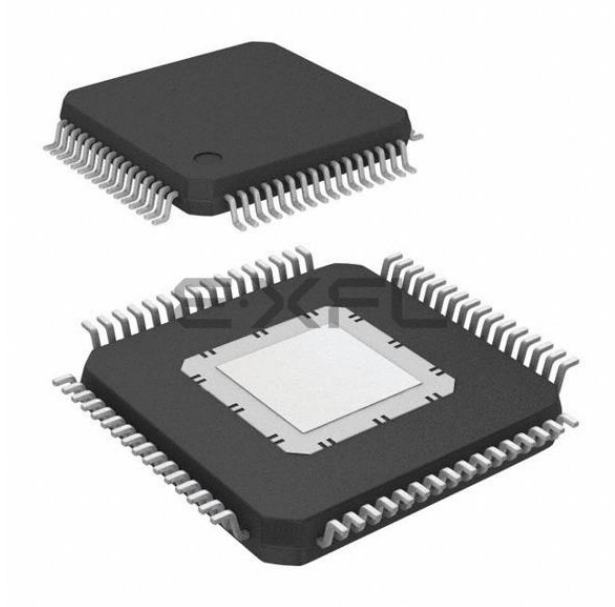


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

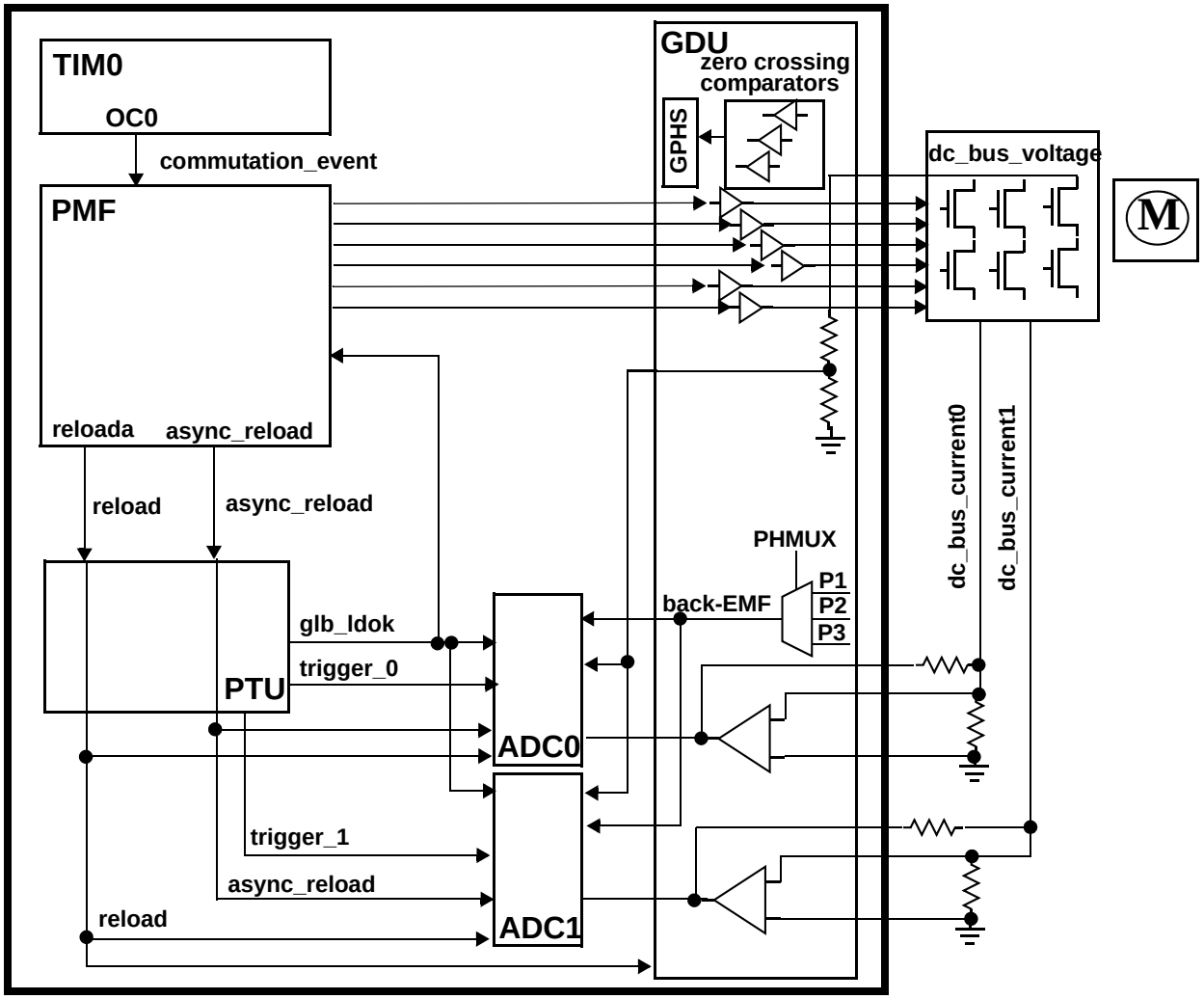
Details	
Product Status	Obsolete
Core Processor	S12Z
Core Size	16-Bit
Speed	50MHz
Connectivity	CANbus, LINbus, SCI, SPI
Peripherals	DMA, POR, PWM, WDT
Number of I/O	31
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	3.5V ~ 40V
Data Converters	A/D 9x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP Exposed Pad
Supplier Device Package	64-HLQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/s912zvml32f1vkh

