

Low Cost PMSM Sensorless Field-Oriented Control Based on KE02

1. Introduction

This application note describes the design of a sensorless vector control algorithm for a three-phase PMSM based on NXP’s MKE02Z64 dedicated motor control devices.

Sinusoidal Permanent Magnet Synchronous Motors (PMSM) are getting more and more popular, replacing brushed DC, universal, and other motors in a wide application area.

The detailed PMSM FOC control algorithm refers to *Sensorless PMSM Field-Oriented Control* (document [DRM148](#)).

This application note includes two FOC implement methods corresponding to two different ADC current sampling methods (three shunt and single shunt). It outlines the MCU peripherals used in the application, the hardware set up, and the software design.

Contents

1.	Introduction	1
2.	MCU peripherals.....	2
2.1.	FlexTimer2 configuration to generate a 6-channel PWM.....	2
2.2.	ADC modules configuration.....	4
2.3.	ADC conversion timing, currents and voltage sampling.....	5
2.4.	Current measurement	7
2.5.	SCI (UART) configuration.....	12
3.	Interrupts	12
3.1.	Interrupt	12
4.	Software description.....	14
4.1.	Anti-wind function.....	14
4.2.	Application state machine	14
5.	Hardware setup	18
5.1.	Hardware setup and configuration	18
6.	Application operation	20
6.1.	FreeMASTER installation on a PC or notebook ...	20
6.2.	Establishing connection between PC and embedded application	20
6.3.	Application operation using FreeMASTER.....	20
7.	References	25
8.	Revision History	25



2. MCU peripherals

Table 1 summarizes the peripherals on the Kinetis MKE02Z64 MCU and their usage by the PMSM sensorless vector control application:

Table 1. Kinetis MKE02Z64 peripherals overview

Kinetis KE02 peripherals			Used in the application		Purpose
Group	Module	Number of modules or channels	Three Shunt current sampling	Single Shunt current sampling	
Analog	ADC	12-bit successive-approximation ADC module with up to 16 channels	4 channels	2 channels	DC-bus voltage and motor phase currents sensing
	Comparators	2 modules	—	—	—
	DAC	1 module	—	—	—
Communications	SPI	Two 8-bit SPI modules	—	-	—
	UART	3 SCI/UART modules	1 module	1 module	FreeMASTER communication
	I ² C	1 I ² C module	—	—	—
Timer	FTM	10 channels	6 channels	6 channels	Generate 6-channels PWM for motor control
	PIT	2 channels	—	—	—
	RTC	1 module	—	—	—

2.1. FlexTimer2 configuration to generate a 6-channel PWM

The FlexTimer Module (FTM) is a two to eight channel timer which supports input capture, output compare, and the generation of PWM signals to control an electric motor and power management applications. The FTM time reference is a 16-bit counter that can be used as an unsigned or signed counter.

On the Kinetis KE02 there are three instances of FTM. One FTM has 6 channels, the other two FTMs both have 2 channels.

Configuration for generating a center-aligned PWM with dead time insertion is described in the application note titled *Using FlexTimer in ACIM/PMSM Motor Control Applications* (document [AN3729](#)).

The configuration is as follows:

- It is necessary to enable the system clock for the FlexTimer module in the Clock Gating Control register:
SIM→SCGC |= SIM_SCGC_FTM2_MASK;
- It is necessary to disable the write protection of some registers before they can be updated:
FTM2→MODE |= FTM_MODE_WPDIS_MASK;
- It is necessary to enable all FTM registers including the FTM-specific registers (second set of registers) are available for use with no restrictions.
FTM2→MODE |= FTM_MODE_FTMEN_MASK;

- It is recommended to enable the internal FlexTimer counter to run in the debug mode:
FTM2→CONF |= FTM_CONF_BDMODE(3);
- When the HW debugging interface (jLink, Multilink,...) is connected to the microcontroller, the MCU is in debug mode. This is not dependent on running code containing breakpoints.
- For Field-Oriented control we should configure FTM to combine and complement mode with 2µs deadline:

```
FTM2->COMBINE = FTM_COMBINE_FAULTEN0_MASK | FTM_COMBINE_SYNCEN0_MASK
| FTM_COMBINE_DTEN0_MASK | FTM_COMBINE_COMP0_MASK |
FTM_COMBINE_COMBINE0_MASK | FTM_COMBINE_FAULTEN1_MASK |
FTM_COMBINE_SYNCEN1_MASK | FTM_COMBINE_DTEN1_MASK
| FTM_COMBINE_COMP1_MASK | FTM_COMBINE_COMBINE1_MASK
| FTM_COMBINE_FAULTEN2_MASK | FTM_COMBINE_SYNCEN2_MASK |
FTM_COMBINE_DTEN2_MASK | FTM_COMBINE_COMP2_MASK
| FTM_COMBINE_COMBINE2_MASK;
FTM2->DEADTIME = FTM_PWM_DEAD_TIME;
```

```
( #define FTM_PWM_DEAD_TIME 40 )
```

- The PWM signals generated by the FlexTimer2 are directly connected to the MOSFET driver. Therefore, it is necessary to set the right polarity of the PWM signals. There is no inverse polarity pin based on HVP platform, therefore:

```
FTM2→POL = 0x00;
```

- The duty cycle is changed by changing the value of the FlexTimer Value registers. These registers are double-buffered, meaning that their values are updated not only by writing the number, but also by confirming the change by setting the Load Enable (LDOK) bit. This ensures that all values are updated at the same instance.

```
FTM2→PWMLOAD = FTM_PWMLOAD_LDOK_MASK;
```

- It is necessary to write the LDOK bit every time the value registers are changed, so not only at the stage of loading them with initial values, but with every update after the duty cycle value is computed in the vector control algorithm.
- Within the application, hardware triggering of the AD converter is employed. The Initialization Trigger signal(For Three shunt current sampling method) or The Match Trigger signal(For Single shunt sampling method) from the FlexTimer is used as the primary triggering signal which is fed into the SIM_SOPT→DELAY that services the timing of the AD conversion initiation.

```
Three Shunt: FTM2→EXTTRIG |= FTM_EXTTRIG_INITTRIGEN_MASK;
```

```
Single Shunt: FTM2→EXTTRIG |= FTM2->EXTTRIG |=
FTM_EXTTRIG_CH0TRIG_MASK|FTM_EXTTRIG_CH2TRIG_MASK|
FTM_EXTTRIG_CH4TRIG_MASK;
```

- In this application, six channel PWM output must be masked when over current or over voltage fault occurred on hardware board. Also a fault interrupt should be generated when a fault occurred. Manual fault clearing is selected in this case.

```
FTM2->MODE |= FTM_MODE_FAULTM(2) | FTM_MODE_FAULTIE_MASK;
```

- FTM2 interrupt is generated on every four PWM reload:

```
FTM2->SC |= FTM_SC_TOIE_MASK;
```

```
FTM2->CONF |= FTM_CONF_NUMTOF(3);
```

- Finally, the output pins of the MCU must be configured, to bring out the signals from the chip. The assignment of signals to output pins is set in the Pin Control Register.

```
SIM->PINSEL |= SIM_PINSEL_FTM2PS3_MASK | SIM_PINSEL_FTM2PS2_MASK |  
SIM_PINSEL_FTM2PS1_MASK | SIM_PINSEL_FTM2PS0_MASK;
```

2.2. ADC modules configuration

The on-chip ADC module is used to sample feedback signals (motor phase currents and DC-bus voltage) that are necessary to successfully perform the vector control algorithm. The FTM2 Trigger Delay register `SIM_SOPT→DELAY` closely cooperates with the ADC and triggers the hardware for sampling.

- It is necessary to enable the system clock for the ADC module in the Clock Gating Control register:

```
SIM→SCGC |= SIM_SCGC_ADC_MASK;
```

- In this application the ADC modules are configured to a 12-bit accuracy. The System clock is set to 20 MHz, so by using available prescaler values the input clock to the ADC module is set to 5 MHz:

```
ADC->SC3 = ADC_SC3_MODE(2) | ADC_SC3_ADIV(2) | ADC_SC3_ADICLK(0);
```

- Then, the hardware trigger must be enabled in the Status and Control Register 2.

```
ADC->SC2 |= ADC_SC2_ADTRG_MASK;
```

This application note includes two FOC implement methods corresponding to two different ADC current sampling methods (three shunt and single shunt):

- For three shunt current sampling method, finally select PHASE A current channel as ADC input:

```
ADC->SC1 = ADC_INPUT_SELECT_PHASE_A;
```

Then we can use software trigger to sample the next two phase current in ADC end of scan interrupt.

- For Single shunt current sampling method, you must sample the DC bus current three times at certain time point in one PWM cycle. In this chip, ADC FIFO operation can be enabled to achieve this purpose.

