



Welcome to [E-XFL.COM](#)

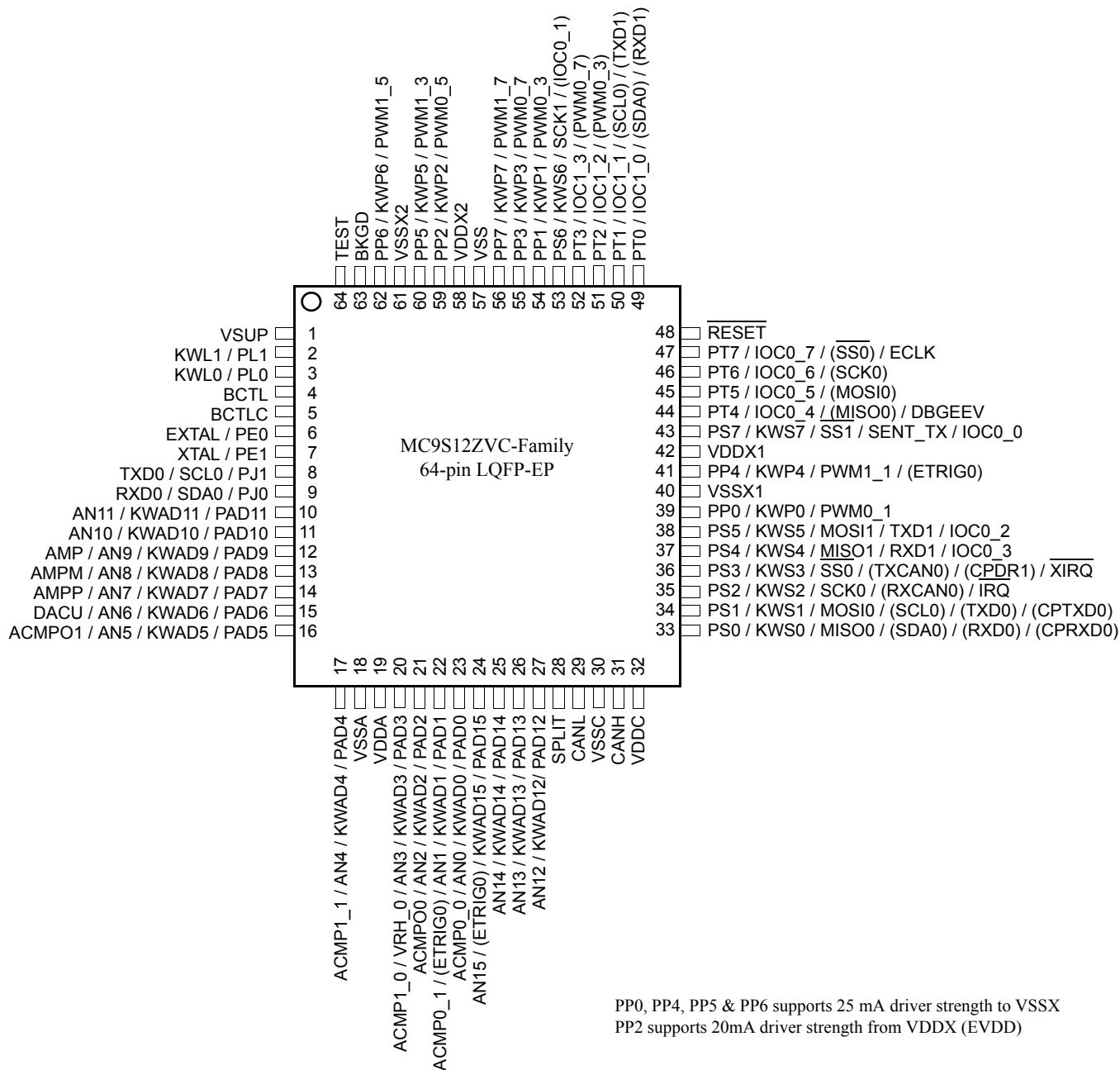
### What is "[Embedded - Microcontrollers](#)"?

