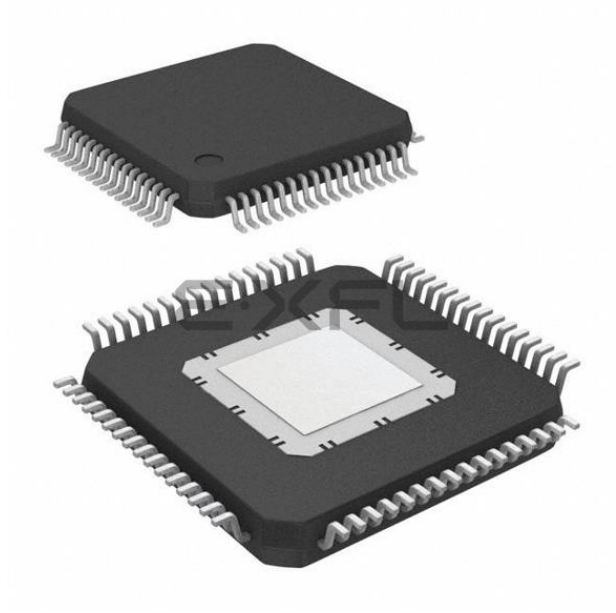


Welcome to E-XFL.COM

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"

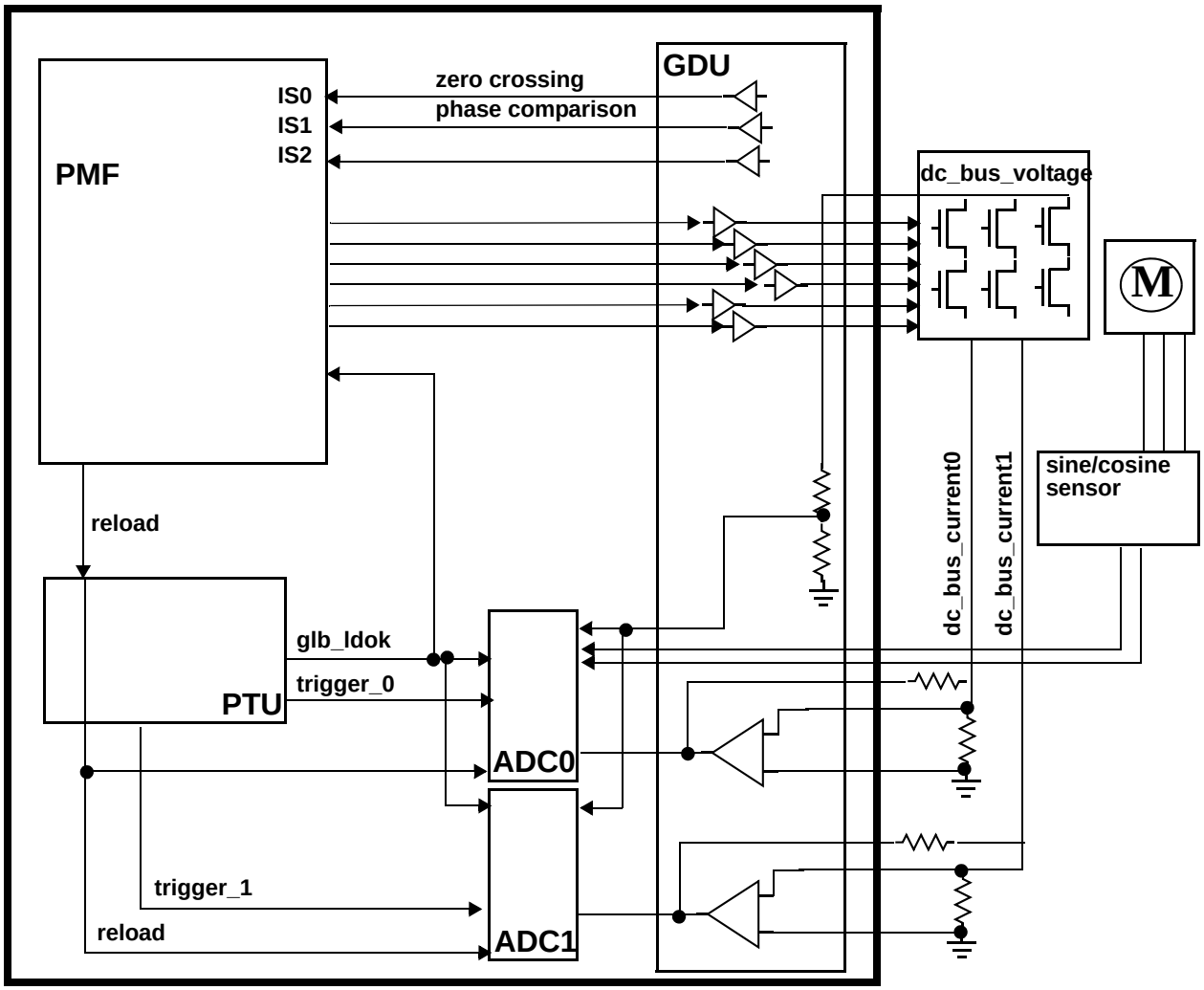
Details	
Product Status	Obsolete
Core Processor	S12Z
Core Size	16-Bit
Speed	50MHz
Connectivity	LINbus, SCI, SPI
Peripherals	DMA, POR, PWM, WDT
Number of I/O	31
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	3.5V ~ 40V
Data Converters	A/D 9x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°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/s912zvml12f1wkh