Chapter 2
Port Integration Module (S12ZVMPIMV3)

7. Read port register PTIT[3:1] to determine starting sector.
8. Startup motor by applying PWM to the related motor phase.
9. In IC1 interrupt ISR calculate the delay to next commutation and store value to output compare register. Update registers with next values of mask and swap.
10. On next output compare event the buffered mask and swap information is transferred to the active PMF registers to execute the commutation.

1.13.5.2 Sensorless Commutation

Figure 1-15. Sensorless BLDC Configuration



To calculate the commutation time in a sensorless motor system the back-EMF zero crossing event of the currently non-fed phase within an electrical rotation cycle must be determined. For fast motor rotation, the ADC is used to measure the back-EMF voltage and the DC bus voltage to determine the zero crossing time. For slow motor rotation the GPHS register can be polled. In either case the zero crossing event is handled

Write: Anytime

Table 4-5. INT_CFADDR Field Descriptions

Field	Description
6–3 INT_CFADDR[6:3]	**Interrupt Request Configuration Data Register Select Bits** — These bits determine which of the 128 configuration data registers are accessible in the 8 register window at INT_CFDATA0–7. The hexadecimal value written to this register corresponds to the upper 4 bits of the vector number (multiply with 4 to get the vector address offset). If, for example, the value 0x70 is written to this register, the configuration data register block for the 8 interrupt vector requests starting with vector at address (vector base + (0x70*4 = 0x0001C0)) is selected and can be accessed as INT_CFDATA0–7.

4.3.2.3 Interrupt Request Configuration Data Registers (INT_CFDATA0–7)

The eight register window visible at addresses INT_CFDATA0–7 contains the configuration data for the block of eight interrupt requests (out of 128) selected by the interrupt configuration address register (INT_CFADDR) in ascending order. INT_CFDATA0 represents the interrupt configuration data register of the vector with the lowest address in this block, while INT_CFDATA7 represents the interrupt configuration data register of the vector with the highest address, respectively.

Address: 0x000018

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	PRIOLVL[2:0]		
W								
Reset	0	0	0	0	0	0	0	1(1)

= Unimplemented or Reserved

Figure 4-5. Interrupt Request Configuration Data Register 0 (INT_CFDATA0)

1. Please refer to the notes following the PRIOLVL[2:0] description below.

Address: 0x000019

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	PRIOLVL[2:0]		
W								
Reset	0	0	0	0	0	0	0	1(1)

= Unimplemented or Reserved

Figure 4-6. Interrupt Request Configuration Data Register 1 (INT_CFDATA1)

1. Please refer to the notes following the PRIOLVL[2:0] description below.

6.4.7.3.1 DBG Breakpoint Priorities And BDC Interfacing

Breakpoint operation is dependent on the state of the S12ZBDC module. BDM cannot be entered from a breakpoint unless the BDC is enabled (ENBDC bit is set in the BDC). If BDM is already active, breakpoints are disabled. In addition, while executing a BDC STEP1 command, breakpoints are disabled.

When the DBG breakpoints are mapped to BDM (BDMBP set), then if a breakpoint request, either from a BDC BACKGROUND command or a DBG event, coincides with an SWI instruction in application code, (i.e. the DBG requests a breakpoint at the next instruction boundary and the next instruction is an SWI) then the CPU gives priority to the BDM request over the SWI request.

On returning from BDM, the SWI from user code gets executed. Breakpoint generation control is summarized in Table 6-62.

Table 6-62. Breakpoint Mapping Summary

BRKCPU	BDMBP Bit (DBGC1[4])	BDC Enabled	BDM Active	Breakpoint Mapping
0	X	X	X	No Breakpoint
1	0	X	0	Breakpoint to SWI
1	0	1	1	No Breakpoint
1	1	0	X	No Breakpoint
1	1	1	0	Breakpoint to BDM
1	1	1	1	No Breakpoint

6.5 Application Information

6.5.1 Avoiding Unintended Breakpoint Re-triggering

Returning from an instruction address breakpoint using an RTI or BDC GO command without PC modification, returns to the instruction that generated the breakpoint. If an active breakpoint or trigger still exists at that address, this can re-trigger, disarming the DBG. If configured for BDM breakpoints, the user must apply the BDC STEP1 command to increment the PC past the current instruction.