The FIFO function is enabled when the `ADC_SC4[AFDEP]` bits are set non-zero, in this application 3 levels FIFO depth is selected:

```
ADC->SC4 = ADC_SC4_AFDEP(2);
```

When FIFO is enabled, the result FIFO is read via `ADC_R`. The ADC conversion completes when the input channel FIFO is fulfilled at the depth indicated by the `AFDEP`. The AD result FIFO can be read via `ADC_R` continuously by the order set in analog input channel `ADCH`, therefore write DC bus ADC channel three times and enable ADC interrupt at the last one.

```

ADC->SC1 = ADC_INPUT_SELECT_DCB_I;
ADC->SC1 = ADC_INPUT_SELECT_DCB_I;
ADC->SC1 = ADC_SC1_AIEN_MASK|ADC_INPUT_SELECT_DCB_I;

```

2.3. ADC conversion timing, currents and voltage sampling

Three shunt current sampling:

FlexTimer2 is configured to trigger an internal hardware signal when its counter is reset after overflow to the initialization value. ADC will be triggered by this FTM unit trigger with a predefined delay. There is only one ADC module on KE02, and long ADC sampling time will be needed to get all three-phase current during one PWM cycle. This will lead to low voltage length and maximum speed limitation because longer zero vector is needed for ADC sampling. In some low-cost application areas cases such as fan and pumps, ADC FIFO operation can be enabled to get different phase current at different PWM cycle. This will help to increase the PWM frequency (reduce electromagnetic noise) and improve maximum motor speed.

In this application, ADC FIFO operation can be used to set the control frequency every four PWM cycle once. So, 4 levels FIFO depth is selected:

```
ADC->SC4 = ADC_SC4_AFDEP(3);
```

The AD result can be read via ADC_R continuously by the order set in analog input channel ADCH, so we should write corresponding ADC channel and enable ADC interrupt at the last one.

```
/*IPM TEMP channel*/
```

```
ADC->SC1 = ADC_INPUT_SELECT_TEMPERATURE;
```

```
/*DC bus voltage channel*/
```

```
ADC->SC1 = ADC_INPUT_SELECT_DCB_V;
```

```
/*Phase A current channel*/
```

```
ADC->SC1 = ADC_INPUT_SELECT_PHASE_A;
```

```
/*Phase B current channel*/
```

```
ADC->SC1 = ADC_SC1_AIEN_MASK|ADC_INPUT_SELECT_PHASE_B;
```

Figure 1 shows the module interconnections and the ADC interrupt generation.

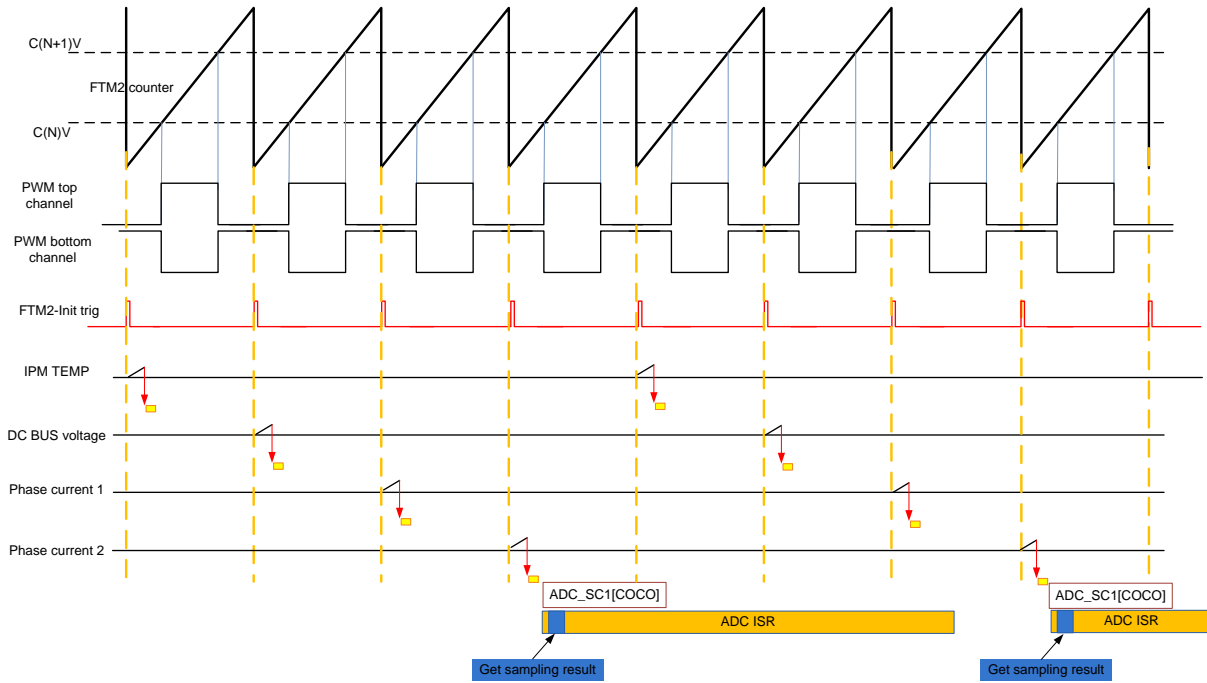


Figure 1. ADC conversion timing diagram of three shunt sampling

Single shunt current sampling:

The FTM is able to generate multiple triggers in one PWM period. Because each trigger is generated for a specific channel, the FTM generates a trigger when the channel (i) match occurs (FTM counter = C(i)V).

In this application the channel 0,2,4 trigger output provides a trigger signal that is used for single shunt current sampling. ADC ISR is enabled in FTM over flow ISR, and then generated when FIFO depth is matched (three level depth in this application). DCBus voltage will be sampled by software trigger in ADC ISR.

Figure 2 shows the module interconnections and the ADC interrupt generation (take SVM sector 1 as an example).

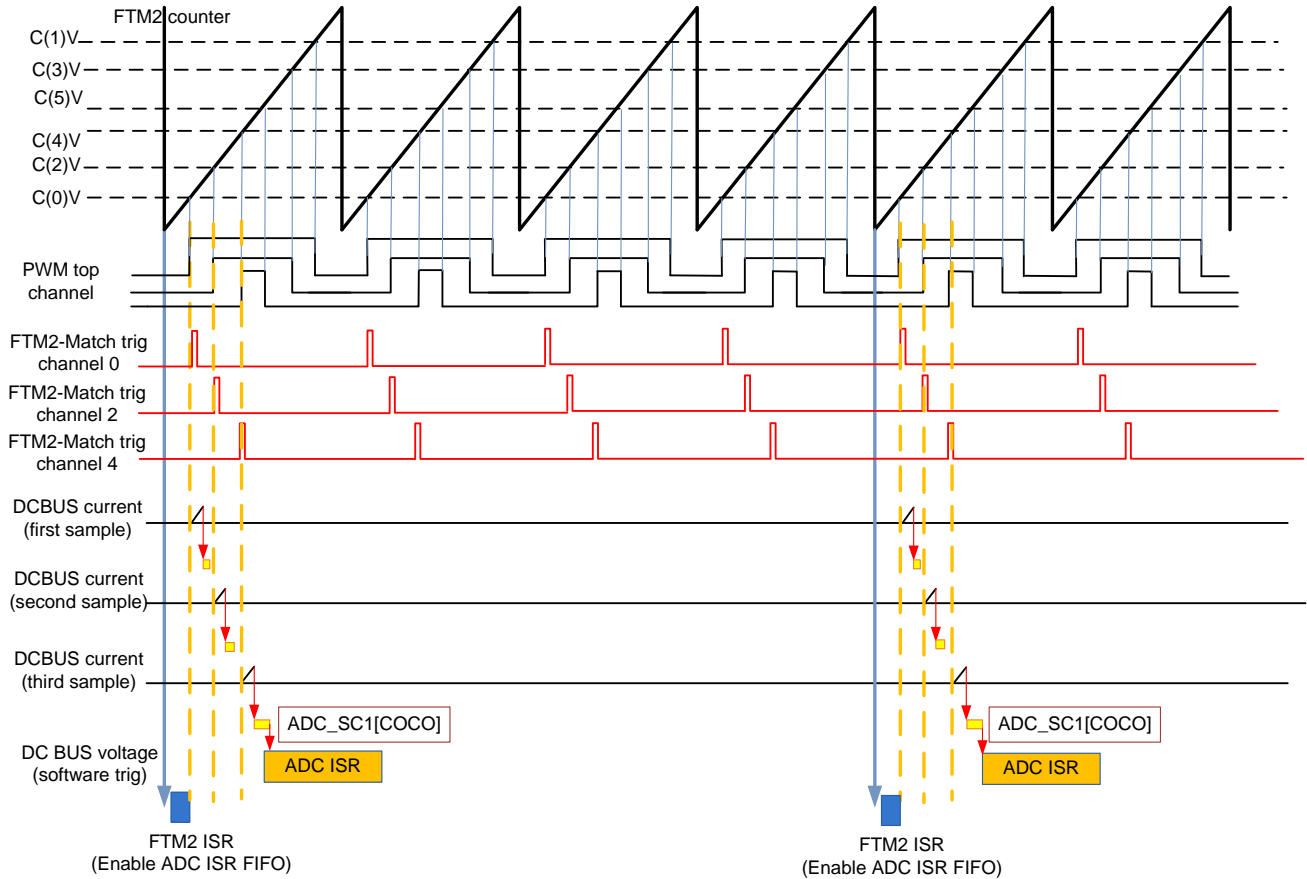


Figure 2. ADC conversion timing diagram of single shunt sampling

2.4. Current measurement

Three shunt current sampling:

Closely related to the ADC conversion trigger timing is the assignment of the ADC channels to the measured analog signals. For computation of the fast (current) control loop of the FOC, it is necessary to know the values of all three motor phase currents. Assuming the motor represents a symmetrical 3-phase system, the sum of all three instantaneous phase currents is zero.

$$0 = I_a + I_b + I_c \tag{Eqn. 1}$$

Because the phase currents are measured the instance when the bottom transistors are conducting, in the case of high duty cycle ratios (current value is in the area of the maximum of the sine curve), the time when the current can be measured is too short. The bottom transistor must be switched on at least for a critical pulse width to get a stabilized current shunt resistor voltage drop. The selection of the channels is determined based on the section when the space vector of the stator current is generated. This assignment is performed at the end of the ADC interrupt service routine. Therefore, it is enough to sample only two phase currents while the third is easily calculated according to the following equations.