"[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	Active
Core Processor	S12Z
Core Size	16-Bit
Speed	32MHz
Connectivity	CANbus, I <sup>2</sup> C, SCI, SPI
Peripherals	DMA, POR, PWM, WDT
Number of I/O	28
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	3.5V ~ 40V
Data Converters	A/D 10x10b; D/A 1x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-LQFP
Supplier Device Package	48-LQFP (7x7)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/s912zvc64f0clfr">https://www.e-xfl.com/product-detail/nxp-semiconductors/s912zvc64f0clfr</a>

Figure 1-4. Pinout MC9S12ZVC-Family 64-pin LQFP-EP<sup>1</sup>

<sup>1</sup> The exposed pad on the package bottom must be connected to a ground pad on the PCB.

Port	Pin	Pin Function & Priority	I/O	Description	Routing Register Bit	Func. after Reset
P	PP7	PWM1_7	O	PWM1 channel 7 (fast)	—	GPIO
		PTP[7]/KWP[7]	I/O	GPIO with pin-interrupt and key-wakeup	—	
	PP6 <sup>6</sup>	PWM1_5	O	PWM1 channel 5 (fast) with over-current interrupt	—	
		PTP[6]/KWP[6]	I/O	GPIO with pin-interrupt, key-wakeup and over-current interrupt	—	
	PP5 <sup>6</sup>	PWM1_3	O	PWM1 channel 3 (fast) with over-current interrupt	—	
		PTP[5]/KWP[5]	I/O	GPIO with pin-interrupt, key-wakeup and over-current interrupt	—	
	PP4 <sup>6</sup>	(ETRIG0)	I	ADC0 external trigger	TRIG0RR1-0	
		PWM1_1	O	PWM1 channel 1 (fast) with over-current interrupt	—	
		PTP[4]/KWP[4]	I/O	GPIO with pin-interrupt, key-wakeup and over-current interrupt	—	
	PP3	PWM0_7	O	PWM0 channel 7	P0C7RR	
		PTP[3]/KWP[3]	I/O	GPIO with pin-interrupt and key-wakeup	—	
	PP2 <sup>5</sup>	PWM0_5	O	PWM0 channel 5 with over-current interrupt	—	
		PTP[2]/KWP[2]/EVDD1	I/O	GPIO with pin-interrupt, key-wakeup and over-current interrupt	—	
	PP1	PWM0_3	O	PWM0 channel 3	P0C3RR	
		PTP[1]/KWP[1]	I/O	GPIO with pin-interrupt and key-wakeup	—	
	PP0 <sup>6</sup>	PWM0_1	O	PWM0 channel 1 with over-current interrupt	—	
		PTP[0]/KWP[0]	I/O	GPIO with pin-interrupt, key-wakeup and over-current interrupt	—	
J	PJ1	TXD0	I/O	SCI0 transmit	SCI0RR	GPIO
		SCL0	I/O	IIC0 serial clock	IIC0RR	
		PTJ[1]	I/O	GPIO	—	
	PJ0	RXD0	I	SCI0 receive	SCI0RR	
		SDA0	I/O	IIC0 serial data	IIC0RR	
		PTJ[0]	I/O	GPIO	—	
L	PL1-0	PTIL[1:0]/KWL[1:0]	I	High-voltage input (HVI) with pin-interrupt and key-wakeup; optional ADC link	—	HVI

<sup>1</sup> Function active when  $\overline{\text{RESET}}$  asserted<sup>2</sup> Input Capture only. Output Compare on PS6.<sup>3</sup> Input Capture routed to alternative signal out of reset (refer to Module Routing Register 3 (MODRR3)). Output Compare unaffected.<sup>4</sup> The interrupt is enabled by clearing the X mask bit in the CPU CCR. The pin is forced to input upon first clearing of the X bit and is held in this state until reset. A stop or wait recovery using XIRQ with the X bit set is not available.<sup>5</sup> High-current capable high-side output with over-current interrupt (EVDD1)<sup>6</sup> High-current capable low-side output with over-current interrupt

## 2.3.4 PIM Generic Register Exceptions

This section lists registers with deviations from the generic description in one or more register bits.

