Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 002-05150

Spec Title: AN205150 - FR MB91265 Microcontroller BLDC Drive with Multi-Function Timer

Replaced by: NONE

AN205150

FR MB91265 Microcontroller BLDC Drive with Multi-Function Timer

This application note will give a short introduction to the principles of the BLDC (Brushless DC) and PMSM (Permanent Magnet Synchronous) motor using Hall sensors and how it can be controlled using the Cypress MB91F267 MCU.

Contents

1 Introduction 1
2 Features 2
2.1 Feature summary 2
2.2 Block diagram of the MFT 3
3 Registers explanation 4
3.1 Registers of the 16-bit Free-Running Timers (FRT0-2) 4
3.2 Registers of the 16-bit Input Capture (IC0-3) 5
3.3 Registers of the 16-bit Output Compare 5
3.4 Registers of the Waveform Generator 6
4 Application Example 8
4.1 BLDC basics 8
4.2 BLDC with block commutation 9
4.3 BLDC with sinusoidal commutation 12
A Appendix 16
A.1 Related Documents 16
A.2 Related software examples 16
Document History 17
Worldwide Sales and Design Support 18
Products 18
PSoC® Solutions 18
Cypress Developer Community 18
Technical Support 18

1 Introduction

Brushless motors, even though known for several decades especially in industrial drives (AC induction, AC synchronous motors), were not commonly used in many applications because of some motor specific requirements. The electronic commutation or even variable frequency sine wave generation was quite costly and complicated using discrete electronics, and high-frequency, high-power switching devices were rare and expensive, as well as computational power. Therefore, these motors were often used in big industrial drives with electromechanical methods (e.g. multi-tap transformers, selectable stator windings, inserting of resistors etc.) of speed switching or control. For small and / or variable speed applications or when three-phase power was not available, brushed DC or universal motors were used. Since computational power of single-chip microcontrollers and DSPs steadily increased, it has reached a level where highly sophisticated control methods can be implemented on a device costing only a few euros. Also power electronics like MOSFETs capable of far more than 100A and channel resistances of only some mΩ are available at prices beneath one Euro, with falling tendency. This also relies to integrated IGBT modules as well as other advanced power electronics. All this together started opening new application fields to brushless three-phase drives, which then started replacing their brushed ancestors. The higher reliability and controllability made electronically controlled brushless three-phase drives popular in applications from some watt up to several MW, from RC cars or small blowers over servo drives of all sizes up to big locomotives or industrial applications.

This application note will give a short introduction to the principles of the BLDC (Brushless DC) and PMSM (Permanent Magnet Synchronous) motor using Hall sensors and how it can be controlled using the Cypress MB91F267 MCU.

OBSOLETE

2 Features

2.1 Feature summary

The MB91F267 (N) is part of Cypress's MB91265 FR60Lite 32-bit RISC MCU series. It offers a maximum operating frequency of 33 MHz, using the internal PLL (x8) and a 4.192 MHz quartz crystal oscillator. The chip has 128kB of internal Dual-Operation Flash, as well as 4kB of internal SRAM. It has numerous features making it especially interesting for motor control as well as general control applications:

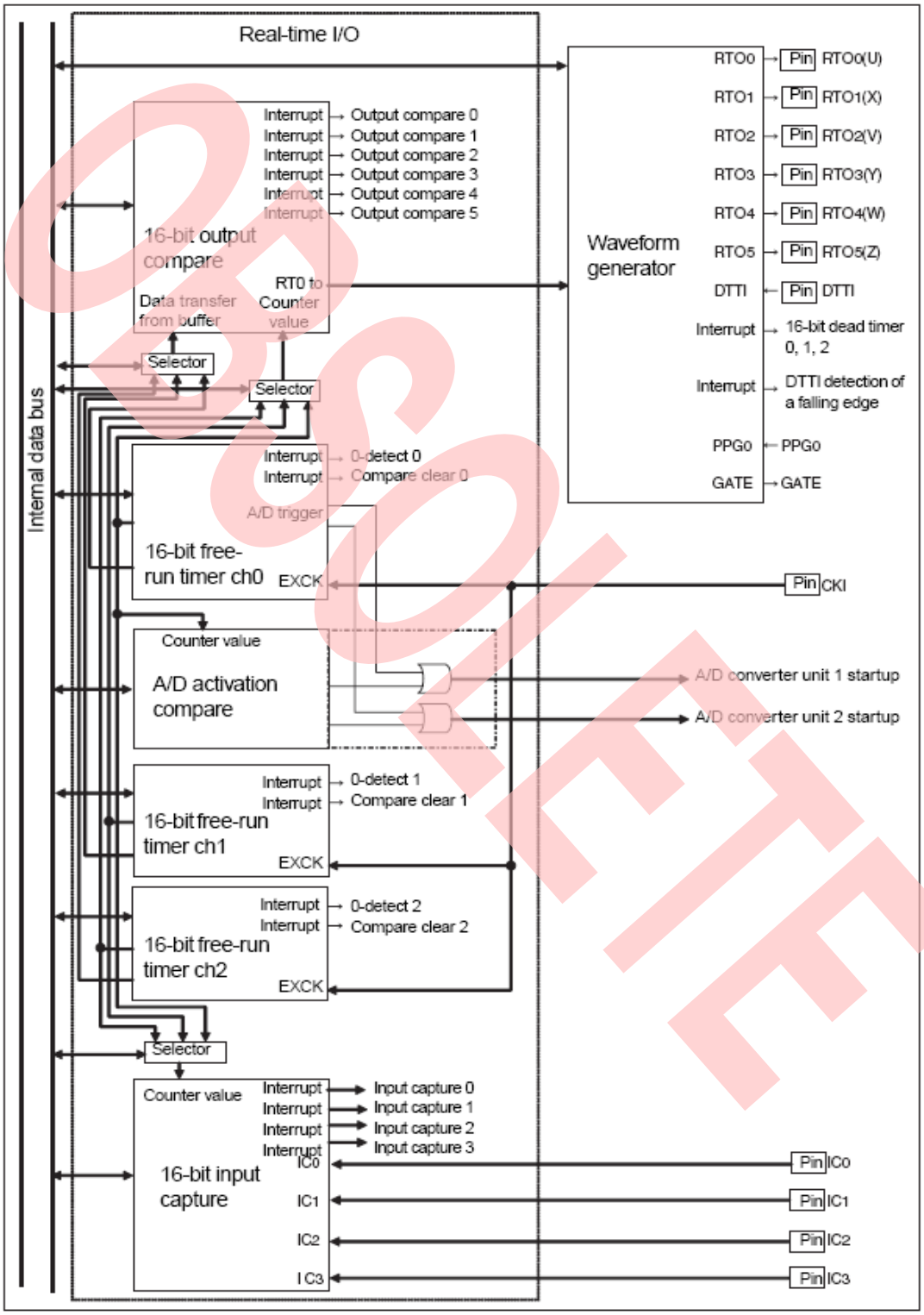
- MAC unit (µDSP): 16-bit x 16-bit + 40-bit single cycle MAC unit parallel to MCU
- 3-cycle (16-bit signed) / 5-cycle (32-bit signed) hardware multiplier
- 16 interrupt priority levels
- 8 external interrupt pins + NMI
- 2x UART with synchronous / asynchronous modes and special baud rate generator
- 1x CAN (MB91F267N only)
- Very fast 8/10-bit AD-converter: 1.2 µs minimum conversion time, two independent units (1x4ch, 1x 7ch)
- 16-bit Pulse-width counter (PWC)
- Three 16-bit reload timers (RLT0-2)
- Multi-function timer with Waveform Generator:
  - Three 16-bit free-running timers (FRT0-2), up- or up/down counting for center-/edge-aligned PWM generation and separate, buffered compare clear registers
  - Six output compare units with buffered compare registers, register update on FRT zero or compare clear match
  - Four input capture units (rising, falling or both edges)
  - Several possibilities to trigger the two ADC units (FRT0 zero, FRT0 compare clear match, separate trigger compare value)
  - Waveform generator with 16-bit dead time timer (3ch) and fault input (DTTI)
  - OCU and ICU channels freely assignable to FRT0-2
  - 8x 8-bit or 4x 16-bit PPG
  - Numerous interrupt sources (19 in total):
    - Zero detect and compare clear match for each FRT channel
    - Output compare match for each OCU channel
    - Input capture event (IC0, IC1, IC2+3)
    - 16-bit dead timer underflow (3ch)
    - DTTI falling edge (fault input)