Figure 3 shows two cases (case I at 30°, case II at 60°) which explain why calculating the third current is necessary.

Sector1,6 $I_a = -I_b - I_c$ Eqn. 2

Sector2,3 $I_b = -I_a - I_c$ Eqn. 3

Sector4,5 $I_c = -I_b - I_a$ Eqn. 4

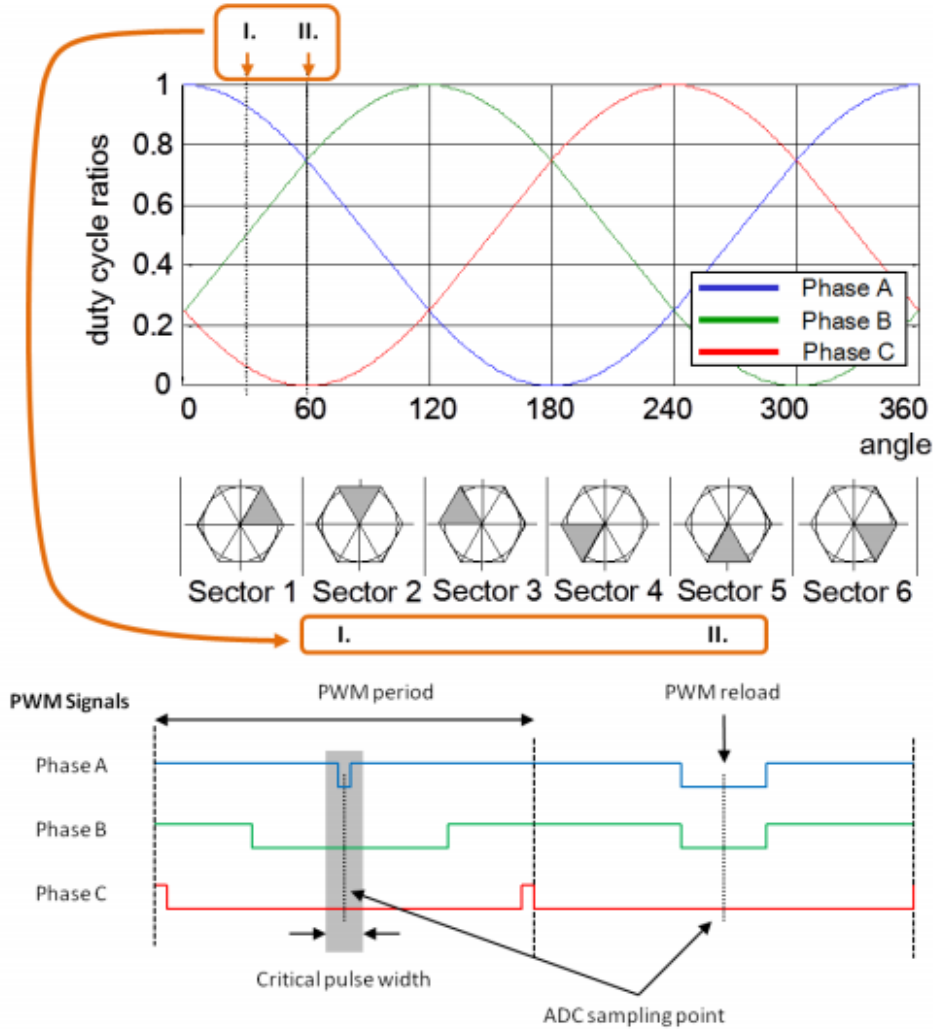


Figure 3. Three shunt current sensing

As the figure shows, at 60 ° all three currents can be sampled because, as mentioned previously, the currents are sampled when the bottom transistors are turned-on. Therefore, the pulse width is sufficient to stabilize the current and to perform the acquisition of the signal value by the AD converter. At 30 °, the pulse is too short, so the current of Phase A cannot be sampled.

Single shunt current sampling:

The vector control algorithm requires the sensing of the three motor phase currents. A standard approach is to sense the phase currents directly through current transformers, or Hall effect sensors, directly coupled to the motor phase lines that carry the current between the switches and the motor. To reduce the number of current sensors and overall cost of the design, the three-phase stator currents are measured

by means of a single DC-Link current shunt sensor; see [Figure 4](#). The DC-Link current pulses are sampled at exactly timed intervals. A voltage drop on the shunt resistor is amplified by an operational amplifier inside the 3-phase driver and shifted up by 1.65 V. The resultant voltage is converted by the ADC; see [Figure 5](#).

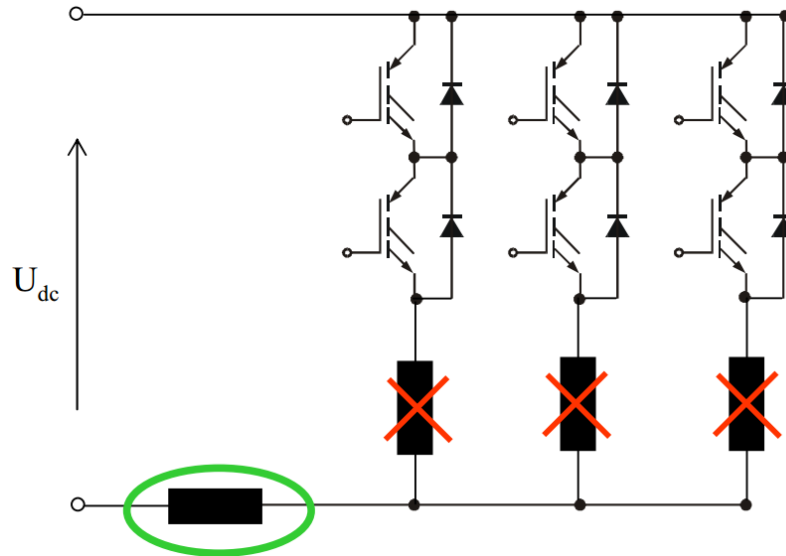


Figure 4. DC-Link current shunt

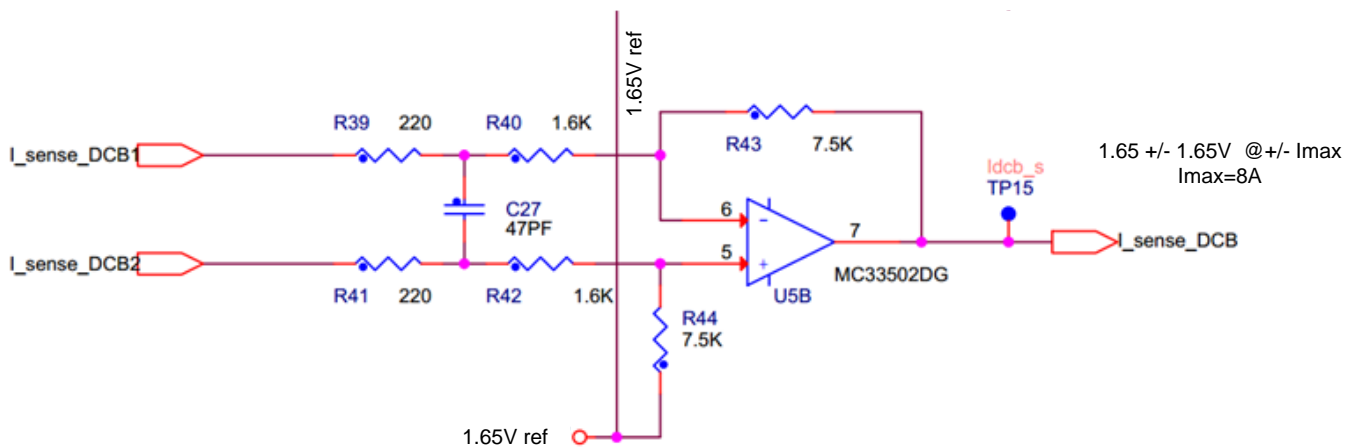


Figure 5. Current amplifier for DC-Link current

Based on the actual combination of switches, the three-phase currents of the stator are reconstructed.

The AD converter measures the DC-link current during the active vectors of the PWM cycle. When the voltage vector V_1 is applied, current flows from the positive rail into the phase A winding, and returns to the negative rail through the B and C phase windings. When the voltage vector V_2 is applied, the DC link current returning to the negative rail equals the T phase current. Therefore, in each sector, two phase current measurements are available, see [Figure 6](#). The calculation of the third phase current value is possible because the three winding currents sum to zero. The voltage vector combination and corresponding reconstructed motor phase currents are shown in [Table 2](#).

