Application Guide

iMOTION™ motor control IC for single motor drive

About this document

Scope and purpose

The IRMCK099 Series motor control ICs are mixed signal devices optimized for permanent magnet motor control. These devices can perform sensorless Field-Oriented Control (FOC) over the full speed range of the motor, including providing torque at zero speed and stable control at deep field weakening speeds.

This Application's Guide will begin with describing the various features of the IRMCK099 IC and firmware, continue with listing application specific registers and conclude with the design of motor control hardware for the final application. This Guide assumes that the user is in possession of an iMotion Reference Design Kit like the MADK and has already completed the installation of required software. The user should also review the software User's Guides included in the respective installation package.

Section 2 starts by describing in detail the features of IRMCK099 firmware and the benefits to the application and Section 3 introduces the MCE processor in more detail and describes all the application specific registers.

Section 4 guides the user through design, testing and optimization of application specific hardware as it relates to the IRMCK099 motor control IC.

Section 5 gives instructions on how to tune the flux estimator, flux PLL, speed and current control loops and optimize the motor start-up parameters. Section 5 concludes with motor drive performance verification and troubleshooting methods.

An additional document, the Reference Manual, also referred to frequently in this document, has detailed information on many topics covered here, as well as full descriptions of the MCE hardware registers and programming methods.

Intended audience

This application guide is intended for customers implementing an inverterized drive.

Table of Contents

1 Introduction

The IRMCK099 Series motor control ICs are mixed signal devices optimized for permanent magnet motor control. These devices can perform sensorless Field-Oriented Control (FOC) over the full speed range of the motor, including providing torque at zero speed and stable control at deep field weakening speeds. This guide corresponds to the case when "IRMCK099 V0.8" is selected in the "Options" page of the TinyWizard.

This Application's Guide will begin with describing the various features of the IRMCK099 Ic and firmware, continue with listing application specific registers and conclude with the design of motor control hardware for the final application. This Guide assumes that the user is in possession of an iMotion Reference Design Kit and has already completed the installation of required software. The user should also review the "EzTune Tiny User's Guide." This Guide will refer to EzTune Tiny features and actions frequently.

Section 2 starts by describing in detail the features of IRMCK099 firmware and the benefits to the application and Section 3 introduces the MCE processor in more detail and describes all the application specific registers.

Section 4 guides the user through design, testing and optimization of application specific hardware as it relates to the IRMCK099 motor control IC. Section 5 gives instructions on how to tune the flux estimator, flux PLL, speed and current control loops and optimize the motor start-up parameters. Section 5 concludes with motor drive performance verification and troubleshooting methods.

An additional document, the Reference Manual, also referred to frequently in this document, has detailed information on many topics covered here, as well as full descriptions of the MCE hardware registers and programming methods.

2 IRMCK099 Firmware

This application guide describes in great detail about the IRMCK099M Firmware and contains the following features:

- *Sensorless FOC control*

  High performance sensorless Field Oriented Control (FOC) of PMSM and IPM motors while utilizing all the benefits of fast switching ADC, integrated op-amps, comparator and motion peripherals of IRMCK099.

- *Enhanced flux based control algorithm which provides quick and smooth start*

  The direct closed-loop control of both torque and stator flux are achieved using proportional-integral controllers and space vector modulation with a new over modulation strategy.

- *Variable inductor sensing for initial rotor angle detection*

  Together with direct closed-loop start, variable inductor sensing improves motor start performance.

- *Single shunt or leg shunt motor current sensing*

  Provide unique single shunt and leg shunt current construction. Integrated op-amps and A/D converter enable a direct shunt resistor interface to the IC while eliminating additional analog/digital circuitry. Single shunt option can use either minimum pulse method or the phase shift method. Phase Shift PWM provides better startup and low speed performance in single shunt configuration.

- *Support 3ph and 2ph PWM modulation*

  Three-phase SVPWM with symmetrical placement of zero vectors (SVPWM) and Two-phase SVPWM ,which allows farther reduction of switching losses with the provision of Type1, Type2 and Type3 for single shunt option and Type3 for leg shunt option.

- *Networking capability with user mode UART*

  Master and slave mode available, with up to 254 nodes and each node has its own address. Broadcast feature available to update all the slaves at once.

- *31 re-programmable parameter blocks*

  31 configuration blocks can be programmed and are of each 128 bytes in size. Each block can be programmed individually or all 31 blocks at the same time using MCEProgrammer.

- *Multiple motor parameter support*

  Each parameter block can be assigned to different motor or hardware platforms with a unique motor id.

- *Current limit de-rating by temperature*

  When the temperature is above a set limit, proper measures must be taken to reduce the internal power dissipation in the system. IRMCK099 will reduce the current, thereby reducing the power when the temperature limit is reached.

- *Protection*

Motor Gatekill	This fault is set when there is over current and will instantly initiate inverter and regulator shutdown. Cannot be masked.
Critical Over Voltage	This fault is set when the voltage is above a set threshold; all low side switches are clamped (zero-vector-braking) to protect the drive and brake the motor. The zero-vector is held until fault is cleared. Cannot be masked.
Over Voltage	This fault is set when the DC Bus voltage is above a set limit.
Under Voltage	This fault is set when the DC Bus voltage is under a set limit.
Flux PLL Out Of Control	This fault is set when the PLL is not locked which could be due to wrong parameter configuration, one or more phase loss.
Zero Speed	This fault is set if motor is not spinning for time duration of more than 2 or 4 seconds
Over Temperature	This fault is set when the temperature is above a set limit.
Rotor Lock	This fault is set when the rotor is locked
MCE Execution	This fault occurs if the MCE firmware does not complete its PWM-cycle processing before the next PWM pulse.
Parameter Load	This fault is set when there is no matching Motor ID is found or when there is no parameter block is programmed in the OTP.
Link Break Protection	This fault is set when there is no UART communication for more than a set time limit.

2.1 Application Pin Out

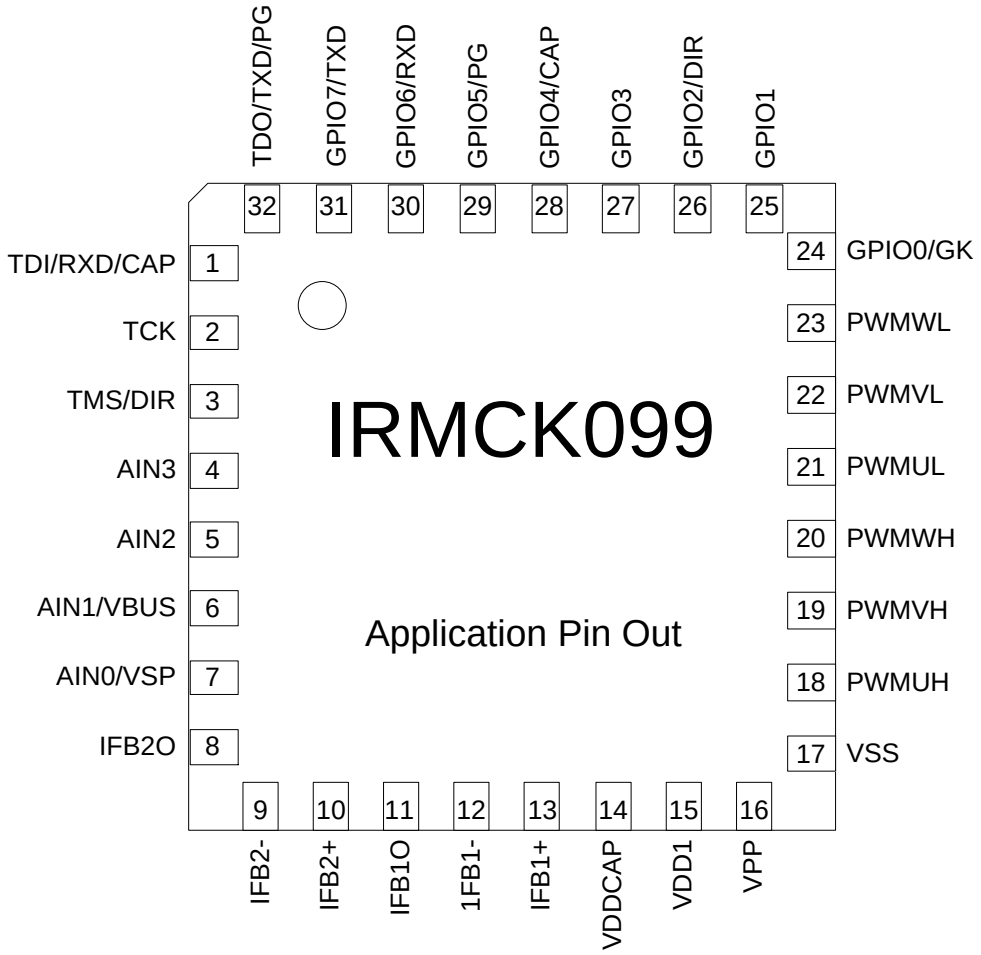


Figure 1: IRMCK099M Pin Out

The Application pinout (full mapping for QFN-32 package)) is shown in Figure 1 and the functions of each pin are described in Table 1.

For package with limited pin access, control bit HwConfig[7] should be configured to 1, see Table 5 for detail.

Pin Number	Pin Name	Function
1	TDI/RXD/CAP	JTAG Test Data Input/ UART Receiver Input/ Capture Input
2	TCK	JTAG Test Clock
3	TMS /DIR	JTAG Test Mode Input / Motor Direction Input (not Valid in UART Control Mode)
4	AIN3	Analog input channel 3, 0-1.2V range, needs to be pulled down to VSS if unused
5	AIN2	Analog input channel 2, 0-1.2V range, needs to be pulled down to VSS if unused
6	AIN1/VBUS	Analog input channel 1, 0-1.2V range, used for DC Bus Voltage Sensing
7	AIN0/VSP	Analog input channel 0, 0-1.2V range, used for Variable Speed Control Input

8	IFB2O	Operational amplifier output for 2nd leg shunt resistor current sensing
9	IFB2-	Operational amplifier negative input for 2nd leg shunt resistor current sensing
10	IFB2+	Operational amplifier positive input for 2nd leg shunt resistor current sensing
11	IFB1O	Operational amplifier output for single or leg shunt resistor current sensing
12	IFB1-	Operational amplifier negative input for single or leg shunt resistor current sensing
13	IFB1+	Operational amplifier positive input for single or leg shunt resistor current sensing
14	VDDCAP	Internal 1.8V output, Capacitor(s) to be connected
15	VDD	3.3V digital and analog power
16	VPP	OTP Programming Voltage (6.75V)
17	VSS	Digital common
18	PWMUH	PWM gate drive for phase U high side, configurable either high or low true
19	PWMVH	PWM gate drive for phase V high side, configurable either high or low true
20	PWMWH	PWM gate drive for phase W high side, configurable either high or low true
21	PWMUL	PWM gate drive for phase U low side, configurable either high or low true
22	PWMVL	PWM gate drive for phase V low side, configurable either high or low true
23	PWMWL	PWM gate drive for phase W low side, configurable either high or low true
24	GPIO0/GK	External Gatekill Input /General Purpose Digital I/O [1]
25	GPIO1	General Purpose Digital I/O [1]
26	GPIO2/DIR	Motor Direction Input (not valid in UART control mode) /General Purpose Digital I/O [1]
27	GPIO3	General Purpose Digital I/O [1], this pin should be left floating or pulled high to 3.3V
28	GPIO4/CAP	General Purpose Digital I/O [1]/ Capture Input[1]
29	GPIO5/PG	Pulse generator output / General Purpose Digital I/O [1]
30	GPIO6/RXD	UART Receive / General Purpose Digital I/O [1]
31	GPIO7/TXD	UART Transmit / General Purpose Digital I/O [1]
32	TDO/TXD/PG	JTAG Test Data Output / UART Transmit / Pulse Generator Output[2]

Note[1] - Functions are available depending on the firmware provided.

Note[2] - JTAG functions can be remapped using HwConfig register explained in this document.

Table 1 IRMCK099M Pin Descriptions

2.2 Typical Application Diagram

In Figure 2, a typical application connection diagram with IRMCK099 functionality details is shown when single shunt option is selected and Figure 3 depicts a typical application connection diagram with leg shunt option.

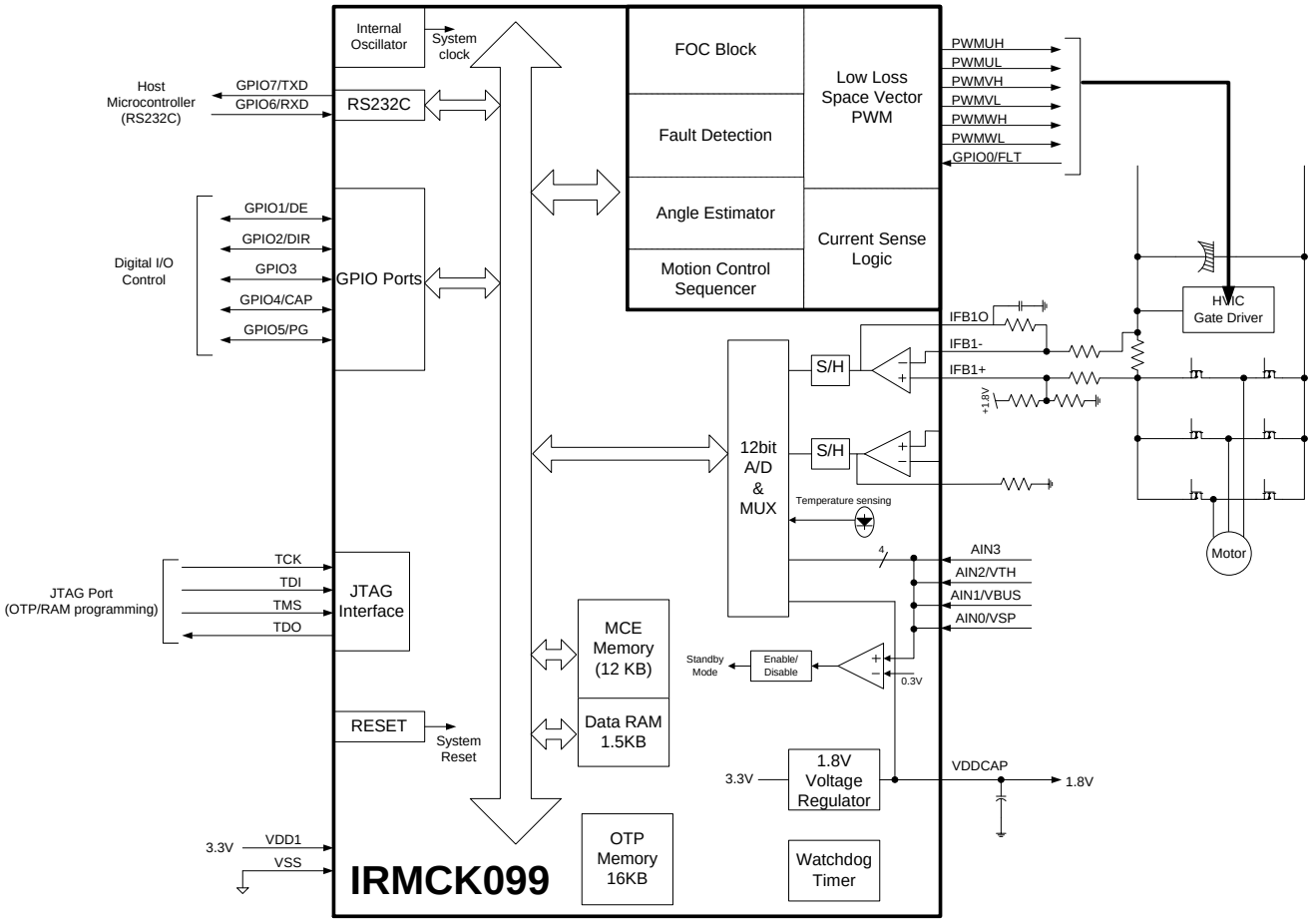


Figure 2 IRMCK099M Single Shunt Application Diagram

When single shunt option is being used in the application, the second op-amp IFB2 is available for the users for any other application use like to calculate DC bus average or ac power calculations. In leg shunt option, both IFB1 and IFB2 are being used for current sensing.

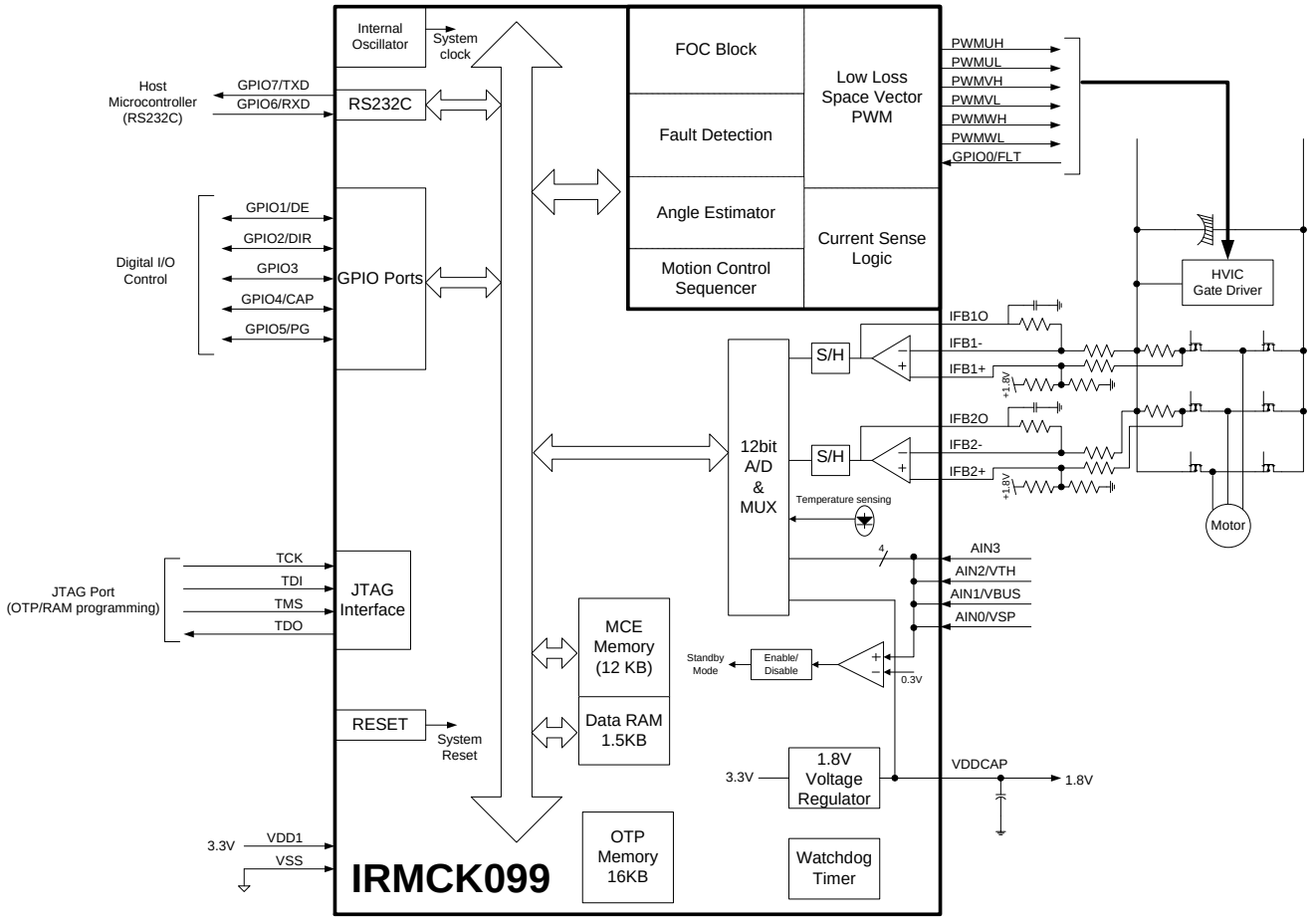


Figure 3 IRMCK099M Leg Shunt Application Diagram

2.3 Analog inputs and ADC channels configuration

IRMCK099 has 8 ADC channels (ch0, ch1, ch2, ch3, ch8, ch9, ch10, ch11). Each ADC channel is connected to a mux which can individually select from 8 input signals (IFB1O_0, IFB1O_1, IFB2O, AIN0, AIN1, AIN2, AIN3 and internal TempSense). See Figure 4 for analog inputs and ADC channels configuration.

IFB1O has two inputs (IFB1O_0 and IFB1O_1), each input has a dedicate sample and hold circuit. This structure is designed for purpose of single shunt current sensing. In single shunt current sensing, each input needs to be samples twice within 1 PWM cycles. It needs totally 4 ADC channels to do the single shunt current sensing. These 4 ADC channels are ch0, ch1, ch2 and ch3 and they are assigned to IFB1O_1, IFB1O_0, IFB1O_0 and IFB1O_1 respectively.

There is no ch4, ch5, ch6, ch7 existing in the IC. Ch8 does exist but this ADC channel is not used.

Ch9 is assigned to IFB2O which is the V phase current input in leg shunt configuration.

Ch10 is assigned to AIN1 which is the DC bus voltage input.

Ch11 is a special rolling channel which samples AIN0, AIN2, AIN3 and internal TempSense. Each input will be sampled every 4 PWM cycles.

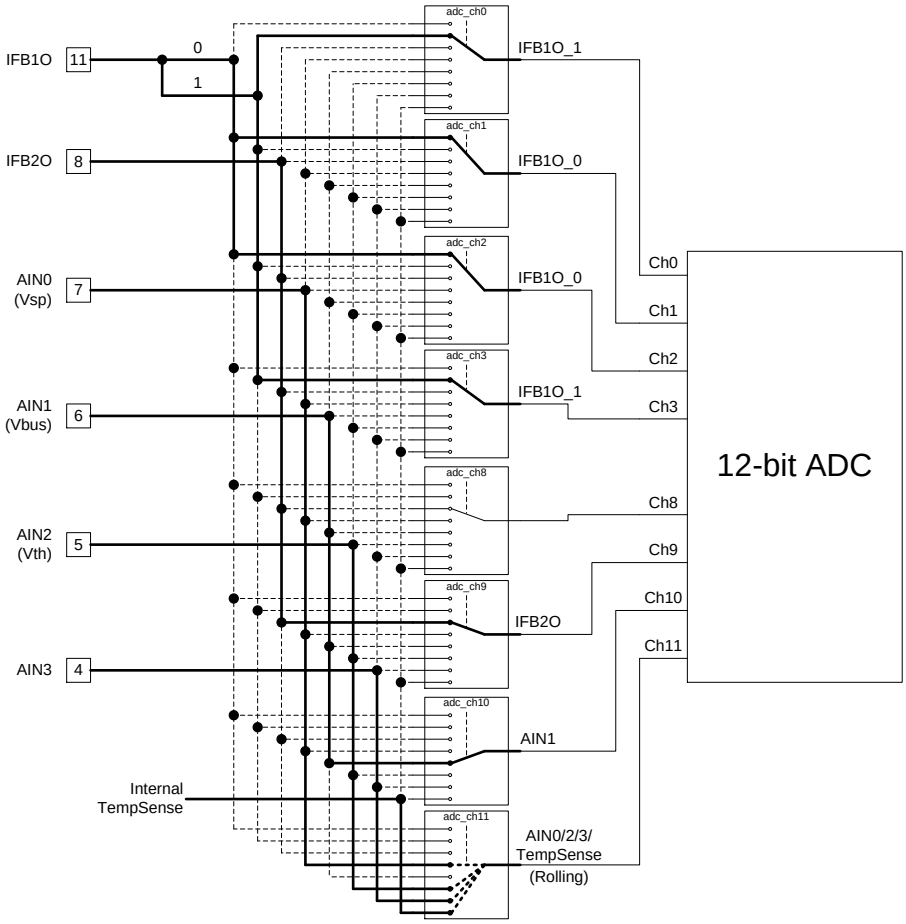


Figure 4: Analog inputs and ADC channels configuration

Figure 5 illustrates ADC channels being sampled in every PWM cycle. For ch11, its input changes every PWM cycle in sequence AIN0→AIN2→AIN3→TempSense.

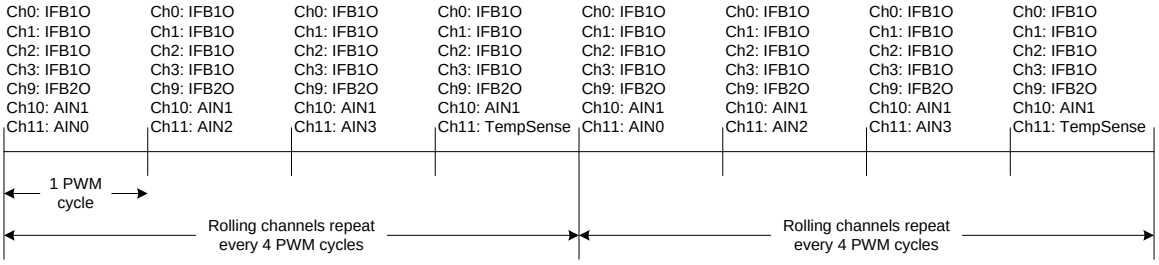


Figure 5: ADC sample strategy

2.4 User Mode UART

The user mode UART communication is designed to provide a simple, reliable and scalable communication protocol for motor control application. The protocol is simple so that it can be easily implemented even in low-end microcontrollers which work as master to control the motor. It supports networking (up to 254 nodes on same network) which is required in some industrial fan/pump application.