Even though the Waveform Generator can generate a variety of different output signals together with other blocks of the Multi-Function timer, this application note will focus on those parts relevant for basic three-phase motor control applications. This mainly includes center or edge aligned PWM generation using the output compare units in independent or complementary mode. Please refer to the HWM for further configurations.

OBSOLETE

2.2 Block diagram of the MFT

Figure 1. Block Diagram of the Multi-Function Timer



OBSOLETE

3 Registers explanation

Most of the registers described here consist of a high- and a low byte register, e.g. CPCLR0 consists of CPCLRH0 and CPCLRL0. Unless otherwise noted, only the 16-bit register name is used here. Please refer to the MB91265 Series Hardware Manual for details on register access modes.

3.1 Registers of the 16-bit Free-Running Timers (FRT0-2)

3.1.1 Timer data register (TCDT0-2)

The Timer Data Registers hold the actual count value of each FRT channel.

3.1.2 Compare clear and compare clear buffer registers (CPCLR0-2, CPCLRB0-2)

The compare clear registers set the value, up to which the free-run timers count. After the compare clear value is reached the counter is either cleared and restarts counting (up-count mode, MODE=0) or counts back down to zero (up/down count mode), depending on the selected mode (TCCSL0-2 bit 5 (MODE)).

The compare clear registers can be used to raise the obtainable PWM base frequency for a given count clock by reducing the amplitude resolution.

The CPCLR and the corresponding buffer registers are located at the same address. Writing affects the buffer register, while reading shows the value of the compare clear register. By this, always the actually used compare clear value is read. If the buffer function is disabled (TCCSL0-2 bit 7, BFE=0) or the FRT is stopped, the write value is immediately transferred to the compare clear register. If the function is enabled, the value in the compare clear buffer register is transferred to the compare clear register the next time the FRT reaches zero.

3.1.3 Timer status control register (TCCSH0-2, TCCSL0-2)

TCCSH0-2:

Bit15 (ECKE): Clock select bit. For PWM generation, normally the internal clock will be selected (ECKE=0)

Bit14 (IRQZF): FRT zero-detect interrupt flag

Bit13 (IRQZE): Zero detect interrupt enable

The corresponding zero-detect interrupt service routine can be used e.g. to update the duty cycle values and transfer them to the compare buffer registers.

Bit12-10 (MSI2-0): These bits can be used to mask a specified amount of zero-detect or compare clear interrupts, so that only every n-th compare clear or zero-detect event (n=1-8) causes an interrupt. See HWM for details.

Bit9 (ICLR): Compare clear interrupt flag

Bit8 (ICRE): Compare clear interrupt enable

TCCSL0-2:

Bit7 (BFE): Compare clear buffer enable (see above)

Bit6 (STOP): When this bit is set, the FRT stops counting. Clearing it starts the FRT.

Bit5 (MODE): Selects up-count mode (MODE=0) or up/down count mode (MODE=1). For symmetric (center-aligned) PWM, the up/down count mode is used, which offers the best switching distribution for most applications, but the resulting base frequency is only half as high as with edge-aligned mode.

Bit4 (SCLR): Setting this bit clears the FRT to “0000” at the next count clock (but does not generate a zero-detect interrupt!)