Chapter 7
ECC Generation Module (SRAM_ECCV1)

Chapter 8
S12 Clock, Reset and Power Management Unit (V10 and V6)

6. Map the sine/cosine input signals to ADC input channels.
7. Configure the EVDD1 pin as output.
8. Optionally use GDU phase comparators for zero crossing detection to correct dead time distortion.
9. Fetch targeted motor speed parameter from external source (e.g. SCI)
10. Configure PMF period and duty cycle.
11. Start motor by applying startup algorithm.
12. Sample the sine/cosine voltages periodically based on PWM cycle to determine motor position.
13. Use FOC algorithm to determine back EMF and motor speed.

Figure 1-17. PMSM Sine/Cosine Control Loop Configuration



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.

When BDM is activated, the CPU finishes executing the current instruction. Thereafter only BDC commands can affect CPU register contents until the BDC GO command returns from active BDM to user code or a device reset occurs. When BDM is activated by a breakpoint, the type of breakpoint used determines if BDM becomes active before or after execution of the next instruction.

NOTE

Attempting to activate BDM using a BGND instruction whilst the BDC is disabled, the CPU requires clock cycles for the attempted BGND execution. However BACKGROUND commands issued whilst the BDC is disabled are ignored by the BDC and the CPU execution is not delayed.

5.4.3 Clock Source

The BDC clock source can be mapped to a constant frequency clock source or a PLL based fast clock. The clock source for the BDC is selected by the CLKSW bit as shown in Figure 5-5. The BDC internal clock is named BDCSI clock. If BDCSI clock is mapped to the BDCCLK by CLKSW then the serial interface communication is not affected by bus/core clock frequency changes. If the BDC is mapped to BDCFCLK then the clock is connected to a PLL derived source at device level (typically bus clock), thus can be subject to frequency changes in application. Debugging through frequency changes requires SYNC pulses to re-synchronize. The sources of BDCCLK and BDCFCLK are specified at device level.

BDC accesses of internal device resources always use the device core clock. Thus if the ACK handshake protocol is not enabled, the clock frequency relationship must be taken into account by the host.

When changing the clock source via the CLKSW bit a minimum delay of 150 cycles at the initial clock speed must elapse before a SYNC can be sent. This guarantees that the start of the next BDC command uses the new clock for timing subsequent BDC communications.

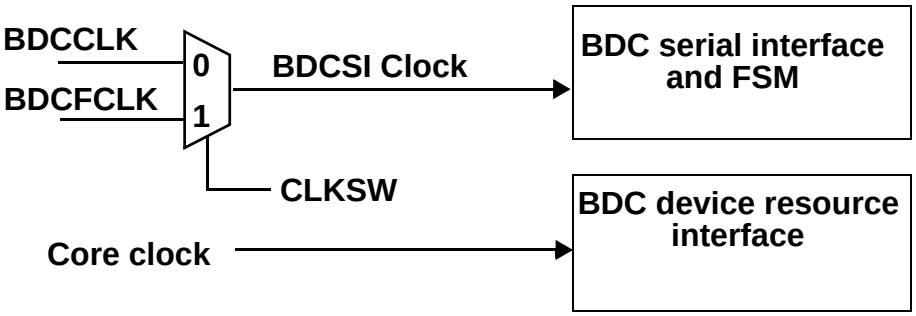


Figure 5-5. Clock Switch

5.4.4 BDC Commands

BDC commands can be classified into three types as shown in Table 5-7.

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
0x010C-0x010F	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0110	DBGACTL	R	0	NDB	INST	0	RW	RWE	reserved	COMPE
		W								
0x0111-0x0114	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0115	DBGAAH	R	DBGAA[23:16]							
		W								
0x0116	DBGAAM	R	DBGAA[15:8]							
		W								
0x0117	DBGAAL	R	DBGAA[7:0]							
		W								
0x0118	DBGAD0	R	Bit 31	30	29	28	27	26	25	Bit 24
		W								
0x0119	DBGAD1	R	Bit 23	22	21	20	19	18	17	Bit 16
		W								
0x011A	DBGAD2	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x011B	DBGAD3	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x011C	DBGADM0	R	Bit 31	30	29	28	27	26	25	Bit 24
		W								
0x011D	DBGADM1	R	Bit 23	22	21	20	19	18	17	Bit 16
		W								
0x011E	DBGADM2	R	Bit 15	14	13	12	11	10	9	Bit 8
		W								
0x011F	DBGADM3	R	Bit 7	6	5	4	3	2	1	Bit 0
		W								
0x0120	DBGBCTL	R	0	0	INST	0	RW	RWE	reserved	COMPE
		W								
0x0121-0x0124	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0125	DBGBAH	R	DBGBA[23:16]							
		W								
0x0126	DBGBAM	R	DBGBA[15:8]							
		W								
0x0127	DBGBAL	R	DBGBA[7:0]							
		W								

Figure 6-2. Quick Reference to DBG Registers

Table 6-42. Comparator Address Bus Matches

Access	Address	ADDR[n]	ADDR[n+1]	ADDR[n+2]	ADDR[n+3]
8-bit	ADDR[n]	Match	No Match	No Match	No Match

If the comparator INST bit is set, the comparator address register contents are compared with the PC, the data register contents and access type bits are ignored. The comparator address register must be loaded with the address of the first opcode byte.

