NXP Semiconductors
Application Note

Document Number: AN12200
Rev. 0, 06/2018

Dual FOC Servo Motor Control on i.MX RT

1. Introduction

This application note describes the dual servo demo with the NXP i.MX RT1020 processor. You can use it as a reference for motor control application developing based on other i.MX RT products.

The i.MX RT1020 is a processor with single ARM Cortex-M7 core, which operates at speeds up to 500 MHz. The great processing capability, real-time feature, and reach integration of abundant peripherals make i.MX RT1020 ideal for lot of high-performance applications, such as industrial computing, motor control, power conversion, smart consumer products, high-end audio systems, home and building automation.

For this demo, the RT1020 processor samples current and voltage of motor, receives encoder signal, and generates PWMs to drive motors.

- Section 2 of this document introduces the peripherals configuration of the dual servo demo.
- Section 3 describes the system structure and software of dual servo demo.
- Section 4 describes how to operate the dual servo demo.

Contents

2. Peripherals Configuration

The following dual servo motor demo uses only the essential peripherals for dual motor control technique implement in the application code.

2.1. Clock controller module (CCM)

The CCM generates and controls the clocks of various modules in the design and manages the low-power modes. This module uses the available clock sources to generate the clock roots.

The clock sources used in the motor control demo are:

- PLL3, also called USB1 PLL with a frequency of 480 MHz.
- PLL6, also called ENET PLL, with a frequency of 500 MHz.

The ARM clock core works at a frequency of 500 MHz and the clock source is PLL6. For this setting, the following registers are set: CBCMR[PRE_PERIPH_CLK_SEL], CBCDR[PERIPH_CLK_SEL], and CBCDR[AHB_PODF] in *clock_config.c*.

The ADC, XBAR, and PWM are clocked from the IPG_CLK_ROOT output which has a frequency of 125 MHz. The CBCDR[IPG_PODF] register must be set for this setting. The IPG_CLK_ROOT is sourced from the AHB_CLK_ROOT. The LPUART is sourced from the PLL3 at a frequency of 480 MHz divided by 6.

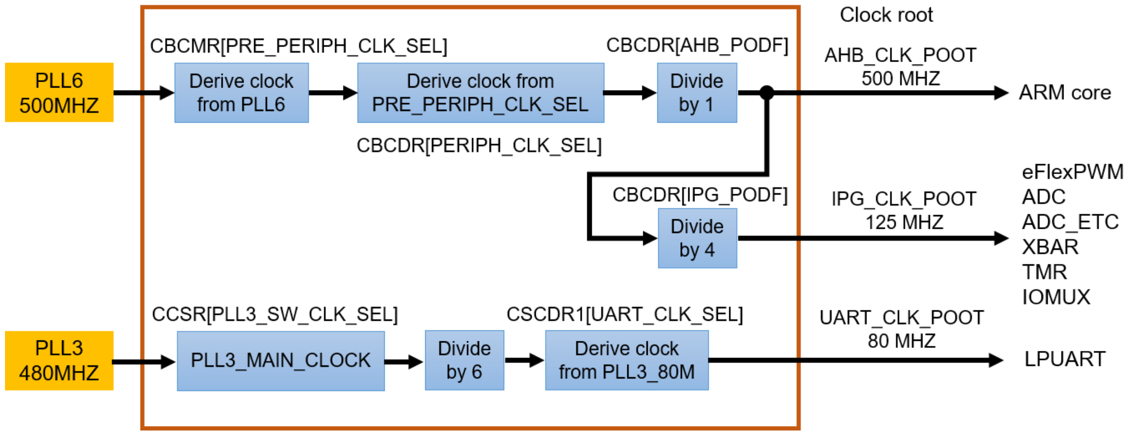


Figure 1. i.MX RT1020 clock source for motor control peripherals

The clock sources for the peripherals used for the motor control are listed in Table 1.

Table 1. i.MX RT1020 clock source for motor control peripherals

—	Clock source	Clock root	Clock root frequency
ARM core	PLL6	AHB_CLK_ROOT	500 MHz
PWM	PLL6	IPG_CLK_ROOT	125 MHz
ADC	PLL6	IPG_CLK_ROOT	125 MHz
ADC_ETC	PLL6	IPG_CLK_ROOT	125 MHz
XBAR	PLL6	IPG_CLK_ROOT	125 MHz
TMR	PLL6	IPG_CLK_ROOT	125 MHz
IOMUX	PLL6	IPG_CLK_ROOT	125 MHz
LPUART	PLL3	UART_CLK_ROOT	80 MHz

2.2. Enhanced flex pulse width modulator (eFlexPWM) configuration

The eFlexPWM contains PWM submodules, each of which is set up to control a single half-bridge power stage. Fault channel support is provided. This PWM module can generate various switching patterns, including highly sophisticated waveforms. The PWM module can control all known motor types.

The eFlexPWM module is a dedicated peripheral enabling the generation of three-phase PWM signals connected to MOSFET H-bridge via pre-drivers.

The three PWM submodules of motor 1 used in this demo are configured as listed below:

- PWM1_Submodule_0
  - IPBus clock source 125 MHz.
  - Running frequency of 16 kHz with 62.5 μs period.
  - INIT register –3906, VAL1 3906.
  - Complementary mode with 0.5 μs dead time.
  - PWM reload and initialization signals generated every opportunity from this submodule to other submodules of PWM1.
  - Trigger one signal from VAL0(0) for providing synchronization with PWM2 of motor 2 via XBAR.
- PWM1_Submodule_1
  - PWM_0 clock source.
  - Running frequency of 16 kHz with 62.5 μs period.
  - INIT register –3906, VAL1 3906.
  - Complementary mode with 0.5 μs dead time.
  - PWM reload and initialization signals generated from submodule 0.
  - Trigger one signal from VAL4(-3744) for providing synchronization with ADC_ETC module via XBAR.
- PWM1_Submodule_2
  - PWM_0 clock source.
  - Running frequency of 16 kHz with 62.5 μs period.
  - INIT register –3906, VAL1 3906.
  - Complementary mode with 0.5 μs dead time.
  - PWM reload and initialization signals generated from submodule 0.

The three PWM submodules of motor 2 used in this demo are configured as listed here:

- PWM2_Submodule_0
  - IPBus clock source 125 MHz.
  - Running frequency of 16 kHz with 62.5 µs period.
  - INIT register –3906, VAL1 3906.
  - Complementary mode with 0.5 µs dead time.
  - EXT_SYNC signal from PWM1 causes initialization.
  - PWM reload signals generated every opportunity from this submodule to other submodules of PWM2.
  - Trigger one signal from VAL4(-3744) for providing synchronization with ADC_ETC module via XBAR.
- PWM2_Submodule_1 & Submodule_2
  - PWM_0 clock source.
  - Running frequency of 16 kHz with 62.5 µs period.
  - INIT register –3906, VAL1 3906.
  - Complementary mode with 0.5 µs dead time.
  - EXT_SYNC signal from PWM1 causes initialization.
  - PWM reload signals generated from submodule 0.