Table 2. Measured currents

Voltage Vector	DC-link Current
V1 (100)	+i _a
V2 (110)	-i _c
V3 (010)	+i _b
V4 (011)	-i _a
V5 (001)	+i _c
V6 (101)	-i _b
V7 (111)	0
V0 (000)	0

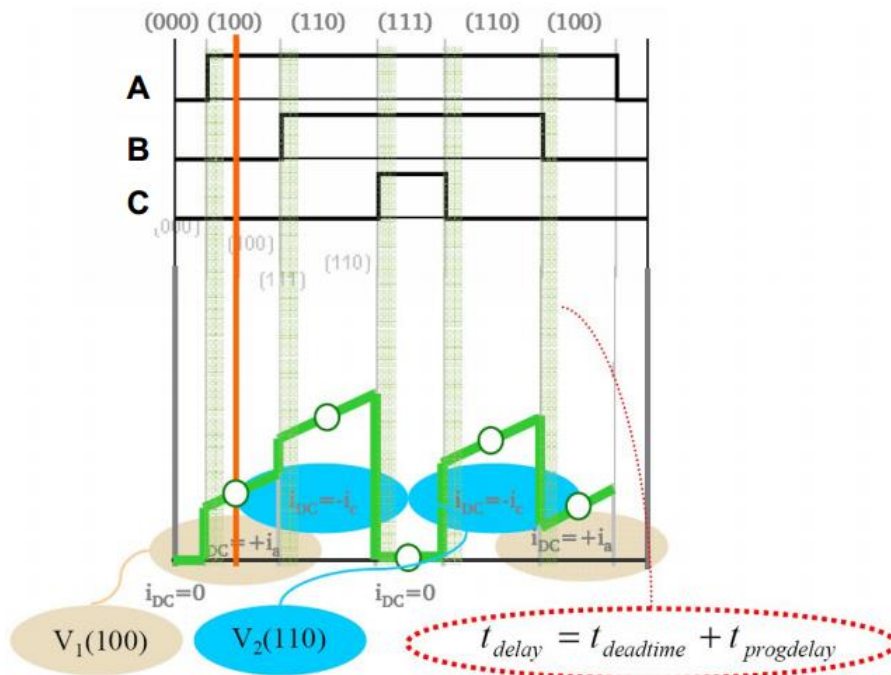


Figure 6. Single shunt Current Sampling Timing

However, the DC-Link current cannot be measured in two cases:

- When the voltage vector is crossing a sector border. In this case, only one sample can be taken; see Figure 7.
- When the modulation index is low - sampling interval is too short and none of the current samples can be taken; see Figure 8.

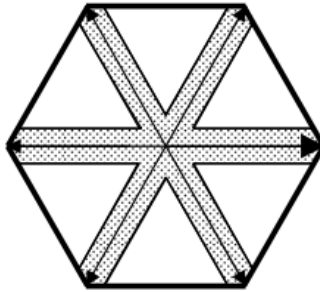


Figure 7. Passing Active Vector

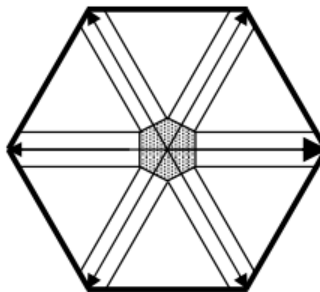


Figure 8. Low Modulation Index

This current measurement limitation can be partly solved using asymmetrical PWM pulses. Two PWM pulses are shifted in order to obtain enough time for current sampling. Nevertheless, duty cycles for all the PWM pulses have to be preserved.

The solution for asymmetrical PWM's use can be applied in both cases. In the first case, the voltage vector crosses a sector border, the center edge of the PWM period is frozen and one critical edge is moved; see [Figure 9](#). In the second case, when the modulation index is low, the center edge remains frozen as well, and both side edges are moved in opposite directions; see [Figure 10](#).

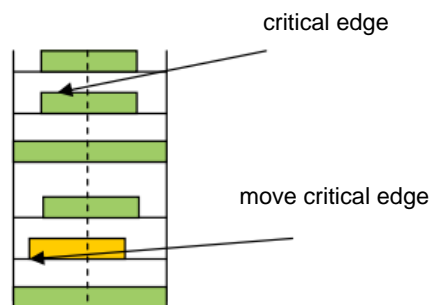


Figure 9. Edge moving when passing Active Vector

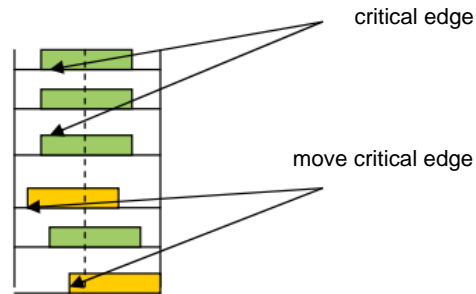


Figure 10. Edge moving when Low Modulation indexes

2.5. SCI (UART) configuration

The SCI is used in the application for the communication between the master system and the embedded application. A master system is the notebook or the PC where the FreeMASTER software is installed in order to control the application and visualization of its state. On the Kinetis KE02, there are three UART modules implemented. Because the hardware solution is based on the Tower modules, the UART1 is used.

The communication speed is set to 9600 Bd and is limited by use of the OpenSDA - CDC serial communication driver.

3. Interrupts

In the application, there are 2 interrupts in total. ADC interrupt serves to execute fast and slow control loop for FOC control algorithms, FTM2 ISR serves to enable ADC ISR every fourth PWM cycle.

3.1. Interrupt

This interrupt request is triggered when the conversion of the ADC module is completed and has the highest priority. In the beginning of the ADC ISR execution an application state machine function is called. If the application is in the Run state, then it is followed by the execution of the fast (current) control loop of the PMSM vector control algorithm, including the position and speed estimation. The slow (speed) control loop is calculated based on the value of software counter that is decremented each time the fast control loop is passed. The interrupt flag is cleared by reading of the result register of the ADC channel that triggered the interrupt. Therefore, at the beginning of each particular state machine function, the results of AD conversion are read. Then the rest current and voltage are sampled by the software trigger in state machine function. The state machine of three shunt sampling and single shunt sampling are almost the same, For the sake of simplicity, the next introduction will take the three shunt sampling method as an example.

The flow chart shown in [Figure 11](#) provides an overview of the program flow during execution of the ADC interrupt service routine when the application is in Run state and Spin sub-state.

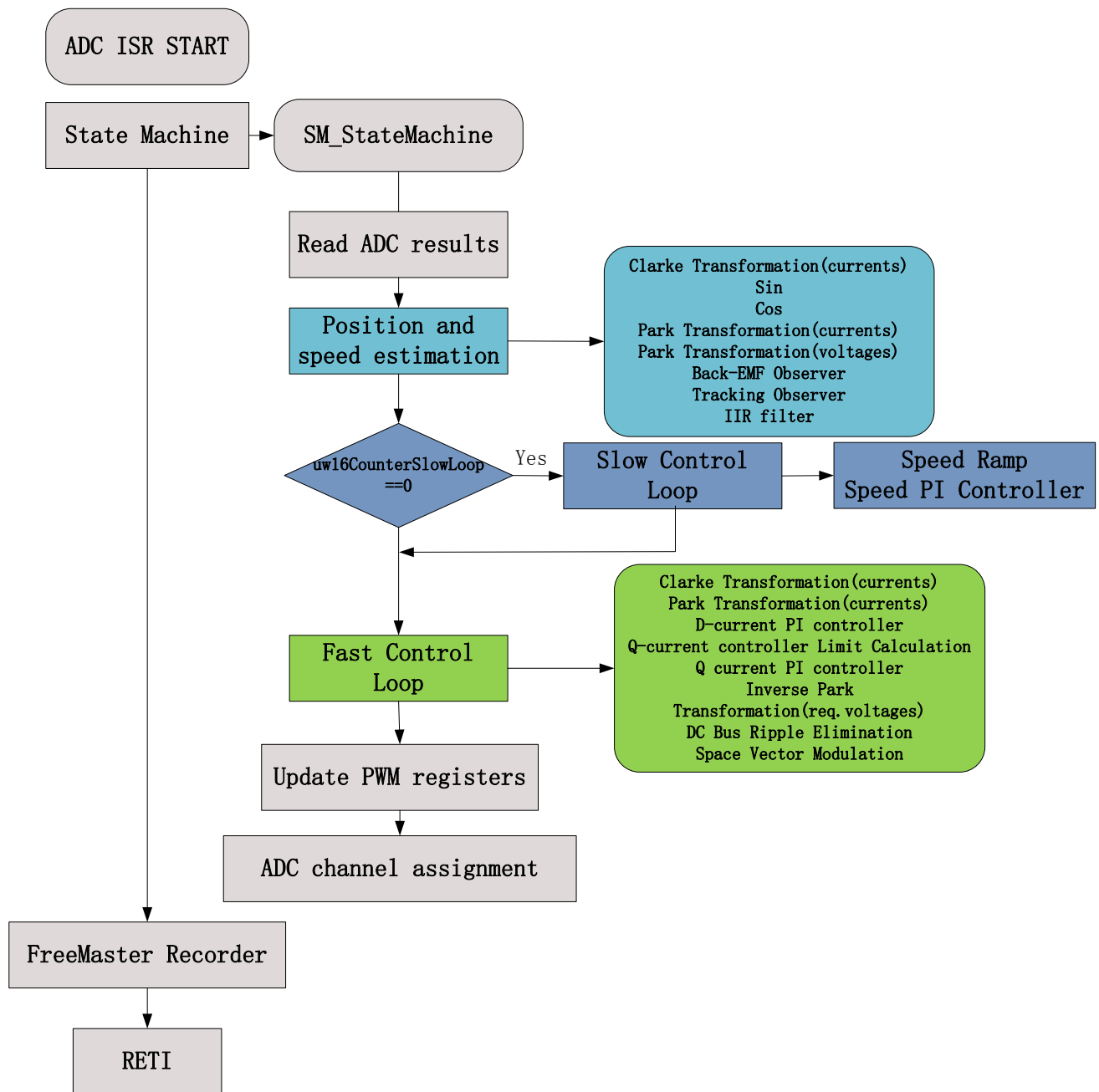


Figure 11. ADC ISR flow chart

4. Software description

4.1. Anti-wind function

The anti-wind function is a required feature for fan applications. Most ceiling fans with large plates are easily forced into rotation by air movement caused by a draft or the operation of other fans in the room. By using this application, a rotating PMS motor generates back electromotive force (back-EMF). To prevent current surges that can yield a system failure, the rotor is gradually stopped before the voltage vector is applied.