6.4.2.2 Address and Data Comparator Match

Comparators A and C feature data comparators, for data access comparisons. The comparators do not evaluate if accessed data is valid. Accesses across aligned 32-bit boundaries are split internally into consecutive accesses. The data comparator mapping to accessed addresses for the CPU is shown in Table 6-43, whereby the Address column refers to the lowest 2 bits of the lowest accessed address. This corresponds to the most significant data byte.

Table 6-43. Comparator Data Byte Alignment

Address[1:0]	Data Comparator
00	DBGxD0
01	DBGxD1
10	DBGxD2
11	DBGxD3

The fixed mapping of data comparator bytes to addresses within a 32-bit data field ensures data matches independent of access size. To compare a single data byte within the 32-bit field, the other bytes within that field must be masked using the corresponding data mask registers. This ensures that any access of that byte (32-bit,16-bit or 8-bit) with matching data causes a match. If no bytes are masked then the data comparator always compares all 32-bits and can only generate a match on a 32-bit access with correct 32-bit data value. In this case, 8-bit or 16-bit accesses within the 32-bit field cannot generate a match even if the contents of the addressed bytes match because all 32-bits must match. In Table 6-44 the Access Address column refers to the address bits[1:0] of the lowest accessed address (most significant data byte).

Table 6-44. Data Register Use Dependency On CPU Access Type

			Memory Address[2:0]						
Case	Access Address	Access Size	000	001	010	011	100	101	110
1	00	32-bit	DBGxD0	DBGxD1	DBGxD2	DBGxD3			
2	01	32-bit		DBGxD1	DBGxD2	DBGxD3	DBGxD0		
3	10	32-bit			DBGxD2	DBGxD3	DBGxD0	DBGxD1	
4	11	32-bit				DBGxD3	DBGxD0	DBGxD1	DBGxD2
5	00	16-bit	DBGxD0	DBGxD1					
6	01	16-bit		DBGxD1	DBGxD2				
7	10	16-bit			DBGxD2	DBGxD3			

9.5.2.8 ADC Interrupt Enable Register (ADCIE)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	SEQAD_IE	CONIF_OIE	Reserved	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-11. ADC Interrupt Enable Register (ADCIE)

Read: Anytime

Write: Anytime

Table 9-13. ADCIE Field Descriptions

Field	Description
7 SEQAD_IE	**Conversion Sequence Abort Done Interrupt Enable Bit** — This bit enables the conversion sequence abort event done interrupt. 0 Conversion sequence abort event done interrupt disabled. 1 Conversion sequence abort event done interrupt enabled.
6 CONIF_OIE	**ADCCONIF Register Flags Overrun Interrupt Enable** — This bit enables the flag which indicates if an overrun situation occurred for one of the CON_IF[15:1] flags or for the EOL_IF flag. 0 No ADCCONIF Register Flag overrun occurred. 1 ADCCONIF Register Flag overrun occurred.

Figure 13-24. Receive/Transmit Message Buffer — Extended Identifier Mapping (continued)

Register Name	Bit 7	6	5	4	3	2	1	Bit0

= Unused, always read ‘x’

Read:

- For transmit buffers, anytime when TXEx flag is set (see Section 13.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 13.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”).
- For receive buffers, only when RXF flag is set (see Section 13.3.2.5, “MSCAN Receiver Flag Register (CANRFLG)”).

Write:

- For transmit buffers, anytime when TXEx flag is set (see Section 13.3.2.7, “MSCAN Transmitter Flag Register (CANTFLG)”) and the corresponding transmit buffer is selected in CANTBSEL (see Section 13.3.2.11, “MSCAN Transmit Buffer Selection Register (CANTBSEL)”).
- Unimplemented for receive buffers.

Reset: Undefined because of RAM-based implementation

Figure 13-25. Receive/Transmit Message Buffer — Standard Identifier Mapping

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
IDR0 0x00X0	R W	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
IDR1 0x00X1	R W	ID2	ID1	ID0	RTR	IDE (=0)			
IDR2 0x00X2	R W								
IDR3 0x00X3	R W								

= Unused, always read ‘x’

13.3.3.1 Identifier Registers (IDR0–IDR3)

The identifier registers for an extended format identifier consist of a total of 32 bits: ID[28:0], SRR, IDE, and RTR. The identifier registers for a standard format identifier consist of a total of 13 bits: ID[10:0], RTR, and IDE.

14.4.2 Memory based trigger event list

The lists with the trigger values are located inside the global memory map. The location of the trigger lists in the memory map is configured with registers PTUPTR and TGxLxIDX. If one of the TGs is enabled then the PTUPTR register is locked. If the TG is enabled then the associated TGxLxIDX registers are locked.