Bit3-0 (CLK): Selects the FRT count clock prescaler (1-256)

3.1.4 A/D trigger control register (ADTRGC)

Bit3, Bit2 (SEL1, SEL2): Selects whether the ADC unit (1 or 2) is triggered by a compare match or at FRT zero detect.

Bit1, Bit0 (AD2E, AD1E): These bits activate the ADC trigger by FRT0.

Additional possibilities to trigger the ADC by FRT0 using a dedicated compare value exist with the ADC activation compare function, as described later in this document.

OBSOLETE

3.2 Registers of the 16-bit Input Capture (IC0-3)

3.2.1 16-bit input capture data register (IPCP0-3)

These registers hold the count value of the selected FRT channel at the last input capture event.

3.2.2 Input capture status control register (ch2+3) (ICSH23, ICSL23)

ICSH23:

Bit9 (IEI3): If the last event on IC3 was a falling edge, '0' is read from this bit. A '1' indicates a rising edge.

Bit8 (IEI2): If the last event on IC2 was a falling edge, '0' is read from this bit. A '1' indicates a rising edge.

ICSL23:

Bit7 (ICP3): Input capture 3 interrupt flag

Bit6 (ICP2): Input capture 2 interrupt flag

Bit5 (ICE3): Input capture 3 interrupt enable

Bit4 (ICE2): Input capture 2 interrupt enable

Bit3, Bit2 (EG31, EG30): Edge selection for IC3 (01: rising, 10: falling, 11: both edges cause an input capture event)

Bit1, Bit0 (EG21, EG20): Edge selection for IC2 (as above)

Note:

The ICP2 bit is misleadingly named 'IP2' in the HWM as well as in older versions of the header files supplied with the software examples.

3.2.3 PPG output control / input capture status control Register (PICSH01, PICSL01)

PICSH01:

Bit15-Bit10 (PGEN5-PGEN0): PPG output enable; Enables or disables the output of PPG0 (depending on RT0-5) to RTO0-5.

Bit9 (IEI1): If the last event on IC1 was a falling edge, '0' is read from this bit. A '1' indicates a rising edge.

Bit8 (IEI0): If the last event on IC0 was a falling edge, '0' is read from this bit. A '1' indicates a rising edge.

PICSL01: Same bit functions as ICSL23, but for input capture 0 and 1.

3.3 Registers of the 16-bit Output Compare

3.3.1 16-bit output compare and output compare buffer registers (OCCP0-5, OCCPB0-5)

The output compare register holds the actual compare value for every output compare unit. If the value of the selected FRT matches the value in an OCCP register, an output compare event is generated on the corresponding channel.

The OCCP and the corresponding buffer registers (OCCPB) are located at the same address. Writing affects the buffer register, while reading shows the value of the compare register. By this, always the actually used compare value is read. If the buffer function is disabled (OCSL0,2,4 BUF1 and BUF0=0) or the FRT is stopped, the write value is immediately transferred to the compare register. If the function is enabled, the value in the compare buffer register is transferred to the compare register the next time the FRT reaches zero or the compare clear value, depending on the settings of the OCSH1,3,5, BTS1 and 0 bits. This function ensures that all duty cycles are updated synchronously and without glitches or other disturbances. Please refer to the HWM for further operation details.

3.3.2 Compare control register (OCSH1,3,5, OCSL0,2,4)

OCSH1,3,5:

Bit14, Bit13 (BTS1,0): Buffer transfer selection bit; These bits define whether the compare buffer registers are transferred to the compare register at zero detect (setting '0') or at compare clear (setting '1'). OCSH1_BTS0 applies to OC0, OCSH1_BTS1 applies to OC1, OCSH3_BTS0 applies to OC3 and so on.

Bit12 (CMOD): Output level reverse mode

OBSOLETE

This bit is very important for the correct function of the MFT. If the particular output channel is in toggle mode, the CMOD bit selects between 'normal' operation and '2-channel mode'. To generate a non-overlapping (complementary) output using the Waveform Generator, CMOD=1 has to be set for RTO1,3,5. If the channel is in set-reset mode, the CMOD bit inverts the polarity of the RTO signal.

Bit11, Bit10 (OTE1,0): Output enable; This bit switches the RTO pins between I/O-port function (when set to '0') or waveform output (setting '1'). The corresponding port data register should be preloaded with a 'safe stop' value (mostly so that all power transistors are turned off), so that no damage can occur when the pins are switched to I/O-port by accident or the DTTI function.

Bit9, Bit8 (OTD1,0): These bits show the actual value of every Waveform generator output (RTO) signal. If the compare operation is stopped, these bits also define the default level for every RTO signal. For complementary, non-overlapping PWM output using the Waveform Generator, set the OTD1 bits to '0' and the OTD0 bits to '1'.

OCSL0,2,4:

Bit7 (IOP1): Compare match interrupt flag for OC1 (OCSL0_IOP1), 3 (OCSL2_IOP1) and 5 (OCSL4_IOP1)

Bit6 (IOP0): Compare match interrupt flag for OC0 (OCSL0_IOP0), 2 (OCSL2_IOP0) and 4 (OCSL4_IOP0)

Bit5 (IOE1): Compare match interrupt enable for OC1,3,5

Bit4 (IOE0): Compare match interrupt enable for OC0,2,4

Bit3 (BUF1): Compare buffer disable bit for OC1,3,5 (0=Buffer enabled, 1=disabled)

Bit2 (BUF0): Compare buffer disable bit for OC0,2,4 (0=Buffer enabled, 1=disabled)

Bit1 (CST1): Compare operation enable bit for OC1,3,5

Bit0 (CST0): Compare operation enable bit for OC0,2,4

3.4 Registers of the Waveform Generator

3.4.1 16-bit dead timer register (TMRR0-2)

The dead timer registers hold the reload values for the three 16-bit dead timers. These registers are used to set the non-overlapping time ('dead time') in complementary, non-overlapping PWM mode. For other functions, please refer to the HWM.