To allocate CPU loading reasonably and lower the probability of the same time energy consumption, it is necessary to achieve a 180-degrees lag between the PWM waves of the two motors. As shown in the Figure 2, the INIT and VAL1 are configured reasonably to make PWM counter run on a regular period. The key to achieve a 180-degrees lag is that whenever the PWM1 counter reaches VAL0, it triggers a signal as EXT_SYNC (external synchronization) to PWM2 to initial the counters of PWM2.

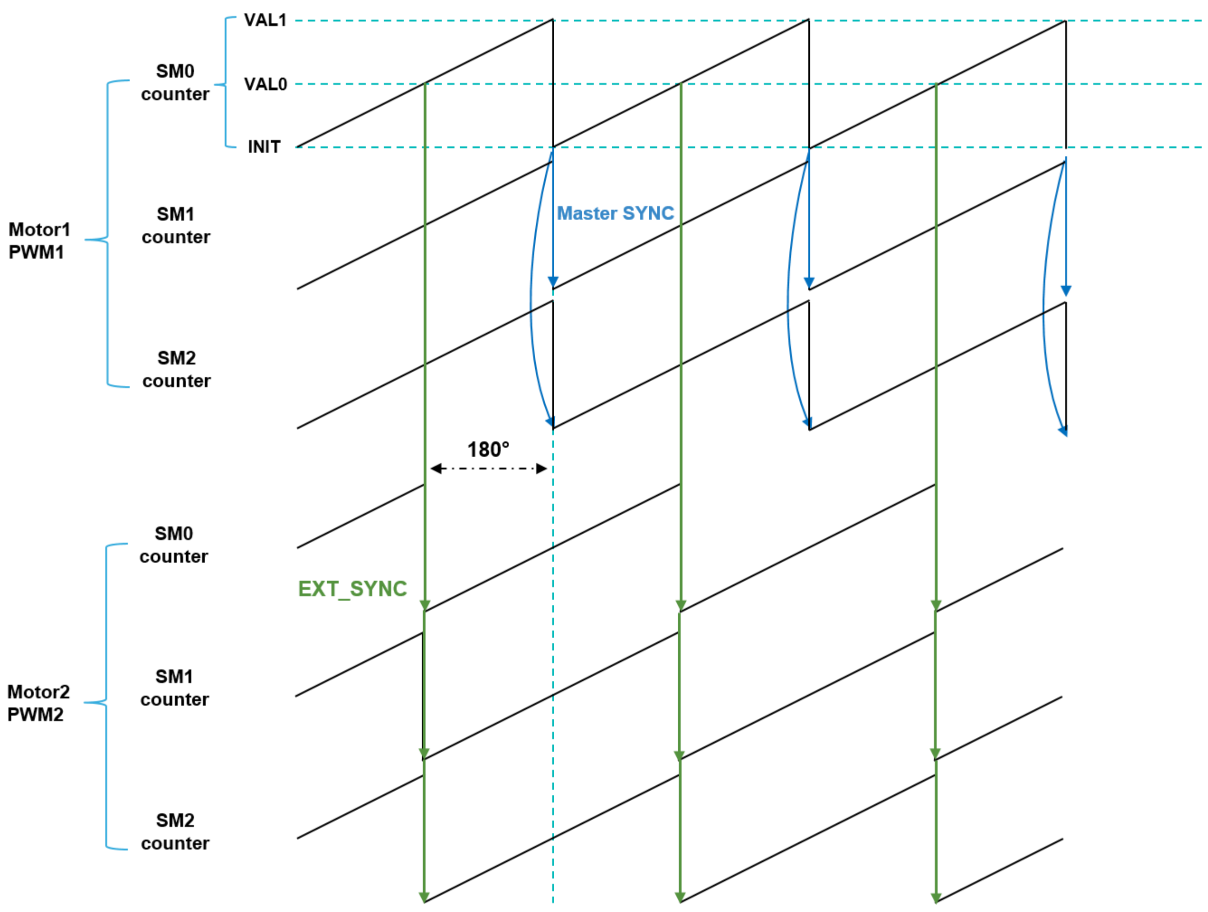


Figure 2. Synchronization between PWMs of dual motor

2.3. Inter-Peripheral Crossbar Switch (XBAR)

XBAR implements an array of M N-input combinational muxes and provide a flexible crossbar switch function that allows any input (typically from eFlexPWM trigger outputs) to be connected to any output (typically to ADC_ETC inputs) under user control.

Figure 3 shows all signals which are transmitted among internal modules via XBAR in this dual servo motor control demo.

The XBAR resource assignments including the input and output number of XBAR are shown in the section 3.4of i.MX RT1020 reference manual. XBAR configuration codes for this demo are listed in below:

From PWM1 to PWM2:

XBARA->SEL22 = XBARA_SEL22_SEL44(0x28U) | XBARA_SEL22_SEL45(0x28U);

XBARA->SEL23 = XBARA_SEL23_SEL46(0x28U) | XBARA_SEL23_SEL47(0x28U);

From PWM1 and PWM2 to ADC_ETC:

XBARA->SEL51 |= XBARA_SEL51_SEL103(0x29U);

XBARA->SEL52 |= XBARA_SEL52_SEL104(0x2CU);

From GPIO PAD to Quad Timer:

XBARA->SEL43 = XBARA_SEL43_SEL86(0x11U) | XBARA_SEL43_SEL87(0x0DU);

XBARA->SEL45 = XBARA_SEL45_SEL90(0x12U) | XBARA_SEL45_SEL91(0x13U);