### 2.3.4.1 Port P Over-Current Protection Enable Register (OCPEP)

Address 0x02F9

Access: User read/write<sup>1</sup>

	7	6	5	4	3	2	1	0
R	0	OCPEP6	OCPEP5	OCPEP4	0	OCPEP2	0	OCPEP0
W								
Reset	0	0	0	0	0	0	0	0

Figure 2-21. Over-Current Protection Enable Register (OCPEP)

<sup>1</sup> Read: Anytime  
Write: Anytime

Table 2-21. OCPEP Register Field Descriptions

Field	Description
6 OCPEP6	<b>Over-Current Protection Enable</b> — Activate over-current detector on PP6 Refer to Section 2.5.4, “Over-Current Protection on PP[6-4,0]” 1 PP6 over-current detector enabled 0 PP6 over-current detector disabled
5 OCPEP5	<b>Over-Current Protection Enable</b> — Activate over-current detector on PP5 Refer to Section 2.5.4, “Over-Current Protection on PP[6-4,0]” 1 PP5 over-current detector enabled 0 PP5 over-current detector disabled
4 OCPEP4	<b>Over-Current Protection Enable</b> — Activate over-current detector on PP4 Refer to Section 2.5.4, “Over-Current Protection on PP[6-4,0]” 1 PP4 over-current detector enabled 0 PP4 over-current detector disabled
2 OCPEP2	<b>Over-Current Protection Enable</b> — Activate over-current detector on EVDD1 Refer to Section 2.5.3, “Over-Current Protection on PP2 (EVDD1)” 1 EVDD1 over-current detector enabled 0 EVDD1 over-current detector disabled
0 OCPEP0	<b>Over-Current Protection Enable</b> — Activate over-current detector on PP0 Refer to Section 2.5.4, “Over-Current Protection on PP[6-4,0]” 1 PP0 over-current detector enabled 0 PP0 over-current detector disabled


**Table 5-3. INT Memory Map**

0x000019	Interrupt Request Configuration Data Register 1 (INT_CFDATA1)	R/W
0x00001A	Interrupt Request Configuration Data Register 2 (INT_CFDATA2)	R/W
0x00001B	Interrupt Request Configuration Data Register 3 (INT_CFDATA3)	R/W
0x00001C	Interrupt Request Configuration Data Register 4 (INT_CFDATA4)	R/W
0x00001D	Interrupt Request Configuration Data Register 5 (INT_CFDATA5)	R/W
0x00001E	Interrupt Request Configuration Data Register 6 (INT_CFDATA6)	R/W
0x00001F	Interrupt Request Configuration Data Register 7 (INT_CFDATA7)	R/W

### 5.3.2 Register Descriptions

This section describes in address order all the INT module registers and their individual bits.

Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x000010	IVBR	R	IVB_ADDR[15:8]							
		W								
0x000011		R	IVB_ADDR[7:1]							0
		W								
0x000017	INT_CFADDR	R	0	INT_CFADDR[6:3]				0	0	0
		W								
0x000018	INT_CFDATA0	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x000019	INT_CFDATA1	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00001A	INT_CFDATA2	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00001B	INT_CFDATA3	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00001C	INT_CFDATA4	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								

 = Unimplemented or Reserved

**Figure 5-2. INT Register Summary**

Address	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x00001D	INT_CFDATA5	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00001E	INT_CFDATA6	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								
0x00001F	INT_CFDATA7	R	0	0	0	0	0	PRIOLVL[2:0]		
		W								