3.4.2 16-bit dead timer control register (DTCR0-2)

This 8-bit register controls the dead timer and Waveform Generator operation. The 16-bit bit numbers result from the location in memory, nevertheless these are 8-bit registers.

Bit15 or 7 (DMOD0/1/2): Output polarity control; Setting this bit reverses the output polarity for every pair of outputs (RT0/1, RT2/3, RT4/5)

Bit14 or 6 (GTEN1/3/5): RT1/3/5 controls the GATE signal if this bit is set

Bit13 or 5 (GTEN0/2/4): RT0/2/4 controls the GATE signal if this bit is set

Bit12 or 4 (TMIF0/1/2): 16-bit dead timer interrupt flag; this flag is set when an underflow occurs in the 16-bit dead timer (i.e. at the end of the non-overlapping time)

Bit11 or 3 (TMIE0/1/2): Interrupt enable bit for the 16-bit dead timer

Bit10-8 or 2-0 (TMD2-0, TMD5-3, TMD8-6): Timer operation mode

Each triplet of TMDx bits in each DTCR register configures the behavior of one 16-bit dead timer channel. The settings used in the examples related to this document are '000' to disable the waveform generator (e.g. for block commutation) or '100' for dead time timer mode (complementary PWM). Please refer to the HWM for further operation modes.

OBSOLETE

3.4.3 Waveform control registers (SIGCR1, SIGCR2)

These 8-registers control the operation of the 16-bit dead timers and the waveform generator.

Note: If the DTTI function is active, the Waveform Generator disables the RTO outputs by setting them to I/O-port mode when the DTTI input is externally pulled low for more than the time defined by the NWS and DCK bits (if the noise cancellation function is enabled). If the noise cancellation function is disabled (NRSL=0), the outputs are switched to I/O-port immediately. The corresponding port data register (PDR3 for the MB91F267) should be pre-loaded with the desired safe value (e.g. all switches off) during initialization. Also, the corresponding port direction register should be set to 'output' for these pins, unless a Hi-Z condition is desired in case of DTTI shutdown.

SIGCR1:

Bit7 (DTIE): DTTI input enable; if this bit is set, the DTTI (fault) input is enabled.

Bit6 (DTIF): DTTI interrupt flag; is set when a DTTI fault condition occurs.

Bit5 (NRSL): Noise cancellation function enable; If this bit is set, the DTTI is only triggered if the DTTI input remains 'Low' for more than the number of clock cycles defined by NWS1/NWS0. This requires the presence of a system clock oscillation!

Bit4-2 (DCK2-0): These bits select the clock prescaler for the 16-bit dead timer.

Bit1-0 (NWS1-0): These bits configure the number of clock cycles for the noise cancellation function (n = 4*2^NWS1:0)

SIGCR2:

Bit0 (DTTI): This bit is OR-connected to the DTTI input pin. It can be used to trigger the DTTI function by software by writing '0' to this bit. Write '1' to the bit to clear DTTI after a software trigger.

3.4.4 A/D activation compare register (ADCOMP1, ADCOMP2)

This registers hold optional compare values to start the two ADC units by FRT0. When the value in the compare register matches the FRT0 counter and ADC trigger by this compare match is enabled in the ADCOMPCx registers, a conversion on the corresponding ADC unit is started.

3.4.5 AD activation control register (ADCOMPC1, ADCOMPC2)

These 8-bit registers configure the behavior of the ADC activation compare function.

ADCOMP1:

Bit2-1 (CE2,CE1): Activate start by A/D activation compare register match for ADC unit 2 (CE2 bit) and unit 1 (CE1 bit)

ADCOMP2:

Bit13-10 (SEL21,20,11,10): These bits define if every match shall trigger the ADC (SEL21,20 or 11,10 = '00'), or only those during up-counting ('01') or down-counting ('10').

3.4.6 Free-run timer selector register (FSR0-2)

These 8-bit registers are used to assign the input capture and output compare channels to the different FRT channels. Every IC/OC channel can be individually connected to one of the three free-run timer channels.

Note: Only FRT0 can trigger the ADC units by hardware, so it is a good choice to use FRT0 for PWM generation (by the output compare units) whenever the ADC needs to by synchronized to the PWM. This often is used to reduce disturbances to the ADC measurements (like phase or DC bus voltage and especially current) caused by PWM switching.

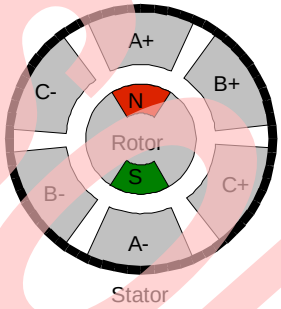
OBSOLETE

4 Application Example

4.1 BLDC basics

The basic construction of a brushless DC motor bears some analogy to a usual brushed DC motor turned inside out. Now that the moving rotor contains the permanent magnets and the stator carries the phase windings, the mechanical commutator is replaced by an electronic one. This saves space in the motor and eliminates the problems caused by the commutator's brushes, like brush wear, abrasion and arcs. The lifetime of a BLDC motor is basically only limited by the bearings. The BLDC offers a better torque-to-volume ratio than its brushed ancestor, because of the saved room usually needed for the commutator. Additionally, the BLDC can dissipate heat much better because of the lower thermal resistance of the windings to the housing. Also, the complete motor can be sealed (e.g. for pumps) or even run directly in liquids. The absence of arc sources makes it ideal for explosive environments. With modern microcontrollers and power electronics, the electronic commutation of BLDC motors is no longer a problem. Most BLDC motors are equipped with three Hall sensors for rotor position feedback, displaced 120 el. degrees to each other, so that the combination of the Hall sensor signals yields the rotor position within 60° el. For more precise position measurement, e.g. optical encoders can be used.