The trigger values inside the trigger list are 16 bit values. Each 16 bit value defines the delay between the reload event and the trigger event in bus clock cycles. A delay value of 0x0000 will be interpreted as End Of trigger List (EOL) symbol. The list must be sorted in ascending order. If a subsequent value is smaller than the previous value or the loaded trigger value is smaller than the current counter value then the TGxTEIF error indication is generated and the trigger generation of this list is stopped until the next reload event. For more information about these error scenario see Section 14.4.5.5, “Trigger Generator Timing Error”.

The module is not able to access memory area outside the 256 byte window starting at the memory address defined by PTUPTR.

Figure 14-23. Global Memory map usage

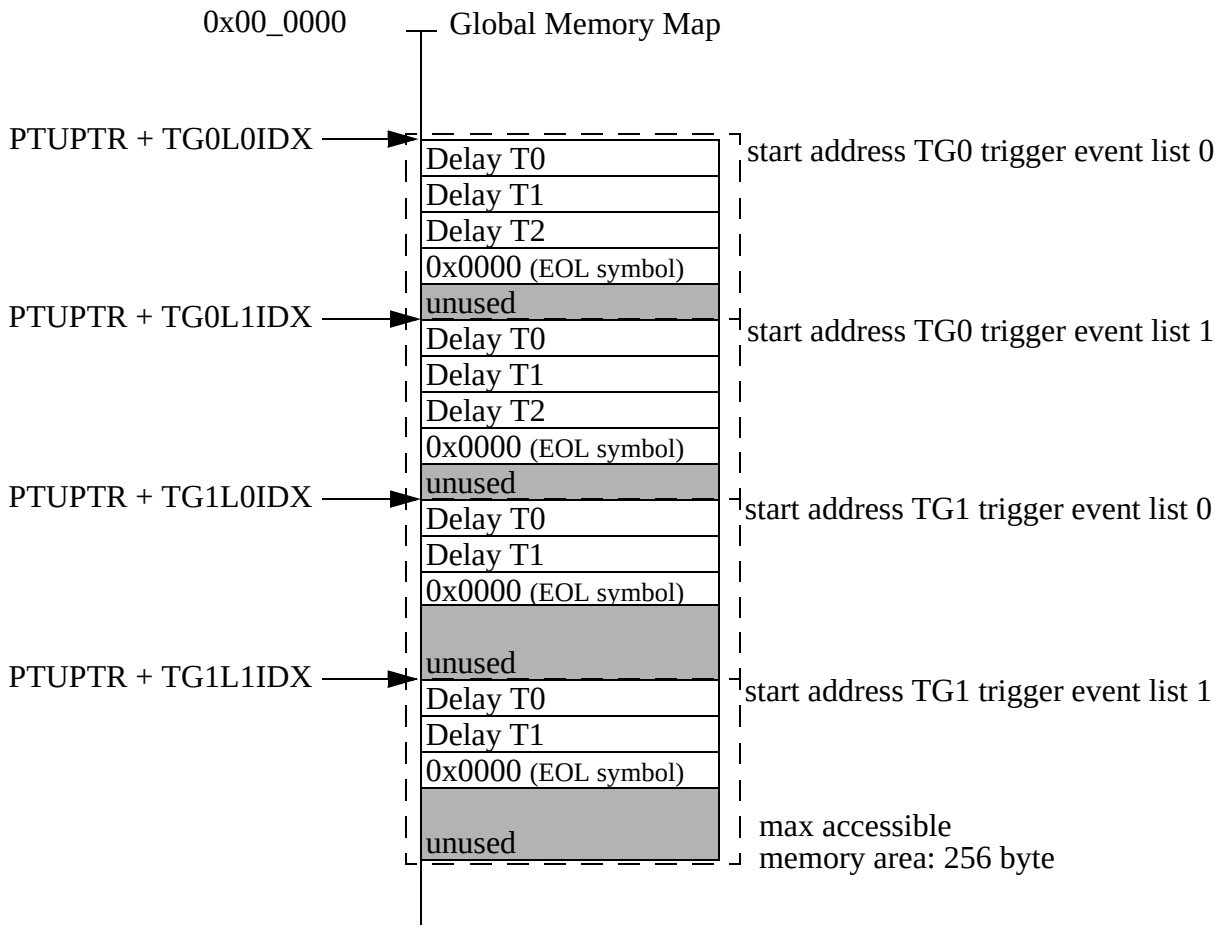


Table 15-23. PMFROIE Descriptions

Field	Description
2 PMFROIEC	Reload Overrun Interrupt Enable C — 0 Reload Overrun Interrupt C disabled 1 Reload Overrun Interrupt C enabled
1 PMFROIEB	Reload Overrun Interrupt Enable B — 0 Reload Overrun Interrupt B disabled 1 Reload Overrun Interrupt B enabled
0 PMFROIEA	Reload Overrun Interrupt Enable A — 0 Reload Overrun Interrupt A disabled 1 Reload Overrun Interrupt A enabled

15.3.2.16 PMF Interrupt Flag Register (PMFROIF)

Address: Module Base + 0x001D

Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	PMFROIFC	PMFROIFB	PMFROIFA
W								
Reset	0	0	0	0	0	0	0	0

Figure 15-19. PMF Interrupt Flag Register (PMFROIF)