UART baudrate is configured by SysConfig[13] and SysConfig[1:0], it has eight selections: 2400bps, 9600bps, 19200bps, 57600bps, 115200bps, 230400bps, 256000bps and 460800bps.

2.4.1 Data Frame

There are three types of data frames: 8-bytes standard message, 12-bytes long message and 16-bytes long message. Standard message is used in most commands while long message is only used in command 8, 9, 10, 11 (and command 12, 13 if SysConfig[14]=1).

The format of the data frame is shown in Figure 6.

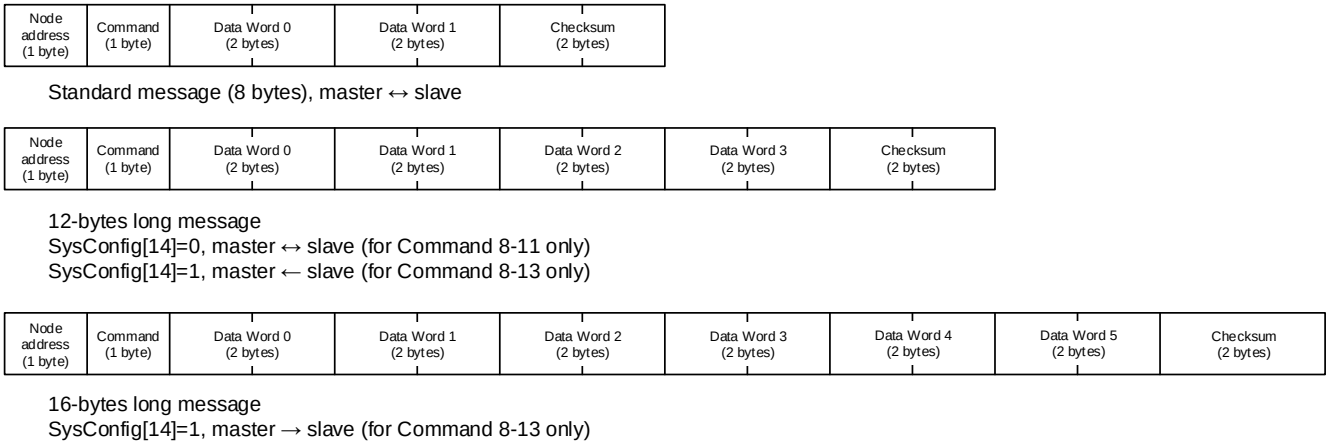


Figure 6: UART Data Frame

2.4.2 Node Address

Node address is the first byte in a data frame. It is designed to allow one master controlling multiple slaves in the same network. Each slave node has its unique node ID which is defined in low 8-bits of NodeID_L register. The slave only acknowledges and responds to the message with same ID. There are two broadcast addresses (0x00 and 0xFF) defined for different usage. If a message is received with address=0x00, all the slaves execute the command but will not send a reply to the master. This is useful in a multiple slave network and the master needs to control all the slaves at the same time, for example, turn on all the motor by sending only one message. If receive a frame with address=0xFF, the slave will execute the command and also send a reply to the master. This is useful in 1-to-1 configuration when the master doesn't know or doesn't need to know the slave node address.

NodeID_L	Command
0x00	All nodes receive and execute command, no response.
0x01 to 0xFE	Only the node that has same address executes the command and replies the master.
0xFF	All nodes receive and execute the command and reply the master. Only used in 1-to-1 configuration. It will cause conflict if multiple nodes connected to the same network

2.4.3 Link Break Protection

Link break protection is to stop the motor if there is no UART communication for certain period of time. In some application, the main controller maintains communication with the motor controller. In case of a loss of communication or line break, it is desired to stop the motor for protection.

This protection feature is enabled by set 1 to SwFaults_Mask[13].

Link break timeout is configured by TLinkBreakBase together with SysConfig[6:5]. TLinkBreakBase is register to configure the base time of link break timeout. SysConfig[6:5] is used to select timeout from x1, x2, x3 or x6 of TLinkBreakBase.

- SysConfig[6:5]=00, $T_{LinkBreak} = TLinkBreakBase * 0.01$, in seconds, default is 5 seconds

- SysConfig[6:5]=01, $T_{LinkBreak} = TLinkBreakBase * 0.02$, in seconds, default is 10 seconds
- SysConfig[6:5]=10, $T_{LinkBreak} = TLinkBreakBase * 0.03$, in seconds, default is 15 seconds
- SysConfig[6:5]=11, $T_{LinkBreak} = TLinkBreakBase * 0.06$, in seconds, default is 30 seconds

2.4.4 Command

UART command is the second byte in a data frame. Bit [6:0] specifies the command code. Bit [7] is the indication bit indicates the direction of the data frame. All data frames sent by master must has bit 7 cleared (=0), all reply data frames sent by slave must has bit 7 set (=1).

Bit[6:0] Command 0-127

0: Read Status

1: Clear Fault

2: Change Control Mode

3: Motor Control, motor run if Targetspeed not 0, motor stop if TargetSpeed=0

4: Not used, slave will not reply to master

5: Register Read

6: Register Write

7: Not used, slave will not reply to master

8: Motor control command for NodeAddress=1 to 4 (or 1 to 6[Note 1]), only node 1 will reply

9: Motor control command for NodeAddress=1 to 4 (or 1 to 6 [Note 1]), only node 2 will reply

10: Motor control command for NodeAddress=1 to 4 (or 1 to 6 [Note 1]), only node 3 will reply

11: Motor control command for NodeAddress=1 to 4 (or 1 to 6 [Note 1]), only node 4 will reply

12 [Note 2]: Motor control command for NodeAddress=1 to 6, only node 5 will reply

13 [Note 2]: Motor control command for NodeAddress=1 to 6, only node 6 will reply

14-127: Not used, slave will not reply to master

Bit[7] Indication bit.

0: This message is sent by master

1: This message is reply message sent by slave

Note 1: If control bit SysConfig[14]=1.

Note 2: Command 12 and 13 are valid commands if SysConfig[14]=1, otherwise these two commands are not used.

2.4.5 Checksum

Checksum is 16-bits and calculated as below:

[Node address : Command] + Data Word 0 + Data Word 1 + (Data Word 2 + Data Word 3 + Data Word 4 + Data Word 5) + Checksum = 0x0000

Note: Only command 8-13 has Data Word 2/3/4/5.

2.4.6 UART message

Command = 0x00 (Read Status)

Master → Slave	Node address (1 byte)	Command = 0x00	Status Code (2 bytes)	0x00	0x00	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x80	Status Code (2 bytes)	Status Reply (2 bytes)		Checksum (2 bytes)

Status code and status reply:

Status Code	Status Reply	Register Address
0x0000	FaultFlags	0x1EE
0x0001	MotorSpeed	0x0E1
0x0002	SequencerState	0x0C1
0x0003	NodeAddr	0x21C
0x0004 – 0x00FF	0x0000	N/A

Command = 0x01 (Clear Fault)

Master → Slave	Node address (1 byte)	Command = 0x01	0x00	0x00	0x00	0x00	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x81	0x00	0x00	0x00	0x00	Checksum (2 bytes)

Command = 0x02 (Change Control Mode)

Master → Slave	Node address (1 byte)	Command = 0x02	0x00	0x00	0x00	00: UART 01: Analog 02: Freq 03: Duty	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x82	0x00	0x00	0x00	00: UART 01: Analog 02: Freq 03: Duty	Checksum (2 bytes)

Command = 0x03 (Motor Control)

Master → Slave	Node address (1 byte)	Command = 0x03	0x00	0x00	TargetSpeed (2 bytes)	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x83	SequencerState (2 bytes)	MotorSpeed (2 bytes)		Checksum (2 bytes)

TargetSpeed=0: motor stop, TargetSpeed≠0: motor start

Command = 0x05 (Register Read)

Master → Slave	Node address (1 byte)	Command = 0x05	Register Address (2 bytes)	0x00	0x00	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x85	Register Address (2 bytes)	Register Value (2 bytes)		Checksum (2 bytes)

Command = 0x06 (Register Write)

Master → Slave	Node address (1 byte)	Command = 0x06	Register Address (2 bytes)	Register Value (2 bytes)	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x86	Register Address (2 bytes)	Register Value (2 bytes)	Checksum (2 bytes)

Command = 0x08 (Motor control command for NodeAddress=1 to 4 (or 1 to 6), only node 1 will reply)

Master → Slave (SysConfig[14]=0)	Node address (1 byte)	Command = 0x08	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	Checksum (2 bytes)	
Master → Slave (SysConfig[14]=1)	Node address (1 byte)	Command = 0x08	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	TargetSpeed for NodeAddr = 5 (2 bytes)	TargetSpeed for NodeAddr = 6 (2 bytes)	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x88	MotorSpeed of NodeAddr = 1 (2 bytes)	IdqFilt of NodeAddr = 1 (2 bytes)	SwFaults of NodeAddr = 1 (2 bytes)	Indirect_rd_val of NodeAddr = 1 (2 bytes)	Checksum (2 bytes)	

TargetSpeed=0: motor stop, TargetSpeed≠0: motor start

Command = 0x09 (Motor control command for NodeAddress=1 to 4 (or 1 to 6), only node 2 will reply)

Master → Slave (SysConfig[14]=0)	Node address (1 byte)	Command = 0x09	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	Checksum (2 bytes)	
Master → Slave (SysConfig[14]=1)	Node address (1 byte)	Command = 0x09	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	TargetSpeed for NodeAddr = 5 (2 bytes)	TargetSpeed for NodeAddr = 6 (2 bytes)	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x89	MotorSpeed of NodeAddr = 2 (2 bytes)	IdqFilt of NodeAddr = 2 (2 bytes)	SwFaults of NodeAddr = 2 (2 bytes)	Indirect_rd_val of NodeAddr = 2 (2 bytes)	Checksum (2 bytes)	

TargetSpeed=0: motor stop, TargetSpeed≠0: motor start

Command = 0x0A (Motor control command for NodeAddress=1 to 4 (or 1 to 6), only node 3 will reply)

Master → Slave (SysConfig[14]=0)	Node address (1 byte)	Command = 0x0A	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	Checksum (2 bytes)	
Master → Slave (SysConfig[14]=1)	Node address (1 byte)	Command = 0x0A	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	TargetSpeed for NodeAddr = 5 (2 bytes)	TargetSpeed for NodeAddr = 6 (2 bytes)	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x8A	MotorSpeed of NodeAddr = 3 (2 bytes)	IdqFilt of NodeAddr = 3 (2 bytes)	SwFaults of NodeAddr = 3 (2 bytes)	Indirect_rd_val of NodeAddr = 3 (2 bytes)	Checksum (2 bytes)	

TargetSpeed=0: motor stop, TargetSpeed≠0: motor start

Command = 0x0B (Motor control command for NodeAddress=1 to 4 (or 1 to 6), only node 4 will reply)

Master → Slave (SysConfig[14]=0)	Node address (1 byte)	Command = 0x0B	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	Checksum (2 bytes)	
Master → Slave (SysConfig[14]=1)	Node address (1 byte)	Command = 0x0B	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	TargetSpeed for NodeAddr = 5 (2 bytes)	TargetSpeed for NodeAddr = 6 (2 bytes)	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x8B	MotorSpeed of NodeAddr = 4 (2 bytes)	IdqFilt of NodeAddr = 4 (2 bytes)	SwFaults of NodeAddr = 4 (2 bytes)	Indirect_rd_val of NodeAddr = 4 (2 bytes)	Checksum (2 bytes)	

TargetSpeed=0: motor stop, TargetSpeed≠0: motor start

Command = 0x0C (Motor control command for NodeAddress=1 to 6, only node 5 will reply)

(This command only valid if SysConfig[14]=1)

Master → Slave (SysConfig[14]=1)	Node address (1 byte)	Command = 0x0C	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	TargetSpeed for NodeAddr = 5 (2 bytes)	TargetSpeed for NodeAddr = 6 (2 bytes)	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x8C	MotorSpeed of NodeAddr = 5 (2 bytes)	IdqFilt of NodeAddr = 5 (2 bytes)	SwFaults of NodeAddr = 5 (2 bytes)	Indirect_rd_val of NodeAddr = 5 (2 bytes)	Checksum (2 bytes)		

TargetSpeed=0: motor stop, TargetSpeed≠0: motor start

Command = 0x0D (Motor control command for NodeAddress=1 to 6, only node 6 will reply)

(This command only valid if SysConfig[14]=1)

Master → Slave (SysConfig[14]=1)	Node address (1 byte)	Command = 0x0D	TargetSpeed for NodeAddr = 1 (2 bytes)	TargetSpeed for NodeAddr = 2 (2 bytes)	TargetSpeed for NodeAddr = 3 (2 bytes)	TargetSpeed for NodeAddr = 4 (2 bytes)	TargetSpeed for NodeAddr = 5 (2 bytes)	TargetSpeed for NodeAddr = 6 (2 bytes)	Checksum (2 bytes)
Slave → Master (Reply)	Node address (1 byte)	Command = 0x8D	MotorSpeed of NodeAddr = 6 (2 bytes)	IdqFilt of NodeAddr = 6 (2 bytes)	SwFaults of NodeAddr = 6 (2 bytes)	Indirect_rd_val of NodeAddr = 6 (2 bytes)	Checksum (2 bytes)		

TargetSpeed=0: motor stop, TargetSpeed≠0: motor start

2.4.7 Connecting multiple nodes to same network

It is possible to connect multiple IRMCK099 controllers to same UART network, see Figure 7 for detail.

For the TXD pin of each IRMCK099 node, it needs to connect a Schottky diode before connect to the same wire, and on the master controller side, a 4.7kOhm pull up resister is required.

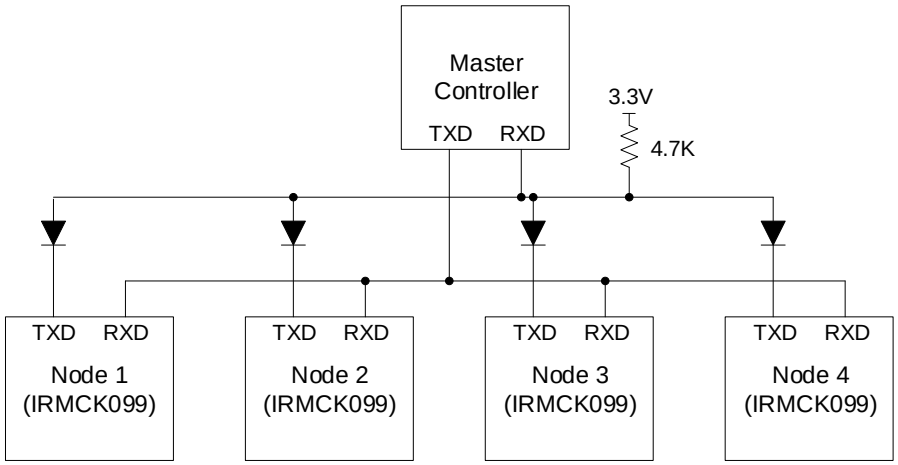


Figure 7: UART network connection

2.5 Multiple Parameter Programming

IRMCK099 has 16k bytes OTP memory, 12k bytes are used to store MCE firmware code and the rest 4k bytes are used to store control parameter and calibration data. There are totally 31 parameter blocks that can be used to store control parameters. Each parameter block is 128 bytes in size. Multiple parameter blocks can be programmed in order to support different motor types. Each parameter block has a field called “MotorID” which is to identify the motor type that it supports.

The parameter block that is programmed into the OTP is the output of TinyWizard saved into .txt file. Every time there is parameter change, new block can be programmed into the unused parameter block using MCEProgrammer.

MCEProgrammer will search if there is any block that has already been programmed in OTP memory and if there was no block that was programmed before, the values will be written to block 1 located at 0x3F7F. If there was already a block programmed, the next available block will be programmed. For example, if there were 3 blocks already programmed, then block 4 will be programmed at 0x3D80.

The starting address of each block can be calculated as shown below:

Starting Block Address in OTP = 0x3000 + 0x80 x (31 –n) where n =1 to 31

For example, parameter block 2 starting MCE address =0x3000 + 0x80 x (0x1D) = 0x3E80

Bit [10:9] in register HwConfig is used to enable/disable the multiple motor support and also configure the source for MotorID selection. The HwConfig value is always used from the latest block or the last block written to the OTP.

Multiple motor support is disabled if HwConfig[10:9]=00, MCE firmware will search for the most recent valid parameter block and use it to configure the registers.

Multiple motor support is enabled if HwConfig[10:9]=01, 10, or 11:

- HwConfig[10:9]=01, MotorID will be read from AIN3 input
- HwConfig[10:9]=10, MotorID will be read from AIN0 input
- HwConfig[10:9]=11, MotorID will be selected by writing a value between 0 to 255 to register MotorIDSel.

If multiple motor support is enabled in the most recent block and MotorID is set (by AIN3/AIN0 or UART), it will start to search for parameter block which has matching MotorID and use it to configure the parameters.

Once the 31 parameter blocks are used up, IC cannot be programmed with new parameter. Due to limited number of parameter blocks, during development, it's better to tune the motor in RAM and program the parameter block only when motor tuning is done. Figure 8 shows the memory address for the parameter block and in Figure 9 the OTP parameter load procedure is shown in a flow chart. The *HwConfig* register is explained in the later sections on how to enable the multiple motor parameters.

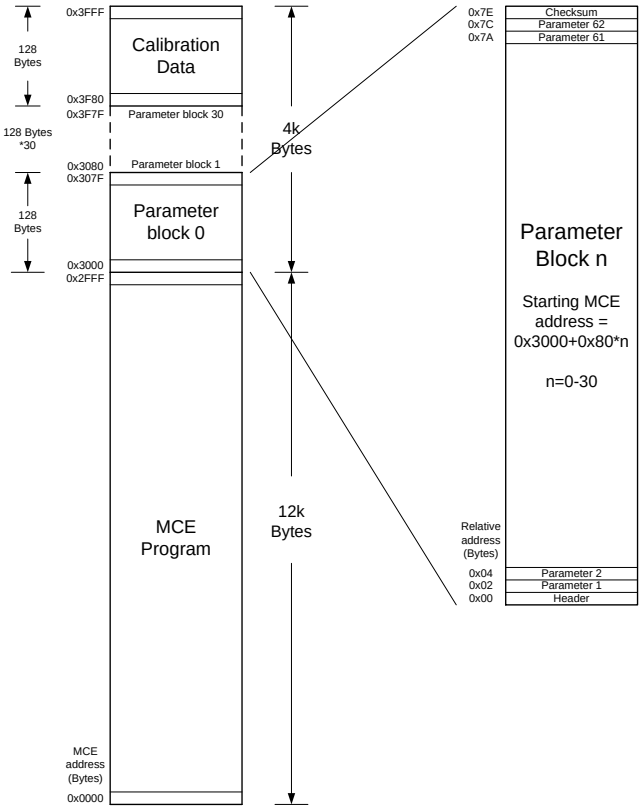


Figure 8: OTP Parameter Block Structure

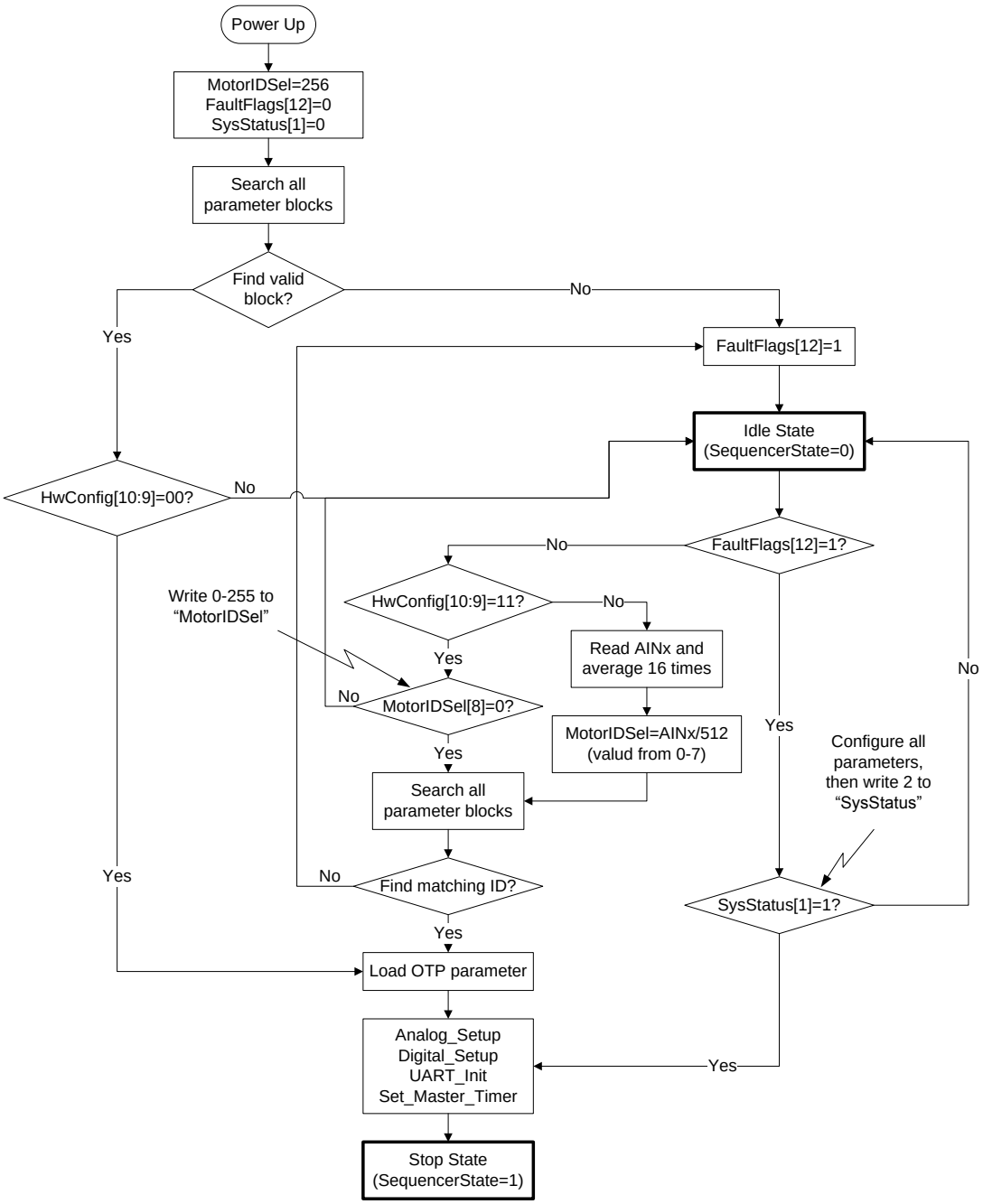


Figure 9: OTP parameter load procedure

2.5.1 Parameter load fault

If no matching MotorID could be found, or no valid parameter block found in OTP, FaultFlags[12] will be set to indicate fault condition and the motor will not be allowed to run. In this case, motor parameters can be configured by UART command, after all the parameters are configured, write 2 to register SysStatus will inform the MCE firmware that the parameters has been configured. If there is no other fault, the firmware will then go to stop state and ready to run the motor.

2.6 Control Input

MCE firmware is able to control the motor from 4 types of inputs. Control input is selected by register SysConfig[3:2].

SysConfig[3:2]=00: UART control

SysConfig[3:2]=01: Vsp analog input

SysConfig[3:2]=10: Frequency input

SysConfig[3:2]=11: Duty cycle input

2.6.1 UART control

In UART control mode, motor start, stop and speed change are controlled by UART command 3.

If IRMCK099 receive UART command 3 with TargetSpeed field (Data word 1, 5th and 6th bytes) other than 0, it will start the motor. If TargetSpeed field is 0, the motor will stop.

To change the motor target speed, master controller should send a new UART command 3 with new TargetSpeed.

TargetSpeed can be positive or negative, motor will spin in reverse direction if TargetSpeed is negative.

If any fault condition happens, motor will stop and stay in fault status. It is up to master controller when to clear the fault and restart the motor.