Figure 2. Three-Phase BLDC Motor



As the motor rotates, the permanent magnets induce a voltage in the motor phase windings, the so called back-EMF (Electro-Magnetic Force). In many BLDC motors, the back-EMF has a nearly trapezoidal shape as a result of their optimization for six-step commutation, while a 'pure' synchronous motor has a nearly sine-wave back-EMF shape. These motors preferably are driven by a sine wave inverter.

The back-EMF waveform can also be used to measure the rotor position in order to save additional sensors. The simplest and most popular method called back-EMF zero-crossing detection uses block commutation as described below, because current only flows through two phases at a time, while the third one is floating and can be used for back-EMF detection. A detailed example for a sensorless BLDC drive using this method can be found in further application notes.

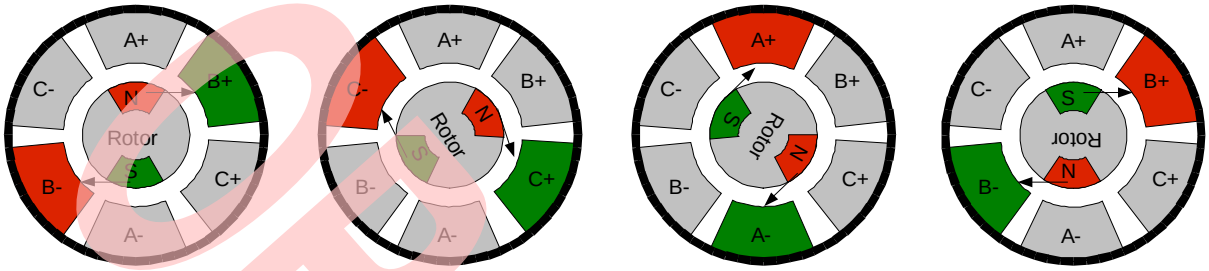
Most BLDC motors are three-phase motors, but also two-phase versions exist. These mainly are used for small pumps or blowers, e.g. in personal computers. Because of its close relationship to other permanent magnet motors, the BLDC is also known by other names, such as *Electronically Commutated* (EC) motor, PMSM (*Permanent Magnet Synchronous Motor*) or *Brushless AC* motor. Often, 'EC' or 'BLDC motor' means the block-commutated variant with trapezoidal back-EMF voltages, because its behaviour is very similar to a brushed DC motor. PMSM or PMAC mostly refer to a motor with sinusoidal back-EMF which is driven by sine-modulated currents. This results in nearly no torque ripple and a very high performance capability.

OBSOLETE

4.2 BLDC with block commutation

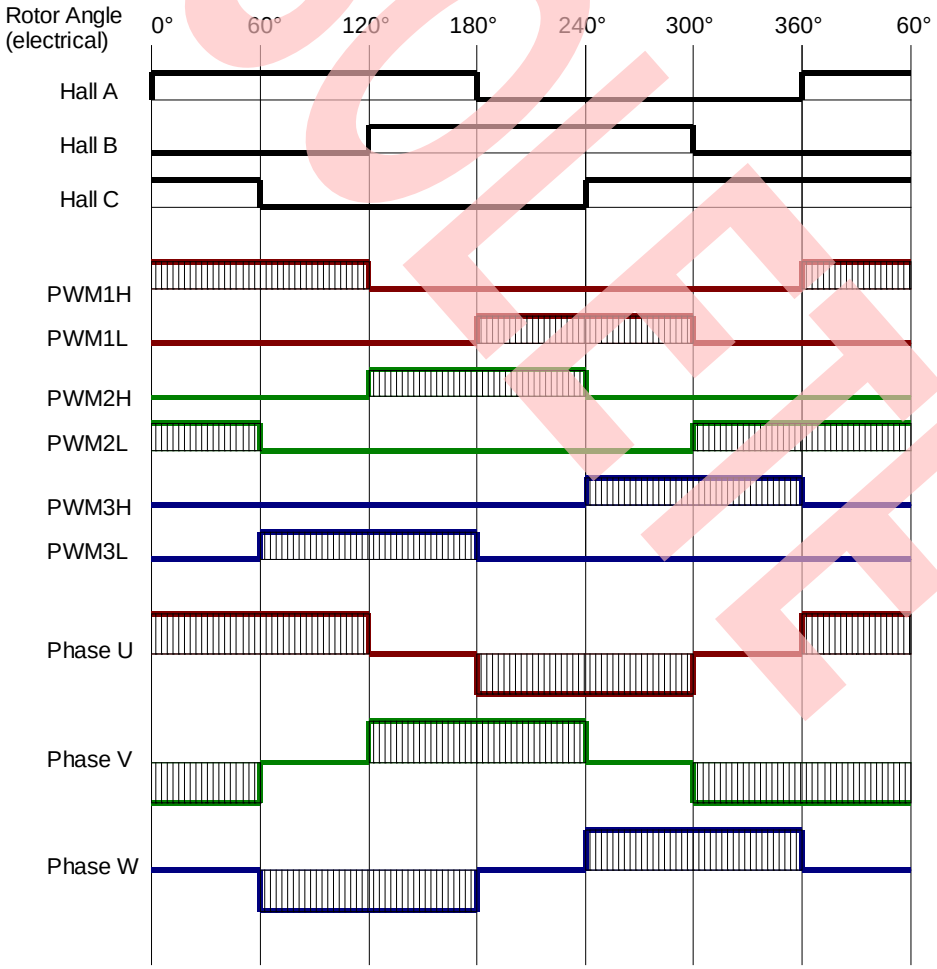
The easiest way to drive a BLDC motor is called block commutation (also known as six-step or trapezoidal drive). It is easy to implement on many microcontrollers, especially when the motor has Hall sensors. In this case, every slope on a Hall sensor signal triggers the next commutation. A part of this sequence is shown below:

Figure 3. Rotating BLDC Motor



Block commutation is often used in simple applications with moderate demands on dynamic behavior and acceptable torque ripple / noise. However, it is possible to optimize the motor construction for trapezoidal drive, reducing the noise and torque ripple.