= Unimplemented or Reserved

Figure 5-2. INT Register Summary

### 5.3.2.1 Interrupt Vector Base Register (IVBR)

Address: 0x000010

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IVB_ADDR[15:1]															0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Figure 5-3. Interrupt Vector Base Register (IVBR)

Read: Anytime

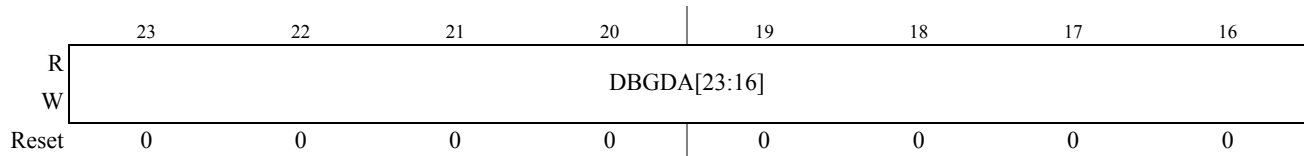
Write: Anytime

Table 5-4. IVBR Field Descriptions

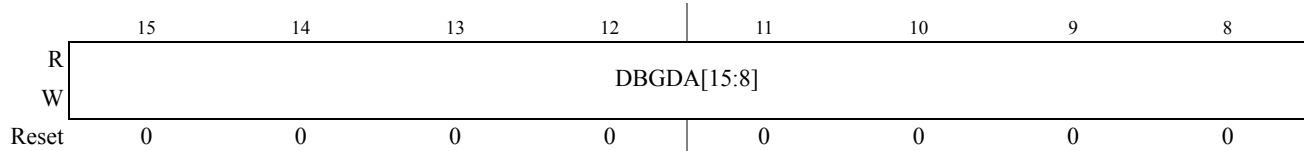
Field	Description
15–1 IVB_ADDR [15:1]	<b>Interrupt Vector Base Address Bits</b> — These bits represent the upper 15 bits of all vector addresses. Out of reset these bits are set to 0xFFFFE (i.e., vectors are located at 0xFFFFE00–0xFFFFFFF). <b>Note:</b> A system reset will initialize the interrupt vector base register with “0xFFFFE” before it is used to determine the reset vector address. Therefore, changing the IVBR has no effect on the location of the reset vector (0xFFFFFC–0xFFFFFFF).

### 6.3.2.15 Debug Comparator D Address Register (DBGDAH, DBGDAM, DBGDAL)

Address: 0x0145, DBGDAH



Address: 0x0146, DBGDAM



Address: 0x0147, DBGDAL

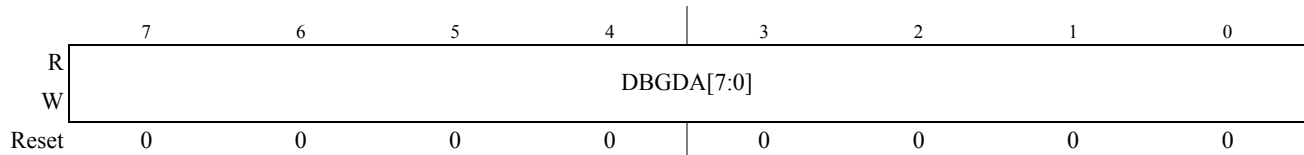


Figure 6-18. Debug Comparator D Address Register

Read: Anytime.

Write: If DBG not armed.

Table 6-26. DBGDAH, DBGDAM, DBGDAL Field Descriptions

Field	Description
23–16 DBGDA [23:16]	<b>Comparator Address Bits [23:16]</b> — These comparator address bits control whether the comparator compares the address bus bits [23:16] to a logic one or logic zero. 0 Compare corresponding address bit to a logic zero 1 Compare corresponding address bit to a logic one
15–0 DBGDA [15:0]	<b>Comparator Address Bits [15:0]</b> — These comparator address bits control whether the comparator compares the address bus bits [15:0] to a logic one or logic zero. 0 Compare corresponding address bit to a logic zero 1 Compare corresponding address bit to a logic one

## 6.4 Functional Description

This section provides a complete functional description of the DBG module.