If configured for SWI breakpoints, the DBG can be re configured in the SWI routine. If a comparator match occurs at an SWI vector address then a code SWI and DBG breakpoint SWI could occur simultaneously. In this case the SWI routine is executed twice before returning.

6.5.2 Debugging Through Reset

To debug through reset, the debugger can recognize a reset occurrence and pull the device BKGD pin low. This forces the device to leave reset in special single chip (SSC) mode, because the BKGD pin is used as the MODC signal in the reset phase. When the device leaves reset in SSC mode, CPU execution is halted and the device is in active BDM. Thus the debugger can configure the DBG for tracing and breakpoints before returning to application code execution. In this way it is possible to analyze the sequence of events emerging from reset. The recommended handling of the internal reset scenario is as follows:

8.7.3 Application Information for PLL and Oscillator Startup

The following C-code example shows a recommended way of setting up the system clock system using the PLL and Oscillator:

```
/* Procedure proposed by to setup PLL and Oscillator */
/* example for OSC = 4 MHz and Bus Clock = 25MHz, That is VCOCLK = 50MHz */

/* Initialize */
/* PLL Clock = 50 MHz, divide by one */
CPMUPOSTDIV = 0x00;

/* Generally: Whenever changing PLL reference clock (REFCLK) frequency to a higher value */
/* it is recommended to write CPMUSYNR = 0x00 in order to stay within specified */
/* maximum frequency of the MCU */
CPMUSYNR = 0x00;

/* configure PLL reference clock (REFCLK) for usage with Oscillator */
/* OSC=4MHz divide by 4 (3+1) = 1MHz, REFCLK range 1MHz to 2 MHz (REFFRQ[1:0] = 00) */
CPMUREFDIV = 0x03;

/* enable external Oscillator, switch PLL reference clock (REFCLK) to OSC */
CPMUOSC = 0x80;

/* multiply REFCLK = 1MHz by 2*(24+1)*1MHz = 50MHz */
/* VCO range 48 to 80 MHz (VCOFRQ[1:0] = 01) */
CPMUSYNR = 0x58;

/* clear all flags, especially LOCKIF and OSCIF */
CPMUIFLG = 0xFF;

/* put your code to loop and wait for the LOCKIF and OSCIF or */
/* poll CPMUIFLG register until both UPOSC and LOCK status are "1" */
/* that is CPMUIFLG == 0x1B */

/*..............continue to your main code execution here..............*/

/* in case later in your code you want to disable the Oscillator and use the */
/* 1MHz IRCCLK as PLL reference clock */

/* Generally: Whenever changing PLL reference clock (REFCLK) frequency to a higher value */
/* it is recommended to write CPMUSYNR = 0x00 in order to stay within specified */
/* maximum frequency of the MCU */
CPMUSYNR = 0x00;

/* disable OSC and switch PLL reference clock to IRC */
CPMUOSC = 0x00;

/* multiply REFCLK = 1MHz by 2*(24+1)*1MHz = 50MHz */
/* VCO range 48 to 80 MHz (VCOFRQ[1:0] = 01) */
CPMUSYNR = 0x58;

/* clear all flags, especially LOCKIF and OSCIF */
CPMUIFLG = 0xFF;
```

9.6 Functional Description

9.6.1 Overview

The ADC12B_LBA consists of an analog sub-block and a digital sub-block. It is a successive approximation analog-to-digital converter including a sample-and-hold mechanism and an internal charge scaled C-DAC (switched capacitor scaled digital-to-analog converter) with a comparator to realize the successive approximation algorithm.

9.6.2 Analog Sub-Block

The analog sub-block contains all analog circuits (sample and hold, C-DAC, analog Comparator, and so on) required to perform a single conversion. Separate power supplies VDDA and VSSA allow noise from the MCU circuitry to be isolated from the analog sub-block for improved accuracy.

9.6.2.1 Analog Input Multiplexer

The analog input multiplexers connect one of the external or internal analog input channels to the sample and hold storage node.

9.6.2.2 Sample and Hold Machine with Sample Buffer Amplifier

The Sample and Hold Machine controls the storage and charge of the storage node (sample capacitor) to the voltage level of the analog signal at the selected ADC input channel. This architecture employs the advantage of reduced crosstalk between channels.

The sample buffer amplifier is used to raise the effective input impedance of the A/D machine, so that external components (higher bandwidth or higher impedance connected as specified) are less significant to accuracy degradation.

During the sample phase, the analog input connects first via a sample buffer amplifier with the storage node always for two ADC clock cycles (“Buffer” sample time). For the remaining sample time (“Final” sample time) the storage node is directly connected to the analog input source. Please see also Figure 9-28 for illustration and the Appendix of the device reference manual for more details.
The input analog signals are unipolar and must be within the potential range of VSSA to VDDA.
During the hold process, the analog input is disconnected from the storage node.