Figure 4. Block Commutation Pattern



OBSOLETE

Setting up a three-phase PWM for block commutation is quite simple and does not necessarily need special waveform generator functions. In this example, it is performed by six independent output compare channels. An example configuration is shown below (center-aligned PWM with PWM_MAX_VAL steps resolution). Center-aligned PWM has the advantage of better distribution of switching pulses over time, since channels with different compare values are always switched on and off at different times. In edge aligned PWM, either all channels are switched on at the same time and switched off according to the duty cycle ('left aligned'), or the turn-on time varies with the duty cycle and all channels are switched off simultaneously ('right aligned'). The MB91F267 allows the usage of both edge- and center-aligned PWM with up to 16-bit resolution. For small drives, the compare clear register can be set lower, moving the PWM fundamental frequency completely out of the hearing range. A compare clear value of 1023 gives a PWM fundamental of more than 15kHz for center-aligned PWM with a 32MHz system clock, still keeping 10-bit amplitude resolution for precise control.

In the configuration shown below, the MOD1x bits of the OCMOD registers are all set to 1, as well as the CMOD bit in the OCSH1,3,5 registers. This causes the PWM output to switch to 'low' at a compare match during up-counting of the FRT, while a match during down-counting switches the output to high. With this configuration, the resulting output duty cycle is directly proportional to the corresponding output compare value, because the output is low when the compare value is below the FRT count value and high when it is above.

The position feedback in this example is achieved by three Hall sensors connected to the Input Capture pins. In the MB91265 Series (as in most other Cypress MCUs) the IC pins have two main advantages over e.g. External Interrupt pins for this application: They can be configured to trigger on both rising and falling edges using the EG10, 11, 20, 21 bits in the PICSL01 register and the EG21, 20, 31, 30 bits in the ICSL23 register. Additionally, the Input Capture function directly allows easy speed measurement using a FRT channel.

Besides the timer and ICU configuration registers (PICSL01, ICSL23, TCCSHn, TCCSLn), the user has to define which ICU/OCU unit shall interface with which Free-Run Timer channel. This is done in the FSR0 to FSR2 registers. If ADC usage on a PWM signal (e.g. phase voltage or current measurement) is planned, it might be a good choice to use FRT0 for the PWM generation and FRT1 or FRT2 for the ICU pins, since FRT1 and FRT2 cannot trigger the ADC units, and synchronizing the ADC to the PWM is often useful if not necessary to minimize errors due to switching noise or oscillations typical for motor control applications.

An example configuration for block commutation could look like this:

```
void init_block_pwm(void){    // configure Multi-function timer for block PWM output with
                              // center-aligned PWM
PDR3=ALL_OFF;                 // the output pins not used for PWM remain in inactive state
FSR0=0x00;                    // all OCUs use FRT0
FSR1=0x00;
CPCLRB0=PWM_MAX_VAL;          // set compare clear register of FRT0
TCCSH0=0x00;                  // no irqs
TCCSL0=0x70;                  // Initialize FRT0, no irqs, no compare clear match, up/down
                              // mode, do not start counting yet, full speed (32MHz)
OCMOD=0x3F;                   // level set operation depending on CMOD bit in OCSHx, no
                              // toggle mode
OCSH1=0x10;                   // Inititalize OCU (Buffer transfer at zero, 0 on rising compare
                              // match, 1 on fallin one
OCSH3=0x10;                   // All RTO pins = IO output (will be switched in commutation
                              // ISR), starting level = 0
OCSH5=0x10;
OCSL0=0x03;                  // No match IRQs, enable compare buffers, enable compare
                              // operation for all registers
OCSL2=0x03;
```

```
OCSL4=0x03;
ADTRGC=0x00;                    // no adc trigger by MFT
DTCR0=0x00;                     // disable waveform generator (not needed for block commutation)
DTCR1=0x00;
DTCR2=0x00;

SIGCR1=0xA3;                    // DTTI enable, cancel 32cycle noise

PICSL01=0x3f;                   // ICU 0+1 clear flags, enable IRQs, both edges for hall sensor
                                // state detection
ICSL23=0x13;                    // ICU 2 clear flags, enable IRQ, both edges

TCCSL0_STOP=0;                  // start FRT0
}
```

Now, the PWM duty cycle can be controlled by writing the desired compare value (in the range of0...PWM_MAX_VAL) to the output compare buffer registers, OCCPB0...5. The new valuesbecome valid at the beginning of the next counting period of the selected FRT, if configured in thismanner (OCSLx_BUF1 and BUF0 bits). This avoids glitches and malfunctions that can occurwhen directly writing to the compare registers while output compare operation is in progress.Writing automatically addresses the OCCPBn register, while reading the same address reads theOCCPn register, so always the actual compare value is read.

The commutation of the BLDC is donein the ICU ISR by writing the appropriate values to the OCSH1,3,5 registers and thereby switchingthe output pins between PWM (OCU) output and I/O port. By pre-loading the I/O Port Data Register (PDR3 for the MB91265 Series), thecorrect safe or idle value for each pin can be defined. These values also are applied when the DTTIfunction of the Waveform Generator stops the PWM output, since this is done by switching theRTO pins to I/O port mode. At this point it might be important to know, that the noise cancelingfunction of the DTTI input uses the system clock. If e.g. overcurrent protection must also beensured in case of oscillator failure, it is important to disable the noise cancellation function.