1. Read: Anytime
   Write: Anytime. Write 1 to clear.

Table 15-24. PMFROIF Field Descriptions

Field	Description
2 PMFROIFC	Reload Overrun Interrupt Flag C — If a reload event occurs when the LDOKC or global load OK bit is not set then this flag will be set. 0 No Reload Overrun C occurred 1 Reload Overrun C occurred
1 PMFROIFB	Reload Overrun Interrupt Flag B — If a reload event occurs when the LDOKB or global load OK bit is not set then this flag will be set. 0 No Reload Overrun B occurred 1 Reload Overrun B occurred
0 PMFROIFA	Reload Overrun Interrupt Flag A — If PMFCFG2[REV1:REV0]=01 and a reload event occurs when the LDOKA or global load OK bit is not set then this flag will be set. If PMFCFG2[REV1:REV0]=10 and a reload event occurs when the LDOKB or global load OK bit is not set then this flag will be set. If PMFCFG2[REV1:REV0]=11 and a reload event occurs when the LDOKC or global load OK bit is not set then this flag will be set. If PMFCFG2[REV1:REV0]=00 no flag will be generated. 0 No Reload Overrun A occurred 1 Reload Overrun A occurred

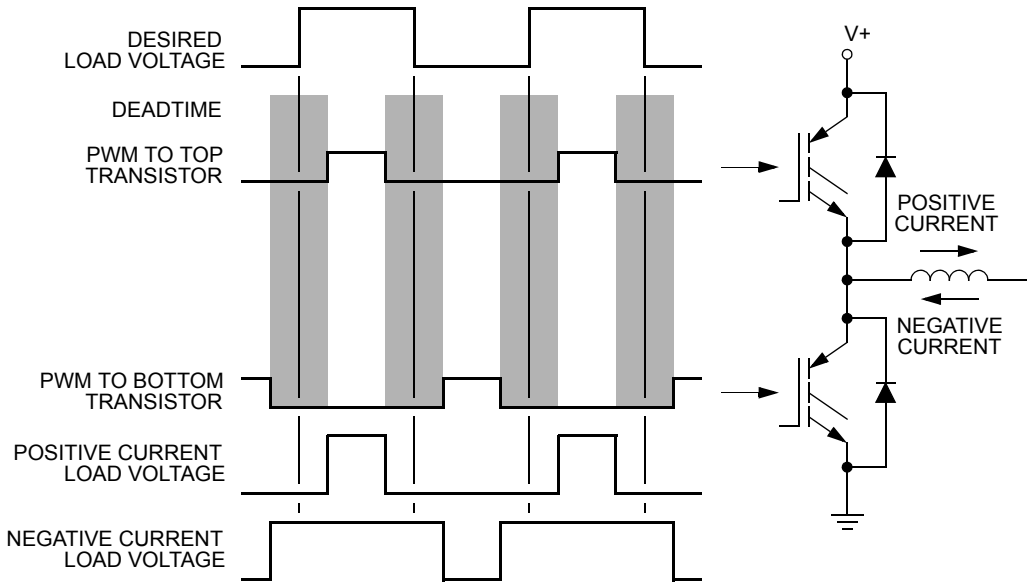


Figure 15-54. Deadtime Distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths were shortened by deadtime insertion, the averaged sinusoidal output will be less than the desired value. However, when deadtime is inserted, it creates a distortion in motor current waveform. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time, this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the motor current for that pair. See Figure 15-54. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in an odd-numbered/even numbered PWM register pair. Either the odd or the even PMFVAL register controls the pulse width at any given time. For a given PWM pair, whether the odd or even PMFVAL register is active depends on either:

- The state of the current status input, $\overline{IS}$, for that driver
- The state of the odd/even correction bit, IPOLx, for that driver if ICC bits in the PMFICCTL register are set to zeros
- The direction of PWM counter if ICC bits in the PMFICCTL register are set to ones

To correct deadtime distortion, software can decrease or increase the value in the appropriate PMFVAL register.

19.4 Functional Description

19.4.1 General

The LIN/HV Physical Layer module implements the physical layer of the LIN/HV interface. In the LIN version, this physical layer can be driven by the SCI (Serial Communication Interface) module or directly through the LPDR register.In the HV Phy version, the input can be routed to an internal timer to measure the frequency and duty cycle of the PWM input signal. If required, the output can directly be controlled by the LPDR register, e.g. to send diagnostic feedback.

19.4.2 Slew Rate and LIN Mode Selection

The slew rate can be selected for Electromagnetic Compatibility (EMC) optimized operation at 10.4 kbit/s and 20 kbit/s as well as at fast baud rate (up to 250 kbit/s) for test and programming. The slew rate can be chosen with the bits LPSLR[1:0] in the LIN Slew Rate Mode Register (LPSLRM). The default slew rate corresponds to 20 kbit/s.

In the HV Phy version, the TxD-dominant timeout must be disabled (LPDTDIS=1) in order e.g. to transmit a PWM pulse.

Changing the slew rate (LPSLRM Register) during transmission is not allowed in order to avoid unwanted effects. To change the register, the LIN/HV Physical Layer must first be disabled (LPE=0). Once it is updated, the LIN/HV Physical Layer can be enabled again.