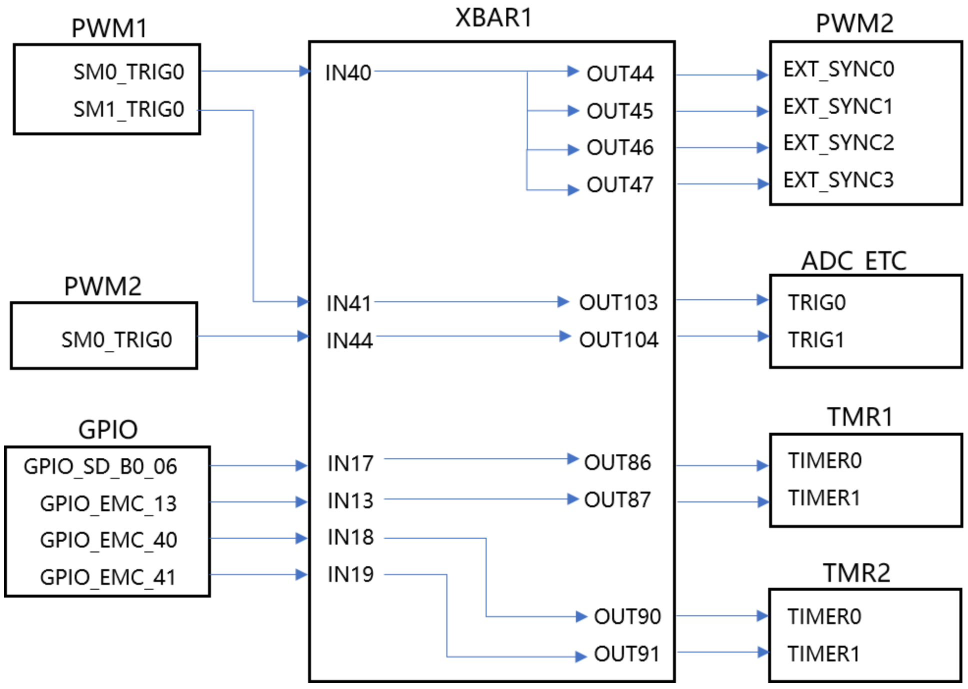


Figure 3. Crossbar interconnection

2.4. Analog sensing — ADC1 and ADC2

ADC1 and ADC2 are used for the motor control analog sensing of currents and DC-bus voltage.

- The clock frequency for ADC1 and ADC2 is 62.5 MHz. It is taken from IPG_CLK_ROOT and divided by 2.
- The ADCs operate as 12-bit with the single-ended conversion and hardware trigger selected. The ADCs are controlled by ADC_ETC based on the triggers generated from the eFlexPWM.
- Enable ADC_HC0s of ADC1 and ADC2, select external channel selection from ADC_ETC as the trigger source.

2.5. ADC External Trigger Control (ADC_ETC)

The ADC_ETC module enables multiple users to share the ADC modules in the Time Division Multiplexing (TDM) way. The external triggers are generated from the Cross BAR (XBAR) or other sources. The ADC scan is started via ADC_ETC.

- Enable external XBAR trigger 0 and 1, both ADCs have their own trigger chains.
- The trigger chain length is set to 2. The back-to-back ADC trigger mode is enabled.
- The SyncMode is on. In the SyncMode, ADC1 and ADC2 are controlled by the same trigger source.

- Both trigger queues have their own finished interrupt. When Queue 4 triggered by TRIG0 is finished, enable DONE0 interrupt. When Queue 5 triggered by TRIG1 is finished, enable DONE1 interrupt.

Figure 4 shows how to use AC_ETC to control dual ADCs in this dual servo motor demo.

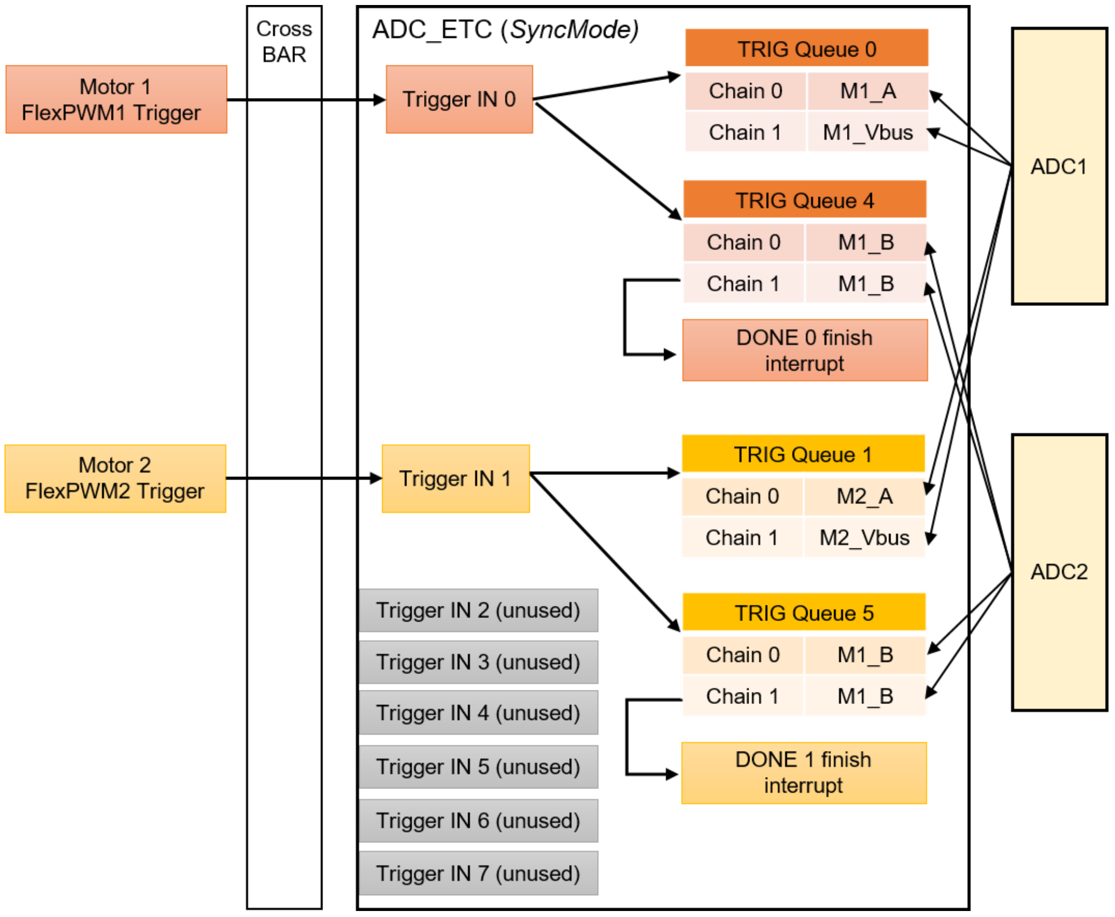


Figure 4. How to use AC_ETC to control dual ADCs

When initialing the ADC_ETC, below initialization process must be operated firstly in sequence before other initialization operations.

- Firstly, SOFTRST (software reset bit) of ADC_ETC global control register must be cleared alone. Otherwise, operations of all other register bit will be invalid.
- Secondly, TSC_BYPASS bit must be cleared if enabling SyncMode. Otherwise, ADC2 will be occupied by TSC.

2.6. IOMUX Controller (IOMUXC)

The IOMUX Controller (IOMUXC), together with the IOMUX, enables the IC to share one pad to several functional blocks. This sharing is done by multiplexing the pad's input and output signals.