2.6.2 Vsp Analog Input Control

In Vsp Analog Input control mode, the motor operations like motor start, motor stop and speed change are controlled by applying an analog voltage signal. MCE firmware uses Ain0 (pin7 in QFN32 package) as the Vsp Analog input. The relationship between Vsp voltage and motor target speed is shown in Figure 10

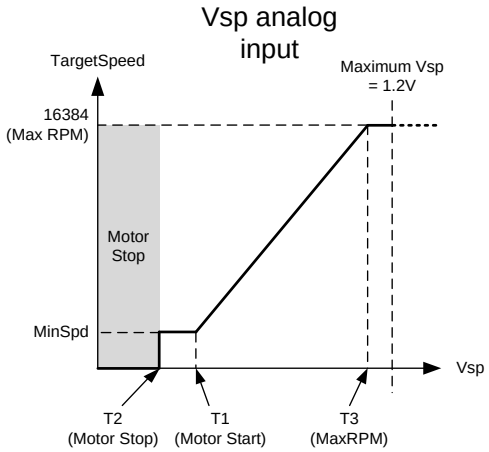


Figure 10 Vsp Analog Input

There are three input thresholds used to define the relationship between input voltage and TargetSpeed.

1. T1 (Input threshold for motor start): if the Vsp analog voltage is above this threshold, motor will start
2. T2 (Input threshold for motor stop): if the Vsp analog voltage is below this threshold, motor will stop
3. T3 (Input threshold for MaxRPM): if the Vsp analog voltage is higher or equal to this threshold, TargetSpeed will be 16384 which is MaxRPM

TinyWizard uses these three input thresholds to calculate the value of two registers: CmdStartStop and CmdGain

$$CmdStartStop[15:8] = Integer\left\{\left(\frac{T2}{0.6} * 255\right) + 0.5\right\}$$

Where T2 = Analog Vsp Motor Stop Voltage in V.

$$CmdStartStop[7:0] = Integer\left\{\left(\frac{T1}{0.6} * 255\right) + 0.5\right\}$$

Where T1 = Analog Vsp Motor Start Voltage in V.

$$CmdGain = Integer\left\{\left(\frac{Speed_{Max} - Speed_{Min}}{Speed_{Max}} * 2^{12}\right) * \left(\frac{2^{14}}{\left(\left(16384 * \frac{T3}{ADC\ Input\ Range}\right) - (CmdStart * 32)\right)}\right) + 0.5\right\}$$

Where T3 = Analog Vsp Motor Max RPM Voltage in V,

$Speed_{Max}$ = Maximum motor speed in RPM,

$Speed_{Min}$ = Minimum motor speed in RPM.

Table 2 Specification for Analog Input voltage

	Vsp Analog input
Recommended input range	0.1V – 1.15V
T1	< 0.6V
T2*	< 0.6V
T3**	< 1.2V

* **T2 must be < T1**

** **T3 must be > T1**

This feature is not available in UART control mode.

2.6.3 Frequency input

In Frequency Input control mode, the motor operations like motor start, motor stop and speed change are controlled by applying a square wave frequency signal on digital IO pin. MCE firmware uses GPIO4 (pin28 in QFN32 package) as the frequency input. The relationship between Frequency and motor target speed is shown in Figure 11

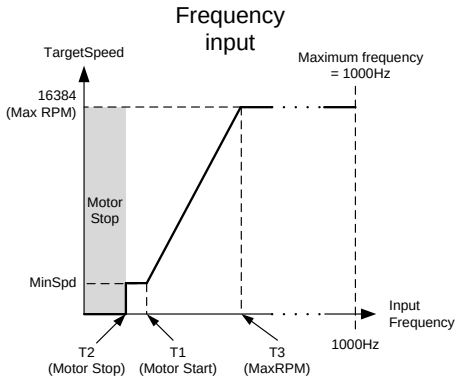


Figure 11 Frequency Input

There are three input thresholds used to define the relationship between frequency input and TargetSpeed.

1. T1 (Input threshold for motor start): if the frequency input is above this threshold, motor will start
2. T2 (Input threshold for motor stop): if the frequency input is below this threshold, motor will stop
3. T3 (Input threshold for MaxRPM): if the frequency input is higher or equal to this threshold, TargetSpeed will be 16384 which is MaxRPM

TinyWizard uses these three input thresholds to calculate the value of two registers: CmdStartStop and CmdGain

$$CmdStartStop[15:8] = Integer\ \{T2 + 0.5\}$$

Where T2 = Motor Stop Speed Frequency in Hz.

$$CmdStartStop[7:0] = Integer\ \{T1 + 0.5\}$$

Where T1 = Motor Start Speed Frequency in Hz.

$$CmdGain = Integer\left\{\left(2^{12} * \frac{\left(16384 - \left(\frac{Speed_{Min}}{Speed_{Max}} * 16384\right)\right)}{(T3 - T1) * 32}\right) + 0.5\right\}$$

Where T1 = Motor Start Speed Frequency in Hz,

T3 = Motor Max Speed Frequency in Hz,

$Speed_{Max}$ = Maximum motor speed in RPM,

$Speed_{Min}$ = Minimum motor speed in RPM.

Table 3 Specification for Frequency Input

	Frequency input
Recommended input range	5Hz – 1000Hz 10% – 90% duty cycle
T1	≤ 255Hz

T2*	≤ 255Hz
T3**	≤ 1000Hz

* **T2 must be < T1**

** **T3 must be > T1**

This feature is not available in UART control mode.

2.6.4 Duty Cycle Input Control

In Duty Cycle Input control mode, the motor operations like motor start, motor stop and speed change are controlled by varying the duty cycle of a rectangular wave signal on digital IO pin. MCE firmware uses GPIO4 (pin28 in QFN32 package) as the duty cycle input. The relationship between duty cycle and motor target speed is shown in Figure 12

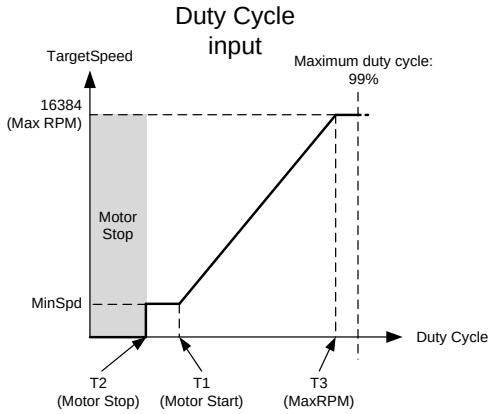


Figure 12: Duty Cycle Input

There are three input thresholds used to define the relationship between duty cycle input and TargetSpeed.

1. T1 (Input threshold for motor start): if the duty cycle input is above this threshold, motor will start
2. T2 (Input threshold for motor stop): if the duty cycle input is below this threshold, motor will stop
3. T3 (Input threshold for MaxRPM): if the input reaches or above this threshold, TargetSpeed will be 16384 which is MaxRPM

$$CmdStartStop[15:8] = Integer\left\{\left(\frac{T2}{50}\right) * 255 + 0.5\right\}$$

Where T2 = Motor Stop Speed Duty Cycle in %.

$$CmdStartStop[7:0] = Integer\left\{\left(\frac{T1}{50}\right) * 255 + 0.5\right\}$$

Where T1 = Motor Start Speed Duty Cycle in %.

$$CmdGain = Integer\left\{\left(\frac{Speed_{Max} - Speed_{Min}}{Speed_{Max}} * 2^{12}\right) * \left(\frac{2^{14}}{\left(\left(16384 * \frac{T3}{100}\right) - (CmdStart * 32)\right)}\right) + 0.5\right\}$$

Where T1 = Motor Start Speed Duty Cycle in %,

T3 = Motor Max Speed Duty Cycle in %,

$Speed_{Max}$ = Maximum motor speed in RPM,

$Speed_{Min}$ = Minimum motor speed in RPM.

TinyWizard uses these three input thresholds to calculate the value of two registers: CmdStartStop and CmdGain.

Table 4 Specification for Duty Cycle Input

	Duty cycle input
Recommended input range	5Hz – 1500Hz 1% – 99% duty cycle
T1	< 50%
T2*	< 50%
T3**	≤ 99%

* **T2 must be < T1**

** **T3 must be > T1**

This feature is not available in UART control mode.

2.6.5 Automatic fault clear and restart

In Vsp, frequency or duty cycle control input mode, if there is fault condition, motor will stop and start a 10seconds counter. After 10 seconds, the control will try to clear the fault. If the fault condition still exists, control will stay in fault condition, once the fault condition is removed, motor will start again.

This feature is not available in UART control mode.

2.6.6 Forced control input change

If required by some debug purpose, it is possible to change the control inputs by sending UART command 2 from master controller (or PC), then a new mode will be effective immediately. If the control input is switched to UART control from the other three inputs, motor status (run/stop, and TargetSpeed) will be unchanged until it receive a new motor control command.

2.7 Bootstrap Capacitor Charge

A zero vector (all low side devices on) is applied for initial turn on of the PWM inverter output. If a Bootstrap gate driver is used, the Bootstrap capacitors (for u, v, w phase) will all be charged simultaneously, which can lead to a nuisance trip. This charging current can be limited by the built-in Pre-charge control function.

Instead of charging all low side devices simultaneously, the gate Pre-charge control will schedule an alternating (u, v, w phase) charging sequence. Each phase charge bootstrap capacitor 1/3 of PWM time. Pre-charge takes 128 PWM cycles.

Figure 13 illustrate the PWM signal during gate charge before motor enter run state.

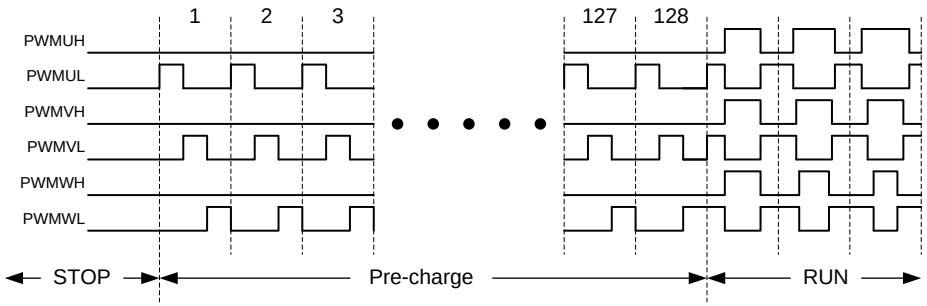


Figure 13: Bootstrap cap pre-charge

Total charge time for each phase can be calculated: $T_{Charge} = \frac{128}{3*F_{PWM}}$. For example, if PWM frequency is 10kHz, charge time of each phase will be: $\frac{128}{3*10000} = 4.267(ms)$.

Bootstrap capacitor value and charging impedance (serial resistor value) should be evaluated to be significantly shorter than total charge time to establish enough voltage on bootstrap capacitor before motor start to run.

2.8 Single Shunt Reconstruction

The IRMCK099 can use a single shunt current reconstruction circuit and methodology to minimize external analog and digital circuitry. In order to implement sensor-less field oriented control, it is crucial to measure the motor winding currents precisely. The single shunt current reconstruction method derives all necessary current feedback by sampling the currents in the shunt resistor, thus eliminating the need for isolation circuits or magnetic current sensors. The space vector modulator generates sample timing signals based on the power inverter state. The IC integrates the A/D converter and amplifier to sample the voltage across the shunt resistor.

The motor current reconstruction circuit measures the DC link current in the shunt resistor during the active vectors of the PWM cycle. Normally in each PWM cycle, there are two different active vectors and DC link current in each active vector represents current on one motor phase. The calculation of the third phase current value is possible because the three winding currents sum to zero.

2.8.1 Minimum Pulse Width

In single shunt reconstruction method, the current through one of the phases can be sensed across the shunt resistor during each active vector. However, under certain operating conditions i.e. when the resultant vector is at sector crossovers or when the length of the resultant vector is low (low modulation index); the duration of one or both active vectors is too narrow to guarantee reliable extraction of winding current data. These operating conditions are shaded in the space vector diagram shown in Figure 14. In order to guarantee reliable extraction of winding current, a minimum pulse width limit is imposed on each active vector in a PWM cycle. This minimum time is set by the MCE registers TcntMin. For an optimal control performance in this mode, SHDelay register must tuned. Please refer to section 5 for tuning procedure.

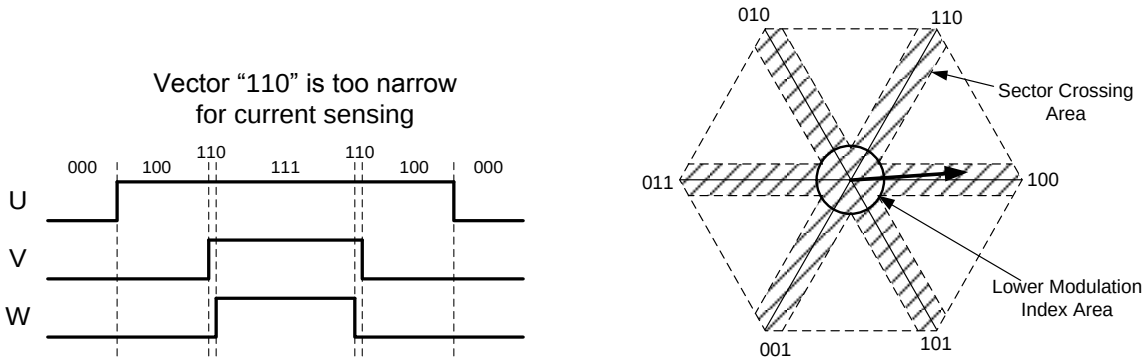


Figure 14: Narrow pulse limitation of single shunt current sensing

2.8.2 Minimum Pulse Limit and Phase Shift PWM

In single shunt configuration, motor current can only be sensed during active vector. In order to allow the ADC to accurately sense the current, each active vector must be ON for a minimum length of time. This imposes a limit on the minimum pulse width of each PWM cycle. This minimum pulse width restriction leads to distortion at lower modulation index or when the resultant voltage vector is transitioning from one sector to another. The resulting distortion may cause audible noise, especially at lower speeds. The shaded regions in the space vector diagram shown in Figure 14 mark the areas which introduce voltage distortion.

Figure 15 illustrates the resulting distortion when the resultant voltage vector is transitioning from one sector to another. Due to minimum pulse width limitation, there is a difference between target output and actual output at sector crossovers. This distortion results in acoustic noise and degradation of control performance.

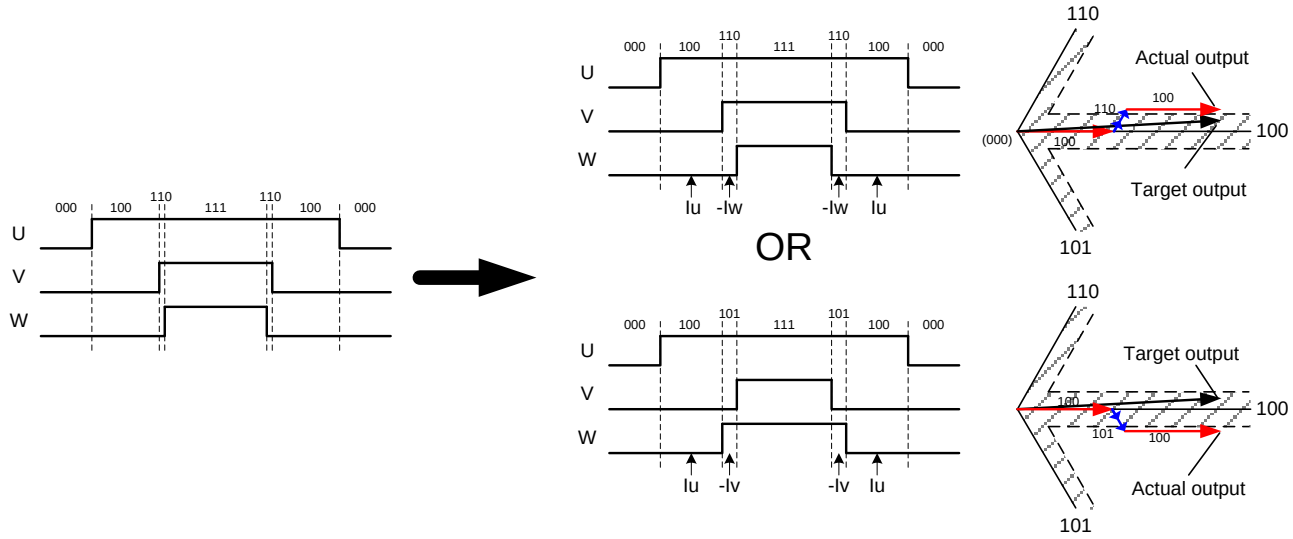


Figure 15: Minimum pulse limit scheme

IRMCK099 provides Phase Shift PWM to eliminate the minimum pulse limitation. In phase shift PWM, the output of each PMW is not always center aligned, it is shifted left to create enough ADC sample time. Figure 16 shows the W phase PWM has been shifted left to create enough time for vector “110”. It can be observed in Figure 16 that the PWM phase shift adds an additional active vector i.e. 101. However, the impact of this additional vector is cancelled due to extension of vector 110 and shrinking of vector 100.

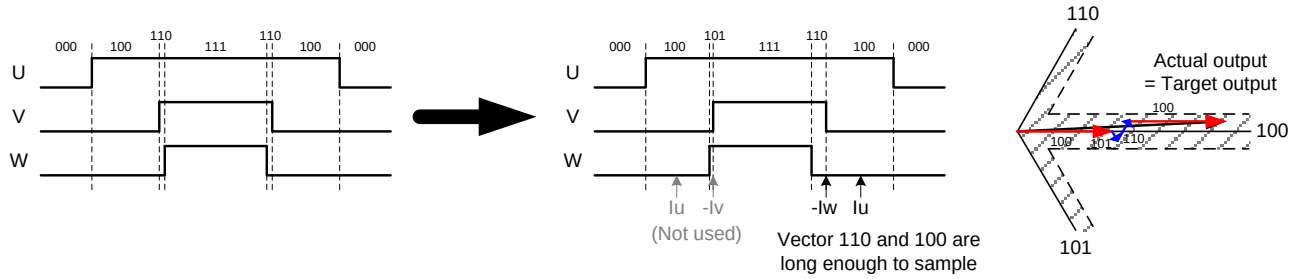


Figure 16: Phase Shift PWM scheme

By using phase shift scheme, the actual output during each cycle will be exactly the same as target output. Control performance at low speed can be improved. The amount of phase shift in the PWM is set by configuring TMinPhaseShift register. For an optimal control performance in this mode, TminPhaseShift and SHDelay register must tuned. Please refer to section 5 for tuning procedure.

2.9 State Handling and Start Control

IRMCK099 firmware manages the internal state and controls the state change. Totally there are 10 states, each state has a value between 0-9 in register SequencerState.

IDLE state (SequencerState=0): This is the first state that the control enters into right after the IC power up. The control will try to load the parameters from OTP parameter block, if it succeeds, bit 1 of register SysStatus will be set automatically and then exits this state. If there is no valid OTP parameter block, it will wait for the master controller to configure the parameters by UART command. After master controller configured all the parameters, master should also write 2 to register SysStatus to inform the control that the configuration process is finished.

STOP state (SequencerState=1): Motor stop state. In this state, if bit SysStatus[0] is not set which means current sensing offset has not been calculated, it will enter OFFSET_CAL state automatically. If bit SysStatus[0] is set, it is ready to start the motor.

OFFSET_CAL state (SequencerState=2): Offset calculation for motor current sensing. This state takes 4096 PWM cycles and averaging the motor current sensing. After 4096 PWM cycles, it goes back to STOP state.

BTS_CHARGE state (SequencerState=3): Boot strap capacitor pre-charge. Once the control receives motor start command, it will first go through this state to charge the boot strap capacitors.

MOTOR_RUN state (SequencerState=4): Normal motor run mode.

FAULT state (SequencerState=5): If there is fault condition, motor will stop (if it was previously running) and enter FAULT state. In UART control mode, fault will only be cleared by writing 1 to register "FaultClear". In other control modes, the control will keep trying to clear the fault after a 10 seconds delay time.

CATCH_SPIN state (SequencerState=6): In this state, flux estimator and flux PLL are working in order to detect the rotor position and measure the motor speed for a spinning motor. Speed regulator is disabled and the Id/Iq current commands are set to 0. At the end of this state, the control directly goes to RUN state if the measured absolute motor speed is above threshold (register DirectStartThr), or if the motor speed is below threshold, it will check register value "IS_Pulses" and decides goes to IS_SENSING state or PARKING state. Total time duration of this state is decided by register TCatchSpin. This state can be skipped if SysConfig[7] is set to 1.

PARKING state (SequencerState=7): Parking state is to align the rotor to a known position by injecting a linearly increased current. The vector angle of output current is configured by register ParkAngle. The final current amplitude is decided by low speed current limit. Total time duration of this state is configured by register ParkTime. It goes to OPEN_LOOP state after parking is done. This state can be skipped if PartTime is set to 0.

OPEN_LOOP state (SequencerState=8): This state is to move the rotor and accelerate from speed 0 to MinSpd by using open loop angle. Flux estimator and flux PLL are working in order to provide smooth transition to MOTOR_RUN state. Speed acceleration of the open loop angle is configured by register OpenLoopRamp. This state can be skipped if OpenLoopRamp is set to 0.

IS_SENSING state (SequencerState=9): This state is to measure the initial rotor angle by method "variable inductor sensing". Basically it applies 6 pulses of different angle to motor windings, then calculates the rotor angle by checking the relationship among the peak current of the 6 pulses. Please refer to Infineon application note AP08018 for detail of the algorithm. After this state, the rotor angle is measured and motor will start in closed loop. The length of each pulse is configured by register IS_Pulses (in PWM cycles). This state can be skipped if IS_Pulses is set to 0.