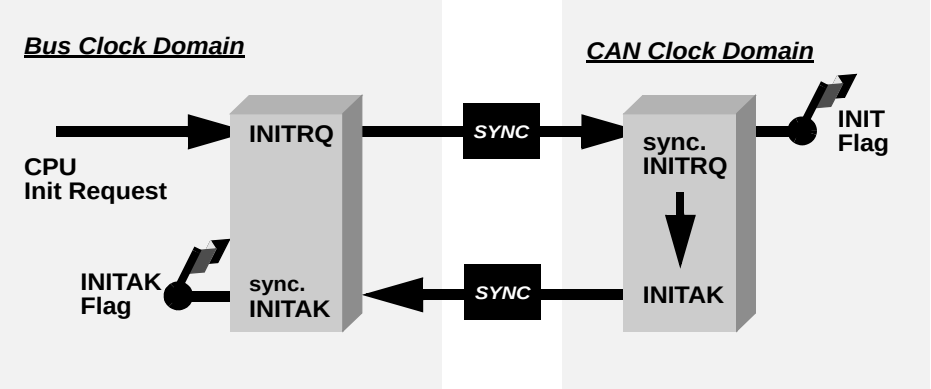


Figure 13-45. Initialization Request/Acknowledge Cycle

Due to independent clock domains within the MSCAN, INITRQ must be synchronized to all domains by using a special handshake mechanism. This handshake causes additional synchronization delay (see Figure 13-45).

If there is no message transfer ongoing on the CAN bus, the minimum delay will be two additional bus clocks and three additional CAN clocks. When all parts of the MSCAN are in initialization mode, the INITAK flag is set. The application software must use INITAK as a handshake indication for the request (INITRQ) to go into initialization mode.

NOTE

The CPU cannot clear INITRQ before initialization mode (INITRQ = 1 and INITAK = 1) is active.

13.4.5 Low-Power Options

If the MSCAN is disabled (CANE = 0), the MSCAN clocks are stopped for power saving.

If the MSCAN is enabled (CANE = 1), the MSCAN has two additional modes with reduced power consumption, compared to normal mode: sleep and power down mode. In sleep mode, power consumption is reduced by stopping all clocks except those to access the registers from the CPU side. In power down mode, all clocks are stopped and no power is consumed.

Table 13-37 summarizes the combinations of MSCAN and CPU modes. A particular combination of modes is entered by the given settings on the CSWAI and SLPRQ/SLPAK bits.

Table 15-31. PWM Reload Frequency B

LDFQB[3:0]	PWM Reload Frequency	LDFQ[3:0]	PWM Reload Frequency
0100	Every 5 PWM opportunities	1100	Every 13 PWM opportunities
0101	Every 6 PWM opportunities	1101	Every 14 PWM opportunities
0110	Every 7 PWM opportunities	1110	Every 15 PWM opportunities
0111	Every 8 PWM opportunities	1111	Every 16 PWM opportunities

Table 15-32. PWM Prescaler B

PRSCB[1:0]	Prescaler Value P_B	PWM Clock Frequency f_{PWM_B}
00	1	f_{core}
01	2	$f_{core}/2$
10	4	$f_{core}/4$
11	8	$f_{core}/8$

15.3.2.26 PMF Counter B Register (PMFCNTB)

Address: Module Base + 0x002A — Access: User read/write(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PMFCNTB														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-31. PMF Counter B Register (PMFCNTB)

1. Read: Anytime. Returns zero if MTG is clear.
   Write: Never

This register displays the state of the 15-bit PWM B counter.

15.3.2.27 PMF Counter Modulo B Register (PMFMODB)

Address: Module Base + 0x002C — Access: User read/write(1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PMFMODB														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-32. PMF Counter Modulo B Register (PMFMODB)

1. Read: Anytime. Returns zero if MTG is clear.
   Write: Anytime if MTG is set.Do not write a modulus value of zero for center-aligned operation. Do not write a modulus of zero or one in edge-aligned mode.

The 15-bit unsigned value written to this register is the PWM period in PWM clock periods.

15.4.2 Prescaler

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the core clock frequency by one, two, four, and eight. Each PWM generator has its own prescaler divisor. Each prescaler is buffered and will not be used by its PWM generator until the corresponding Load OK bit is set and a new PWM reload cycle begins.