Each XBAR pin on pad has both input and output function which can be selected by IOMUXC. In this dual servo demo, encoder signals transmit from GPIO pad to TMR via XBAR and some necessary configuration in IOMUXC are listed below:

- Set XBAR1_INOUT17, XBAR1_INOUT13, XBAR1_IN18, XBAR1_IN19 as input by configure IOMUXC_GPR_GPR6 register.
- Select the XBAR as the input source of TMR by configure IOMUXC_GPR_GPR6 register.

2.7. Quad timer (TMR)

There are two sets of 16-bit timer module (TMR) contains four identical counter/timer groups which are suitable for decoding encoder signal. Each 16-bit counter/timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, two status and control registers, and one control register.

Figure 5 shows how to use one TMR module to implement encoder signal counting, revolution counting and speed measurement for motor 1. The configuration of motor 2 is as same as motor 1.

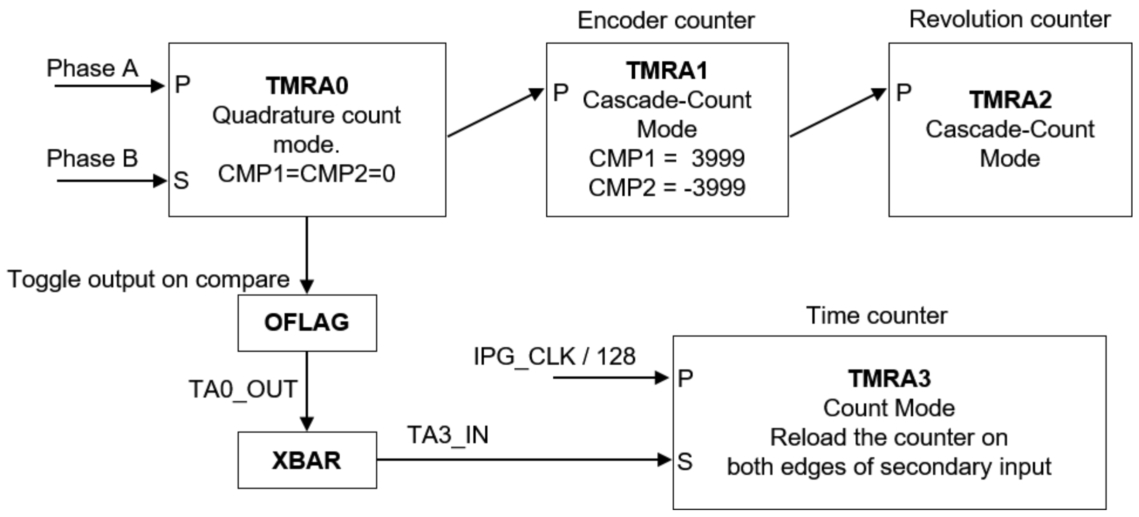


Figure 5. TMR configuration for motor 1

- TMRA0 is used to decode the quadrature inputs, but it does not actually count the number of encoder signal:
  - Quadrature mode.
  - Select encoder phase A and B as primary and secondary source.
  - Set LENGTH bit. The counter counts to the compare value and then re-initializes itself to the value specified in the LOAD register.
  - Because its upper and lower limits are both set to 0, its output is cascaded count up and count down signals each time the quadrature inputs indicate a change in count.
  - Toggle OFLAG output on successful compare.
- TMRA1 is used as encoder counter receiving its counting instructions from TMRA0:

  - Cascade-Count mode.
  - Select counter 0 output as primary source.
  - Set LENGTH bit.
  - COMP1 = 3999. COMP2 = -3999. Due to encoder lines in this demo is 1000 lines, the max counter number is 3999.
  - Output is cascaded; count up and count down signals each time its counter reaches the max counter number.
- TMRA2 is used as revolution counter receiving its counting instructions from TMRA1:
  - Cascade-Count mode.
  - Select counter 0 output as primary source.
- TMRA3 is used as time counter:
  - Count mode.
  - Select IPG_CLK divide by 128 prescaler as primary source, select TMRA0 output as secondary source via XBAR.
  - Set CAPTURE_MODE bit. Load capture register on both edges of input from secondary source.
  - Set ROC bit. Reload the counter on a capture event.

2.8. FreeMASTER communication—LPUART

LPUART (Low-Power Universal Asynchronous Receiver and Transmitter) is used for the FreeMASTER communication between the RT1020 and the PC.

- The baud rate is set to 115200 bit/s.
- The receiver and transmitter are both enabled.
- The other settings are set to default.
- In "*freemaster_cfg.h*" file, add baud rate register address on RT1020, e.g. UART4.

  #define FMSTR_SCI_BASE   (0x40190010u)

3. System structure and software

3.1. System structure

Figure 6 presents the system structure block diagram of this dual servo demo.