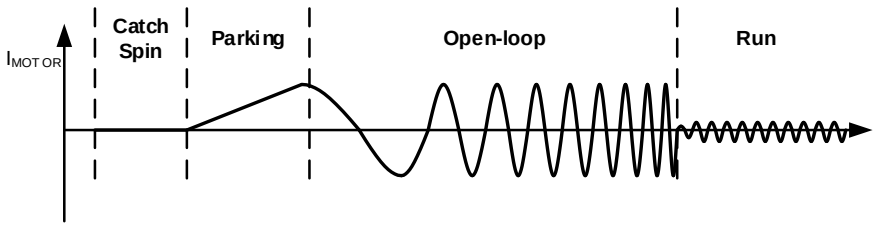


Figure 17: Motor start waveform

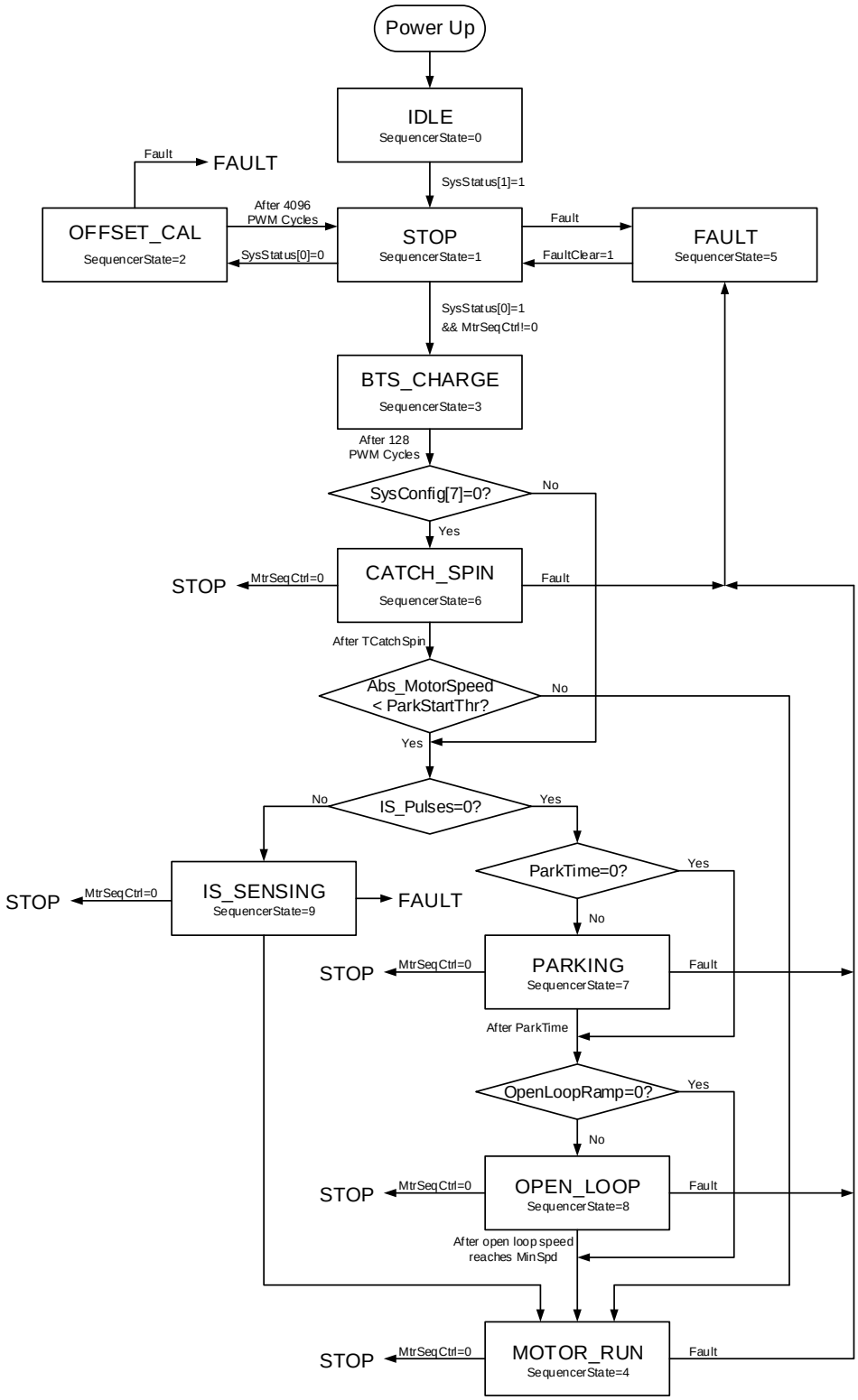


Figure 18: State handling and start control flow chart

2.10 Speed Control Loop and Sensorless FOC

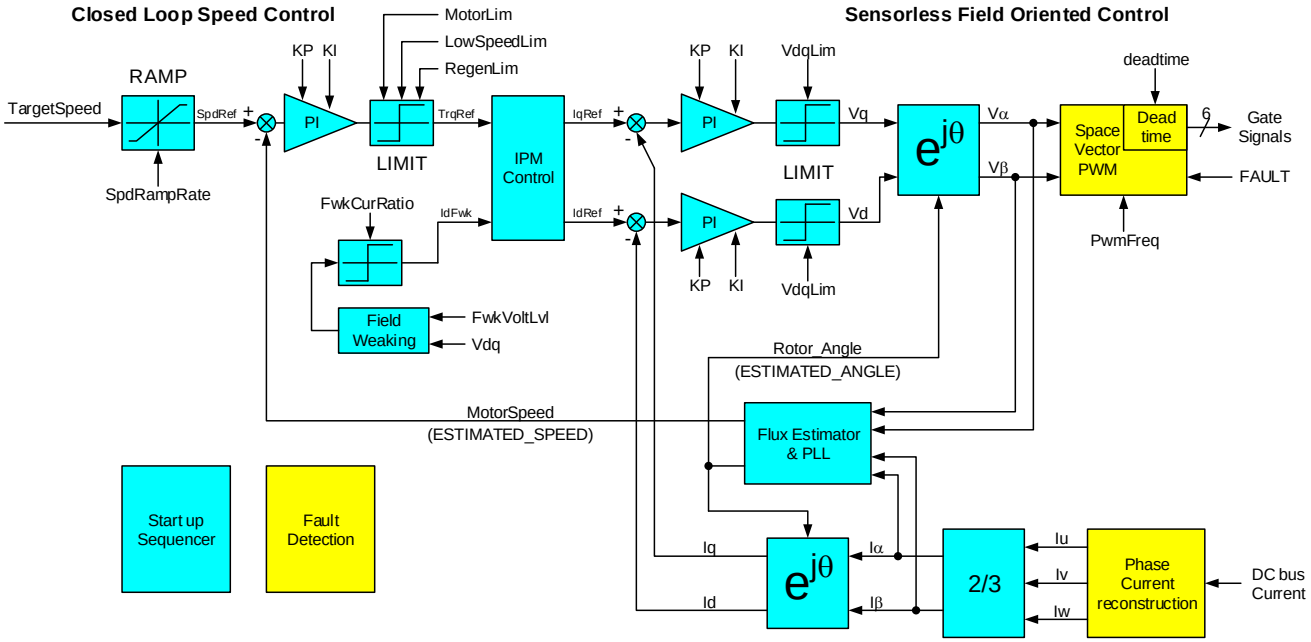


Figure 19: Top level diagram of speed control loop and sensorless FOC

2.11 Temperature Sensing and Protection

IRMCK099 has internal temperature sensor which sense the IC temperature, AIN2 can also be assigned as temperature input by connecting a 47kOhm NTC thermistor. Only one sensor is used as temperature protection feature, it can be selected by bit HwConfig[11].

Pull up and pull down resistors are important for sensing accuracy and maintain the appropriate input voltage to AIN2. Failing to do this may cause erroneous sensing temperature, most importantly; the ADC might not work properly.

The relationship between AIN2 voltage and NTC thermistor resistance:

$$R_{NTC} = V_{AIN2} \Big/ \left(\frac{1.8V - V_{AIN2}}{R_{High}} - \frac{1.8V}{R_{Low}} \right), (in\ K\Omega)$$

Or: $$V_{AIN2} = \left(\frac{R_{NTC}}{R_{High}} - \frac{R_{NTC}}{R_{Low}} \right) \cdot 1.8V \Big/ \left(1 + \frac{R_{NTC}}{R_{High}} \right), (in\ V)$$

Where R_{High} is pull up resistor value in $K\Omega$, it must be $4.87K\Omega$, and R_{Low} is pull down resistor value in $K\Omega$, it must be $9.76K\Omega$.

NTC thermistor temperature is calculated using below equation:

$$T_{NTC} = 1 \Big/ \left(\ln(\frac{R_{NTC}}{R_0}) / B + 1/(25 + 273.16) \right) - 273.16, (in\ ^\circ C)$$

Where R_0 is NTC thermistor resistance at 25°C, it must be $47K\Omega$, and B is NTC constant which must be 4050.

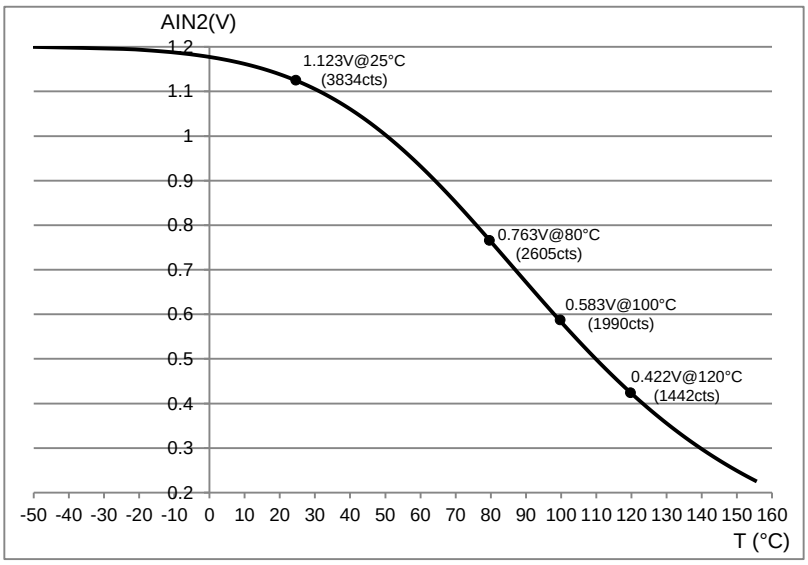


Figure 20: NTC temperature vs AIN2 input voltage

IRMCK099 features automatic motor current derating and thermal shutdown protection (See Figure 21). After temperature rise above Tderating, maximum motor current will begin to decrease to protect the controller and motor. Motor current will be set to 0 if temperature is above Tshutdown, if Over-Temperature Shutdown is enabled, motor will stop and report fault. If the current derating feature is not required or when there is no temperature sensor connected in the application board, user can set Tderating to be very high.

Motoring current, regenerative current, as well as field weakening current are all applied to motor current derating.

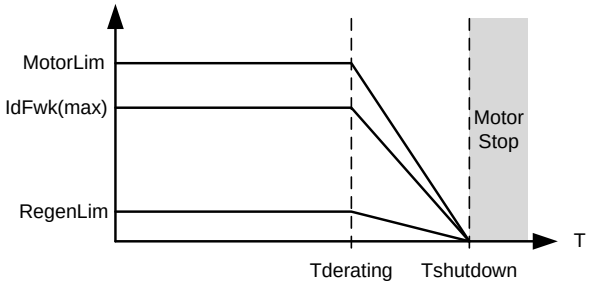
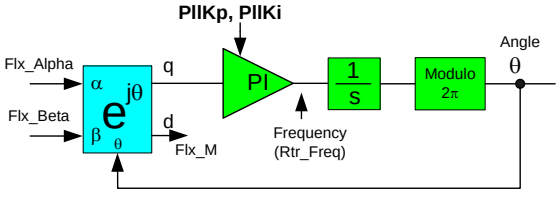


Figure 21: Motor current derating

2.12 Flux PLL out-of-control protection

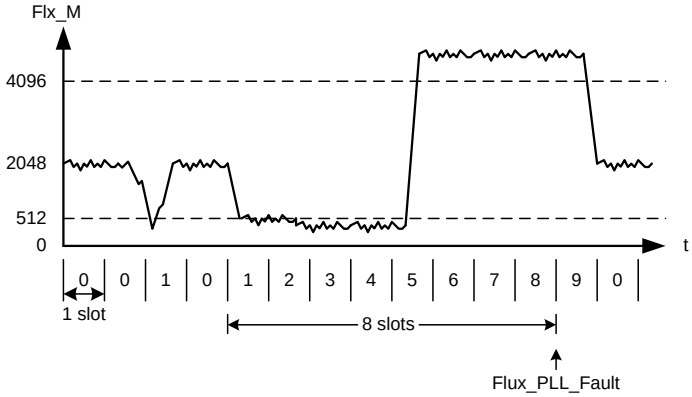


Simplified block diagram of a Flux PLL

Flux PLL out-of-control protection is the mechanism designed to detect this fault condition. When the PLL is locked to correct rotor angle, Flx_M, which is the output of vector rotate, should be a DC component and normalized at 2048 counts.

Instead, if the PLL is not locked to correct rotor angle, Flx_M becomes either unstable or its value is far off from 2048 counts.

MCE firmware keep monitoring Flx_M, within certain time slot (configured by SysConfig[9:8], if detected its value below 512 or above 4096, and if this happens in 8 continuous time slots, flux PLL is considered “out-of-control”. See Figure 22 for detail.



SysConfig[9:8]	Slot time (sec)	Total time (sec)
00	0.05	0.4
01	0.25	2
10	0.5	4
11	1	8

Figure 22: Flux PLL out-of-control protection

Flux PLL out-of-control protection can be enabled or disabled by set SwFaults_Mask[4] to 0 or 1.

This protection is also able to detect phase loss condition.

2.13 Zero speed protection

Zero speed protection is to protect when the motor is stuck at an angle and not able to spin. It could happen when the motor starts or in rotor lock condition. The reason why it happen is, when the motor is running at very low speed, the angle error becomes big. If the error is 90 degree and controller tries to apply torque current on Q axis, the actual current will be put on D axis which can cause motor to be stuck at a certain angle. Figure 23 illustrates the estimated angle and actual angle when rotor is stuck.

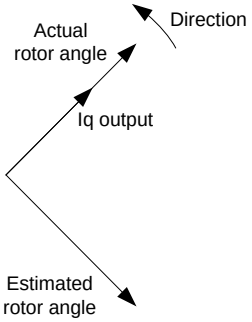


Figure 23: Wrong estimated rotor angle causes motor stuck

If this condition happens and zero speed protection is enabled, it will report the fault condition and stop the motor. If zero speed protection is disabled, the control will automatically adjust the rotor angle and exit from this condition.

Zero speed detection time is 2 seconds or 4 seconds which is configured by control bit SysConfig[10].

2.14 Rotor lock protection

Rotor lock fault is detected if speed PI output (TrqRef) being saturated for 10, 20, 30 or 60 seconds (configured by SysConfig[12:11]. When the motor speed is above 25% maximum RPM, rotor lock check is disabled, this is to avoid erroneous fault report at higher speed.

Rotor lock detection is not 100% guaranteed to report the fault especially when the motor is running at low speed. The reason is, in rotor lock condition, the PLL might be locked at higher speed which may not cause speed PI output to be saturated.

2.15 Low speed current limit

Some applications (such as fan) doesn't require high current at low speed, in another words, full torque is only required above certain speed. IRMCK099 provides low speed current limit feature which reduce current limit in low speed region. This feature provides smooth and quiet start up, and it also can reduce rotor lock current.

When motor speed is below minimum speed ($|MotorSpeed| \leq MinSpd$), motor current is limited by bit[7:0] of register "LowSpeedLim".

Motor current limit increases with motor speed increase, actual limit current calculated by MCE firmware and its gain is specified by bit [15:8] of register "LowSpeedLim".

When motor is running at high speed ($|MotorSpeed| \geq Low\ Speed\ Threshold$), motor limit current becomes "MotorLim".

Figure 24: Low speed current limit (Motoring limit) illustrates how low speed current limit works.

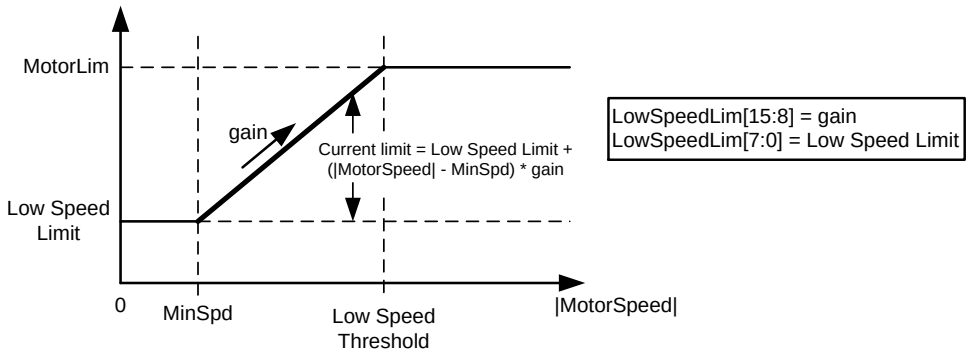


Figure 24: Low speed current limit (Motoring limit)

2.16 Catch Spin

"Catch-Spin" is a feature designed for situations where the motor may already be spinning. Catch-spin cannot be done if the motor back EMF voltage is higher than the DC bus voltage; this usually occurs when the motor is above rated speed. Hence, the catch-spin is generally effective up to the rated speed of the motor. The catch-spin starting process is part of the sequencer and executes at start-up if SysConfig[7] = 0.

In catch-spin, the controller tracks the back EMF in order to determine if the motor is turning, and if so, in which direction. Catch Spin sequence begins after the bootstrap capacitor charging stage is completed. During Catch-Spin, both IqRef and IdRef are set to 0, meanwhile flux PLL attempts to lock to the actual motor speed (MotorSpeed) and rotor angle (RotorAngle). Speed reference (SpdRef) is continuously updated to be the same as MotorSpeed. Catch spin time, defined by register TCatchSpin, has a default value of 50 (=0.5 second). Once catch spin time is elapsed, normal speed control starts, current motor speed will become the initial speed reference and also set as the speed ramp starting point. Depending on the set target speed, motor will decelerate (via regenerative braking) or accelerate to reach the desired speed.

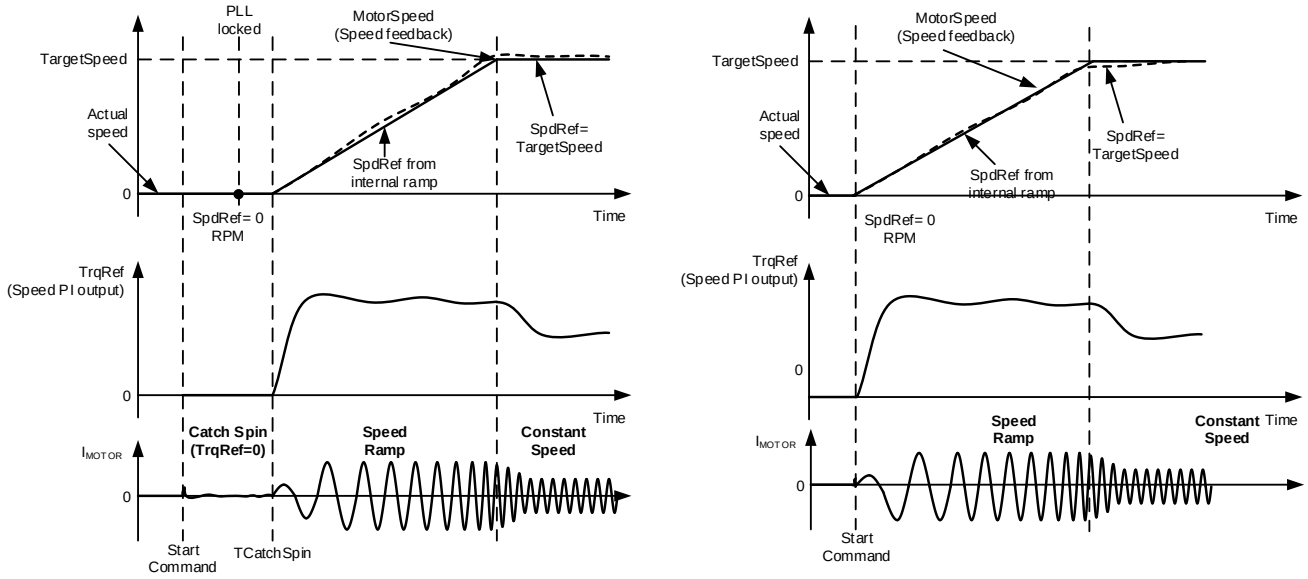
Depending upon the direction of rotation, there are 3 types of catch-spin sequences.

1. Zero Speed Catch-Spin.
2. Forward Catch-Spin.
3. Reverse Catch-Spin.

2.16.1 Zero Speed Catch Spin

If the motor is stationary, then the catch-spin sequence is termed as 'Zero Speed Catch-Spin'. Figure 25(A) shows an example for 'Zero Speed Catch – Spin'. In this example, at the start command, the motor is stationary. After the start command, 'Zero Speed Catch Spin' sequence begins. During the catch spin sequence, no motoring current is injected. After the catch spin time has elapsed, the motor speed at that instance (which is 0 RPM) becomes initial speed reference and starting point for speed ramp reference. The motor continues to accelerate, following the speed ramp reference to reach the set target speed.

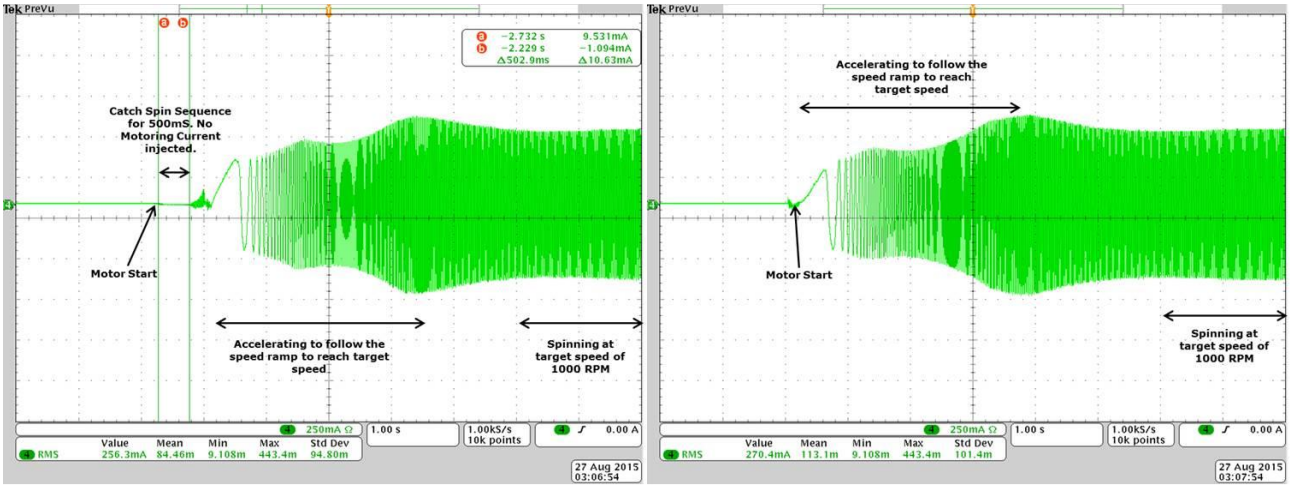
If catch spin is disabled, normal speed control starts immediately after the start command, without waiting for PLL to be locked. As shown in Figure 25(B), after the start command, motoring current is injected directly as there is no catch spin sequence. The motor starts accelerating, following the speed ramp reference to reach the set target speed.



A. Motor start with Catch Spin enabled
SysConfig[7]=0

B. Motor start with Catch Spin Disabled
SysConfig[7]=1

Figure 25: Zero Speed Catch Spin - Motor start with/without catch spin



A. Motor Start Catch Spin Enabled
SysConfig[7]=0

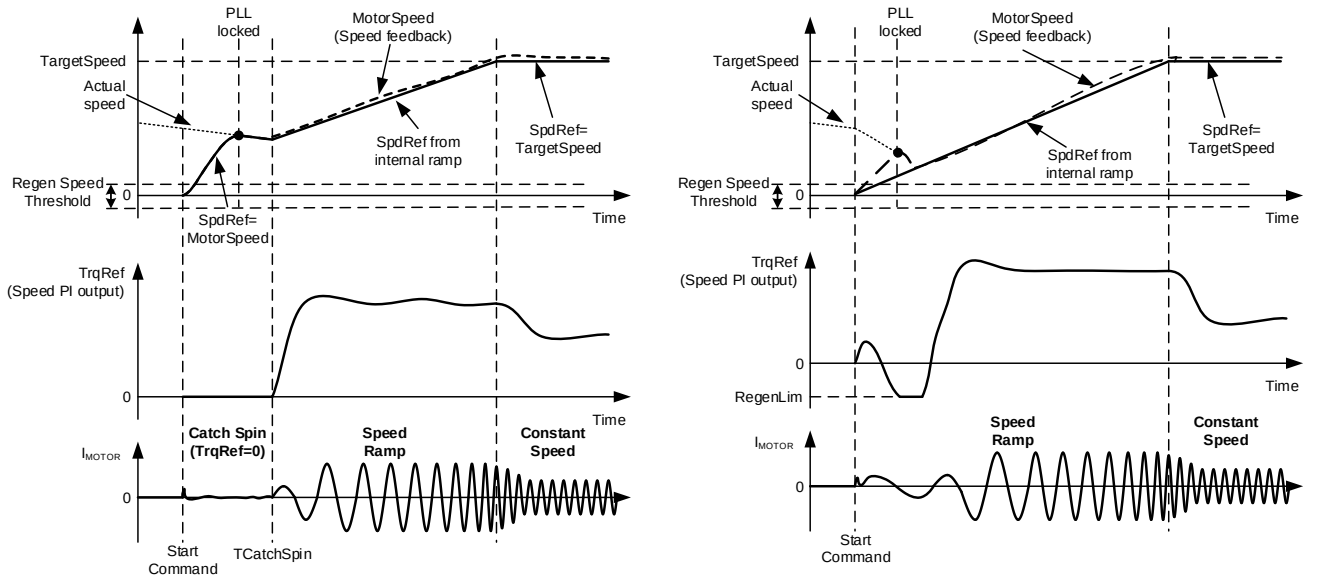
B. Motor Start Catch Spin Disabled
SysConfig[7]=0

Figure 26 Motor Phase Current Waveform - Zero Speed Catch Spin - Motor start with/without catch spin

2.16.2 Forward Catch Spin

If the motor is spinning in the same direction as desired, then the catch-spin sequence is termed as ‘Forward Catch-Spin’. Figure 27 (A) shows an example for ‘Forward Catch – Spin’. In this example, at the start command the motor is already spinning (in the desired direction). During the catch spin sequence, no motoring current is injected. After the catch spin time has elapsed, assuming the flux PLL locks to the actual motor speed, the motor speed at that instance becomes initial speed reference and starting point for speed ramp reference. The motor continues to accelerate or decelerate, following the speed ramp reference to reach the set target speed.