15.4.3 PWM Generator

Each PWM generator contains a 15-bit up/down PWM counter producing output signals with software-selectable

- Alignment — The logic state of each pair EDGE bit determines whether the PWM pair outputs are edge-aligned or center-aligned
- Period — The value written to each pair PWM counter modulo register is used to determine the PWM pair period. The period can also be varied by using the prescaler
- With edge-aligned output, the modulus is the period of the PWM output in clock cycles
- With center-aligned output, the modulus is one-half of the PWM output period in clock cycles
- Pulse width — The number written to the PWM value register determines the pulse width duty cycle of the PWM output in clock cycles
  - — With center-aligned output, the pulse width is twice the value written to the PWM value register
  - — With edge-aligned output, the pulse width is the value written to the PWM value register

15.4.3.1 Alignment and Compare Output Polarity

Each edge-align bit, EDGEx, selects either center-aligned or edge-aligned PWM generator outputs.

PWM compare output polarity is selected by the CINV*n* bit field in the source control (PMFCINV) register. Please see the output operations in Figure 15-42 and Figure 15-43.

The PWM compare output is driven to a high state when the value of PWM value (PMFVAL*n*) register is greater than the value of PWM counter, and PWM compare is counting downwards if the corresponding channel CINV*n*=0. Or, the PWM compare output is driven to low state if the corresponding channel CINV*n*=1.

The PWM compare output is driven to low state when the value of PWM value (PMFVAL*n*) register matches the value of PWM counter, and PWM counter is counting upwards if the corresponding channel CINV*n*=0. Or, the PWM compare output is driven to high state if the corresponding channel CINV*n*=1.

16.2 External Signal Description

The SCI module has a total of two external pins.

16.2.1 TXD — Transmit Pin

The TXD pin transmits SCI (standard or infrared) data. It will idle high in either mode and is high impedance anytime the transmitter is disabled.

16.2.2 RXD — Receive Pin

The RXD pin receives SCI (standard or infrared) data. An idle line is detected as a line high. This input is ignored when the receiver is disabled and should be terminated to a known voltage.

16.3 Memory Map and Register Definition

This section provides a detailed description of all the SCI registers.

16.3.1 Module Memory Map and Register Definition

The memory map for the SCI module is given below in Figure 16-2. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the SCI module and the address offset for each register.

Table 16-11. SCISR1 Field Descriptions (continued)

Field	Description
3 OR	**Overrun Flag** — OR is set when software fails to read the SCI data register before the receive shift register receives the next frame. The OR bit is set immediately after the stop bit has been completely received for the second frame. The data in the shift register is lost, but the data already in the SCI data registers is not affected. Clear OR by reading SCI status register 1 (SCISR1) with OR set and then reading SCI data register low (SCIDRL). 0 No overrun 1 Overrun **Note:** OR flag may read back as set when RDRF flag is clear. This may happen if the following sequence of events occurs: 1. After the first frame is received, read status register SCISR1 (returns RDRF set and OR flag clear); 2. Receive second frame without reading the first frame in the data register (the second frame is not received and OR flag is set); 3. Read data register SCIDRL (returns first frame and clears RDRF flag in the status register); 4. Read status register SCISR1 (returns RDRF clear and OR set). Event 3 may be at exactly the same time as event 2 or any time after. When this happens, a dummy SCIDRL read following event 4 will be required to clear the OR flag if further frames are to be received.
2 NF	**Noise Flag** — NF is set when the SCI detects noise on the receiver input. NF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear NF by reading SCI status register 1(SCISR1), and then reading SCI data register low (SCIDRL). 0 No noise 1 Noise
1 FE	**Framing Error Flag** — FE is set when a logic 0 is accepted as the stop bit. FE bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. FE inhibits further data reception until it is cleared. Clear FE by reading SCI status register 1 (SCISR1) with FE set and then reading the SCI data register low (SCIDRL). 0 No framing error 1 Framing error
0 PF	**Parity Error Flag** — PF is set when the parity enable bit (PE) is set and the parity of the received data does not match the parity type bit (PT). PF bit is set during the same cycle as the RDRF flag but does not get set in the case of an overrun. Clear PF by reading SCI status register 1 (SCISR1), and then reading SCI data register low (SCIDRL). 0 No parity error 1 Parity error

16.4.6.5.2 Fast Data Tolerance

Figure 16-29 shows how much a fast received frame can be misaligned. The fast stop bit ends at RT10 instead of RT16 but is still sampled at RT8, RT9, and RT10.

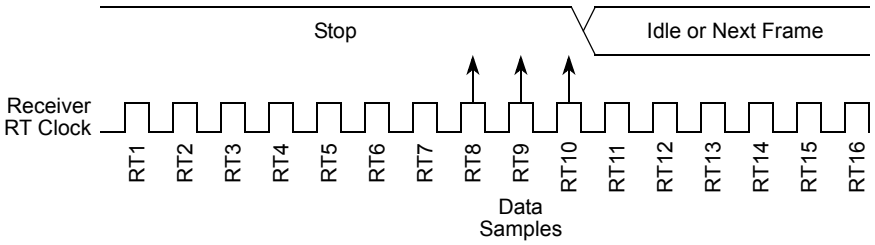


Figure 16-29. Fast Data

For an 8-bit data character, it takes the receiver 9 bit times x 16 RTr cycles + 9 RTr cycles = 153 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 16-29, the receiver counts 153 RTr cycles at the point when the count of the transmitting device is 10 bit times x 16 RTt cycles = 160 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is:

((160 – 153) / 160) x 100 = 4.375%

For a 9-bit data character, it takes the receiver 10 bit times x 16 RTr cycles + 9 RTr cycles = 169 RTr cycles to finish data sampling of the stop bit.