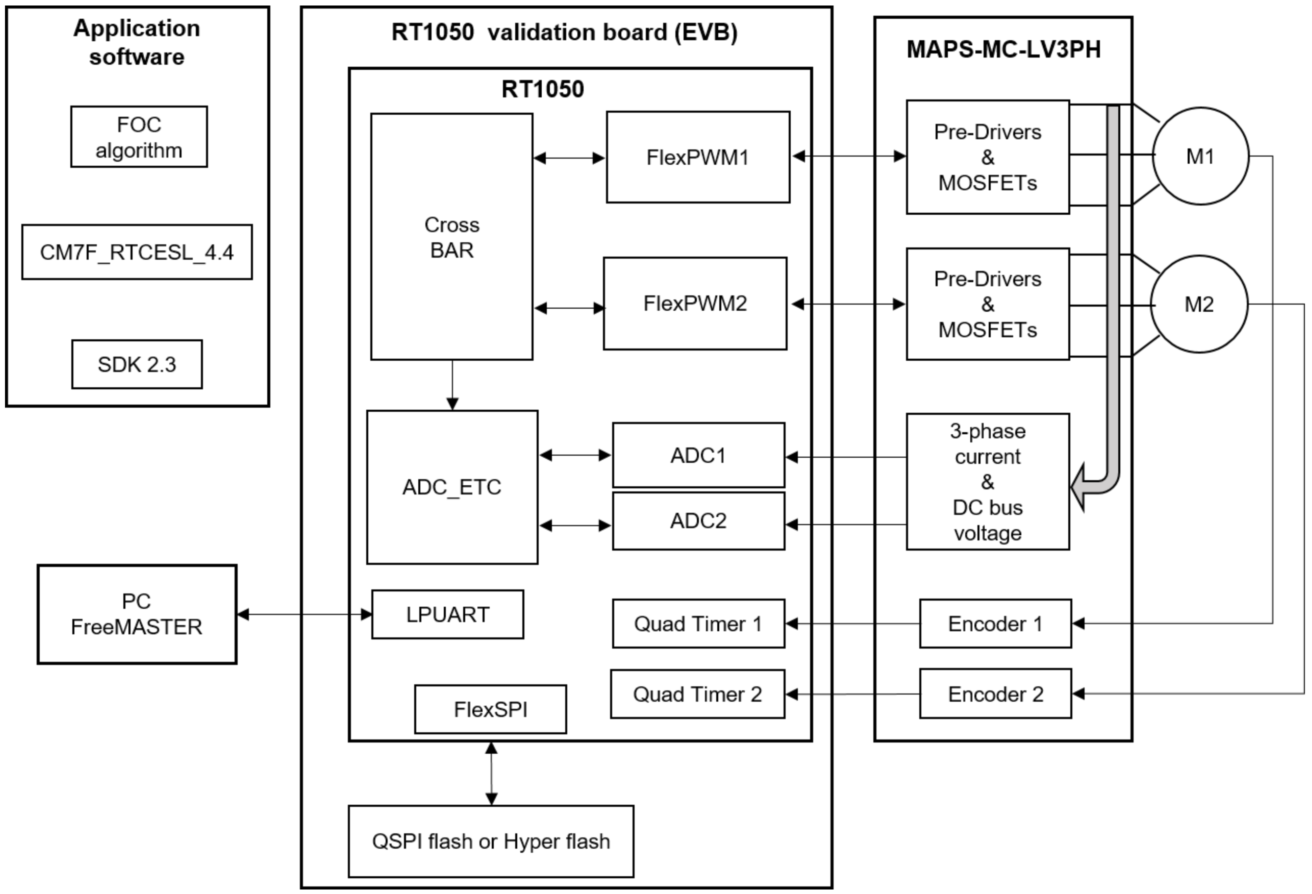


Figure 6. System structure block diagram

- RT1050 validation board (EVB) designed by NXP contains RT1050 chip, QSPI flash or hyper flash and connectors.
- MAPS-MC-LV3PH designed by NXP is dual servo motor driver board which contains driver bridges, analog sampling circuit and encoder interface.
- M1 and M2 are the dual servo motor included 1000 lines encoder and hall sensor.
- QSPI flash or hyper-flash provides code space for non-debugging running configuration. The i.MX RT1050 connects both with via the FlexSPI controller.
- Application software is running on RT1020 which includes field-oriented control (FOC) algorithm, CM7_RTCESL_4.4 (Real Time Control Embedded Software Motor Control and Power Conversion Libraries) and SDK 2.3.
- FreeMASTER run-time debugging tool communicates with RT1050 via LPUART to operate the demo by user.

3.2. Servo control structure

As shown in the Figure 7, the control blocks diagram of servo control in this demo are the classical three-loop structures.

The innermost loop is current control loop (fast loop) which contains analog signal sampling, FOC algorithm and PWM duty updating.

The middle loop is speed control loop. The comparison between the desired speed and the measured speed obtained from the speed measurement method generates a speed error. The speed error is input to the speed PI controller, generating a new desired value for the torque-producing component of the stator current.

The outermost loop is the position control loop. The position command that is entered from the high-level application layer. The comparison between the actual position speed command and the measured position generates a position error. The position error is input to the position controller, generating a new reference speed.

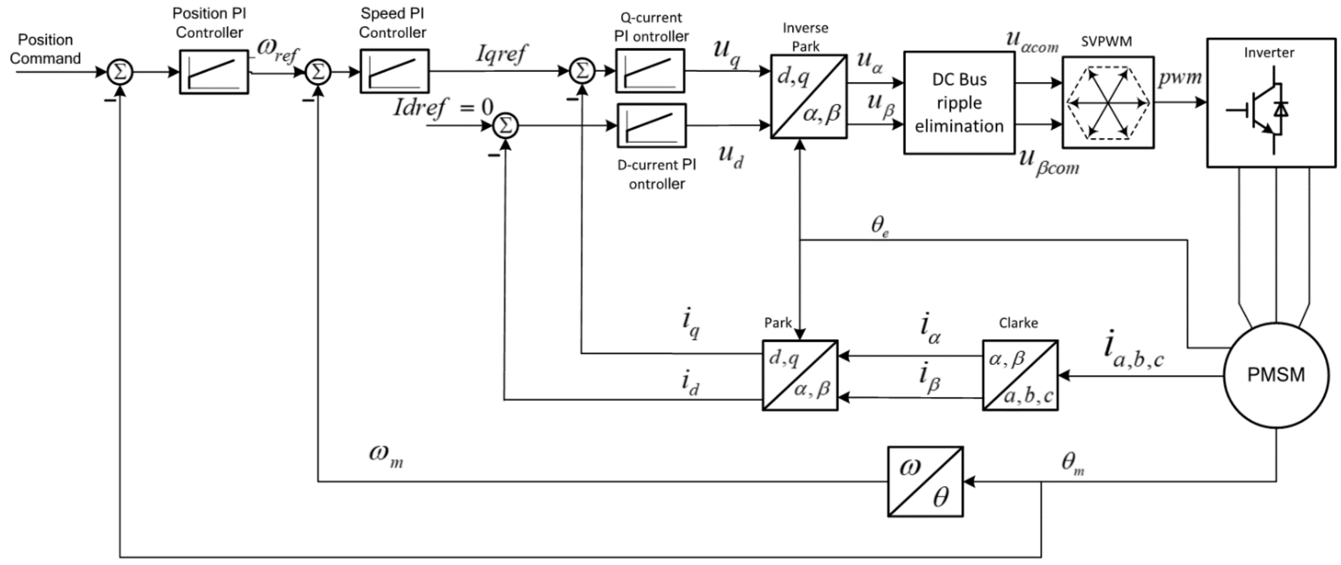


Figure 7. control blocks diagram

3.3. Synchronization

As discussed in section 2.2, there is a 180-degrees phase lag between the PWMs of dual motors, and on this basis ADC_ETC uses the triggers from eFlexPWM to implement time division sampling of analog signals of dual motors.

Figure 8 presents the synchronization between ADC and PWMs.

Once eFlexPWM1 counter reaches submodule 1 VAL4, ADC_ETC will be triggered to control ADCs to sampling analog signal of motor 1. After ADC conversation, ADC_ETC_DONE0 interrupt is enabled to run motor 1 control algorithm.

Once eFlexPWM2 counter reaches submodule 0 VAL4, ADC_ETC will be triggered to control ADCs to sampling analog signal of motor 2. After ADC conversation, ADC_ETC_DONE0 interrupt is enabled to run motor 2 control algorithm.