The way to brake the rotor is simple. At the beginning of the process, the bottom transistors of all three phases are opened at 10 %. If the rotor is moving, a small current excited by the generated back-EMF starts to flow. This current can be measured by shunt resistors and detected by the MCU. When the current starts flowing, the motor starts to generate braking torque which lowers the speed of the rotation. The energy of the moving rotor is dissipated as heat, mainly in the stator winding. If the current is higher than the pre-defined threshold (10 % of nominal value in this application) then the MCU waits until the current decreases. If the current decreases to below the threshold, the PWM duty cycle will be increased until the current again reaches the threshold. In this way, the motor is gently braked to a standstill, when the bottom transistors are gradually opened to 100 %. The flowchart in Figure 12 represents the braking process.

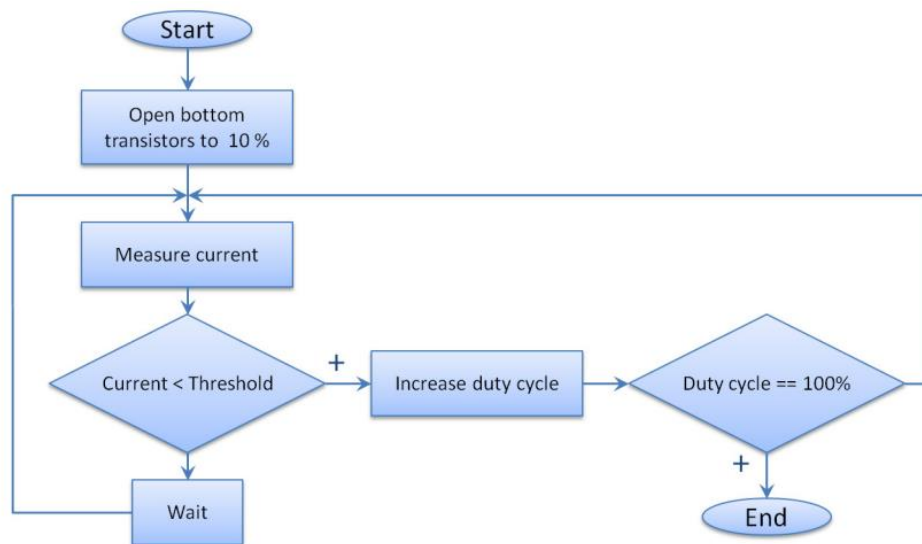


Figure 12. Braking process

The robustness of the system can be further enhanced. A timer can be implemented to limit the braking process, for example in the case when there is still considerable current detected even at 100 % duty cycle. The system can then issue a fault flag and the start-up of the motor will be disabled.

4.2. Application state machine

The application state machine differs in a few details from the original state machine described in *Sensorless PMSM Field-Oriented Control* (document [DRM148](#)). The Run state is enhanced to include one additional sub-states, Brake state, which enable the anti-wind feature. The Ready sub-state was also

modified. In the original application, during the Ready sub-state all the PWM outputs were enabled and set to 50 %. When the motor is running, this causes a current surge with potential system failure.

Figure 13 represents the Run state with sub-states and transitioning conditions. The Run sub-states are called when the state machine is in the Run state. The Run sub-state functions are as follows:

- Ready:
 - The bottom transistors are open to 10 % while the top transistors remain off
 - The PWM output remains disabled
 - The phase currents and DC-bus voltage are measured
 - The ADC channels are set up
 - Certain variables are initialized
- Brake
 - The PWM output is enabled, if the motor is rotating, it is braked
 - The phase currents are measured
 - The duty cycle of bottom transistors is increased up to 100 %
- Calib:
 - The duty cycle is set to 50 % and top and bottom transistors are enabled
 - The current channels offset is measured and filtered
- Align:
 - The currents are measured
 - The ADC channels are set up
 - The rotor alignment algorithm is called
 - The PWM is updated
 - The DC-bus voltage is measured
 - After the alignment time expiration, the system is switched to Startup
- Startup:
 - The currents are measured
 - The ADC channels are set up
 - The Back-EMF observer algorithm is called to estimate the speed and position
 - The estimated speed is filtered
 - The FOC algorithm is called
 - The PWM is updated
 - The DC-bus voltage is measured and filtered
 - The open-loop start-up algorithm is called
- Spin:
 - The currents are measured
 - The ADC channels are set up
 - The Back-EMF observer algorithm is called to estimate the speed and position
 - The estimated speed is filtered

- The FOC algorithm is called
- The PWM is updated
- The motor spins
- The DC-bus voltage is measured
- The speed ramp and the speed PI controller algorithms are called
- The speed command is evaluated
- Freewheel:
 - The PWM output is disabled, and the modulo is set to 50 %
 - The current is measured
 - The ADC channels are set up
 - The DC-bus voltage is measured
 - The system waits in this sub-state for a certain period of time which is determined by the rotor
 - Inertia: that is, it waits until the rotor stops
 - The system evaluates the conditions and proceeds into either the Align or Ready sub-state

The Run sub-states also have functions that are called between sub-state transitions. The sub-state transition functions are as follows:

- Ready > Brake—non-zero speed command; entering the Brake state
 - Top transistors are enabled
- Brake > Calib
 - Top and bottom transistors are enabled
- Calib > Align—entering the Align state
 - Certain variables are initialized (voltage, speed, position)
 - The alignment time is set up
- Align > Ready—zero speed command
 - Entering the Ready state
- Align > Startup—alignment done
 - The filters and control variables are initialized
 - Entering the Startup state
- Startup > Spin—entering the Spin state
- Startup > Freewheel—no action occurs. Can be used to handle the start-up fail condition for a
 - more robust application
- Spin > Freewheel—zero speed command
 - PWM output is disabled
 - Entering the Freewheel state
- Freewheel > Ready—Freewheel time expired
 - The PWM output is enabled
 - Entering the Ready state

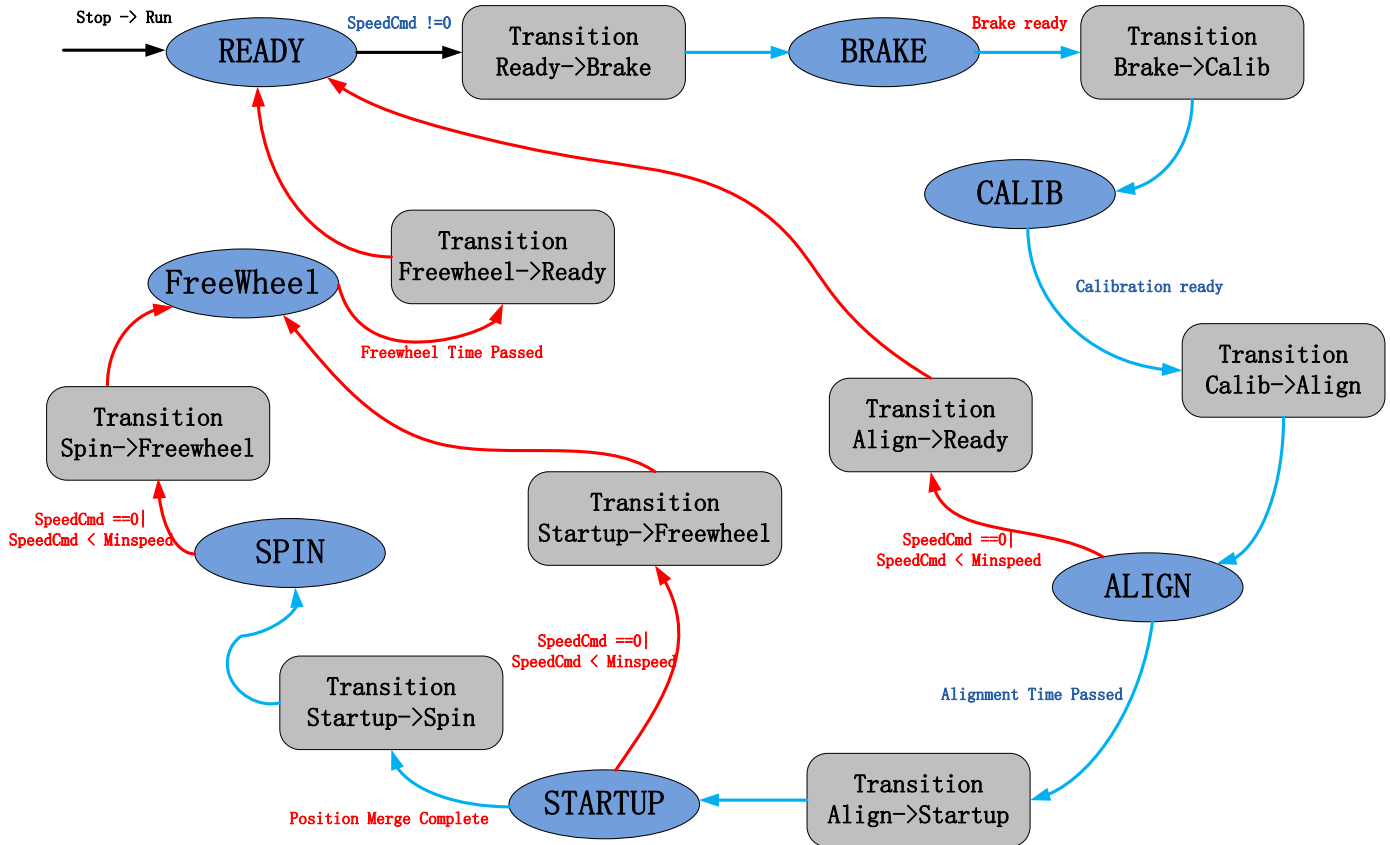


Figure 13. Application flow in Run state

5. Hardware setup

The **HVP-MC3PH** (High Voltage Motor Control Platform) development system is used as the hardware platform for the PMSM sensorless on Kinetis KE02. It consists of the following modules:

- High Voltage Motor Control Platform (HVP-MC3PH)
- Kinetis MKE02Z64 daughter card

All modules of the system are available for order through nxp.com or from distributors, so the user can build the hardware platform for which the application is targeted.

5.1. Hardware setup and configuration

Building the system using the modules of the system is straightforward. The peripheral modules and the MCU module are in the daughter board, which is plugged into the HVP slot connectors, then communicates with the PC through a mini USB connector, see [Figure 14](#):

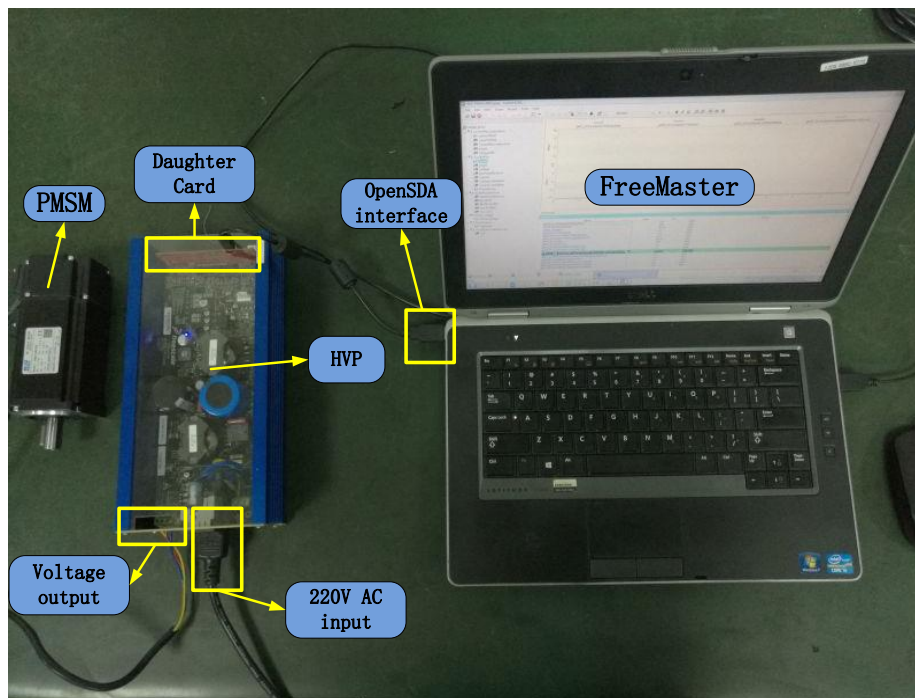


Figure 14. Application concept

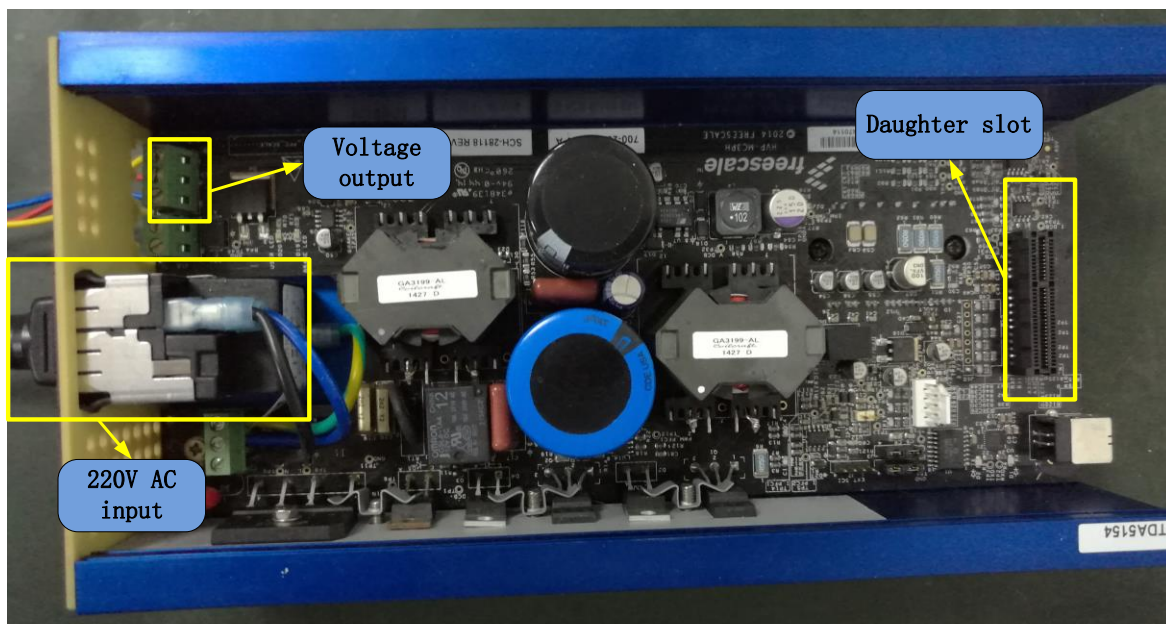


Figure 15. HVP platform

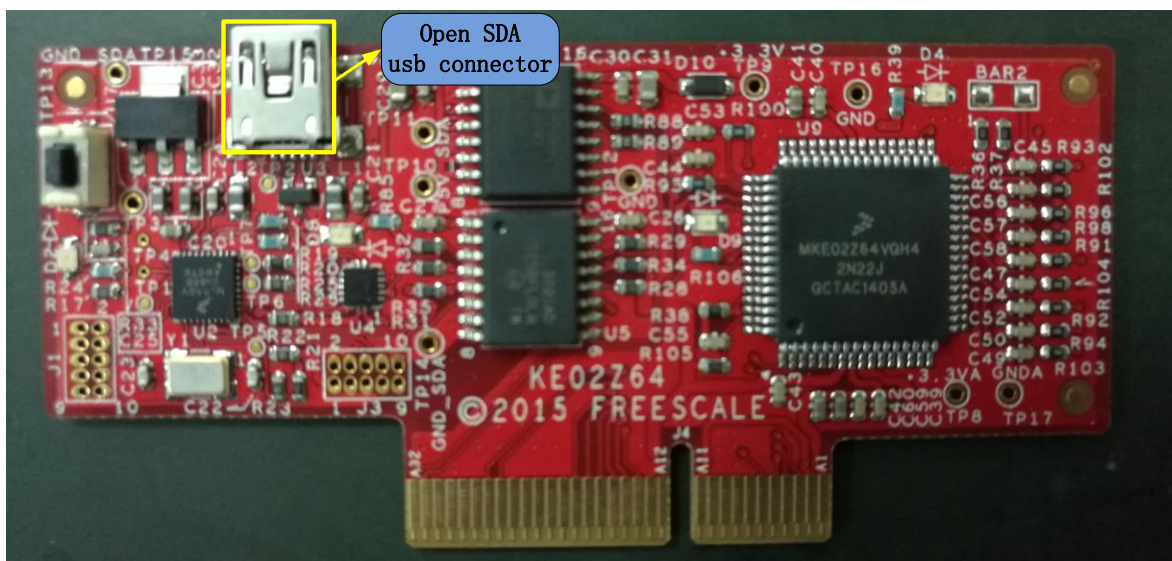


Figure 16. MKE02Z64 daughter card

6. Application operation

The application can be operated through the FreeMASTER software, which also enables visualizing the application variables. The FreeMASTER application consists of two parts, the PC application used for variables visualization, and the set of software drivers running in the embedded application. The data between the PC and the embedded application is passed through the RS232 interface.

6.1. FreeMASTER installation on a PC or notebook

The FreeMASTER PC application can be downloaded from the NXP webpage: nxp.com/freemaster. From the Download tab select FreeMASTER 2.0 Application Installation. To run the installation, click the Run button. Follow the instructions on the screen to complete the installation process.

6.2. Establishing connection between PC and embedded application

The FreeMASTER enables the use of multiple communication interfaces between the embedded application and the PC or notebook (UART (RS232), CAN, Ethernet, USB, BDM, and so on). For this application the RS232 was used because the software overhead for the data transfer represents the lowest additional load on the CPU. Currently, notebooks are not equipped with a COM port, so for this purpose the Kinetis KE02 daughter card has in place a USB-to-RS232 interface (OpenSDA - CDC Serial Port). By connecting the Kinetis KE02 module with a notebook through the USB cable, a virtual serial port will be established in the Windows system.