With the misaligned character shown in Figure 16-29, the receiver counts 169 RTr cycles at the point when the count of the transmitting device is 11 bit times x 16 RTt cycles = 176 RTt cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

((176 – 169) /176) x 100 = 3.98%

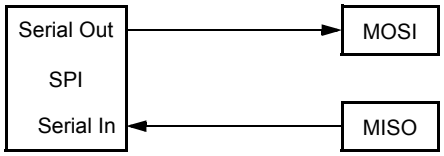
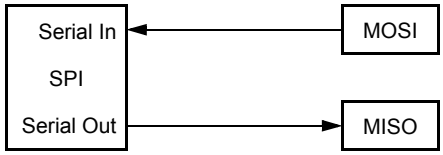
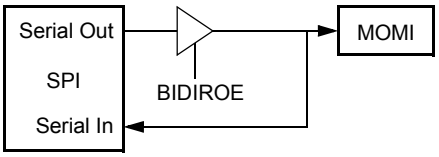
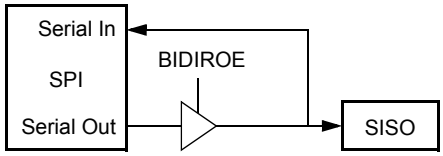
NOTE

Due to asynchronous sample and internal logic, there is maximal 2 bus cycles between startbit edge and 1st RT clock, and cause to additional tolerance loss at worst case. The loss should be 2/SBR/10*100%, it is small.For example, for highspeed baud=230400 with 25MHz bus, SBR should be 109, and the tolerance loss is 2/109/10*100=0.18%, and fast data tolerance is 4.375%-0.18%=4.195%.

16.4.6.6 Receiver Wakeup

To enable the SCI to ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCI control register 2 (SCICR2) puts the receiver into standby state during which receiver interrupts are disabled.The SCI will still load the receive data into the SCIDRH/L registers, but it will not set the RDRF flag.

Table 17-11. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
Normal Mode SPC0 = 0		
Bidirectional Mode SPC0 = 1		

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

- The SCK is output for the master mode and input for the slave mode.
- The $\overline{SS}$ is the input or output for the master mode, and it is always the input for the slave mode.
- The bidirectional mode does not affect SCK and $\overline{SS}$ functions.

NOTE

In bidirectional master mode, with mode fault enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode. In this case MISO becomes occupied by the SPI and MOSI is not used. This must be considered, if the MISO pin is used for another purpose.

17.4.6 Error Conditions

The SPI has one error condition:

- Mode fault error

17.4.6.1 Mode Fault Error

If the $\overline{SS}$ input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SCK lines simultaneously. This condition is not permitted in normal operation, the MODF bit in the SPI status register is set automatically, provided the MODFEN bit is set.

In the special case where the SPI is in master mode and MODFEN bit is cleared, the $\overline{SS}$ pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. In case

Table 18-3. GDUE Register Field Description

Field	Description
1 GCPE	**GDU Charge Pump Enable** — This bit enables the charge pump. This bit cannot be modified after GWP bit is set. See Section 18.4.4, “Charge Pump 0 Charge pump is disabled 1 Charge pump is enabled
0 GFDE	**GDU FET Pre-Driver Enable** — This bit enables the low-side and high-side FET pre-drivers. It must also be set in order to use the boost converter and the current sense amplifiers. This bit cannot be modified after GWP bit is set.See Section 18.4.2, “Low-Side FET Pre-Drivers and Section 18.4.3, “High-Side FET Pre-Driver. 0 Low-side and high-side drivers are disabled 1 Low-side and high-side drivers are enabled **NOTE** It is not allowed to set and clear GFDE bit periodically in order to switch on and off the FET pre-drivers. In order to switch on and off the FET pre-drivers the PMF module has to be used to mask and un-mask the PWM channels.

20.4.5.3 Valid Flash Module Commands

Table 20-29 present the valid Flash commands, as enabled by the combination of the functional MCU mode (Normal SingleChip NS, Special Singlechip SS) with the MCU security state (Unsecured, Secured).