OBSOLETE

4.3 BLDC with sinusoidal commutation

Even though BLDC motors exist which are specially designed for six-step commutation and therefore perform quite well using this method, block commutation does not deliver sufficient performance for all applications and oftendoes not fully make use of the motor's capabilities. For smooth and silent operation, sinusoidal waveforms are applied to the motor windings. Often encoders or resolvers are used for precise position feedback in demanding applications, but sinusoidal commutation is also possible solely relying on Hall sensor feedback. Even though this does not reach the dynamic performance of encoder-based systems (especially at low speed) because of the lack of position resolution, it can significantly reduce noise and vibrations and increase dynamic behavior compared to block commutation.

In the following configuration, a three-phase sine waveform will be generated using the same hall sensor configuration as above. Three output compare units using one Free-Run Timer set the momentary output duty cycles for each motor phase winding, while the Waveform Generator creates the appropriate non-overlapping waveforms with dead-time for the power stage and outputs the drive signals for the high- and low-side transistors of each half bridge. In a non-overlapping waveform, first one transistor is turned off, and after a dead time of approx. 500ns (depending on the inverter topology) the complementary transistor is turned on. This prevents current spikes which may occur if both transistors of a half bridge conduct at the same time during switch-over.

For the generation of the sine wave output, the OCU values have to be updated more frequently than once per sector given by the Hall sensors. Therefore, a phase increment is calculated from the ratio of PWM cycle time to hall sector cycle time, so that the angle at the beginning of the next sector equals the starting angle of the last sector + n-times the phase increment, with n = PWM cycles / Hall sector time. Of course, this assumes that the rotational speed of the motor does not change too quickly within this period (e.g. by load variations), or else a phase jump will occur at the next hall sector change.

Based on the actual rotor angle α, a new sine value for the first phase is fetched from the sine table in the FRT compare clear ISR, multiplied with the desired modulation index and written in the corresponding output compare buffer register, in this case OCCPB1. Then, the same is done with α+120° and α+240° for the other two phases. If desired, an offset can be added in order to realize an additional phase advance component for a BLDC at high rotational speeds. The calculation of the new compare values in the compare clear ISR has the advantage of leaving enough time for the calculations to complete (PWM_MAX_VAL * 31.25ns) before the value is transferred from the compare buffer register to the output compare register. Thereby, all three duty cycles are updated at the same time.

```
void init_sine_pwm(void){     // configure Multi-function timer for sinusoidal PWM output

  FSR0=0x00;                  // Set all OCUs to use FRT0
  FSR1=0x00;
  CPCLRB0=PWM_MAX_VAL;        // set compare clear register of FRT0
  OCCPB0=OCCPB2=OCCPB4=PWM_MAX_VAL;   // Output duty cycle is set by OCCPB1, 3, 5 only
  TCCSH0=0x20;                // enable zero detect irq
  TCCSL0=0x70;                // Initialize FRT0, no compare clear match, up/down mode, do
                              // not start counting yet, full speed (32MHz)
  OCMOD=0x00;                 // toggle mode for all output pins
  OCSH1=0x1d;                 // Initialize OCU: Buffer transfer at zero, toggle on compare
                              // match
  OCSH3=0x1d;                 // All RTO pins = waveform pins, RTO0,2,4 starting with 1,
                              // RTO1,3,5 starting with 0
  OCSH5=0x1d;
  OCSL0=0x03;                 // No compare match IRQs, enable compare buffers, enable compare
                              // operation for all registers
  OCSL2=0x03;
  OCSL4=0x03;
```

```
  ADTRGC=0x01;                // start adc1 conversion on zero detect (when pwm is on)
  TMRR0=MIN_DEADTIME;         // set dead time timer to about 500ns (enough for typical MOSFET
                              // power stages)
  TMRR1=MIN_DEADTIME;
  TMRR2=MIN_DEADTIME;
   DTCR0=0x04;               // enable waveform generator to output non-overlapping waveform
                              // pairs for inverter control
  DTCR1=0x04;
  DTCR2=0x04;

 SIGCR1=0xA3;                // DTTI enable, cancel 32cycle noise

  TCCSL0_STOP=0;              // start FRT0

}
```

Depending on the load conditions at motor startup, it can be necessary to start the motor using block commutation, and switch to sine commutation when speed is reliably measured. Otherwise, the incremental rotor angle estimation might not get in sync with the rotor, so that it can start to oscillate.

In the software examples, the sine modulation is generated with third harmonics injection by default. This means, that the sine table holds a sine wave with a small amount of additional third harmonics. Since the example's lookup table is calculated at startup and held in RAM, different variants can easily be compared by varying the generation formula. The effect of third harmonics injection is shown in the following figures:

Figure 5. Pure Sine Modulation

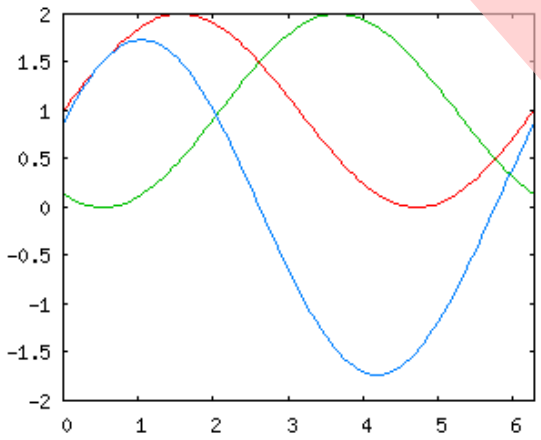
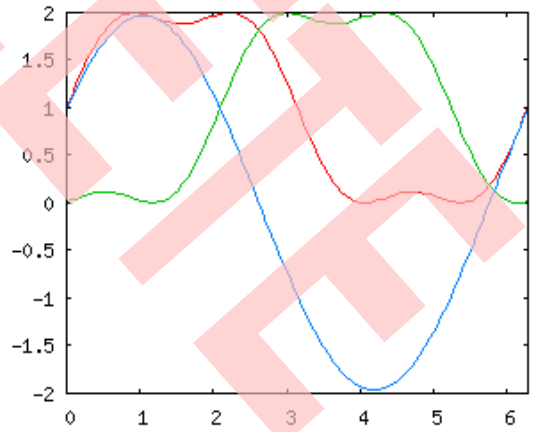


Figure 6. 3rd Harmonics Injection



The red and green lines show two inverter output voltages a and b, while the blue line shows the resulting phase voltage (a-b). It can be seen, that the third harmonics injection leads to a higher (about 15%) phase voltage at a given output amplitude (the DC bus voltage). The phase voltage still remains sinusoidal. The output waveforms are comparable to those generated by Space Vector Modulation (SVM), but need less calculations. Details to SVM can be found in literature.