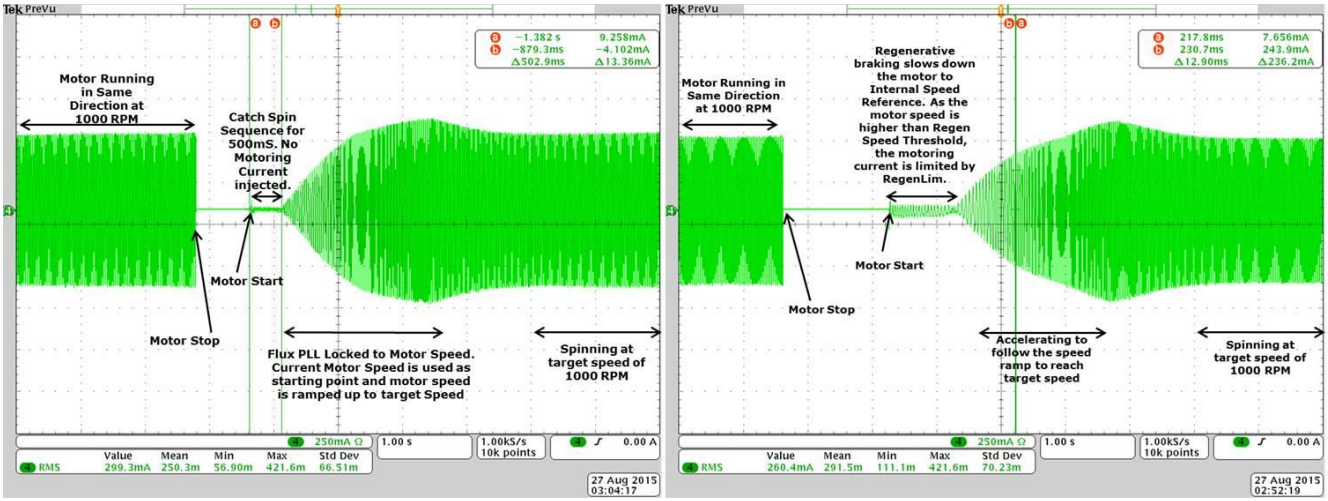
If catch spin is disabled, normal speed control starts immediately after the start command, without waiting for PLL to be locked. Usually the control would still be able to start a spinning motor, but motor speed may not increase/decrease seamlessly. As shown in Figure 27 (B), after the start command, the actual motor speed is higher than speed reference (SpdRef). Hence, the motor is decelerated (using regenerative braking) to force the motor to follow the speed reference (SpdRef). As the speed of the motor is higher than Regen Speed Threshold (RegSpdThr), the negative torque injected in the motor to achieve deceleration is limited by the value in RegenLim register. Once the motor speed matches the speed reference, the motor starts accelerating, following the speed ramp reference to reach the set target speed.



A. Motor start with Catch Spin enabled SysConfig[7]=0

B. Motor start with Catch Spin Disabled SysConfig[7]=1

Figure 27: Forward Catch Spin - Motor start with/without catch spin



A. Motor Start Catch Spin Enabled SysConfig[7]=0

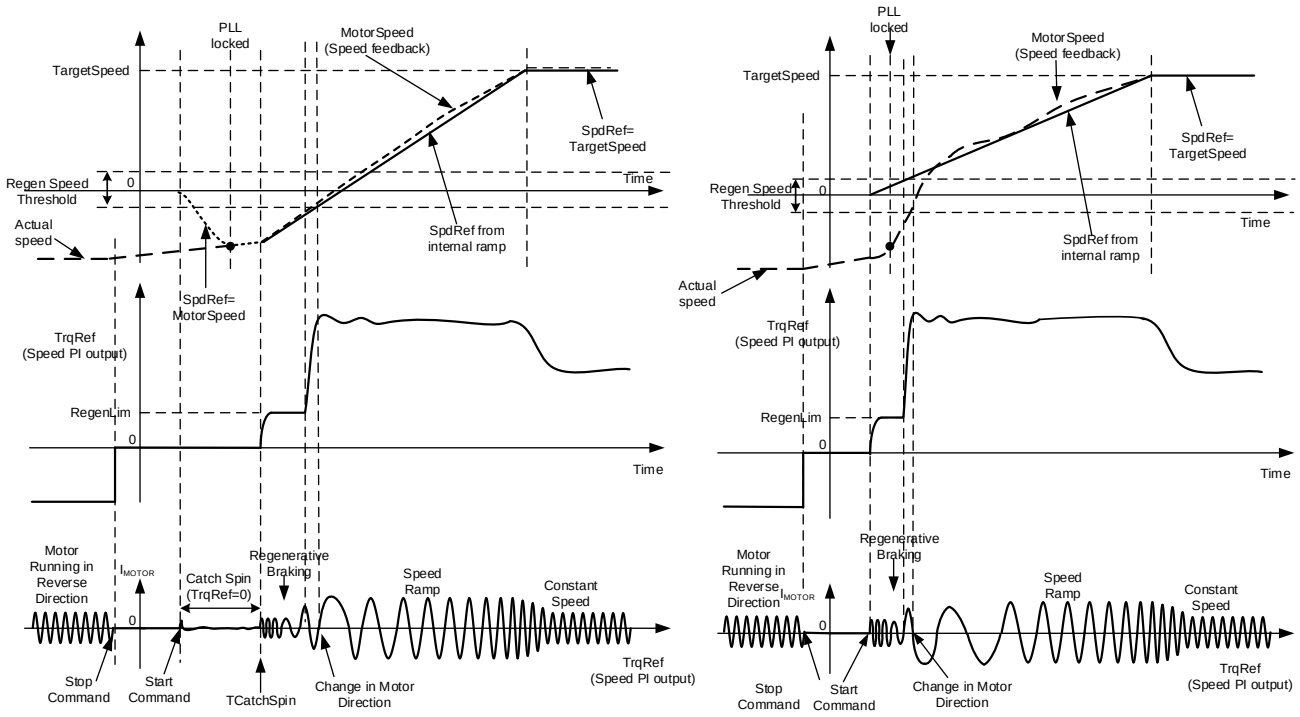
B. Motor Start Catch Spin Disabled SysConfig[7]=0

Figure 28 Motor Phase Current Waveform - Forward Catch Spin - Motor start with/without catch

2.16.3 Reverse Catch-Spin

If the motor is spinning in the opposite direction as desired, then the catch-spin sequence is termed as ‘Reverse Catch-Spin’. Figure 29 (A) shows an example of ‘Reverse Catch Spin’. In this example, at the start command, the motor is already spinning (in the opposite direction). During the catch spin sequence, no motoring current is injected. After the TCatchSpin time has elapsed, the motor is still spinning in opposite direction at a speed higher than Regen Speed Threshold (RegenSpdThr), thus an injected torque, limited by the value defined in RegenLim register, forces the motor to decelerate via regenerative braking. Once the speed of the reverse spinning motor falls below Regen Speed Threshold (RegenSpdThr), the injected torque is limited by MotorLim (RegenLim<=MotorLim). The injected torque forces the motor to come to a stop and start accelerating in the desired spin direction, following the speed ramp reference to reach the set target speed.

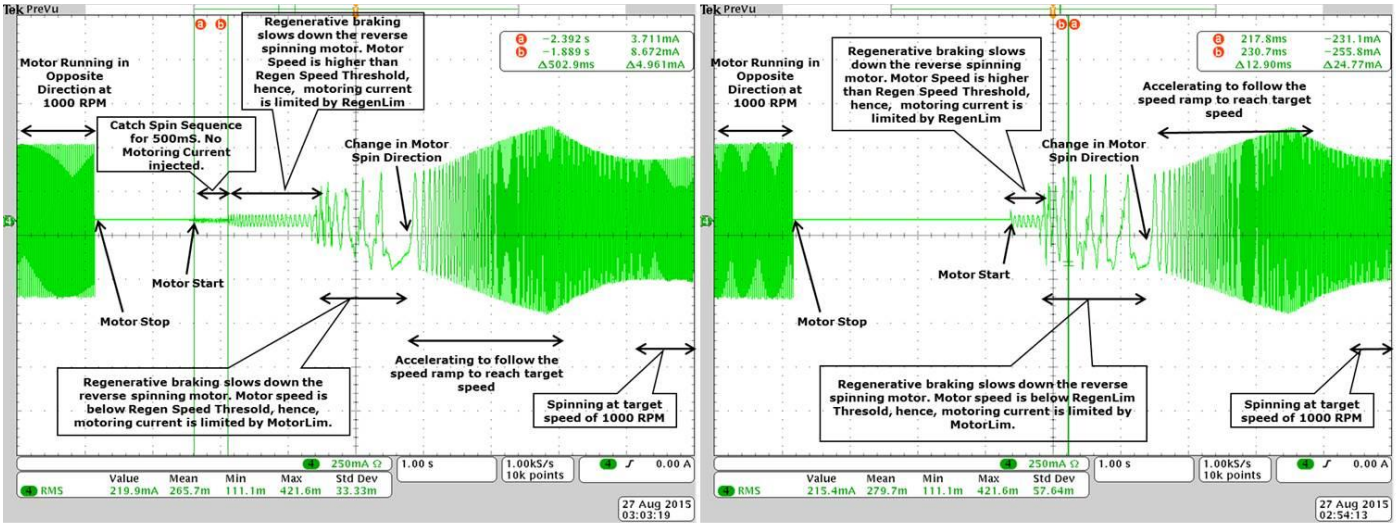
If catch spin is disabled, normal speed control starts immediately after the start command, without waiting for PLL to be locked. Usually the control would still be able to start a spinning motor, but motor speed may not increase/decrease seamlessly. As shown in Figure 27 (B), after the start command, the motor is still spinning at a speed higher than Regen Speed Threshold (RegenSpdThr), hence the injected torque limited by the value defined in RegenLim register, forces the reverse spinning motor to decelerate via regenerative braking. Once the speed of the reverse spinning motor falls below Regen Speed Threshold (RegenSpdThr), the injected torque is limited by MotorLim (RegenLim<=MotorLim). The injected torque forces the motor to come to a stop and start accelerating in the desired spin direction, following the speed ramp reference to reach the set target speed.



A. Motor start with Catch Spin enabled
SysConfig[7]=0

B. Motor start with Catch Spin disabled
SysConfig[7]=1

Figure 29 Reverse Catch Spin - Motor start with/without catch spin



A. Motor Start Catch Spin Enabled
SysConfig[7]=0

B. Motor Start Catch Spin Disabled
SysConfig[7]=0

Figure 30 Motor Phase Current Waveform - Reverse Catch Spin - Motor start with/without catch

3 Register Description

This chapter describes the MCE registers used in IRMCK099 MCE firmware.

3.1 System Register Group

3.1.1 HwConfig

Address:	*0x0BD*	*Size:*	*16 bits*	*Range:*	*Unsigned input with bit field definitions*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: Application hardware configuration register:

[15:13] not used

[12] **Enable OpAmp2**

This bit is to enable/disable Op-amp2 in single shunt configuration. Please note it should be set to 1 in leg shunt configuration because it is used as V phase current sensing.

0 Disable OpAmp2, IFBO2 pin used as regular AIN pin

1 Enable OpAmp2

[11] **TempSenseInput**

0 Internal temperature sensor

1 External NTC sensor on AIN2

[10:9] **Multiple Motor**

00 No multiple motor support, use most recent valid parameter block to configure the motor

01 Multiple motor support, MotorID is selected by AIN3

10 Multiple motor support, MotorID is selected by AIN0

11 Multiple motor support, MotorID will be configured by UART command

[8] **EnableStandby**

0 Disable low power standby mode

1 Enable low power standby mode (only available in Vsp control)

[7] **IPMPackage**

0 Standard pin assignments, for QFN32 discrete package only

1 Special pin assignments for IPM module which integrated with IRMCK099, not recommended for QFN32 discrete package

[6] **IPMPkgSel**

This bit is to select the package type of IPM. Different IPM package has different PWM pin definition (See Table 5 for detail). This bit is ignored if EnableTinyIPM=0.

0 Smart IPM in QFN package

1 Smart IPM in MiniSIP/MiniDIP package

[5] **JTAGAltFunction**

This bit is to select the alternate function on JTAG pins definition (See Table 5 for detail). This bit is ignored if EnableTinyIPM=0.

0 UART on JTAG

1 Capture/PG on JTAG

Pin name	Discrete QFN32			IPM: QFN12x12 package			IPM: MiniSIP/MiniDIP package		
	Pin #	HwConfig[7:5] =0x0	HwConfig[7:5] =0x1	Pin #	HwConfig[7:5] =100	HwConfig[7:5] =101	Pin #	HwConfig[7:5] =110	HwConfig[7:5] =111
		Pin Function	Pin Function		Pin Function	Pin Function		Pin Function	Pin Function
GPIO0	24	GK		N/A*	GK		Not in use		
GPIO1	25	OUT1		Not in use			N/A*	PWMWL	
GPIO2	26	DIR	OUT2					PWMVL	
GPIO3	27	OUT3						PWMUL	
GPIO4	28	CAPTURE						PWMWH	
GPIO5	29	PG						PWMVH	
GPIO6	30	RXD						PWMUH	
GPIO7	31	TXD						GK	
GPIO8	18	PWMUH		N/A*	PWMUL		Not in use		
GPIO9	19	PWMVH			PWMVL				
GPIO10	20	PWMWH			PWMWL				
GPIO11	21	PWMUL			PWMWH				
GPIO12	22	PWMVL			PWMVH				
GPIO13	23	PWMWL			PWMUH				
TDI	1	TDI		32	TDI/RXD	TDI/CAPTURE	15	TDI/RXD	TDI/CAPTURE
TCK	2	TCK		33	TCK	TCK	16	TCK	TCK
TMS	3	TMS		34	TMS/DE	TMS/DIR	17	TMS/DE	TMS/DIR
TDO	32	TDO		31	TDO/TXD	TDO/PG	14	TDO/TXD	TDO/PG

N/A*: These pins are connecting to internal gate driver, not available from external

Table 5 IRMCK099 pin mapping

[4] **Minimum Pulse Enable**

0 Use Phase Shift mechanisms for narrow pulse

1 Use Minimum Pulse mechanisms for narrow pulse

[3:2] **PWM mode**

00 3 phase PWM only

01 2 phase PWM type 1, for single shunt only

10 2 phase PWM type 2, for single shunt only

11 2 phase PWM type 3

[1] **Non-Inverting Amplifier**

0 Motor current sensing op-amp is configured inverting amplifier. Internal over-current comparator cannot protect over-current with inverting configuration. External over-current protection circuit must be used and connect to pin24.

1 Motor current sensing op-amp is configured as non-inverting amplifier. Internal over-current comparator is able to protect over-current with non-inverting configuration. For single shunt, there is no need to add external over-current protection circuit. For leg shunt, it still needs external over-current protection circuit.

[0] **ShuntConfig**

0 Single shunt current sensing

1 Leg shunt current sensing

3.1.2 SysConfig

Address: *0x0BE* *Size:* *16 bits* *Range:* *Unsigned input with bit field definitions* *Reset value:* *0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: System configuration register

[15] not used

[14] **LongMsgSel**

There are two types of long UART messages: 12 bytes and 16 bytes messages. This bit is to select long UART message type. Please note even it 16-bytes long message is selected, the return long message is always 12 bytes.

0 Long UART message is 12 bytes long (command number 8-11)

1 Long UART message is 16 bytes long (command number 8-13), return message is still 12 bytes

[13] **BaudRate_Ext**

This bit is to extend the UART baud rate range. See description of SysConfig[1:0] for detail.

[12:11] **Rotor Lock time**

00 Rotor lock detect time is configured by register TRotorLock, default = 10 seconds

01 If detected rotor lock, fault bit will be set after 20 seconds

10 If detected rotor lock, fault bit will be set after 30 seconds

11 If detected rotor lock, fault bit will be set after 60 seconds

[10] **Zero Speed Time**

0 Zero speed time is 2 seconds

1 Zero speed time is 4 seconds

[9:8] **PLL out-of-control slot time**

00 50ms each time slot, total detect time is 0.4 second

01 250ms each time slot, total detect time is 2 seconds

10 500ms each time slot, total detect time is 4 seconds

11 1000ms each time slot, total detect time is 8 seconds

[7] **Disable Catch Spin**

0 Catch Spin is enabled, catch spin time is configured by register TCatchSpin

1 Catch Spin is disabled

[6:5] **LinkBreakTimeOut**

00 Timeout = TLinkBreakBase, default is 5 seconds

01 Timeout = TLinkBreakBase x 2, default is 10 seconds

10 Timeout = TLinkBreakBase x 3, default is 15 seconds

11 Timeout = TLinkBreakBase x 6, default is 30 seconds

[4] **DcBusCompEn**

0 Disable DC bus compensation

1 Enable DC bus compensation

[3:2] **ControlInput**

00 Motor controlled by UART

01 Motor controlled by analog Vsp input

10 Motor controlled by frequency input

11 Motor controlled by duty cycle input

[1:0] **UART_BaudRate**

00 2400bps (BaudRate_Ext=0), or 115200bps (BaudRate_Ext=1)

01 9600bps (BaudRate_Ext=0), or 230400bps (BaudRate_Ext=1)

10 19200bps (BaudRate_Ext=0), or 256000bps (BaudRate_Ext=1)
11 57600bps (BaudRate_Ext=0), or 460800bps (BaudRate_Ext=1)

3.1.3 CmdStartStop

Address:	*0x112*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *With bit field definitions*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: In Vsp/Frequency/Duty Cycle input mode, this register specifies the input threshold for motor start and motor stop. This register combines two 8-bits parameters into one 16-bit register. The bit definition as below:

[15:8] **Threshold for motor start**

[7:0] **Threshold for motor stop**

It's recommended always use TinyWizard tool to calculate the value of this register.

3.1.4 CmdGain

Address:	*0x113*	*Size:*	*16 bits*	*Range:*	*0-65535*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: In Vsp/Frequency/Duty Cycle input mode, this register specifies the slope between the input threshold for motor start and threshold for MaxRPM.

It's recommended always use TinyWizard tool to calculate the value of this register.

3.1.5 MtrSeqCtrl

Address:	*0x13D*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0-4*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This register is the command control register. It controls the system state with the following values:

0 stop the motor

2 start the motor

3.1.6 AngleSelect

Address:	*0x1DE*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0-2*	*Reset value:*	*2*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This register selects the rotor angle from three sources:

0 Open loop angle, angle rotating speed is configured by register TargetSpeed. If TargetSpeed=0, open loop angle will be fixed but can be changed by writing a value to register RotorAngle.

2 Flux angle

3.1.7 CtrlModeSelect

Address:	*0x1F8*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-2*	*Reset value:*	*2*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This register selects one of three control modes:

0 Open loop voltage control mode, voltage command is Vd_Cmd and Vq_Cmd
1 Current control mode, current command is IdRef and IqRef
2 Speed control mode, speed command is TargetSpeed
3 Closed loop voltage control mode, voltage command is

3.1.8 SequencerState

Address:	*0x0C1*	*Size:*	*16 bits*	*Range:*	*0-5*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: This register reflects the main state of the firmware:

0 Power on state
1 Stopped state
2 Measuring offset current
3 Charging boot strap capacitors
4 Motor running
5 Fault state
6 Catch spin
7 Parking
8 Open loop acceleration

3.1.9 SysStatus

Address:	*0x0CB*	*Size:*	*16 bits*	*Range:*	*0-1024*	*Reset value:*	*0*

Register Type: MCE Read/Write Register.

Scaling or Notation: See description.

Description: This register reflects the main state of the FOC block:

[15:3] not used

[2] **TrqRef Reach Limit**

In “RUN” state, this bit set indicates whether TrqRef (speed PI output) reaches motoring current limit. Value of this bit will keep unchanged when the control is in “FAULT” state. This bit will be cleared when the control is in “STOP” state.

[1] **Parameter Configured**

[0] **AD Offset Calculated**

3.1.10 FW_Version

Address:	*0x110*	*Size:*	*16 bits*	*Range:*	*0-65535*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: MCE firmware version. It is updated by IR every time there is firmware change.

3.1.11 ParameterVer

Address:	*0x111*	*Size:*	*16 bits*	*Range:*	*0-255*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: Parameter version. Each MCE firmware version uses specified parameter version. When the IC powered up and tries to load parameter from parameter block, low byte of the parameter block header must match the parameter version. If the parameter version doesn't match, loading parameter from parameter block will fail and default value will be used.

3.1.12 CustomerID_1

Address:	*0x21E*	*Size:*	*32 bits*	*Range:*	*0-4294967295*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: Customer ID is a number pre-assigned for specific customer, customer may use this number to identify their application is using IRMCK099.

Value of CustomerID_1 = 0x00140098.

3.1.13 UserVersion

Address:	*0x220*	*Size:*	*24 bits*	*Range:*	*0x0 -0xFFFFFF*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: User version is a 24-bits value used to identify user's product version. User can assign any 24-bits value (0x000000 to 0xFFFFFF in hex) and program it into the parameter block.

3.1.14 MotorIDSel

Address:	*0x222*	*Size:*	*16 bits*	*Range:*	*0-255*	*Reset value:*	*256*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: If multiple motor support is enabled, this register contains the desired MotorID which is configured by AIN3, AIN0 or select by UART.

Only 8 motor IDs are supported if using AIN3 or AIN0, the valid MotorID is 0-7.

Any MotorID between 0 and 255 is supported if using UART command to write to this register.

After this register is set, MCE firmware will search all the parameter blocks, the most recent parameter block which has matching MotorID will be used to configure the parameters.

3.2 Start Control Register Group

3.2.1 DirectStartThr

Address:	*0x115*	*Size:*	*16 bits*	*Range:*	*0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 16384 = Motor Max RPM

Description: At the end of catch spin state, this register is the absolute motor speed threshold that decides whether to go through IS_Sensing + Parking + OpenLoop start or directly go to closed loop run state.

If DirectStartThr=0, after catch spin, it will directly go to closed loop run state.

3.2.2 ParkAngle

Address:	*0x116*	*Size:*	*16 bits*	*Range:*	*-32768-32767*	*Reset value:*	*5461*

Register Type: MCE Write Register.

Scaling or Notation: -32768→+32767 is -180°→+180°

Description: This register configures the current angle during parking state. Reset value of parking angle is set to 5461 (30°) which is the center of sector 0.

3.2.3 ParkTime

Address:	*0x117*	*Size:*	*16 bits*	*Range:*	*0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 0.01 second

Description: This register configures total parking time. During parking state, parking current increases linearly from 0 to low speed current limit (register LowSpeedLim[7:0]).

If ParkTime=0, parking state will be skipped.

3.2.4 OpenLoopRamp

Address:	*0x118*	*Size:*	*16 bits*	*Range:*	*Unsigned input*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This register configures the open loop acceleration rate. During open loop state, motor current is regulated at low speed current limit (register LowSpeedLim[7:0]), rotation speed accelerates linearly from 0 to MinSpd.

Total duration of open loop:

$$T_{OpenLoop} = \frac{MinSpd*1024*10}{OpenLoopRamp*PwmFreq}$$, in 1 millisecond

If OpenLoopRamp=0, open loop state will be skipped.

3.2.5 IS_Pulses

Address:	*0x11F*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0-8191*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 1 sensing pulse under nominal DC bus voltage (DcBusVolts=2048)

Description: This register specifies the number of PWM cycles for each inductor sensing pulse under nominal DC bus voltage. Actual number of PWM cycles is DC bus compensated in order to keep constant volt-second which in turns keep the same peak sensing current.

Inductor sensing measures the current of last 2 PWM cycles, if the register is configured at 1, only one PWM cycle will actually carry current. So it is advised to configure this register ≥ 2.

Write 0 to this register disable the inductor sensing feature.

3.2.6 IS_IqInit

Address:	*0x123*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-8191*	*Reset value:*	*1024*

Register Type: MCE Write Register.

Scaling or Notation: 4096 = 100% motor rated RMS current

Description: This register specifies the initial torque been applied after done IS_SENSING stage and before entering MOTOR_RUN.

Right after IS_SENSING stage, the flux PLL has not locked to the rotor angle, it takes some time and also needs motor speed to be high enough. This means in the beginning of MOTOR_RUN state, flux PLL is not working properly and it relies on initial torque to accelerate the motor in order for the PLL to lock. To achieve reliable and smooth start, some tuning to flux estimator and flux PLL is required.

3.2.7 IS_Duty

Address:	*0x124*	*Size:*	*16 bits*	*Range:*	*Signed input 0-8191*	*Reset value:*	*4096*

Register Type: MCE Write Register.

Scaling or Notation: 8192 = 100% PWM duty cycle

Description: This register specifies the PWM duty cycle during IS_SENSING. For better current sensing quality, in single shunt current sensing, duty cycle of inductor sensing should not be too low otherwise active vector will be too short to sense the current. In leg shunt current sensing, duty cycle should not be too high otherwise there will not be enough time to sense the current during 000 zero vector.

3.2.8 TCatchSpin

Address:	*0x1F2*	*Size:*	*16 bits*	*Range:*	*Signed input 0-32767*	*Reset value:*	*50*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 0.01 second

Description: This register specifies the catch spin synchronization time duration before engaging motor start acceleration. During this period, the internal controller tries to sync rotor position with zero torque current reference. The power up reset value is 50, which means 0.5 second catch spin time.

Catch spin can be disabled by set register SysConfig[7] to 1, or write value 0 to TCatchSpin.

3.3 PWM Register Group

3.3.1 PwmFreq

Address:	*0x0EA*	*Size:*	*16 bits*	*Range:*	*Unsigned input*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: 1 = 0.1kHz F_{PWM}. 160 = 16kHz F_{PWM}
Description: This register configures the motor PWM frequency in 0.1kHz increment.