Table 20-29. Flash Commands by Mode and Security State

FCMD	Command	Unsecured		Secured	
		NS (1)	SS(2)	NS (3)	SS(4)
0x01	Erase Verify All Blocks	*	*	*	*
0x02	Erase Verify Block	*	*	*	*
0x03	Erase Verify P-Flash Section	*	*	*	
0x04	Read Once	*	*	*	
0x06	Program P-Flash	*	*	*	
0x07	Program Once	*	*	*	
0x08	Erase All Blocks		*		*
0x09	Erase Flash Block	*	*	*	
0x0A	Erase P-Flash Sector	*	*	*	
0x0B	Unsecure Flash		*		*
0x0C	Verify Backdoor Access Key	*		*	
0x0D	Set User Margin Level	*	*	*	
0x0E	Set Field Margin Level		*		
0x10	Erase Verify EEPROM Section	*	*	*	
0x11	Program EEPROM	*	*	*	
0x12	Erase EEPROM Sector	*	*	*	
0x13	Protection Override	*	*	*	*

1. Unsecured Normal Single Chip mode

2. Unsecured Special Single Chip mode.

3. Secured Normal Single Chip mode.

4. Secured Special Single Chip mode.

Table 22-10. PWMCTL Field Descriptions

Note: Bits related to available channels have functional significance. Writing to unavailable bits has no effect. Read from unavailable bits return a zero

Field	Description
7 CON67	Concatenate Channels 6 and 7 0 Channels 6 and 7 are separate 8-bit PWMs. 1 Channels 6 and 7 are concatenated to create one 16-bit PWM channel. Channel 6 becomes the high order byte and channel 7 becomes the low order byte. Channel 7 output pin is used as the output for this 16-bit PWM (bit 7 of port PWMP). Channel 7 clock select control-bit determines the clock source, channel 7 polarity bit determines the polarity, channel 7 enable bit enables the output and channel 7 center aligned enable bit determines the output mode.
6 CON45	Concatenate Channels 4 and 5 0 Channels 4 and 5 are separate 8-bit PWMs. 1 Channels 4 and 5 are concatenated to create one 16-bit PWM channel. Channel 4 becomes the high order byte and channel 5 becomes the low order byte. Channel 5 output pin is used as the output for this 16-bit PWM (bit 5 of port PWMP). Channel 5 clock select control-bit determines the clock source, channel 5 polarity bit determines the polarity, channel 5 enable bit enables the output and channel 5 center aligned enable bit determines the output mode.
5 CON23	Concatenate Channels 2 and 3 0 Channels 2 and 3 are separate 8-bit PWMs. 1 Channels 2 and 3 are concatenated to create one 16-bit PWM channel. Channel 2 becomes the high order byte and channel 3 becomes the low order byte. Channel 3 output pin is used as the output for this 16-bit PWM (bit 3 of port PWMP). Channel 3 clock select control-bit determines the clock source, channel 3 polarity bit determines the polarity, channel 3 enable bit enables the output and channel 3 center aligned enable bit determines the output mode.
4 CON01	Concatenate Channels 0 and 1 0 Channels 0 and 1 are separate 8-bit PWMs. 1 Channels 0 and 1 are concatenated to create one 16-bit PWM channel. Channel 0 becomes the high order byte and channel 1 becomes the low order byte. Channel 1 output pin is used as the output for this 16-bit PWM (bit 1 of port PWMP). Channel 1 clock select control-bit determines the clock source, channel 1 polarity bit determines the polarity, channel 1 enable bit enables the output and channel 1 center aligned enable bit determines the output mode.
3 PSWAI	PWM Stops in Wait Mode — Enabling this bit allows for lower power consumption in wait mode by disabling the input clock to the prescaler. 0 Allow the clock to the prescaler to continue while in wait mode. 1 Stop the input clock to the prescaler whenever the MCU is in wait mode.
2 PFRZ	**PWM Counters Stop in Freeze Mode** — In freeze mode, there is an option to disable the input clock to the prescaler by setting the PFRZ bit in the PWMCTL register. If this bit is set, whenever the MCU is in freeze mode, the input clock to the prescaler is disabled. This feature is useful during emulation as it allows the PWM function to be suspended. In this way, the counters of the PWM can be stopped while in freeze mode so that once normal program flow is continued, the counters are re-enabled to simulate real-time operations. Since the registers can still be accessed in this mode, to re-enable the prescaler clock, either disable the PFRZ bit or exit freeze mode. 0 Allow PWM to continue while in freeze mode. 1 Disable PWM input clock to the prescaler whenever the part is in freeze mode. This is useful for emulation.

22.3.2.7 PWM Clock A/B Select Register (PWMCLKAB)

Each PWM channel has a choice of four clocks to use as the clock source for that channel as described below.

- The channel is disabled

In this way, the output of the PWM will always be either the old duty waveform or the new duty waveform, not some variation in between. If the channel is not enabled, then writes to the duty register will go directly to the latches as well as the buffer.

NOTE

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active duty due to the double buffering scheme.

See Section 22.4.2.3, “PWM Period and Duty” for more information.

