPROM hardware failure when the error is accessed in read or write mode.
When this alarm is active, the access to the EEPROM is restricted. To reset the alarm the default parameters set should be reloaded in the EEPROM. By using the PC GUI and the parameters setting window, it becomes possible to clean the EEPROM content. The first step is to write the magic number “33” in the first parameter n°00. The second step is to reset the board by pressing the reset button on the PCB or switching off the power supply. At this point a coherent set of parameters is loaded and the alarm should disappear.
Finally, if the alarm is produced by a hardware failure of the EEPROM itself, then the board needs to be repaired.

Alarm code 2:
The alarm 2 is called “hardware overcurrent” and described in the software by “FAULT_ALL”. This alarm is produced by the MCU peripheral called Port Output Enable (POE) in case of external overcurrent signal. The hardware overcurrent is producing a falling edge input on the POE pin. Furthermore, if the hardware level of the PWM output pin is not coherent with the level imposed by software, the alarm 2 will also be triggered.
The LED DL4 is quickly blinking on the main board to indicate that an alarm is set.
The only way to clear the alarm is to reset the board by using the reset button on the PCB or by switching off the supply and on again.
Finally, one of the root causes of the Alarm 2 is a hardware defect or a wrong behavior of the current control. So please also check the setting of the current PI coefficients that are stored in EEPROM or used in real-time.

Alarm code 3:
The alarm 3 is called “loss of phase” and described in the software by “TRIP_ALL”. This alarm is produced when the sensor-less position detection algorithm is producing inconsistent results. It means that the rotor position is unknown due to a lack of accuracy, so the motor is stopped.
The LED DL4 is quickly blinking on the main board to indicate that an alarm is set.
This alarm can be reset by setting the speed reference to zero on the PC GUI.

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

```
#define     EQP_ALL       1     // EEPROM alarm code
#define     FAULT_ALL     2     // overcurrent hardware alarm code (POE)
#define     TRIP_ALL      3     // loss of phase alarm code
```

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

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

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