3.3.2 deadtime

Address:	*0xE87*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-1023*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: $1 = 10ns$
Description: This register configures PWM dead time.

3.3.3 active_pol

Address:	*0xE88*	*Size:*	*6 bits*	*Range:*	*Unsigned input with bit field definitions*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: See description.
Description: This register configures active polarity for PWM outputs.

[5:4] not used, must be set to "0"
[3] **PWM1H**
0 High side PWM active low
1 High side PWM active high
[2:1] not used, must be set to "0"
[0] **PWM1L**
0 Low side PWM active low
1 Low side PWM active high

3.4 Current Feedback Register Group

3.4.1 TMinPhaseShift

Address:	*0x020*	*Size:*	*16 bits*	*Range:*	*0-4096*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: 65536 = 100% T_{PWM}
Description: In Phase Shift PWM mode, TMinPhaseShift configure the minimum time of an active vector for single shunt current sensing.

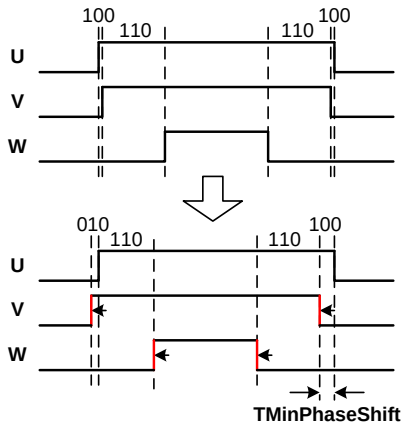


Figure 31: TMinPhaseShift

3.4.2 SHDelay

Address:	*0x024*	*Size:*	*16 bits*	*Range:*	*0-65535*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 32768 = 100% T_{PWM}

Description: SHDelay specifies the time delay from PWM output to ADC sample time for current sensing. The delay time is depending on the hardware design, usually it should consider propagation delay of gate driver circuit and turn on (turn off) delay of switching devices.

In Phase Shift PWM mode, in order to avoid sample the shunt resistor signal while device is switching, SHDelay should be configured smaller than actual hardware delay. Some board design may allow bigger SHDelay value without causing much current sensing noise (bigger SHDelay value may help for a smaller TMinPhaseShift value).

In Minimum Pulse PWM mode, SHDelay should be configured same as actual hardware delay.

3.4.3 TCntMin

Address:	*0x10D*	*Size:*	*16 bits*	*Range:*	*0-2048*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 16384 = 100% T_{PWM}

Description: This parameter specifies the minimum PWM pulse width if minimum pulse width method is being used.

3.4.4 Pwm2PhThr

Address:	*0x10A*	*Size:*	*16 bits*	*Range:*	*0-65535*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 16384 = Motor Max RPM

Description: Switch over speed from 3 phase PWM to 2 phase PWM. When the motor's absolute speed reach or above Pwm2PhThr, PWM will change to the mode configured in HwConfig[3:2].

When the motor speed reduced to below Pwm2PhThr-256, PWM scheme will return to 3 phase PWM.

If the value of Pwm2PhThr is 256 or below, and HwConfig[3:2] is configured 1,2 or 3, after PWM mode change to 2 phase PWM, it will not return to 3 phase PWM automatically unless stop the motor and start again.

3.4.5 PwmGuardBand

Address:	*0x114*	*Size:*	*16 bits*	*Range:*	*0-1023*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 10ns

Description: In leg shunt configuration, this parameter provides a guard band such that PWM switching at high modulation cannot migrate into the beginning and end of a PWM cycle. The guard band insertion can improve feedback noise immunity for signals sampled near the beginning and end of a PWM cycle.

Guard band insertion will reduce the maximum achievable inverter output voltage.

3.4.6 Iu

Address:	*0x0C5*	*Size:*	*16 bits*	*Range:*	*Signed output -2048-2047*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts

Description: This register provides reconstructed motor phase U current (offset eliminated and ADC compensated).

This current is calculated from the DC bus link current feedback (single shunt configuration) or U phase shunt resistor (leg shunt configuration). It is samples on every PWM cycle.

3.4.7 Iv

Address:	*0x0C6*	*Size:*	*16 bits*	*Range:*	*Signed output -2048-2047*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts

Description: This register provides reconstructed motor phase V current (offset eliminated and ADC compensated).

This current is calculated from the DC bus link current feedback (single shunt configuration) or V phase shunt resistor (leg shunt configuration). It is samples on every PWM cycle.

3.4.8 Iw

Address:	*0x0C7*	*Size:*	*16 bits*	*Range:*	*Signed output -2048-2047*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts

Description: This register provides reconstructed motor phase W current (offset eliminated and ADC compensated).

This current is calculated from Iu and Iv by equation $Iw = -(Iu + Iv)$. Its value is updated on every PWM cycle.

3.5 DC Bus Voltage Register Group

3.5.1 DcBusVoltsFilt

Address:	*0x065*	*Size:*	*16 bits*	*Range:*	*Unsigned output 0-4095*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts.

Description: This register provides filtered (T_{PWM}*32 time constant) and AD compensated DC bus voltage feedback.

3.5.2 DcBusReport

Address:	*0x22D*	*Size:*	*16 bits*	*Range:*	*0-65535*	*Reset value:*	*12350*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: DC bus report value. Voltage unit of DC bus report is configurable by register VoltReportScl. MCE firmware calculates the report value as below:

$$DcBusReport = DcBusVoltsFilt * VoltReportScl * 2^{-10}$$

For example, if 10mV/count is wanted for DcBusReport, with 2MOhm/4.87KOhm DC bus voltage divider, VoltReportScl should be 12350.

3.5.3 DcBusFilt2

Address:	*0x0B7*	*Size:*	*16 bits*	*Range:*	*0-4095*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts.

Description: This register provides second filtered and AD compensated DC bus voltage feedback. The second filtered DC bus feedback has configurable low pass filter time constant (see description of write register "DcBusFilt2Tau") and it is being used by DC bus voltage protection. Please note it is not being used by FOC control, which means changing low pass filter time constant doesn't cause any changes to motor control performance.

3.5.4 DcBusFilt2Tau

Address:	*0x17E*	*Size:*	*16 bits*	*Range:*	*1-16384*	*Reset value:*	*512*

Register Type: MCE Write Register.

Scaling or Notation: $\tau = \frac{16384}{DcBusFilt2Tau}, in\ T_{PWM}$

Description: Low pass filter time constant for second DC bus feedback. The input of the low pass filter is DC bus feedback raw value.

3.6 ADC Register Group

3.6.1 Ain0Filt

Address:	*0x0AB*	*Size:*	*16 bits*	*Range:*	*Unsigned output 0-4095*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts.

Description: Filtered (T_{PWM}*1024 time constant) and AD compensated AIN0 analog input.

AIN0 is used by MCE firmware as Vsp speed input.

3.6.2 Ain2Filt

Address:	*0x0AD*	*Size:*	*16 bits*	*Range:*	*Unsigned output 0-4095*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts.

Description: Filtered (T_{PWM}*1024 time constant) and AD compensated AIN2 analog input.

AIN2 is used by MCE firmware as external temperature sensor input.

3.6.3 Ain3Filt

Address:	*0x0B9*	*Size:*	*16 bits*	*Range:*	*Unsigned output 0-4095*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts.

Description: Filtered (T_{PWM}*1024 time constant) and AD compensated AIN3 analog input.

AIN3 is not currently used by MCE firmware. Users can use it for their own purpose and read its value from this register. Please note that the input voltage should not exceed 1.2V.

3.6.4 OpAmp2

Address:	*0x178*	*Size:*	*16 bits*	*Range:*	*Unsigned output 0-4095*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: In ADC counts.

Description: In single shunt configuration, OpAmp2 contains the ADC result of IFBO2 pin (second op-amp). It is not currently used by MCE firmware. Users can use it for their own purpose and read its value from this register. Please note that the input voltage should not exceed 1.2V. Second op-amp can be enabled/disabled by control bit HwConfig[5].

In leg shunt configuration, second op-amp is used for V phase current sensing. The value of this register will not be updated.

3.6.5 TempSense

Address:	*0x0FF*	*Size:*	*16 bits*	*Range:*	*Signed output*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: 1 = 0.25°C

Description: This register is the calculated temperature in Celsius. If HwConfig[11]=0, it is the temperature of IRMCK099's internal sensor. If HwConfig[11]=1, it is the external NTC thermistor temperature.

3.7 Sensorless FOC Register Group

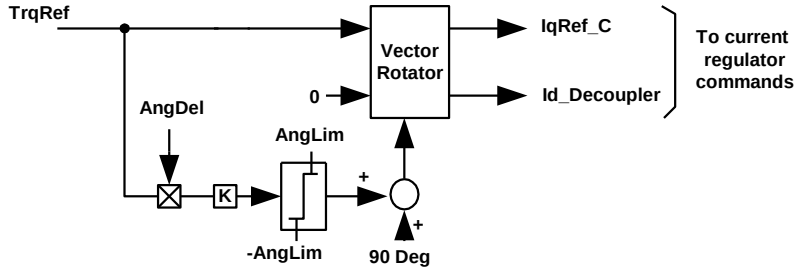
3.7.1 AngDel

Address:	*0x05C*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-255*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: $Angle\ advancement = AngDel \times 0.35156 \times \frac{I\_Motor}{Rated\ Motor\ Amps}, in\ degree$

Description: This parameter provides gain adjustment for current angle advancement. The current angle advancement is added to a fixed defaulted phase (90 Deg) and the rotor angle to form the relative phasing of the current vector. Current angle advancement is required for Permanent Magnet motor with rotor saliency (Interior Permanent Magnet Motors). A value of zero represents zero angle advancement and therefore the current vector is placed at 90 degrees with respect to the rotor angle. Details on angle advancement function are given in the Application Developer's Guide under the Interior Permanent Magnet Motor Control section. Diagram below shows the implementation of the angle advancement function and the related controller parameters.



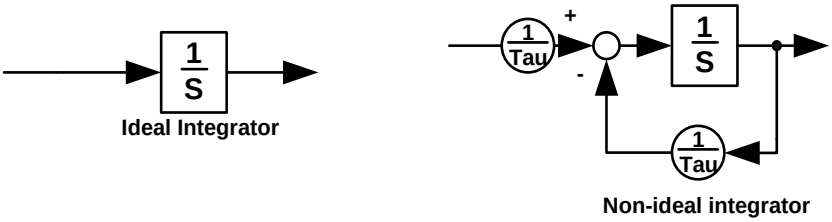
3.7.2 FlxTau

Address:	*0x06C*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-8191*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: Motor flux is calculated by integration of estimated voltages. Pure (ideal) integrator cannot be used due to dc offset problem. The integration is done using non-ideal integrator (low pass filter) as shown in the diagram below. The flux integration time constant (Tau) is an entry of the iMotion TinyWizard. Typical range of non-ideal integrator time constant is in the range of 0.01 to 0.025 sec.



This parameter provides the adjustment for the flux estimator bandwidth. FlxTau is inversely proportional to the "Flux estimator time constant" entered in TinyWizard. The relationship of the Flux estimator time constant and FlxTau is given by:

$$Flux\ estimator\ time\ constant = \frac{2^{18} \times T_{PWM}}{FlxTau} - T_{PWM}, in\ seconds$$

where FlxTm is the Flux estimator time constant,

PwmPeriod = 1/(PWM switching frequency).

FlxTm and PwmPeriod are in seconds.

This parameter is also used as low pass filter time constant for rotor frequency (Rtr_Freq, which is the output of flux PLL) as well as some internal filtering.

3.7.3 AngLim

Address:	*0x071*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-255*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: $1 = 0.17578\ degree$

Description: This parameter provides the maximum limit on the current angle phase advancement specified by register AngDel. (See also AngDel.)

3.7.4 PllKp

Address:	*0x086*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-8191*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This parameter specifies the Angle Frequency Generator tracking proportional gain. The Angle Frequency Generator is mainly a phase lock loop (PLL) device. A larger value of PllKp will increase tracking bandwidth at the expense of increasing speed or frequency ripple.

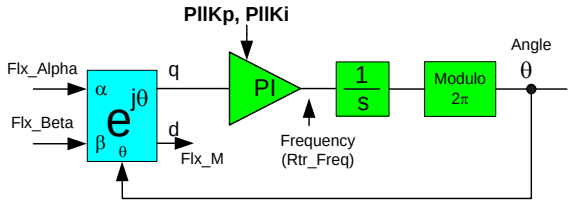
3.7.5 PllKi

Address:	*0x087*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-8191*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This parameter specifies the Angle Frequency Generator tracking integral gain. The Angle Frequency Generator is mainly a phase lock loop (PLL) device. The diagrams below show both a simplified and the detailed PLL architecture. As can be seen in this diagram, PllKi relates internal PLL tracking error (q) to frequency (Rtr_Freq). A larger value of PllKi will increase tracking bandwidth at the expense of increasing speed or frequency ripple.



Simplified block diagram of a Flux PLL

3.7.6 KpIregD

Address:	*0x094*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: $modulation\ index = KpIregD \times 0.01748 \times \frac{IdErr}{RatedMotorAmps}, [\%]$

Where: IdErr is absolute current error of d-axis in Amps

Description: This parameter specifies the proportional gain of the d-axis current regulator. The parameter relates current error to modulation index. 100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding rms motor line to line voltage at 100% modulation is $V_{DC}/\sqrt{2}$.

3.7.7 KpIreg

Address:	*0x095*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: $modulation\ index = KpIreg \times 0.01748 \times \frac{IqErr}{RatedMotorAmps}, [\%]$

Where: IqErr is absolute current error of q-axis in Amps

Description: This parameter specifies the proportional gain of the d-axis current regulator. The parameter relates current error to modulation index. 100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding rms motor line to line voltage at 100% modulation is $V_{DC}/\sqrt{2}$.

3.7.8 rotor_angle_adv

Address:	*0x09B*	*Size:*	*16 bits*	*Range:*	*Signed Input -32768-32767*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: -32768→+32767 is -180°→+180°

Description: If AngleSelect is set to 0, use this register to specify the internal open loop angle.

3.7.9 AtanTau

Address:	*0x0A5*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description

Description: This parameter provides angle compensation (frequency dependent) for the phase shift introduced by flux integration time constant (see register FlxTau). Inside the Sensorless FOC module, frequency (Rtr_Freq) is multiplied by a time constant (AtanTau) to form a compensating angle. This angle represents the phase shift introduced by the non-ideal flux integrators (low pass filter). Pure (ideal) integrator cannot be used due to dc offset problem. The flux integration time constant is an entry of the iMotion TinyWizard. Typical range of integrator time constant is in the range of 0.01 to 0.025 sec.

3.7.10 KxIreg

Address:	*0x0A9*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description

Description: This parameter specifies the integral gain of the d-axis and q-axis current regulator. The parameter relates current error-second (current error integration) to modulation index. The scaling depends on the current regulator execution rate which is directly related to the pwm frequency.

3.7.11 IdqFiltBw

Address:	*0x126*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-16383*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: $\tau = \frac{16384}{IdqFiltBw}, in\ T_{PWM}$

Description: This register configures the low pass filter time constant for Id and Iq.

3.7.12 PllFreqLim

Address:	*0x12A*	*Size:*	*16 bits*	*Range:*	*0-255*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description

Description: This parameter specifies the frequency limit of the PLL integral gain output. The relationship between the actual frequency in Hz and this parameter is given by:

$A = \frac{PllFreqLim \cdot FreqPwm}{8192}$, in Hz

where:

A is the actual frequency in Hz;

FreqPwm is the inverter pwm frequency in Hz;

3.7.13 VdqLim

Address:	*0x137 (MCE)*	*Size:*	*16 bits*	*Range:*	*unsigned input 0 – 4974*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 4974 = 100 [% modulation]

Description: This parameter specifies the current regulator output limit. 100% modulation corresponds to the maximum achievable value of the SVPWM. Please note 100% modulation exceeds SVPWM linear range and output will be over-modulated. If over modulation is not expected, maximum modulation should be limited below 86.6% (4974*86.6%=4307). The corresponding rms motor line to line voltage at 100% modulation is $V_{DC}/\sqrt{2}$.

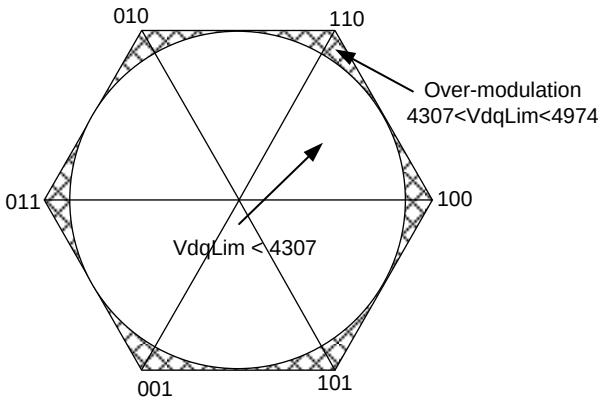


Figure 32: Over modulation

3.7.14 TargetSpeed

Address:	*0x15E*	*Size:*	*16 bits*	*Range:*	*Signed input/output -32768-32767*	*Reset value:*	*2048*

Register Type: MCE Write Register.

Scaling or Notation: 16384 = Motor Max RPM

Description: Target motor speed.

If the motor is running in Vsp analog input, frequency input or duty cycle control, this register will be updated by MCE firmware and writing to it has no effect.

3.7.15 FwkKx

Address:	*0x16E*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-16384*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This register configures the gain of field weakening.

3.7.16 FwkCurRatio

Address:	*0x16F*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-4096*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 4096 = 100% MotorLim.

Description: This register limits the –Id current for field weakening.

3.7.17 FwkVoltLvl

Address:	*0x170*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-16384*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 4974 = 100% PWM duty cycle.

Description: This parameter specifies the modulation threshold to start field weakening. It must be set below VdqLim and it's also recommended to set this value below SVPWM linear range (4974*0.866=4307). Lower threshold gives more voltage margin which provides better control performance but it will enter field weakening mode earlier.

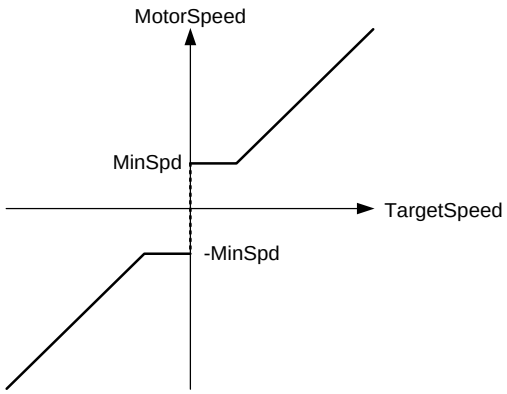
3.7.18 MinSpd

Address:	*0x1A2*	*Size:*	*16 bits*	*Range:*	*0-16384*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 16384 = Motor Max RPM

Description: This register configures the minimum motor speed. Motor will run at MinSpd when the target motor speed is below MinSpd.



3.7.19 KpSreg

Address:	*0x1C6*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: See description.
Description: This parameter specifies the proportional gain of the speed regulator.

3.7.20 KxSreg

Address:	*0x1C8*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: See description.
Description: This parameter specifies the integral gain of the speed regulator.

3.7.21 MotorLim

Address:	*0x1CC*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-16383*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: 4096 = 100% motor rated current
Description: This parameter specifies the maximum total motor current (d axis and q axis).

3.7.22 LowSpeedLim

Address:	*0x1CD*	*Size:*	*16 bits*	*Range:*	*Unsigned input With bit field definitions*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: See description.
Description: This register specifies the motor current limit at low speed. This register combines two 8-bits parameters into one 16-bit register. The bit definition as below:

[15:8] **Gain**

[7:0] **Low Speed Limit**

When the motor speed is within low speed limit range (between MinSpd and Low Speed Threshold), actual motor current limit is calculated by MCE firmware:

$$I_{LIMIT} = LowSpeedLimit * 2^4 + Gain * (|MotorSpeed| - MinSpd) * 2^{-5}$$

See 2.15 for detail description of this feature.

3.7.23 RegenLim

Address:	*0x1CF*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-16383*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 4096 = 100% motor rated current

Description: This parameter specifies the maximum total motor current (d axis and q axis) while motor is running in regenerative mode.

RegenLim should be set to a low value if the drive has no break resistor otherwise regenerative current will raise the DC bus voltage and cause fault condition.

3.7.24 SpdFiltBW

Address:	*0x1D4*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-16383*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: $\tau = \frac{16384}{SpdFiltBw}, in\ T_{PWM}$

Description: This register configures the low pass filter time constant for MotorSpeed. Please note that the input of MotorSpeed low pass filter is Rtr_Freq, which has already been filtered by FreqBW.

3.7.25 RegenSpdThr

Address:	*0x1DA*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 16384 = Motor Max RPM

Description: When the motor is actually spinning opposite direction, MotorLim will be applied when the motor speed is close to zero (either positive or negative) instead of RegenLim. This register specifies the switch over speed threshold between RegenLim and MotorLim.

3.7.26 SpeedScale

Address:	*0x10C*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description

Description: This register is the internal scaling factor between rotor frequency and MotorSpeed.

The value of this register is calculated by Wizard tool from user's input (PWM frequency, motor poles, Motor MaxRPM).

3.7.27 VoltScl

Address:	*0x066*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description

Description: This register is the internal scaling factor between voltage and flux.

The value of this register is calculated by Wizard tool from user's input (PWM frequency, motor poles, DC bus scaling and back emf Ke).

3.7.28 SpdRampRate

Address:	*0x16C*	*Size:*	*16 bits*	*Range:*	*Unsigned input*	*Reset value:*	*0*
					0-16383		

Register Type: MCE Write Register.

Scaling or Notation: See description

Description: This parameter specifies the ramp rate of speed reference.

$$Speed\ ramp\ rate = \frac{SpdRampRate * F_{PWM} * MaxRPM}{2^{25}}, in\ RPM/s$$

3.7.29 RotorAngle

Address:	*0x09A*	*Size:*	*16 bits*	*Range:*	*Signed output*	*Reset value:*	*0*
					-32768-32767		

Register Type: MCE Read Register.

Scaling or Notation: -32768→+32767 is -180°→+180°

Description: This is the estimated rotor angle. It is used for the Field-Oriented control reference frame.

3.7.30 Flx_M

Address:	*0x09C*	*Size:*	*16 bits*	*Range:*	*Signed output*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: 2048 = 100% rated flux

Description: This signal represents the fundamental flux amplitude.

3.7.31 IdRef

Address:	*0x0A6*	*Size:*	*16 bits*	*Range:*	*Signed output*	*Reset value:*	*0*
					-32768-32767		

Register Type: MCE Read Register.

Scaling or Notation: 4096 = 100% motor rated RMS current

Description: Current command on D axis. This is the reference input of current regulator on D axis.

In speed control mode, this register is controlled by speed regulator. It is read register and writing to it has no effect.

In current control mode, this register is used as current input. It becomes a write register.

3.7.32 IqRef

Address:	*0x0A7*	*Size:*	*16 bits*	*Range:*	*Signed output*	*Reset value:*	*0*
					-32768-32767		

Register Type: MCE Read Register.

Scaling or Notation: 4096 = 100% motor rated RMS current

Description: Current command on Q axis. This is the reference input of current regulator on Q axis.

In speed control mode, this register is controlled by speed regulator. It is read register and writing to it has no effect.

In current control mode, this register is used as current input. It becomes a write register.

3.7.33 MotorSpeed

Address:	*0x0E1*	*Size:*	*16 bits*	*Range:*	*Signed output -32768-32767*	*Reset value:*	*0*

Register Type: MCE Read Register.
Scaling or Notation: 16384 = Motor Max RPM
Description: Filtered motor running speed. Filter timer constant is set by register SpdFiltBW. Its value will be reset to 0 when the control is not in RUN state.

3.7.34 abs_MotorSpeed

Address:	*0x11D*	*Size:*	*16 bits*	*Range:*	*Signed output 0-32767*	*Reset value:*	*0*

Register Type: MCE Read Register.
Scaling or Notation: 16384 = Motor Max RPM
Description: Absolute value of MotorSpeed.

3.7.35 IdFilt

Address:	*0x0DB*	*Size:*	*16 bits*	*Range:*	*Signed output -32768-32767*	*Reset value:*	*0*

Register Type: MCE Read Register.
Scaling or Notation: 4096 = 100% motor rated RMS current
Description: Id current after low pass filter. LPF gain is set by register IdqFiltBw.

3.7.36 IqFilt

Address:	*0x0DF*	*Size:*	*16 bits*	*Range:*	*Signed output -32768-32767*	*Reset value:*	*0*

Register Type: MCE Read Register.
Scaling or Notation: 4096 = 100% motor rated RMS current
Description: Iq current after low pass filter. LPF gain is set by register IdqFiltBw.