NOTE

Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time. If the polarity bit is one, the output starts high and then goes low when the duty count is reached, so the duty registers contain a count of the high time. If the polarity bit is zero, the output starts low and then goes high when the duty count is reached, so the duty registers contain a count of the low time.

To calculate the output duty cycle (high time as a% of period) for a particular channel:

- Polarity = 0 (PPOL x =0)

  Duty Cycle = [(PWMPERx-PWMDTYx)/PWMPERx] * 100%
- Polarity = 1 (PPOLx = 1)

  Duty Cycle = [PWMDTYx / PWMPERx] * 100%

For boundary case programming values, please refer to Section 22.4.2.8, “PWM Boundary Cases”.

Module Base + 0x001C = PWMDTY0, 0x001D = PWMDTY1, 0x001E = PWMDTY2, 0x001F = PWMDTY3
Module Base + 0x0020 = PWMDTY4, 0x0021 = PWMDTY5, 0x0022 = PWMDTY6, 0x0023 = PWMDTY7

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	1	1	1	1	1	1	1	1

Figure 22-14. PWM Channel Duty Registers (PWMDTYx)

[1] This register is available only when the corresponding channel exists and is reserved if that channel does not exist. Writes to a reserved register have no functional effect. Reads from a reserved register return zeroes.

Read: Anytime

Write: Anytime

- For channels 0, 1, 4, and 5 the clock choices are clock A.
- For channels 2, 3, 6, and 7 the clock choices are clock B.

22.6 Interrupts

The PWM module has no interrupt.

4. At temperatures above 25°C the current may be naturally limited by the driver, in this case the limitation circuit is not engaged and the flag is not set.

D.2 Dynamic Electrical Characteristics

Table D-2. Dynamic electrical characteristics of the LIN/HV PHY

Characteristics noted under conditions 5.5V ≤ V_{LINSUP} ≤ 18 V unless otherwise noted(1) (2) (3). Typical values noted reflect the approximate parameter mean at T_A = 25°C under nominal conditions unless otherwise noted.							
Num	**C**	**Ratings**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
1		Minimum duration of wake-up pulse generating a wake-up interrupt	t_{WUFR}	56	72	120	µs
2		TxD-dominant timeout (in IRC clock periods) (4)	t_{DTLIM}	16388	—	16389	t_{IRC}
3		Propagation delay of receiver	t_{rx_pd}	—	—	6	µs
4		Symmetry of receiver propagation delay rising edge w.r.t. falling edge	t_{rx_sym}	-2	—	2	µs
LIN PHYSICAL LAYER: DRIVER CHARACTERISTICS FOR NOMINAL SLEW RATE - 20.0KBIT/S							
5		Rising/falling edge time (min to max / max to min)	t_{rise}	—	6.5	—	µs
6		Over-current masking window (IRC trimmed at 1MHz)	t_{OCLIM}	15	—	16	µs
7	M	Duty cycle 1 $T_{HRec(max)} = 0.744 \times V_{LINSUP}$ $T_{HDom(max)} = 0.581 \times V_{LINSUP}$ V_{LINSUP} = 5.5V...18V t_{Bit} = 50us $D1 = t_{Bus\_rec(min)} / (2 \times t_{Bit})$	D1	0.396 (5)	—	—	
8	M	Duty cycle 2 $T_{HRec(min)} = 0.422 \times V_{LINSUP}$ $T_{HDom(min)} = 0.284 \times V_{LINSUP}$ V_{LINSUP} = 5.5V...18V t_{Bit} = 50us $D2 = t_{Bus\_rec(max)} / (2 \times t_{Bit})$	D2	—	—	0.581[5]	
LIN PHYSICAL LAYER: DRIVER CHARACTERISTICS FOR SLOW SLEW RATE - 10.4KBIT/S							
9		Rising/falling edge time (min to max / max to min)	t_{rise}	—	13	—	µs
10		Over-current masking window (IRC trimmed at 1MHz)	t_{OCLIM}	31	—	32	µs
11	M	Duty cycle 3 $T_{HRec(max)} = 0.778 \times V_{LINSUP}$ $T_{HDom(max)} = 0.616 \times V_{LINSUP}$ V_{LINSUP} = 5.5V...18V t_{Bit} = 96us $D3 = t_{Bus\_rec(min)} / (2 \times t_{Bit})$	D3	0.417[5]	—	—	

Appendix L
Ordering Information

Customers can choose either the mask-specific partnumber or the generic, mask-independent partnumber. Ordering a mask-specific partnumber enables the customer to specify which particular maskset they receive whereas ordering the generic partnumber means that the currently preferred maskset (which may change over time) is shipped. In either case, the marking on the device always shows the generic, mask-independent partnumber and the mask set number. The below figure illustrates the structure of a typical mask-specific ordering number.

NOTES

Not every combination is offered. Table 1.2.1 lists available derivatives.
The mask identifier suffix and the Tape & Reel suffix are always both omitted from the partnumber which is actually marked on the device.