NOTE

For 20 kbit/s and Fast Mode communication speeds, the corresponding slew rate ***MUST*** be set; otherwise, the communication is not guaranteed (violation of the specified LIN duty cycles). For 10.4 kbit/s, the 20 kbit/s slew rate ***can*** be set but the EMC performance is worse. The up to 250 kbit/s slew rate must be chosen ***ONLY*** for fast mode, not for any of the 10.4 kbit/s or 20 kbit/s LIN compliant communication speeds.

19.4.2.1 10.4 kbit/s and 20 kbit/s

When the slew rate is chosen for 10.4 kbit/s or 20 kbit/s communication, a control loop is activated within the module to make the rise and fall times of the LIN bus independent from VLINSUP and the load on the bus.

19.4.2.2 Fast Mode (not LIN compliant)

Choosing this slew rate allows baud rates up to 250 kbit/s by having much steeper edges (please refer to electricals). As for the 10.4 kbit/s and 20 kbit/s modes, the slope control loop is also engaged. This mode is used for fast communication only, and the LIN electricals are not supported (for example, the LIN duty cycles).

21.4.2.5 Reserved Register

Module Base + 0x0004 Access: User read/write(1)

	7	6	5	4	3	2	1	0
R W	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Reset	x	x	x	x	x	x	x	x

Figure 21-6. Reserved Register

1. Read: Anytime
   Write: Only in special mode

NOTE

This reserved register is designed for factory test purposes only and is not intended for general user access. Writing to this register when in special modes can alter the modules functionality.

21.4.2.6 Reserved Register

Module Base + 0x0005 Access: User read/write(1)

	7	6	5	4	3	2	1	0
R W	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Reset	x	x	x	x	x	x	x	x

Figure 21-7. Reserved Register

1. Read: Anytime
   Write: Only in special mode

NOTE

This reserved register is designed for factory test purposes only and is not intended for general user access. Writing to this register when in special modes can alter the modules functionality.

21.4.2.7 CAN Physical Layer Interrupt Enable Register (CPIE)

Module Base + 0x0006 Access: User read/write(1)

	7	6	5	4	3	2	1	0
R	0	0	0	CPVFIE	CPDTIE	0	0	CPOCIE
W								
Reset	0	0	0	0	0	0	0	0

Figure 21-8. CAN Physical Layer Interrupt Enable Register (CPIE)