3.7.37 IdqFilt

Address:	*0x0E3*	*Size:*	*16 bits*	*Range:*	*0-65535*	*Reset value:*	*0*

Register Type: MCE Read Register.
Scaling or Notation: 4096 = 100% motor rated RMS current
Description: IdqFilt is actual motor phase RMS current. It is calculated as below:

$$IdqFilt = LPF(\sqrt{Id^2 + Iq^2})$$

LPF gain is configured by register IdqFiltBw.

3.7.38 IdFwk

Address:	*0x1B2*	*Size:*	*16 bits*	*Range:*	*0-16384*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: 4096 = 100% motor rated current
Description: IdFwk represents the -Id current during field weakening.

3.8 Protection Register Group

3.8.1 SwFaults_Mask

Address:	*0x0C3*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *With bit field definitions*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: For each bit, 0 – Ignore the associated fault; 1 – enable processing of the associated fault.
Description: This register specifies disabling of fault handling as follows:

[15:14]	reserved, must be set to "0"
[13]	Enable user mode UART link break fault
[12]	Enable parameter load fault
[11]	reserved, must be set to "0"
[10]	Enable MCE execution fault
[9:8]	reserved, must be set to "0"
[7]	Enable rotor lock fault
[6]	Enable over temperature fault
[5]	Enable zero speed fault
[4]	Enable Flux PLL out of control fault
[3]	Enable DC bus undervoltage fault
[2]	Enable DC bus overvoltage fault
[1:0]	reserved, must be set to "0"

When a fault is disabled (bit set to "1"), the fault condition is ignored and the motor keeps running. However, even when a fault is disabled, its occurrence is reported in the FaultFlags register, until the condition that caused the fault disappears.

3.8.2 Tderating

Address:	*0x10E*	*Size:*	*16 bits*	*Range:*	*Signed input*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: 1 = 0.25°C
Description: Starting temperature of motor current derating. Maximum motor current will decrease if rise above this threshold. See section 2.10 for detail.

3.8.3 Tshutdown

Address:	*0x10F*	*Size:*	*16 bits*	*Range:*	*Signed input*	*Reset value:*	*0*

Register Type: MCE Write Register.
Scaling or Notation: 1 = 0.25°C
Description: Over-temperature shutdown threshold. If the temperature reaches this point, motor current will be set to 0. See section 2.10 for detail.

3.8.4 TRotorLock

Address:	*0x1F4*	*Size:*	*16 bits*	*Range:*	*0-65535*	*Reset value:*	*1000*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 0.01 second

Description: If SysConfig[12:11]=00, user can change the value of this register to customize the rotor lock detect time (default = 10 seconds).

Please note if rotor lock detect time is configured too short, it may trigger the fault during acceleration or momentary high load condition.

If SysConfig[12:11]=01, 10 or 11, this register value is updated every time before motor start. Its value is 2000, 3000 or 6000.

3.8.5 CriticalOvLevel

Address:	*0x13A*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-4095*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: In ADC counts.

Description: Detection level for Critical Overvoltage. If this threshold is exceeded, all low side switches are clamped (zero-vector-braking) to protect the drive and to brake the motor. The Zero-vector is held until fault is cleared.

3.8.6 DcBusOvLevel

Address:	*0x13B*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-4095*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: In ADC counts.

Description: This parameter defines the dc bus over voltage trip level. A dc bus over voltage fault will be generated if dc bus voltage exceeds this threshold. Refer to the description of the register FaultFlags for more information.

3.8.7 DcBusLvLevel

Address:	*0x13C*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-4095*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: In ADC counts.

Description: This parameter defines the dc bus under voltage trip level. A dc bus under trip voltage fault will be generated if dc bus voltage falls below this threshold. Refer to the description of the register FaultFlags for more information.

3.8.8 FaultClear

Address:	*0x1A3*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-1*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 1 = Clear all faults; 0 = No action.

Description: Writing 1 to this register clears all faults. Once clear has been done, the register will be cleared to 0 by MCE firmware. If fault condition doesn't exist, fault clear will be successful and the drive will enter stopped state. If fault condition still exists, the drive will remain in fault state.

3.8.9 gatekill_const_1

Address:	*0xF0B*	*Size:*	*8 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*13*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 10ns

Description: Persistence filter time for PWM gate kill input (in clock cycles)

3.8.10 SwFaults

Address:	*0x05B*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *With bit field definitions*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: This register is derived from FaultFlags by the following bitwise logical operation:

$SwFaults = FaultFlags \cdot SwFaults\_Mask$

SwFaults is cleared by FaultClear. For bit field definition, refer to Faultflags.

3.8.11 RotorLock_Counter

Address:	*0x097*	*Size:*	*16 bits*	*Range:*	*0-6000*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: 1 = 0.01 second

Description: Rotor lock time counter. If the speed PI output (TrqRef) is saturated which may indicate a rotor lock condition, this register begins to count up. When count up to 1000, 2000, 3000 or 6000 (depends on rotor lock detect time), rotor lock fault will be set and motor will stop if the rotor lock fault is enabled. Counter value will be reset to 0 if the control exits from rotor lock condition.

3.8.12 FaultFlags

Address:	*0x1EE*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *With bit field definitions*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: This register provides drive fault status. Most faults are handled by a fault handling routine operating at the PWM inverter switching frequency with the exception of Gate Kill faults. Gate Kill and Critical Overvoltage faults are handled within the Faults module and will instantly initiate inverter and regulator shutdown. The FaultFlags register indicates currently pending fault conditions. The FaultClear register is used to reset fault conditions.

For all bit fields defined below, a value of 1 indicates that the corresponding fault condition is pending.

[15:14]	Unused	reserved, always zero
[13]	**LinkBreakFlt**	User mode UART link break fault

[12]	**ParaLoadFlt**	Parameter load fault
[11]	Unused	reserved, always zero
[10]	**MCEExeFlt**[1]	MCE execution fault
[9:8]	Unused	reserved, always zero
[7]	**RotorLockFlt**	Rotor lock fault
[6]	**OverTempFlt**	Over temperature fault
[5]	**ZeroSpdFlt**	Zero speed fault
[4]	**FluxPllFlt**	Flux PLL out of control fault
[3]	**DcLvFault**	DC bus undervoltage fault
[2]	**DcOvFault**	DC bus overvoltage fault
[1]	**DcCritOvFault**[2]	DC bus Critical overvoltage fault
[0]	**MotorGateKill**[2]	Motor gatekill fault

[1]MCE execution overrun fault. This fault occurs if the MCE firmware does not complete its PWM-cycle processing before the next PWM pulse.

[2]DC bus critical overvoltage and Gatekill fault cannot be masked by SwFaults_Mask

3.9 UART Communication Register Group

3.9.1 TLinkBreakBase

Address:	*0x219*	*Size:*	*16 bits*	*Range:*	*0-5461*	*Reset value:*	*500*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 0.01 second

Description: This register configures the base time of link break protection. See 2.4.3 for detail.

Default value is 500 which mean 5 seconds.

3.9.2 Comm_Status

Address:	*0x0C4*	*Size:*	*16 bits*	*Range:*	*Unsigned output with bit field definitions*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: Communication status read register:

[15:7] not used

[6] **ActiveLayer**

0 Processing data link layer in current PWM cycle

1 Processing protocol layer and application layer in current PWM cycle

[5:3] not used

[2] **Tx_busy**

This bit indicates UART is sending a data frame. Protocol layer set this bit to inform data link layer to start sending data, and data link layer will clear this bit after finished sending out whole data frame.

[1] **Frame_pending**

This bit indicates a valid data frame has been received by data link layer (with checksum validated) and pending protocol layer to process the data. Protocol layer clear this bit after process the data frame.

[0] not used

3.9.3 NodeAddr

Address:	*0x21C*	*Size:*	*16 bits*	*Range:*	*0-254*	*Reset value:*	*0*

Register Type: MCE Read Register.

Scaling or Notation: See description.

Description: Node address for UART communication. The node address is configured by user and been programmed into the parameter block. Valid address is 1 to 254.

If the IC has no parameter block programmed, this register will read 0. In this case, UART communication can still be established by using node address = 0x00 (no reply) or 0xFF (with reply).

3.10 Miscellaneous Register Group

3.10.1 PwmPhaseMask

Address:	*0x0BC*	*Size:*	*16 bits*	*Range:*	*Unsigned input with bit field definitions*	*Reset value:*	*0xFFFF*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This register can be used in PWM test mode to disable PWM outputs on one or more phases.

[15:12] not used

[11:8] **W phase PWM mask**

0000 Disable W phase PWM output

1111 Enable W phase PWM output

[7:4] **V phase PWM mask**

0000 Disable V phase PWM output

1111 Enable V phase PWM output

[3:0] **U phase PWM mask**

0000 Disable U phase PWM output

1111 Enable U phase PWM output

3.10.2 Out1_3_value

Address:	*0x0ED*	*Size:*	*16 bits*	*Range:*	*0-7*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: See description.

Description: This register is to configure the output value of OUT1/OUT2/OUT3 GPIO pins. Please note if HwConfig[7:5]=000, only OUT1 and OUT3 are functioning, OUT2 (GPIO2) pin is configured as DIR input thus cannot be output.

If HwConfig[7:5]=001, all OUT1, OUT2 and OUT3 are functioning.

3.10.3 MaxRunTime

Address:	*0x10B*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-65535*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 1 PWM cycle

Description: This register is to configure the maximum time (in PWM cycles) for the motor to run. Controller will put into stop state after the number of PWM cycles been elapsed during run state. The purpose of this register is to limit the run time in test mode.

The valid number to limit the motor RUN time is 1-65535.

Write 0 to it disable this feature.

3.10.4 TFluxFault

Address:	*0x1F3*	*Size:*	*16 bits*	*Range:*	*Signed input 1-32767*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 1 = 0.01 second

Description: This register specifies the time slot duration for PLL out-of-control protection. See 2.12 Flux PLL out-of-control protection for detail.

Please note the value of this register is updated automatically by firmware according to control bits SysConfig[9:8] every time motor start. If user want to use different value other than the selections provided by SysConfig[9:8], it can only be changed after motor start and only be valid before next time motor starts.

3.10.5 Vd_Cmd

Address:	*0x1F9*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-16384*	*Reset value:*	*2*

Register Type: MCE Write Register.

Scaling or Notation: $Output\ duty\ cycle = \frac{Vd\_Cmd \times 2048}{4974 \times DcBusVoltsFilt} \times 100\%$

Description: Vd command when the drive is working in voltage control mode.

3.10.6 Vq_Cmd

Address:	*0x1FA*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-16384*	*Reset value:*	*2*

Register Type: MCE Write Register.

Scaling or Notation: $Output\ duty\ cycle = \frac{Vq\_Cmd \times 2048}{4974 \times DcBusVoltsFilt} \times 100\%$

Description: Vq command when the drive is working in voltage control mode.

3.10.7 PGDeltaAngle

Address:	*0x0F5*	*Size:*	*16 bits*	*Range:*	*0-16384*	*Reset value:*	*0*

Register Type: MCE Write Register.

Scaling or Notation: 512 = 1 PG pulse every 360 electrical degree (every electrical cycle)

Description: This register configures the PG output.

$$PGDeltaAngle = 256 * \frac{Motor\ poles}{PPR}$$

Write 0 to PGDeltaAngle will disable the PG output.

PPR is expected PG Pulses Per Revolution. For example, we expect 4 PPR for an 8 poles motor (1 pulse per electrical cycle), then: $PGDeltaAngle = 256 * \frac{8}{4} = 512$

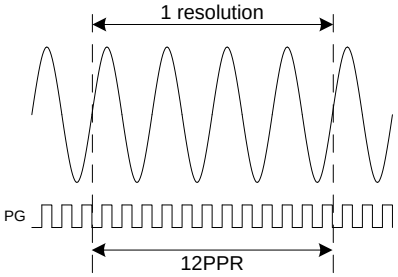


Figure 33: PG pulse generation

PG output is updated every PWM cycle, so the maximum PG output frequency is ½ Fpwm.

The maximum value for PGDeltaAngle is 16384, which means 1 PG pulse take 32 electrical cycle (16384/512=32), on an 8 poles motor, the PG output will be 0.125PPR.

If PGDeltaAngle is 2^n (2,4,8,16….8192,16384), PG pulse will be synchronized with rotor angle. For example, if PGDeltaAngle=512 for an 8 poles motor (4PPR). There are 4 PG pulses every 4 electrical cycles and the PG transition (high to low or low to high) will happen at 0 and 180 electrical degree.

3.10.8 Indirect_rd_addr

Address:	*0x104*	*Size:*	*16 bits*	*Range:*	*Unsigned input 0-65535*	*Reset value:*	*0*

Register Type:	MCE Write Register.
Scaling or Notation:	see description
Description:	MCE firmware provides indirect RAM read method for debug purpose. This register is the write register to specify the RAM address to read. At the end of each PWM cycle, it reads RAM location according to this address and put the RAM value into read register “indirect_rd_val”. Writing 0 to it disable the indirect read method.

3.10.9 Systick

Address:	*0x092*	*Size:*	*16 bits*	*Range:*	*Unsigned output 0-65535*	*Reset value:*	*0*

Register Type:	MCE Read Register.
Scaling or Notation:	1=10ms
Description:	Systick is a free running counter which increments by 1 every 10ms.

3.10.10 Running_Counter

Address:	*0x098*	*Size:*	*32 bits*	*Range:*	*Unsigned output 0-0xFFFFFFFF*	*Reset value:*	*0*

Register Type:	MCE Read Register.
Scaling or Notation:	1 = 1 PWM cycle
Description:	This is a 32-bit free running up counter. When the controller finish boot-strap charge state and enter into run state, it will start count up from 0. If the control exits from run state, this register keeps the value that represents how many PWM cycles elapsed during the previous run.

After the register value reaches 0xFFFFFFFF, it will continue count up from 0.

3.10.11 Indirect_rd_val

Address:	*0x105*	*Size:*	*16 bits*	*Range:*	*Unsigned output 0-65535*	*Reset value:*	*0*

Register Type: MCE Read Register.
Scaling or Notation: see description
Description: Indirect read data register. See 3.10.8 for detail.

4 Hardware Design

4.1 Shunt resistor current sensing

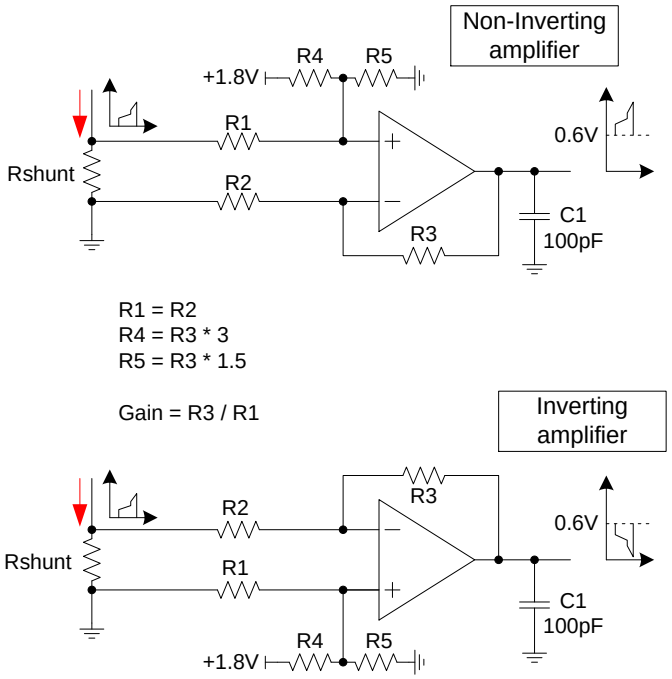


Figure 34: Current sensing circuit

Figure 34 showing recommended non-inverting and inverting current sensing circuit. Both non-inverting and inverting circuits are differential amplifier with 0.6V offset (IFBO=0.6V when shunt current is 0). For single shunt current sensing, the non-inverting configuration makes it possible to use internal Itrip comparator and eliminate the external over-current protection. Differential amplifier removes the common mode noise due to transient potential difference between IC GND and power GND. 0.6V zero current offset is necessary for bi-directional current sensing with 1.2V ADC input range.

Amplifier gain is calculated: $Gain = R3/R1$

If R4 is connecting to 1.8V, in order to get 0.6V zero current offset: $R4 = R3 * 3$ and $R5 = R3 * 1.5$

Resistors value on IRMC0099_R3.0 control board is: R1=R2=5.90K, R3=7.68K, R4=23.2K, R5=11.5K. op-amp gain is: $7.86K/5.90K = 1.302$. This gain should be good enough for most 100V-240V application and user only need to change shunt resistor value and power rating to fit different inverter output current. For low voltage application, op-amp gain may need to be increased to reduce shunt resistor losses.

Shunt resistor current sensing is the critical part to make sensorless FOC control working especially single shunt. A good PCB layout can improve current sensing signal and greatly thus improve the control performance. Hardware designer should consider following while designing a PCB board:

1. Use parallel traces (Shunt+ and Shunt-) from shunt resistor to op-amp, the traces don't need to be wide but they should be close to each other as much as possible.
2. Place components that connecting to analog GND (for example: C1 and R5) in same PCB area which is close to IC analog GND, then connect them to a solid polygon.
3. Minimize power stage trace impedance by using short and wide trace (or polygon), the main loop including DC bus cap→DC bus + of three phase bridge→high side and low side switching devices→shunt resistor→GND return to DC bus cap. This helps reducing ringing time and amplitude during power devices switch.
4. The connection between IC ground and DC bus cap negative (or negative terminal of shunt resistor) should be low impedance and not shared with SMPS ground return path.

4.2 DC bus voltage sensing

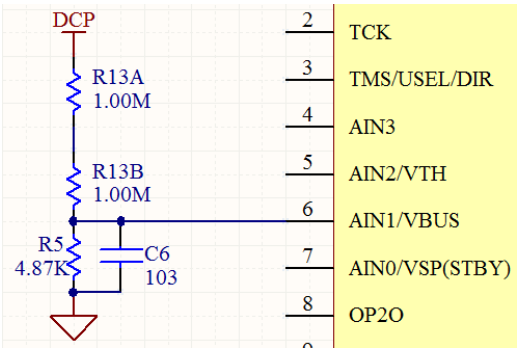


Figure 35: DC bus sensing circuit

DC bus voltage sensing is simply done by voltage divider. C6 must be placed close to IC input pin.

4.3 Vsp input

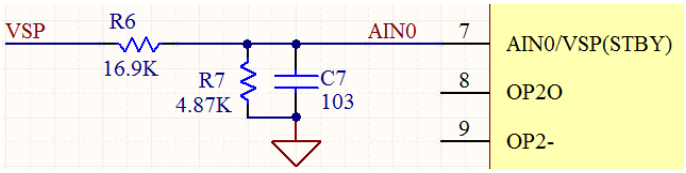


Figure 36: Vsp input

MCE firmware has fixed AIN0 input voltage for motor start/stop(standby) and most application requires higher voltage. Voltage divider can be used to expand the Vsp input voltage range.

IR provides reference design board with 16.9K and 4.87K voltage divider. Below is the Vsp voltage using 16.9K and 4.87K voltage divider:

	AIN0	**Vsp (16.9K and 4.87K)**
Stop Motor	0.256V	1.1V
Start Motor	0.315V	1.4V
Linear speed range (Min Speed)	0.467V	2.1V
Linear speed range (Max speed)	1.15V	5.1V
Maximum Vsp voltage	1.20V	5.4V

4.4 External NTC thermistor

External NTC thermistor can be used as current derating and over-temperature protection. MCE firmware convert AIN2 into Celsius (See description of register TempSense)

Please note the NTC thermistor and external circuit must meet below requirements:

- Resistance (25°C) = 47kOhm, B-Constant = 4050 (Example: Murata NCP15WB473F03RC)
- Pull up to 1.8V by a 4.87kOhm resistor
- Pull down to GND by a 9.76kOhm resistor

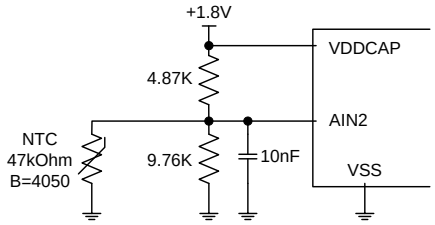


Figure 37: External NTC thermistor circuit

4.5 Reference design – Control board

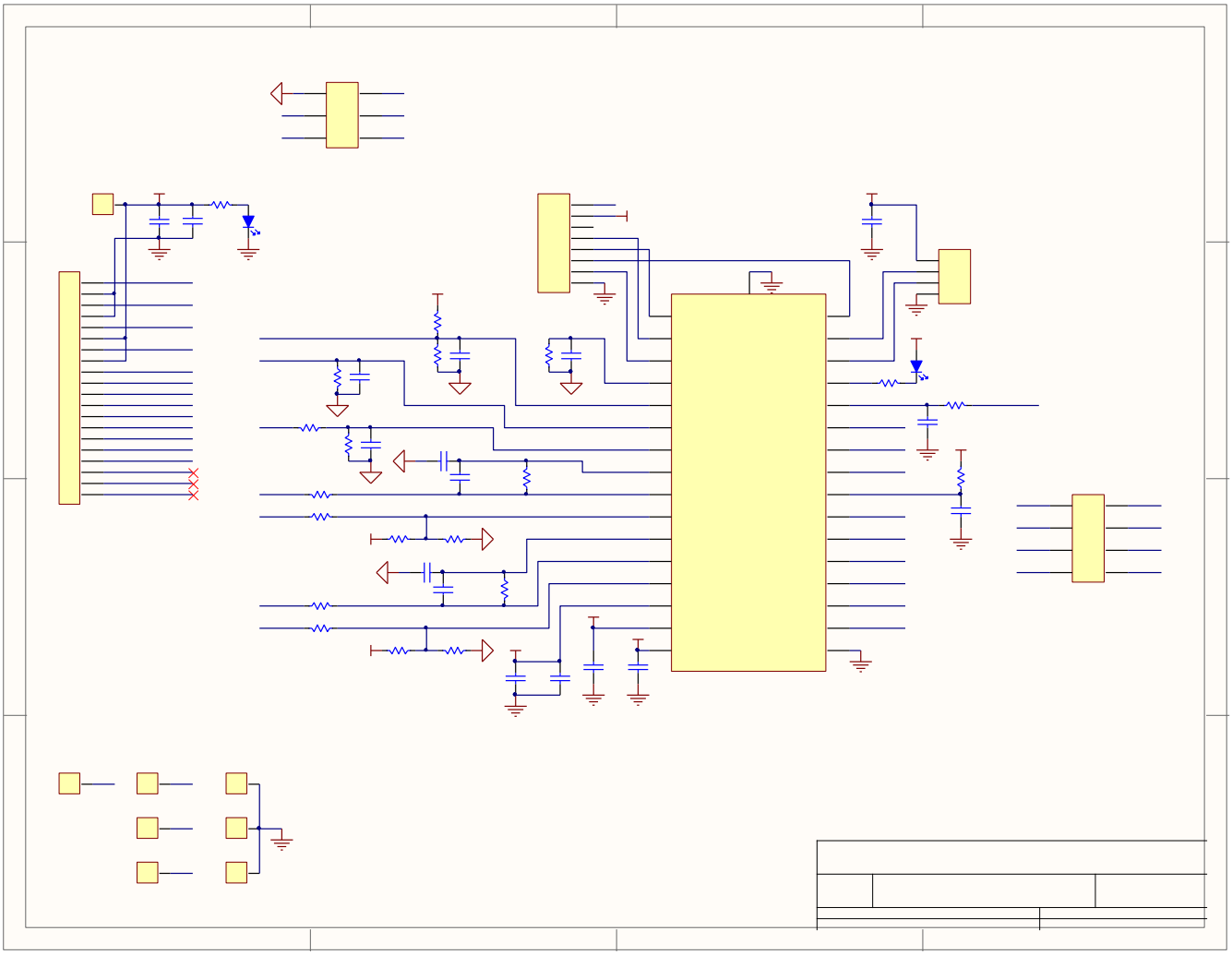


Figure 38: IRMC0099_R3.1 control board schematic

IRMC0099_R3.1 control board can be paired with either single shunt or leg shunt power board. IRMD505-1-D_R1.0 power board can connect to IRMC0099_R3.1 control board directly through 20-pin connector. This is a single shunt board. There is no external circuit for over-current protection; over-current is protected by comparator inside IRMCK099. IPM module has NTC thermistor and can be used as temperature sensor.

For the ease of demonstrating the features and capabilities of IRMCK099M, the control board and the power board are designed separately. The end application will always be using one board and please check IR iMotion applications group for the availability of reference design kits.