6.3. Application operation using FreeMASTER

To run the FreeMASTER application double click on the SNS_PMSM_KE02.pmp file located in the local project folder. The FreeMASTER application starts and the environment will be automatically created, as it is defined in the *.pmp file.

6.3.1. Setting up the communication

When the notebook is connected through the USB cable with the Kinetis KE02 daughter card, the operating system assigns the number of the COM port to the OpenSDA - CDC Serial Port. This number is assigned randomly, therefore it is necessary to set the right communication port each time the connection is established (re-plugging the USB cable might cause a different port number assignment). To set the port number, click the menu item Project \ Options ... then assign the port number in the Comm tab of the opened window. The correct port number selection is confirmed by the text OpenSDA – CDC Serial Port next to the list box with available serial port numbers.

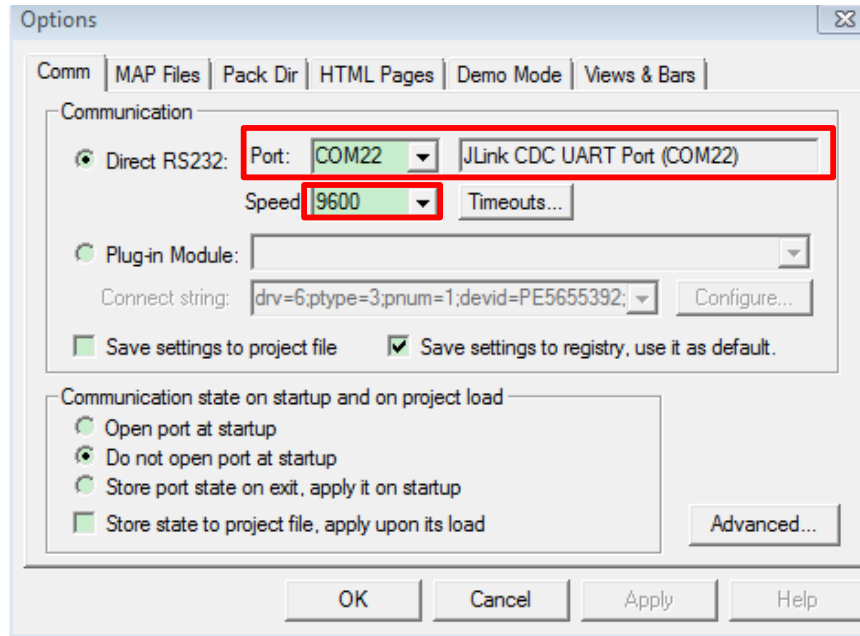


Figure 17. FreeMASTER communication settings

6.3.2. Start/stop the communication

The next step is to switch the application to the RUN state. In the FreeMASTER window in the variable watch grid click on the drop-down list next to the application switch variable name, and select ON, as shown in Figure 18:

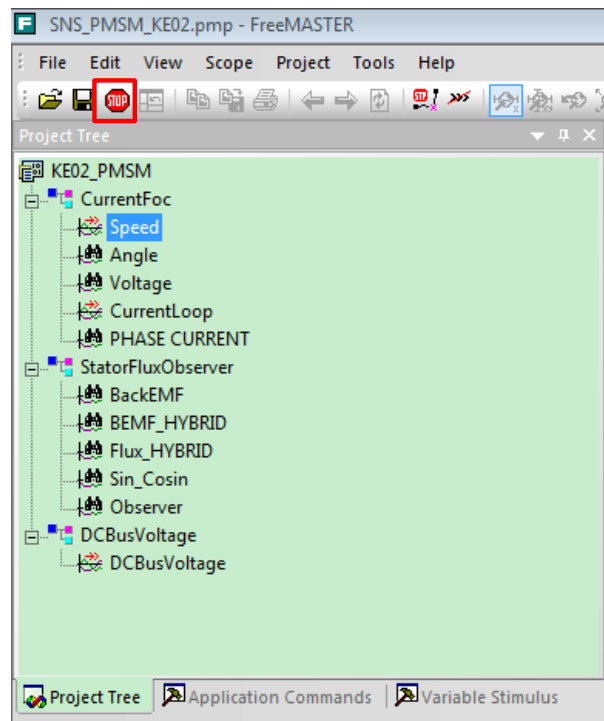


Figure 18. Initiating communication with the embedded side

6.3.3. Start/stop the application, required speed setting

The next step is to switch the application to the RUN state. In the FreeMASTER window in the Variable Watch grid click on the value textbox next to the Application Switch variable name, and enter “1” as shown in [Figure 19](#):

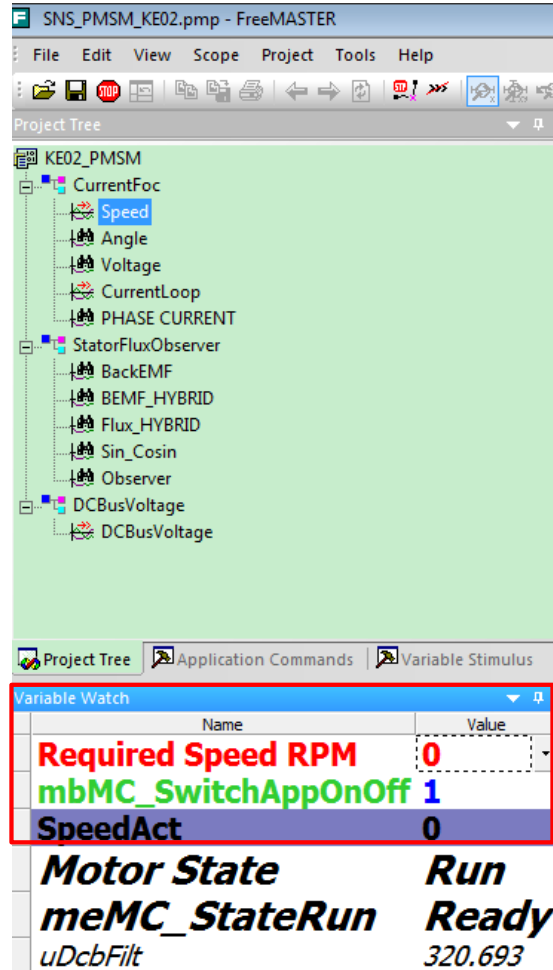


Figure 19. Start the application

After the application is set to the Run state, the required speed can now be changed to some non-zero value. The procedure is similar to the previous step. In the Variable Watch grid enter a positive or negative value next to the Speed Required variable name.

6.3.4. Operation of the application from the control page

The application can also be operated from the control page, see the description in [Figure 20](#).

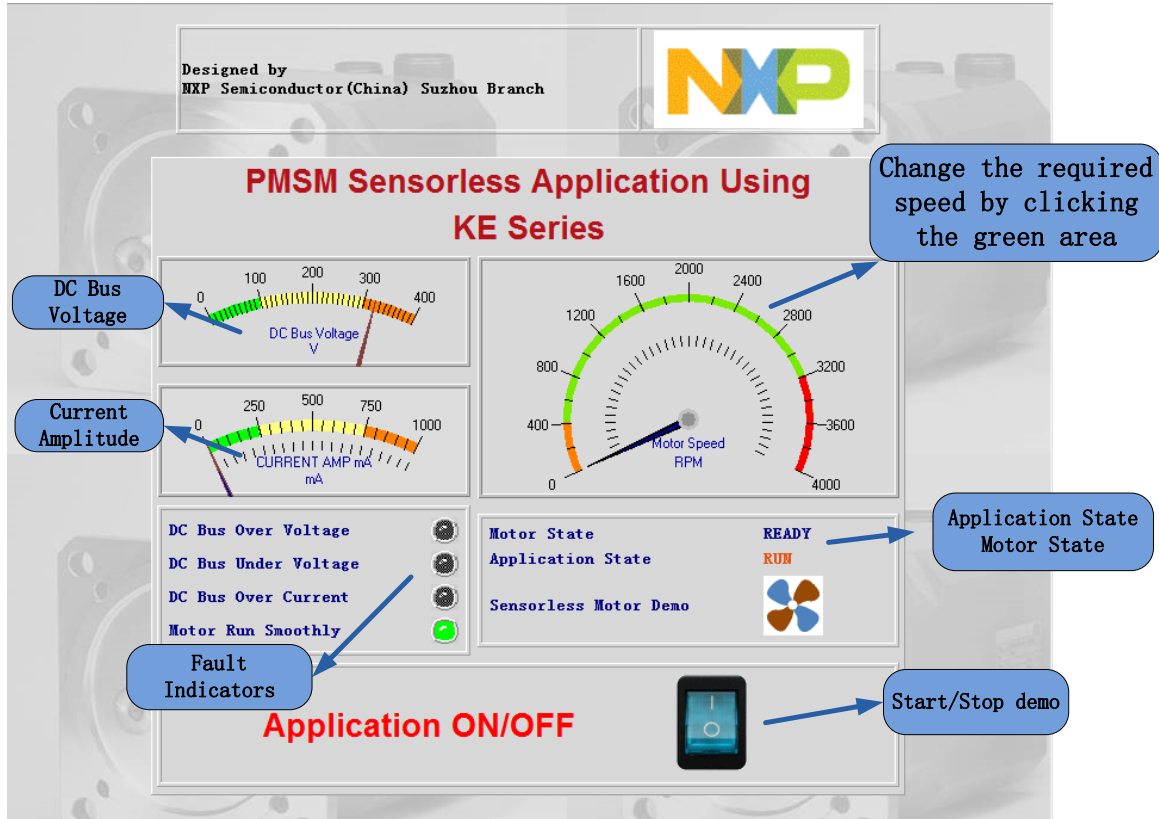


Figure 20. PMSM sensorless vector control page

Click the “Application ON/OFF” button will set the application state to run state, then 1000 rpm required speed will be set. The demo motor will run to 1000 rpm later. Then the required speed can be changed by clicking the green area of the tachometer. Click this button again to stop the demo.