1. Read: Anytime
   Write: Anytime

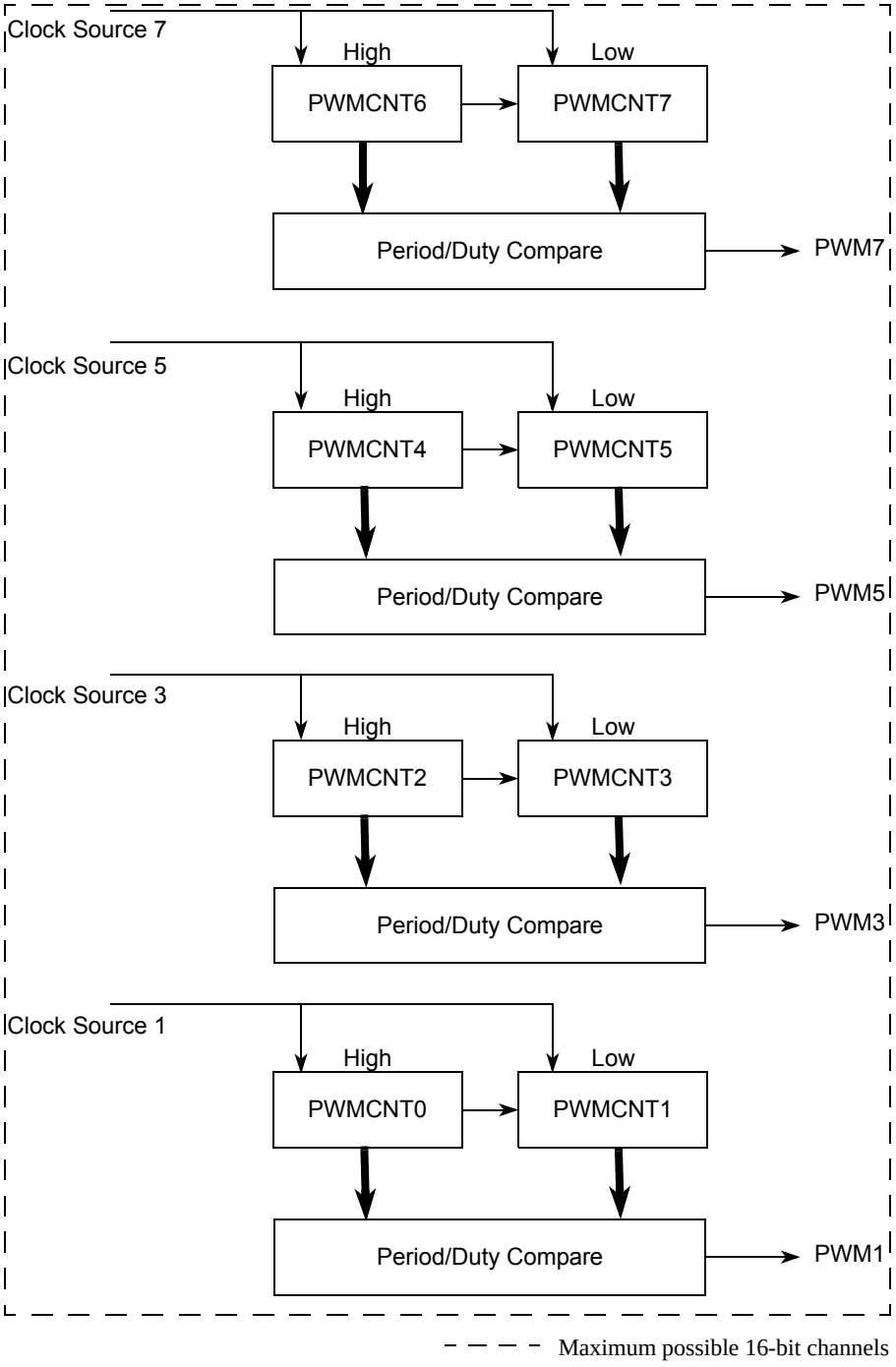


Figure 22-21. PWM 16-Bit Mode

Once concatenated mode is enabled (CONxx bits set in PWMCTL register), enabling/disabling the corresponding 16-bit PWM channel is controlled by the low order PWMEx bit. In this case, the high order bytes PWMEx bits have no effect and their corresponding PWM output is disabled.

C.1.2 ADC Accuracy

Table C-3. specifies the ADC conversion performance excluding any errors due to current injection, input capacitance and source resistance.

C.1.2.1 ADC Accuracy Definitions

For the following definitions see also **Figure C-2.**.
Differential non-linearity (DNL) is defined as the difference between two adjacent switching steps.

$$DNL(i) = \frac{V_i - V_{i-1}}{1LSB} - 1$$

The integral non-linearity (INL) is defined as the sum of all DNLs:

$$INL(n) = \sum_{i=1}^{n} DNL(i) = \frac{V_n - V_0}{1LSB} - n$$

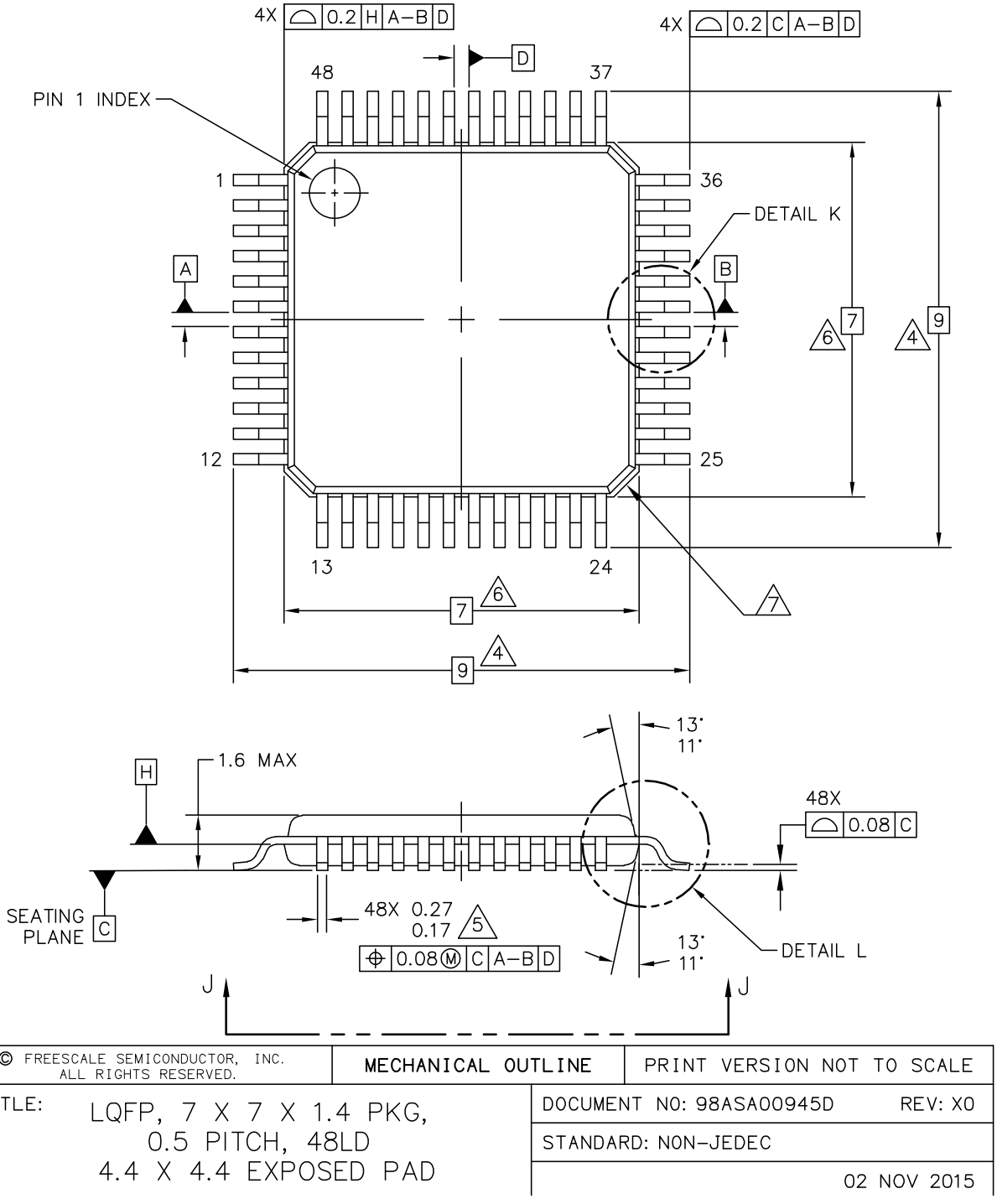
H.3 Dynamic Electrical Characteristics

Table H-3. Dynamic Electrical Characteristics

Characteristics noted under conditions 5.5V ≤ VSUP ≤ 18 V, -40°C ≤ T_J ≤ 175°C unless otherwise noted. Typical values noted reflect the approximate parameter mean at T_A = 25°C under nominal conditions unless otherwise noted.						
Num	**Ratings**	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
SIGNAL EDGE RISE AND FALL TIMES (CANH, CANL)						
1	Propagation Loop Delay TXD to RXD (Recessive to Dominant) Slew Rate 6 Slew Rate 5 Slew Rate 4 Slew Rate 2 Slew Rate 1 Slew Rate 0	t_{LRD}	 — 	 146 112 89 83 72 64	 (255)	ns
2	Propagation Delay TXD to CAN (Recessive to Dominant) Slew Rate 6 Slew Rate 5 Slew Rate 4 Slew Rate 2 Slew Rate 1 Slew Rate 0	t_{TRD}	 — 	 98 63 43 38 28 23	 — 	ns
3	Propagation Delay CAN to RXD (Recessive to Dominant, using slew rate 0)	t_{RRD}	—	42	—	ns
4	Propagation Loop Delay TXD to RXD (Dominant to Recessive) Slew Rate 6 Slew Rate 5 Slew Rate 4 Slew Rate 2 Slew Rate 1 Slew Rate 0	t_{LDR}	 — 	 366 224 153 139 114 102	 (255)	ns
5	Propagation Delay TXD to CAN (Dominant to Recessive) Slew Rate 6 Slew Rate 5 Slew Rate 4 Slew Rate 2 Slew Rate 1 Slew Rate 0	t_{TDR}	 — 	 280 152 90 81 56 46	 — 	ns
6	Propagation Delay CAN to RXD (Dominant to Recessive, using slew rate 0)	t_{RDR}	—	56	—	ns

K.1 48LQFP-EP Mechanical Information

Figure K-1. 48LQFP-EP Mechanical Information



Appendix M
Detailed Register Address Map

Registers listed are a superset of all registers in the S12ZVM-Family.

The device overview section specifies module (version) assignment to individual devices.

M.1 0x0000–0x0003 Part ID

Address	Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x0000	PARTID0	R	0	0	0	0	0	0	0	0
		W								
0x0001	PARTID1	R	0	0	0	1	Derivative Dependent (see Table 1-6)			
		W								
0x0002	PARTID2	R	0	0	0	0	0	0	0	0
		W								
0x0003	PARTID3	R	Revision Dependent							
		W								

M.2 0x0010–0x001F S12ZINT

Address	Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0010	IVBR	R	IVB_ADDR[15:8]							
		W								
0x0011		R	IVB_ADDR[7:1]							0
		W								
0x0012-0x0016	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x0017	INT_CFADDR	R	0	INT_CFADDR[6:3]				0	0	0
		W								
0x0018	INT_CFDATA0	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x0019	INT_CFDATA1	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x001A	INT_CFDATA2	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x001B	INT_CFDATA3	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x001C	INT_CFDATA4	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x001D	INT_CFDATA5	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x001E	INT_CFDATA6	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								

Global Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x02D6	PIES	R	0	0	PIES5[5]	PIES4[5]	PIES3	PIES2	PIES1	PIES0
		W								
0x02D7	PIFS	R	0	0	PIFS5[5]	PIFS4[5]	PIFS3	PIFS2	PIFS1	PIFS0
		W								
0x02D8–0x02DE	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x02DF	WOMS	R	0	0	WOMS5[5]	WOMS4[5]	WOMS3	WOMS2	WOMS1	WOMS0
		W								
0x02E0–0x02EF	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x02F0	PTP	R	0	0	0	0	0	PTP2[5]	PTP1	PTP0
		W								
0x02F1	PTIP	R	0	0	0	0	0	PTIP2[5]	PTIP1	PTIP0
		W								
0x02F2	DDRP	R	0	0	0	0	0	DDRP2[5]	DDRP1	DDRP0
		W								
0x02F3	PERP	R	0	0	0	0	0	PERP2[5]	PERP1	PERP0
		W								
0x02F4	PPSP	R	0	0	0	0	0	PPSP2[5]	PPSP1	PPSP0
		W								
0x02F5	Reserved	R	0	0	0	0	0	0	0	0
		W								
0x02F6	PIEP	R	OCIE1	0	0	0	0	PIEP2[5]	PIEP1	PIEP0
		W								
0x02F7	PIFP	R	OCIF1	0	0	0	0	PIFP2[5]	PIFP1	PIFP0
		W								