4.6 Reference design – Power board

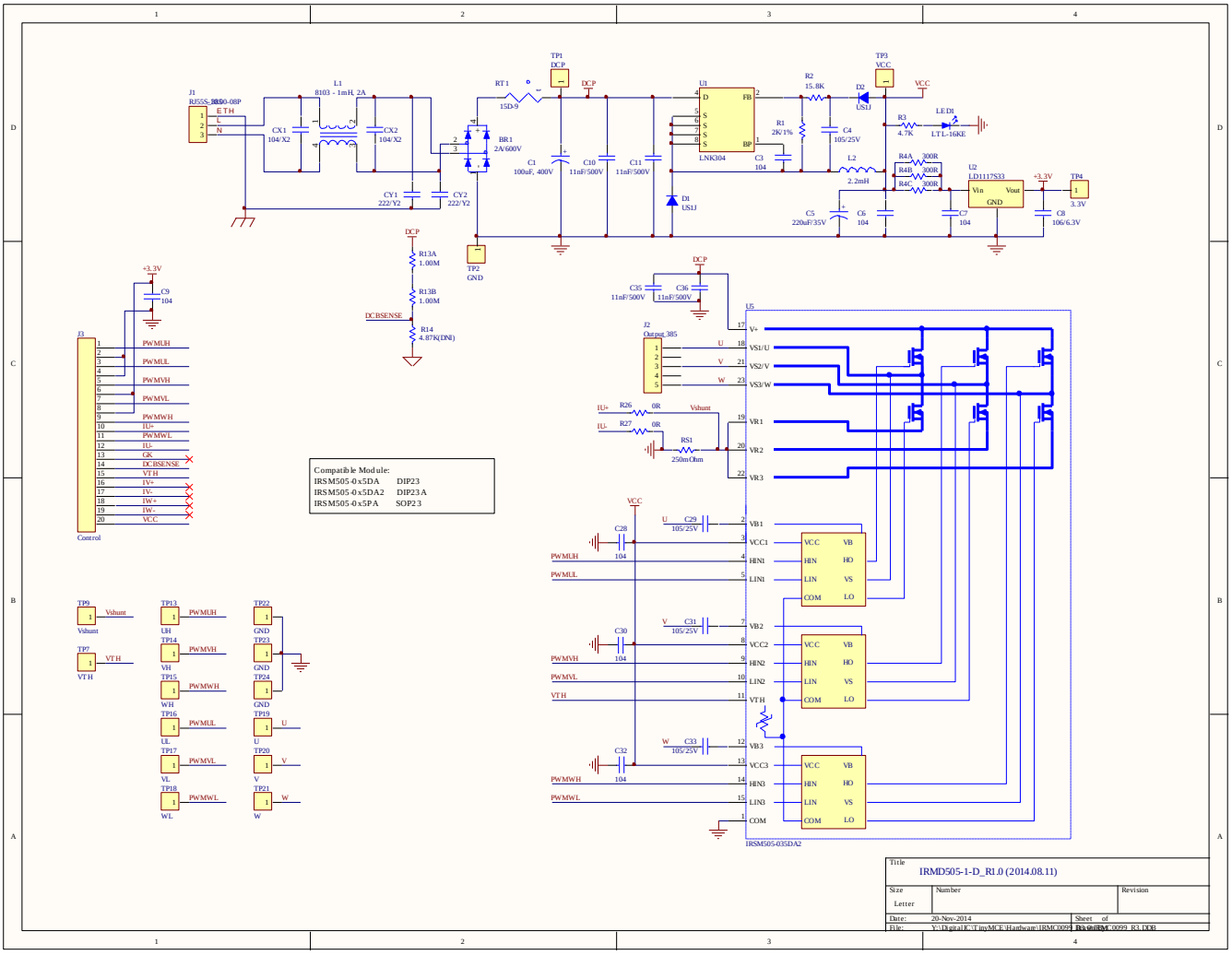


Figure 39: IRMD505-1-D_R1.0 power board schematic

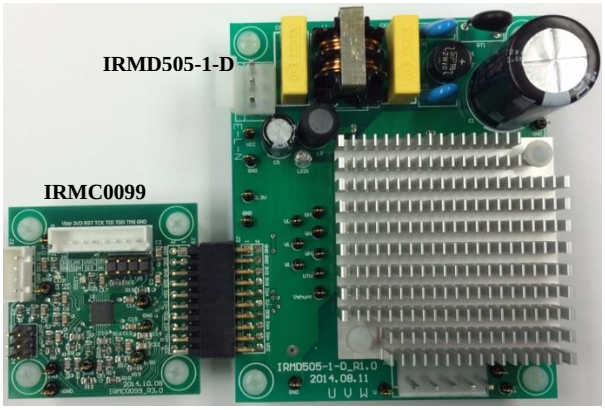


Figure 40: IRMCP0099-1-D Reference Design Kit

5 Motor Tuning

IR provides TinyWizard to calculate hardware parameters, motor parameters, control parameters/features, protection parameters/features as well as features for the complete system. This is the first step that users need to do before running a motor. Regarding motor parameter measurement (Resistance, Lq, Ld and back EMF constant Ke), please refer to chapter 2.1 of IRMCx100 Application Developer's Guide.

Correct motor parameter is important for sensorless FOC control to be able to run the motor in steady state. IRMCK099 MCE firmware uses improved flux based sensorless algorithm which makes it much easier to start a motor. Although the motor can start, depends on application requirement, motor startup and dynamic performance may still need to be tuned in real load condition.

Below are some common problems and basic tuning technics when using IRMCF099 MCE firmware:

1. **How to check if the current sensing is good?**

   It's better to run the motor without load, start the motor and set to a speed that motor can run smoothly. Use oscilloscope to measure motor RMS current. In Eztune GUI, output current display usually is slightly higher than measured motor current due to sensing noise, the difference should be small and close to measured motor current as much as possible.

   If current sensing noise is not good, here list the possible causes:

   - Bad PCB layout
   - Power devices switch too fast which cause too much noise
   - Current sensing parameters doesn't match the hardware, related parameters:
     - Deadtime
     - PwmGuandBand (leg shunt only)
     - TCntMin (single shunt only)
     - SHDelay
     - TMinPhaseShift (single shunt only)

   In single shunt configuration, phase shift PWM provides better control performance. TMinPhaseShift and SHDelay are two key parameters to achieve good single shunt current sensing in phase shift PWM mode

   To achieve good single shunt current sensing signal, TMinPhaseShift and SHDelay should be configured following below guideline:

   $$TMinPhaseShift > Dead\ time + Ringing$$

   $$SHDelay < Hardware\ delay\ time$$

   $$TMinPhaseShift + SHDelay > Hardware\ delay\ time + Dead\ time + Ringing$$

   Please note that TMinPhaseShift may cause acoustic noise so that it should be set to a value as small as possible.

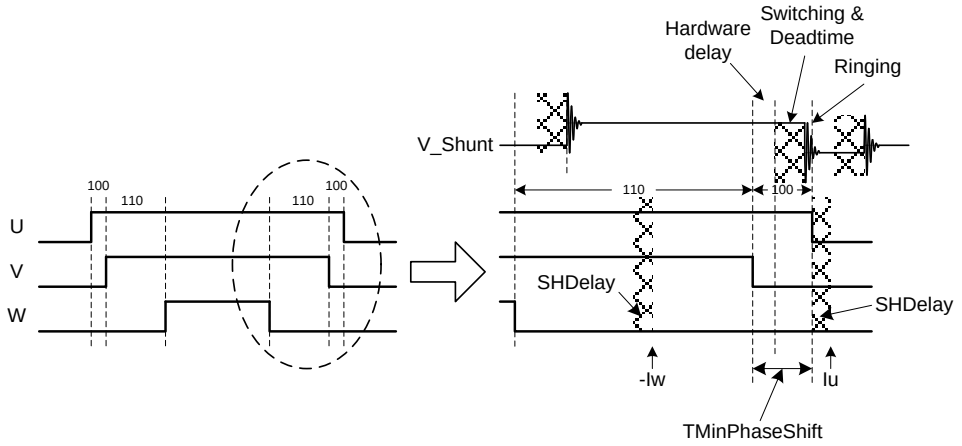


Figure 41: Single Shunt Current Sensing for Phase Shift PWM

There are three timings: Dead time, hardware delay time and ringing time. Dead time is already known since we set it in MCEWizard. What we need to measure on the hardware board is hardware delay time and ringing time.

Example of setting proper TMinPhaseShift and SHDelay:

Below is an example showing how to measure the hardware and fine tune these two parameters.

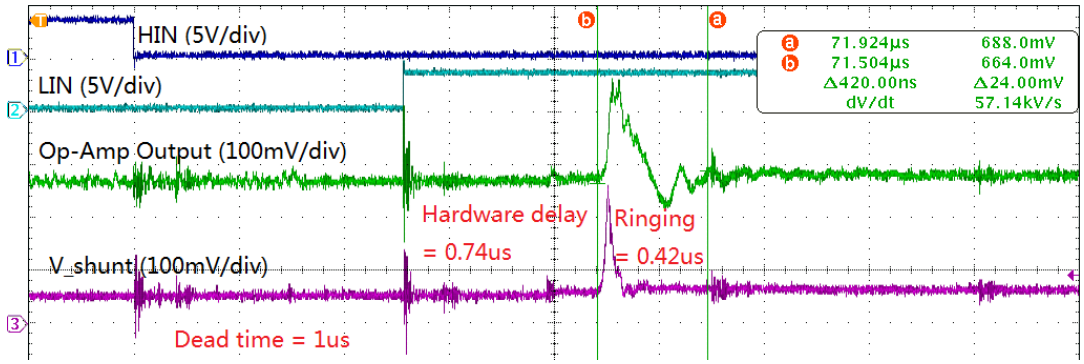


Figure 42: Measuring hardware delay and ringing time

$$TMinPhaseShift > 1us + 0.42us = 1.42us$$

$$SHDelay < 0.74us$$

$$TMinPhaseShift + SHDelay > 0.74us + 1us + 0.42us = 2.16us$$

We can easily configure TMinPhaseShift=2.2us and SHDelay=0 to meet above criteria. But the optimum value should with minimum TMinPhaseShift value to minimize acoustic noise cause by phase shift PWM. The optimum value should be:

$$TMinPhaseShift = 1.6us$$

$$SHDelay = 0.6us$$

2. Current regulator tuning

The iMotion current controller utilizes field-oriented, synchronously rotating reference frame type regulators. Field-orientation provides significant simplification to the control dynamics of the current loop. There are two current regulators (one for the d-channel and one for the q-channel) employed for current regulation. The q-channel (torque) control structure is identical to the d-channel (flux). The current control dynamics of the d-channel is depicted in Figure 43. The motor windings can be represented by a first order lag with a time constant □ = L/R. This time constant is a function of the motor inductance and equivalent resistance (R = cable + winding). For a surface mounted permanent magnet motor, the d and q channel inductances are almost equal. In the case of an interior permanent magnet (IPM) motor, the q-channel inductance is normally higher than the d-channel inductance.

In the current control continuous time domain model, Figure 43, the forward gain A models the conversion of the digital controller output to voltage (including inverter gain) and the feedback gain B models the transformation of the current feedback (Amps) to internal digital counts via an A/D converter. The calculation of the PI compensator gains (KI_{Ireg}, Kp_{Ireg_D}) is done by using a pole-zero cancellation technique as illustrated in Figure 44 where the current controller is rearranged to give transfer function block C(s). Setting Kp_{Ireg_D} / KI_{Ireg} of C(s) equal to the time constant of the motor ($\tau = L/R$), the controller zero will cancel the motor pole (pole-zero cancellation). Therefore, the model of the controller dynamics can be further simplified as shown in Figure 45. The equivalent transfer function of Figure 45 is a first order lag with time constant τ_C. By selecting an appropriate current regulator response (typically 1 to 5 msec) for a particular application, the current regulator gains can be readily obtained. It may be noticed that using the pole zero cancellation technique, the motor inductance enters into proportional gain calculations and the resistance enters into integral gain calculations.

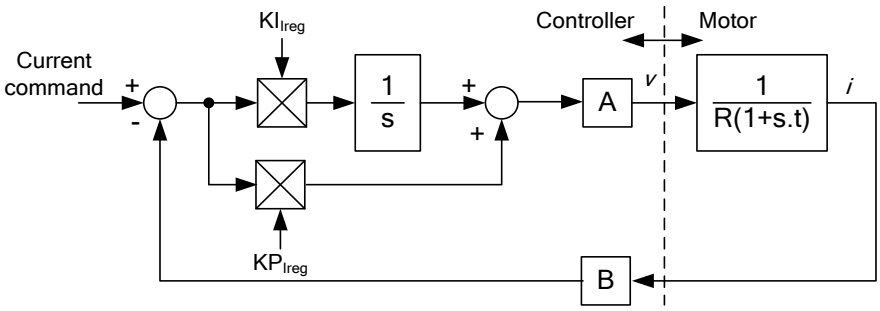


Figure 43: Current controller dynamics

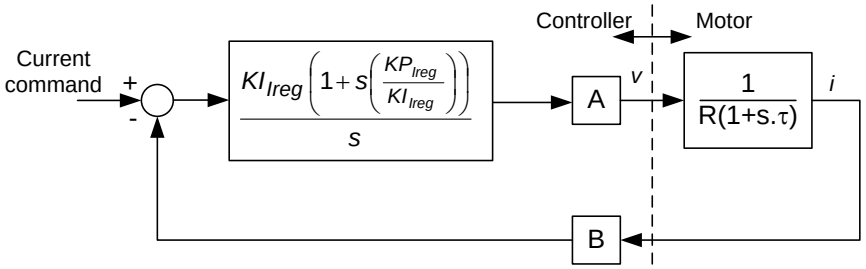


Figure 44: Pole zero cancellation

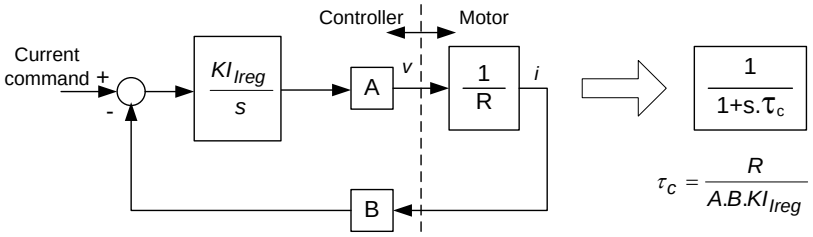


Figure 45: Simplified current control dynamics due to pole zero cancellation

Based on the pole-zero cancellation technique the controller gains in the continuous time domain model are evaluated by:

$$Kp_{Ireg} = \frac{L_q \cdot CurrentRegBW}{A \cdot B}$$

$$KI_{Ireg} = \frac{R \cdot CurrentRegBW}{A \cdot B}$$

Where A and B are the voltage and current scaling.

In the digital controller implementation, the integrator is a digital accumulator and so the discrete time domain model for the PI compensator must be used for the integrator. In this case the digital integrator gain, Kx_{Ireg}, includes a scaling factor for the compensator sampling time.

$$Kx_{Ireg} = KI_{Ireg} \cdot T$$

T is the controller sampling time, which in this case is equal to the PWM period.

The voltage scaling, A, must account for gains in the forward rotation and the space vector modulator. The three phase inverter produces a peak line voltage equal to the dc bus voltage V_{dc}, so at 100% modulation the rms phase voltage is $V_{dc}/\sqrt{2}/\sqrt{3}$. The modulator produces 100% modulation for a digital input of 8192 while the forware rotation function has a gain of 1.64676. Therefore, the current loop voltage scaling A is given by this equation:

$$A = \frac{V_{dc}/\sqrt{6}}{8192/1.64676}\ (in\ V_{rms}/cts)$$

The current loop feedback scaling, B, is defined by the shunt resistor, the amplifier gain, the A/D converter gain and the current feedback scaling register, IfbkScl. However, TinyWizard calculates IfbkScl so that a count of 4096 is equivalent to the motor rated rms current. Therefore, the current loop feedback scaling is simply given by:

$$B = \frac{4096}{I_{RATED}}\ (in\ cts/A_{rms})$$

The controller gains calculated for the current loop typically yield numbers that are less than one and so the current loop PI regulators include post multiplication scaling on the Kp and Kx inputs to increase the precision of the regulator gains. The multiplier on the Kp input is followed by a shift of 14 bits while the regulator on the Kx input is shifted by 19 bits. Therefore, the control gains calculated for this digital implementation are given by:

$$Kp_{Ireg} = \frac{L_q \cdot CurrentRegBW \cdot 2^{14}}{A \cdot B}$$

$$Kx_{Ireg} = \frac{R \cdot CurrentRegBW \cdot T \cdot 2^{19}}{A \cdot B}$$

Current regulator step response can be measured by using current control mode. Follow below steps to put the control into current control mode for current regulator step response diagnostic:

Step 1 – park the rotor to 0°:

a. Connect the motor and measure U phase current from oscilloscope.
b. AngleSelect = 0, disconnect flux rotor angle and use internal open loop angle.
c. CtrlModeSelect = 1, this is set to current control mode and disable the speed regulator.
d. rotor_angle_adv = 0, set open loop angle to 0.
e. TargetSpeed = 0, set open loop angle rotating speed to 0, thus angle will remain 0 during the test.
f. IdRef = 1024, apply 25% rated current to D axis.
g. MtrSeqCtrl = 2, start the drive, the control will regulate the current at 0° and the rotor will be aligned at 0°. The current is flowing out from U phase and flow into V and W phase.

We want to measure the step response without rotor movement. Step 1 is to park the rotor to certain angle so that the following steps will not cause any rotor movement. If the load inertia is high (such as fan blade), rotor will oscillate around parking angle and it may take long time to stop oscillating. If possible, use hand to stop oscillation and help it park at 0°.

Step 2 – apply initial 10% Id current:

a. IdRef = 410, apply 10% rated current to D axis.

Step 3 – apply 50% Id current:

a. IdRef = 2048, step change Id reference to 50%.

This is the step response we want to observe. Capture the U phase current waveform by using oscilloscope.

Step 4 – Stop the drive and recover the control to sensorless speed control mode:

a. MtrSeqCtrl = 4, stop the drive.
b. AngleSelect = 2, use flux rotor angle.
c. CtrlModeSelect = 2, set to speed mode.

Figure 46 shows measured step response with different current regulator bandwidth settings. Step response time constant is defined as the time duration from current start to rise until it reaches 63.2% $(1 - 1/e)$ of final current (not including over shooting).

At lower current regulator bandwidth, actual step response time constant is quite close to theoretical value (9.88ms vs 10ms, 4.84ms vs 5ms, 2.4ms vs 2.5ms). At high current regulator bandwidth, actual time constant becomes much smaller than theoretical value (1.02ms vs 1.25ms, 0.428ms vs 0.625ms) and over-shoot start to appear. To achieve better step response performance, it is recommended to reduce Kx_{Ireg} for high current regulator bandwidth.

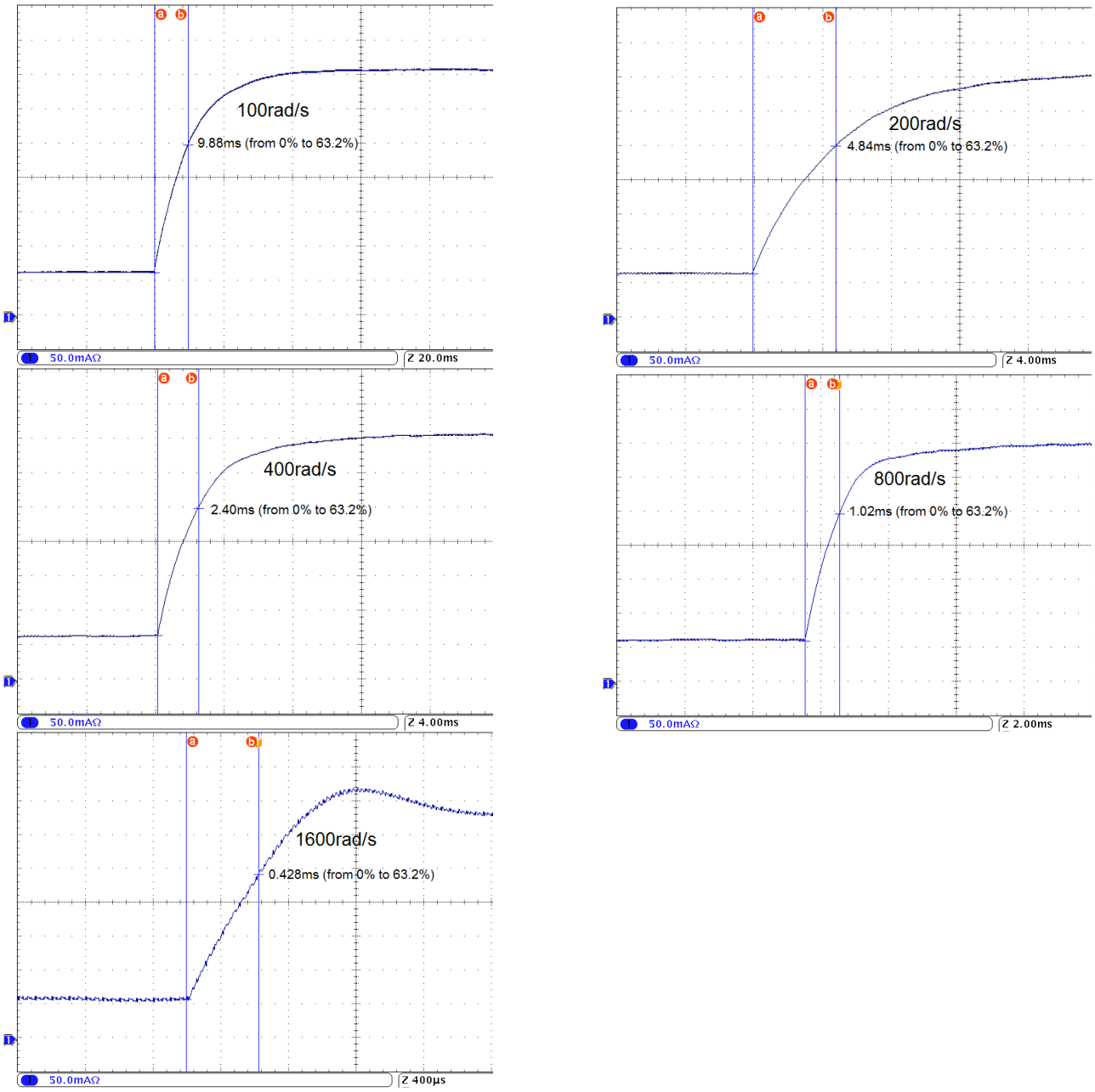


Figure 46: Current regulator step response (100/200/400/800/1600rad/s)

3. Difficult to start the motor

- Make sure current sensing is good
- Make sure motor parameter is correct
- Adjust speed regulator PI gain and speed feedback filter time constant
- Adjust minimum speed
- Adjust speed accelerate and decelerate ramp
- Adjust flux estimator time constant
- Increase motor current limit

4. Motor speed not stable

- If speed is not stable at low speed, check if current sensing is good
- If motor speed oscillate, reduce speed regulator PI, especially I gain
- If motor speed change too much when load change, increase speed PI gain, especially P gain
- If two phase modulation is enabled, make sure 3ph to 2ph switch over speed is high enough, or temporarily disable 2 phase PWM

5. Motor current not stable in field weakening

- Adjust FwkKx together with speed regulator PI gain
- Adjust current regulator PI gain. In field weakening mode, make D axis current regulator higher bandwidth than Q axis, try increase KpIregD 2x higher or more than KpIreg.

6. Reducing acoustic noise

There are many reasons cause acoustic noise. Here are the most common reasons:

- Noise from current sensing circuit. Try to improve current sensing circuit, such as optimizing PCB layout, adjust op-amp load capacitor and feedback capacitor value, optimizing current sensing parameters, etc.
- Noise from high current regulator bandwidth, there is always noise from current sensing, improper current regulator may amplify the noise. To reduce noise from current regulator, try reduce current regulator PI gain, while doing this, make sure the control performance (especially at startup and high load) still good enough
- Noise from low PWM frequency or two phase PWM. Try increase PWM frequency. If the hardware is not suitable for higher PWM frequency, turn off two phase PWM and use 3-phase PWM only.
- Noise from minimum pulse scheme or phase shift PWM scheme (single shunt configuration). Noise caused by minimum pulse scheme can be reduced by reducing register value of TCntMin. Noise caused by phase shift PWM scheme can be reduced by reducing register value of TMinPhaseShift. Please note in either case, SHDelay value also need to be adjusted. It's not possible to eliminate noise in single shunt, if the application requires very low acoustic noise, change to leg shunt may solve the problem.
- Noise from over-modulation. When the motor is running at high speed, over-modulation can be used to maximize DC bus utilization. The drawback of over-modulation is that the output voltage is not sinusoidal, it contains high order harmonics which causes acoustic noise. If in this case, disable over-modulation in TinyWizard.

Edition 2016-04-07

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

IRMCK099_AppGuide

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.