The routine to update the duty cycle registers can look for example like this (The variable rotor_angle holds the actual rotor angle in degrees, scaled by 64):

OBSOLETE

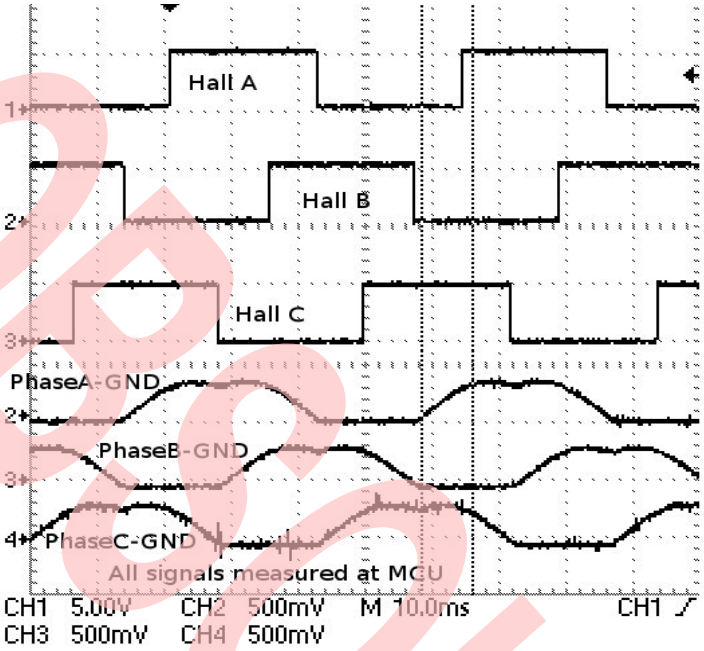
```
void __interrupt frt0_zero(void){     // This ISR sets the duty cycle for each output
                                      // according to the actual rotor position
  unsigned short int phase_a, duty;   // which is determined by the hall sensors and a speed
                                      // dependent increment
  if (dir > 0) {
    rotor_angle += phase_step;        // increment rotor angle at every pwm pulse
    while (rotor_angle>=23040) rotor_angle-=23040;    // keep angle in range of 0..359
                                      // degrees (scaled x64)
    phase_a = (rotor_angle>>6)+phase_advance;   // downscale the rotor angle to degrees
  }
  else {
    if (rotor_angle >= phase_step) rotor_angle -= phase_step;
    else rotor_angle = 23040-(phase_step-rotor_angle);
    phase_a = (rotor_angle>>6);       // downscale the rotor angle to degrees
     if (phase_a >= phase_advance) phase_a -= phase_advance;
     else phase_a = 360-(phase_advance-phase_a);
  }

  while (phase_a>=360) phase_a-=360;          // Catch possible overflow from phase advance
                                              // addition
  duty = ocu_value + 1;
  __DI();
  OCCPB1 = (duty * sinetable[phase_a])>>16; // calculate next ocu val for phase1
  OCCPB3 = (duty * sinetable[(phase_a>=120) ? phase_a-120 : phase_a+240])>>16;
                                              // calculate next ocu val for phase2
  OCCPB5 = (duty * sinetable[(phase_a>=240) ? phase_a-240 : phase_a+120])>>16;
                                              // calculate next ocu val for phase3
  __EI();
  TCCSH0_IRQZF=0;                             // clear interrupt flag
}
```

OBSOLETE

The following figure shows the correct alignment of the hall sensor signals and the phase voltages against ground (measured at J8 of the MotorKit-91F267-MC):

Figure 7. Hall Signals and Phase Voltages



In the picture above, the motor is spun counter-clockwise externally while connected to the MotorKit-MB91F267-MC.

If the motor is still not running sufficiently quiet, the noise might be caused by misalignment of the Hall sensors. If they are not equally positioned 120° from each other, the sectors indicated by the Hall sensors are not equally long, which causes errors in commutation timing. The effect of these errors can easily be reduced by a simple filter in the Hall sensor ISR, something like

```
rotor_angle = ( 3 * rotor_angle + new_angle_from_hall_sensor) >> 2
```

Depending on the motor, this can have a clearly audible effect. However, this slows down the systems response for load changes and might make the motor easier to stall at low speeds, so activating the filter when the motor reaches a certain speed can help.

OBSOLETE

A Appendix

A.1 Related Documents

MB91265 Series Hardware Manual:

hm91265-cm71-10130-2e.pdf

A.2 Related software examples

motorkit-91f267-mc_simple_bldc-v10

motorkit-91f267-mc_simple_sine_bldc-v10

OBSOLETE

Document History

Document Title: AN205150 - FR MB91265 Microcontroller BLDCDrive with Multi-Function Timer

Document Number:002-05150

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	MKEA	02/09/2007	Initial Release
			10/29/2007	Minor changes, Diagram updated
*A	5036148	MKEA	01/05/2016	Migrated Spansion Application Note from MCU-AN-300009-E-V11 to Cypress format
*B	5835199	AESATP12	07/27/2017	Updated logo and copyright.
*C	6041041	NOFL	01/22/2018	This spec is obsolete.

OBSOLETE

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2007-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.