NOTE

There should be approximately 3 seconds’ delay before starting the demo again to take time transition from Freewheel to ready.

6.3.5. Scopes and Recorders

One of the main benefits of the FreeMASTER application is to visualize the values of the variables in real time. For this purpose, there are two possibilities. The user can select between the Scope and the Recorder. While the Scope feature downloads a stream of the data continuously in real time, the Recorder stores the data in a buffer located in the RAM of the embedded MCU and, after a trigger condition is met, the data is transferred in blocks through the communication interface and visualized in the FreeMASTER window. The sampling period of the Scope is limited by the speed of the communication interface, and therefore is used to slowly change quantities such as the speed. The sampling period of the oscilloscope-like Recorder is in the microseconds range to enable visualization of quickly changing quantities, such as the phase currents or the duty cycles. In this application, the recorder buffer is updated each time the fast control loop is executed, that is every 200 us.

Figure 21 and Figure 22 show the speed tracer performance (scope) and the voltage duty cycle (recorder).

Application operation

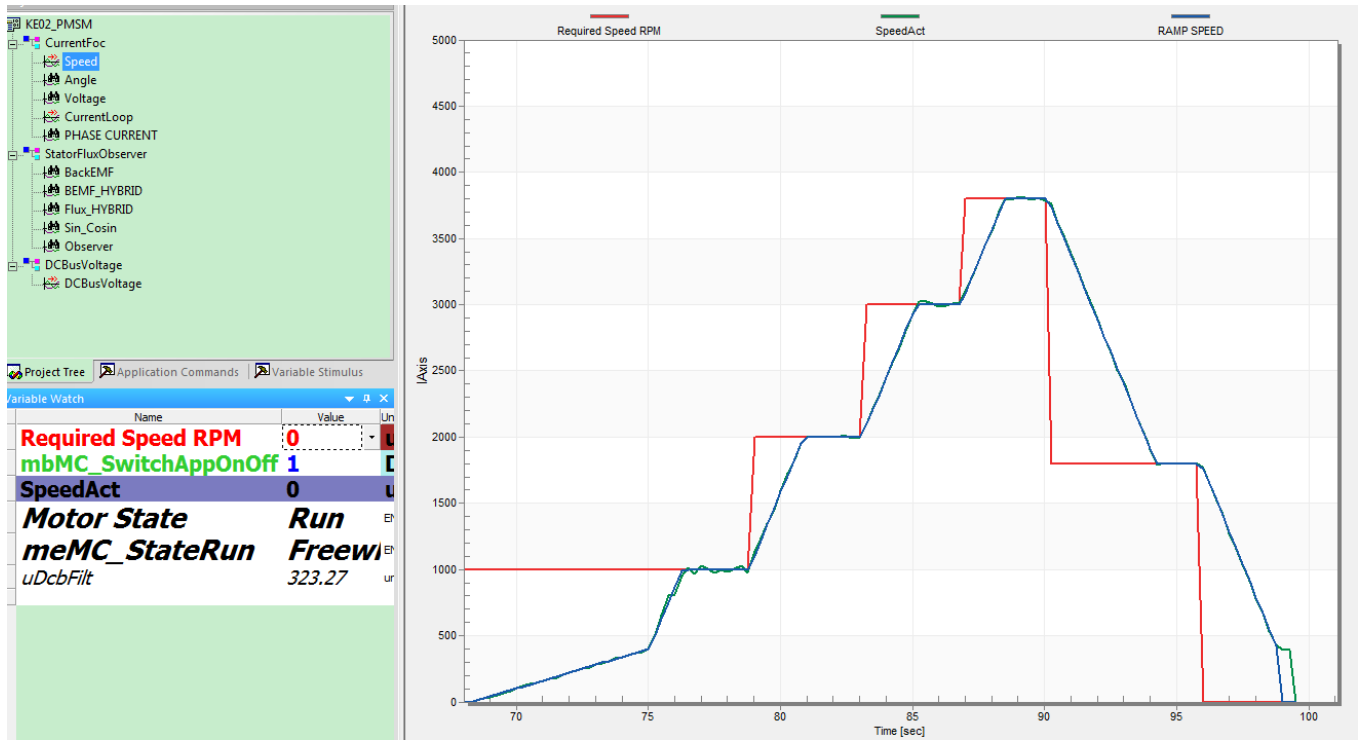


Figure 21. PMSM speed tracer performance in Scope

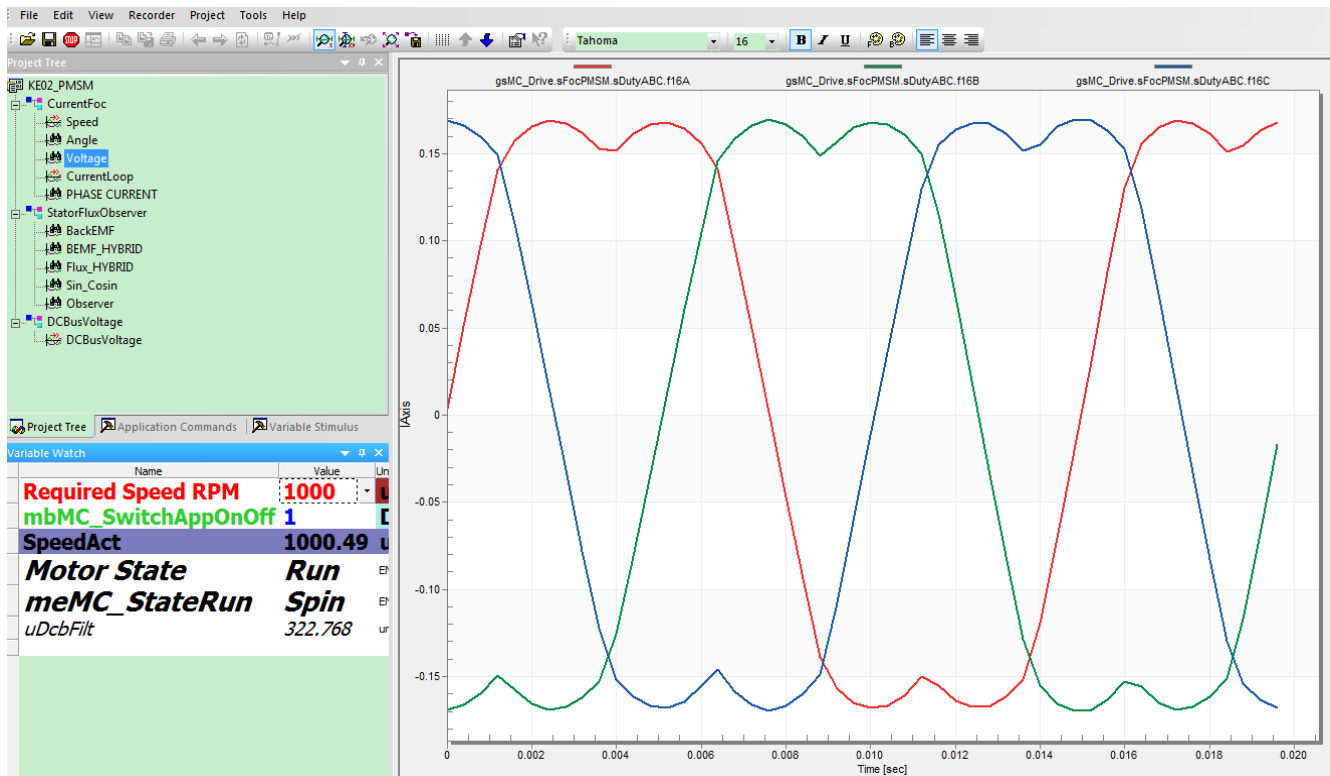


Figure 22. Voltage duty cycle in Recorder

7. References

- *KE02 Sub-Family Reference Manual* (document [MKE02P64M40SF0RM](#))
- *High Voltage Motor Control Platform Schematic* (document [HVP-MC3PH](#))
- *Kinetis MKE02Z64 daughter card Schematic*
- *Sensorless PMSM Field-Oriented Control*, (document [DRM148](#))
- *Using FlexTimer in ACIM/PMSM Motor Control Applications* (document [AN3729](#)).

NXP documents are available online at nxp.com.

8. Revision History

Table 3. Revision history

Revision number	Date	Substantive changes
0	06/2016	Initial release
1	05/2017	Chapter 2.3 updated

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:

nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale, the Freescale logo, and Kinetis are trademarks of NXP B.V. All other product or service names are the property of their respective owners.

ARM, the ARM powered logo, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2016-2017 NXP B.V.

Document Number: AN5294

Rev. 1

05/2017