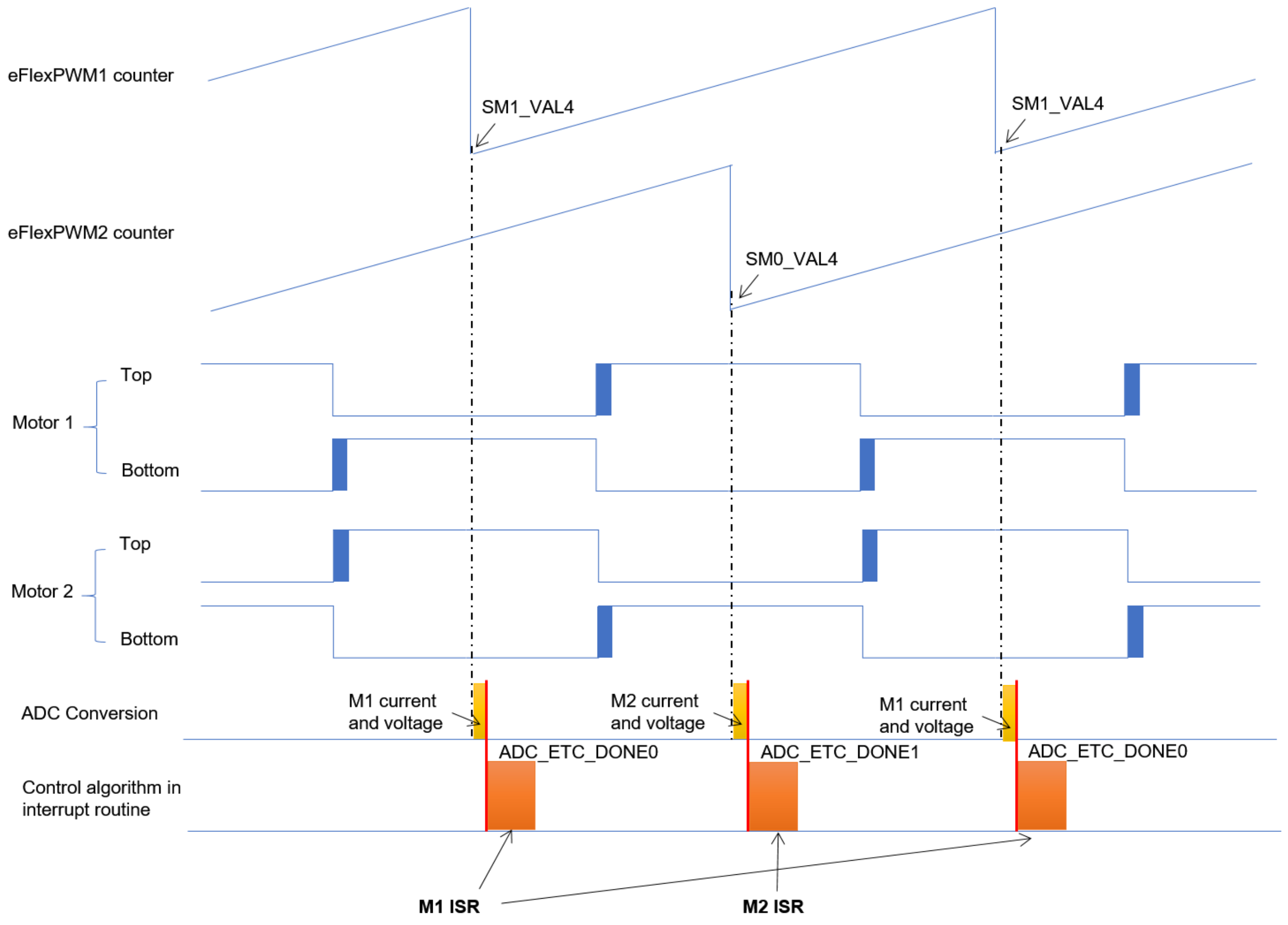


Figure 8. Synchronization between ADC and PWMs

3.4. Project file structure

The total number of source (*.c) and header files (*.h) in the project exceeds one hundred. Therefore, only the key project files will be described in detail, and the rest will be described in groups.

The main project folder is divided into seven directories:

- \boards\dualservo — contains initialization configuration files for hardware board.
- \boards\dualservo\iar — contains compiler necessary files.
- \boards\dualservo\motor_control — contains motor control algorithm files and state machine files.
- \boards\ dualservo\parameter — contains parameter header files and configuration file.
- \CMSIS — Cortex Microcontroller Software Interface Standard.
- \devices\MIMXRT1021 — RT1020 software Development Kit.
- \FM_ControlPage — FreeMASTER control page files.

- \middleware\ freemaster— FreeMASTER support files.
- \middleware\ CM7_RTCESL_4.4_IAR— Real Time Control Embedded Software Motor Control and Power Conversion Libraries.

Files in the folders:

- M1_statemachine.c and M1_statemachine.h contain the software routines executed when the application is in a particular state or state transition
- State_machine.c and state_machine.h contain the application state machine structure definition and manage switching between the application states and application state transitions
- Motor_structure.c and motor_structure.h contain the structure definitions and subroutines dedicated for execution of the motor control algorithm (vector control algorithm, position and speed estimation algorithm, speed control loop)
- Motor_def.h contains the main control and fault structure definition.

4. Demo operation

4.1. Setup dual servo demo

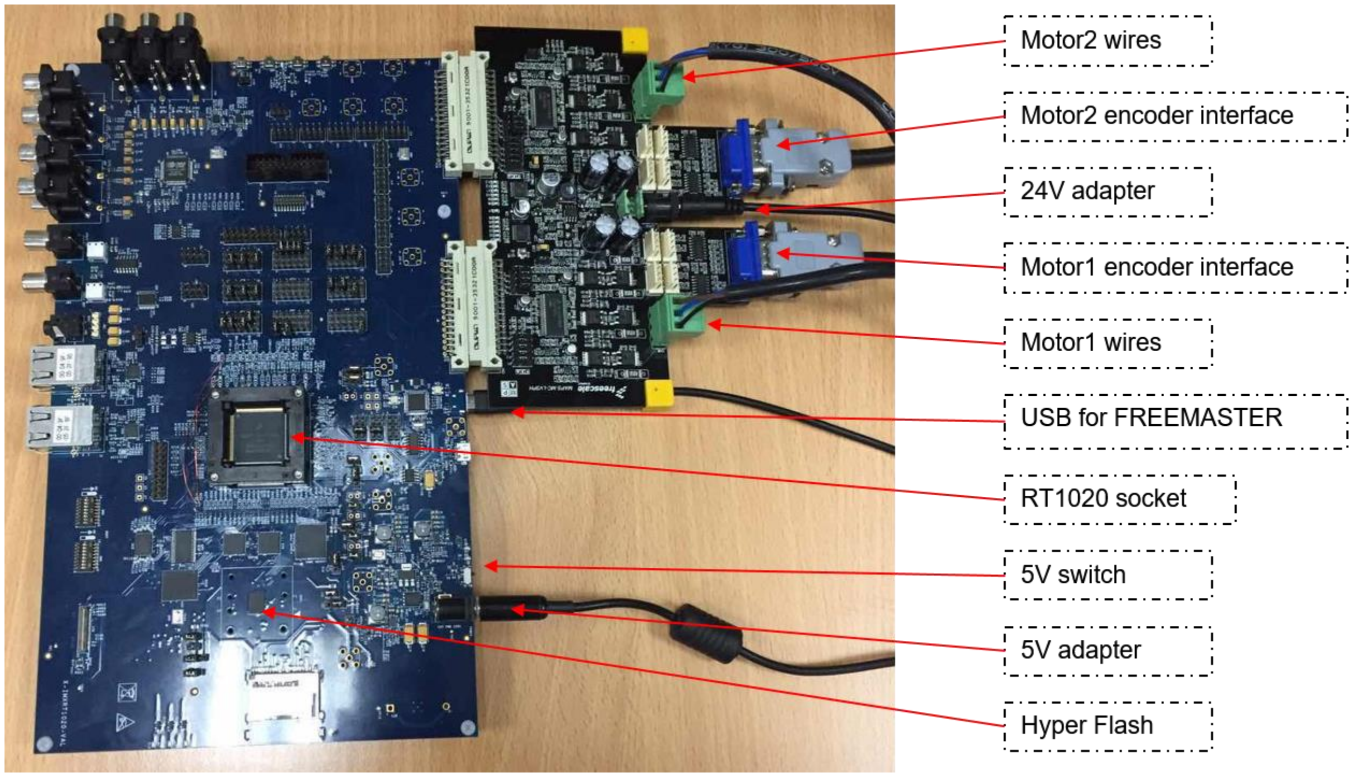


Figure 9. Setup dual servo demo

To setup dual servo demo, follow the steps below:

NOTE

Make sure adapters are powered off before all steps.

1. According to Figure 9, plug RT1020 EVB & motor board together and connect all interface.
2. Power on 24 V adapter to power on motor board.
3. Power on 5 V adapter.
4. Toggle 5 V switch to power on RT1020 EVB. Now RT1020 is booting from hyper flash.
5. Plug USB into PC.
6. Open "*FM_DualServo.pmp*" in the software package. (FREEMATER version need to be not lower than 2.0.5)
7. Click the **STOP** button to enable the communication between PC and RT1020.
8. Click the **DualServo** page.
9. Click the **Start** button to enable demo.
10. Then operate the demo by click other buttons on the control page.

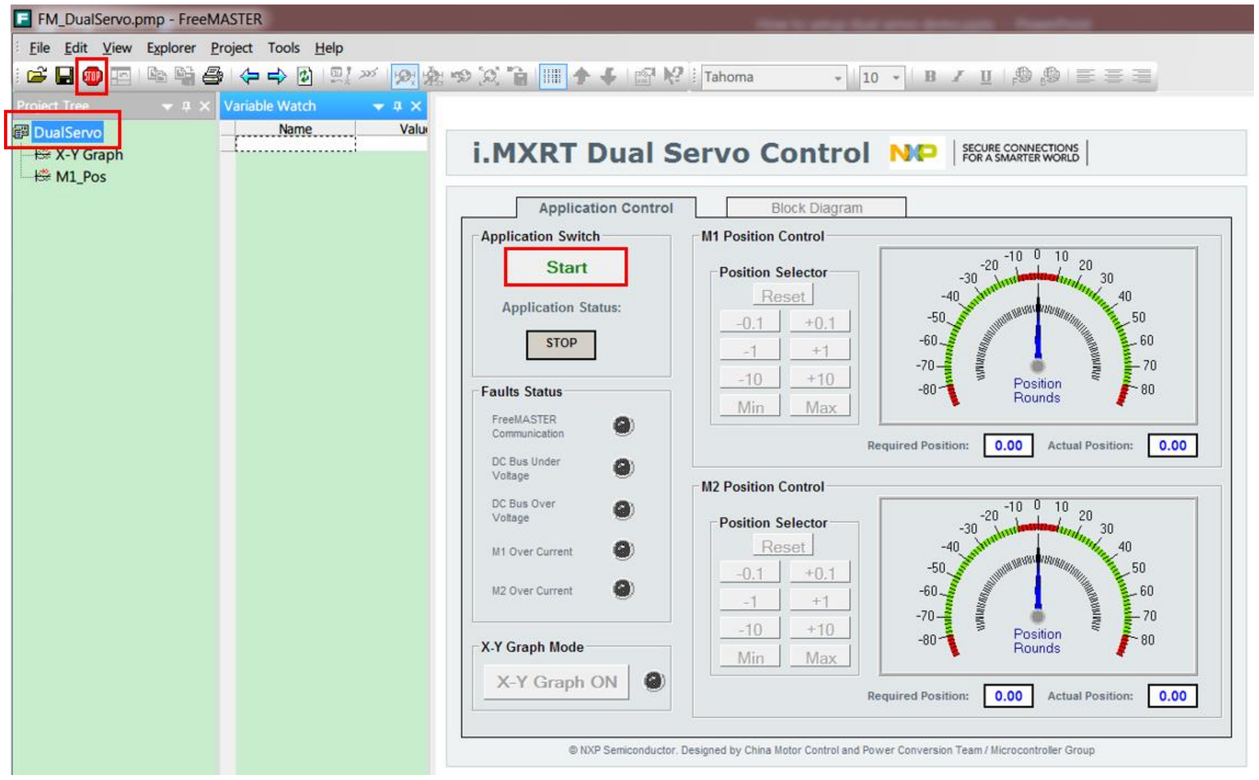
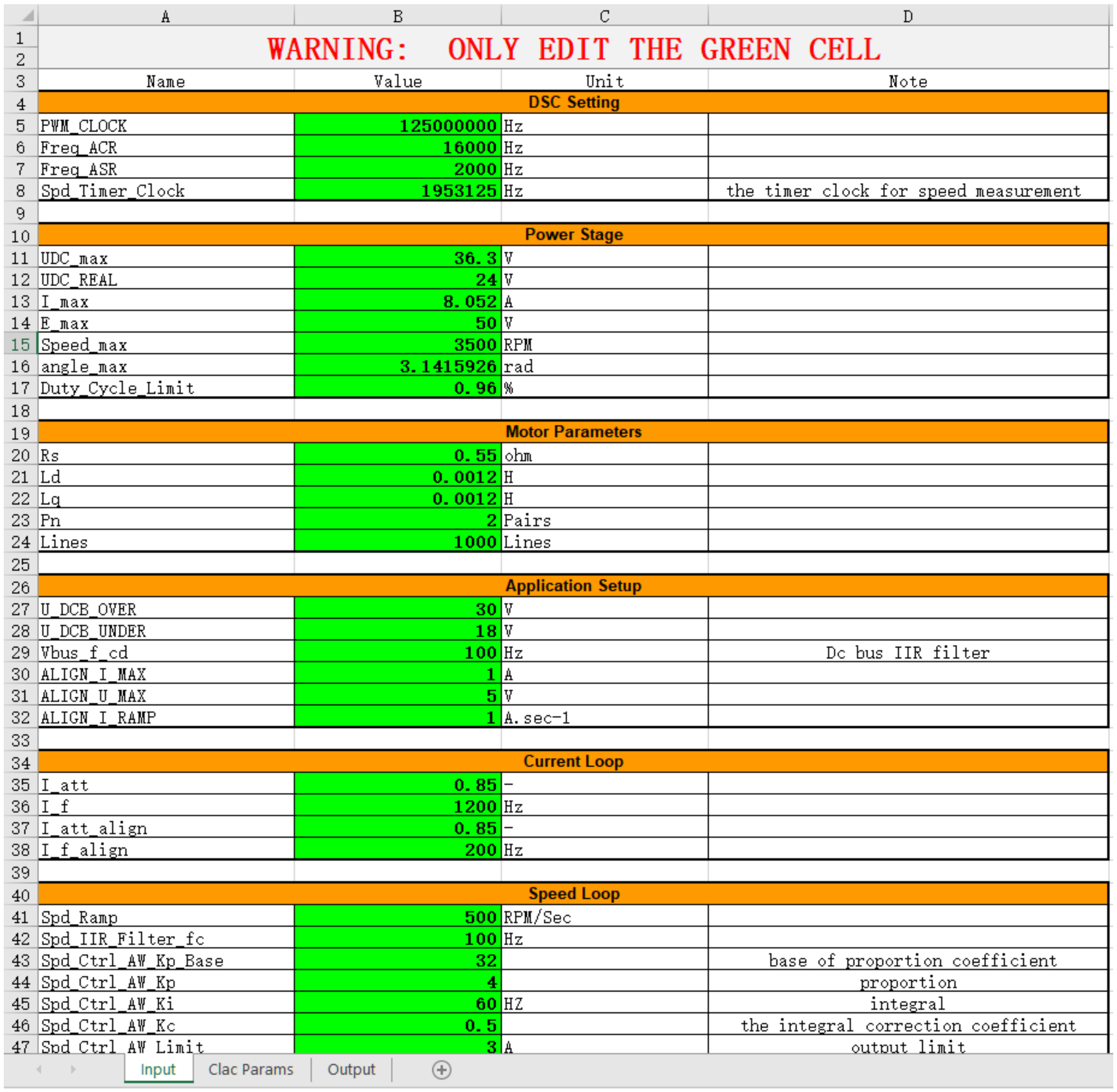


Figure 10. FreeMASTER control page

4.2. Parameter configuration

If the parameters of users' servo motor are different from those of default motor in this demo, the paraments should be re-configured for matching the different motor.

- Open the “input” page of “M1 Parameter Calculation.xlsx” file. As shown in the Figure 11, all the parameters in green cell are available to be modified to match the actual applications.
- The final parameters in the output page are calculated automatically from the input parameters.
- Copy the entire page named “output” into the header file (M1_Params.h or M2_Params.h). Now parameters re- configuring is done.



WARNING: ONLY EDIT THE GREEN CELL

Name	Value	Unit	Note
DSC Setting			
PWM_CLOCK	125000000	Hz	
Freq_ACR	16000	Hz	
Freq_ASR	2000	Hz	
Spd_Timer_Clock	1953125	Hz	the timer clock for speed measurement
Power Stage			
UDC_max	36.3	V	
UDC_REAL	24	V	
I_max	8.052	A	
E_max	50	V	
Speed_max	3500	RPM	
angle_max	3.1415926	rad	
Duty_Cycle_Limit	0.96	%	
Motor Parameters			
Rs	0.55	ohm	
Ld	0.0012	H	
Lq	0.0012	H	
Pn	2	Pairs	
Lines	1000	Lines	
Application Setup			
U_DCB_OVER	30	V	
U_DCB_UNDER	18	V	
Vbus_f_cd	100	Hz	Dc bus IIR filter
ALIGN_I_MAX	1	A	
ALIGN_U_MAX	5	V	
ALIGN_I_RAMP	1	A.sec-1	
Current Loop			
I_att	0.85	-	
I_f	1200	Hz	
I_att_align	0.85	-	
I_f_align	200	Hz	
Speed Loop			
Spd_Ramp	500	RPM/Sec	
Spd_IIR_Filter_fc	100	Hz	
Spd_Ctrl_AW_Kp_Base	32		base of proportion coefficient
Spd_Ctrl_AW_Kp	4		proportion
Spd_Ctrl_AW_Ki	60	HZ	integral
Spd_Ctrl_AW_Kc	0.5		the integral correction coefficient
Spd_Ctrl_AW_Limit	3	A	output limit

Figure 11. FreeMASTER control page

4.3. CPU load and memory usage

The following information apply to the demo application built using the IAR Embedded Workbench® IDE in the Debug RAM and FLASH configurations. Table 2 shows the memory usage and CPU load. The memory usage is calculated from the linker .map file (IAR IDE), including the 4-KB FreeMASTER recorder buffer allocated in RAM. The CPU load is measured using the SysTick timer.

In this case, it applies to the fast loop frequency of 16 kHz and the slow loop (speed & position loop) frequency of 2 kHz.

Table 2. RT1020 dual servo demo CPU load and memory usage

—	Fast loop (Debug RAM)	Slow loop (Debug RAM)	Fast loop (Flash)	Slow loop (Flash)
CPU load	2.172 us (1086 clocks)	3.468us (1734 clocks)	Depend on flash type	Depend on flash type
ROM code memory [bytes]	—	—	33604	33064
RAM code memory [bytes]	33708	33708	—	—
ROM data memory [bytes]	—	—	448	448
RAM data memory [bytes]	8148	8148	7700	7700

5. References

Following documents may offer further reference.

- i.MX RT1020 Processor Reference Manual Rev. C (document IMXRT1020RM)
- PMSM Field-Oriented Control on MIMXRT1050 EVK (document AN12169)

6. Revision history

The following table summarizes the changes since the initial release.

Table 3. Revision history

Revision number	Date	Substantive changes
0	06/2018	Initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

•

Document Number: AN12200
Rev. 0
06/2018