### 6.4.1 DBG Operation

The DBG module operation is enabled by setting ARM in DBGCR1. When armed it can be used to generate breakpoints to the CPU. The DBG module is made up of comparators, control logic, and the state sequencer, [Figure 6-1](#).

The comparators monitor the bus activity of the CPU. Comparators can be configured to monitor opcode addresses (effectively the PC address) or data accesses. Comparators can be configured during data

## Chapter 7

# ECC Generation Module (SRAM\_ECCV1)

### 7.1 Introduction

The purpose of ECC logic is to detect and correct as much as possible memory data bit errors. These soft errors, mainly generated by alpha radiation, can occur randomly during operation. "Soft error" means that only the information inside the memory cell is corrupt; the memory cell itself is not damaged. A write access with correct data solves the issue. If the ECC algorithm is able to correct the data, then the system can use this corrected data without any issues. If the ECC algorithm is able to detect, but not correct the error, then the system is able to ignore the memory read data to avoid system malfunction.

The ECC value is calculated based on an aligned 2 byte memory data word. The ECC algorithm is able to detect and correct single bit ECC errors. Double bit ECC errors will be detected but the system is not able to correct these errors. This kind of ECC code is called SECDED code. This ECC code requires 6 additional parity bits for each 2 byte data word.

#### 7.1.1 Features

The SRAM\_ECC module provides the ECC logic for the system memory based on a SECDED algorithm. The SRAM\_ECC module includes the following features:

- SECDED ECC code
  - Single bit error detection and correction per 2 byte data word
  - Double bit error detection per 2 byte data word
- Memory initialization function
- Byte wide system memory write access
- Automatic single bit ECC error correction for read and write accesses
- Debug logic to read and write raw use data and ECC values

### 7.2 Memory Map and Register Definition

This section provides a detailed description of all memory and registers for the SRAM\_ECC module.

#### 7.2.1 Register Summary

Figure 7-1 shows the summary of all implemented registers inside the SRAM\_ECC module.

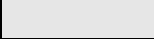


### 9.4.2.3 ADC Status Register (ADCSTS)

It is important to note that if flag DBECC\_ERR is set the ADC ceases operation. In order to make the ADC operational again an ADC Soft-Reset must be issued. An ADC Soft-Reset clears bits CSL\_SEL and RVL\_SEL.

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R	CSL_SEL	RVL_SEL	DBECC_ERR	Reserved	READY	0	0	0
W								
Reset	0	0	0	0	1	0	0	0

 = Unimplemented or Reserved

**Figure 9-6. ADC Status Register (ADCSTS)**

Read: Anytime

Write:

- Bits CSL\_SEL and RVL\_SEL anytime if bit ADC\_EN is clear or bit SMOD\_ACC is set
- Bits DBECC\_ERR and READY not writable

**Table 9-5. ADCSTS Field Descriptions**

Field	Description
7 CSL_SEL	<b>Command Sequence List Select bit</b> — This bit controls and indicates which ADC Command List is active. This bit can only be written if ADC_EN bit is clear. This bit toggles in CSL double buffer mode when no conversion or conversion sequence is ongoing and bit LDOK is set and bit RSTA is set. In CSL single buffer mode this bit is forced to 1'b0 by bit CSL_BMOD. 0 ADC Command List 0 is active. 1 ADC Command List 1 is active.
6 RVL_SEL	<b>Result Value List Select Bit</b> — This bit controls and indicates which ADC Result List is active. This bit can only be written if bit ADC_EN is clear. After storage of the initial Result Value List this bit toggles in RVL double buffer mode whenever the conversion result of the first conversion of the current CSL is stored or a CSL got aborted. In RVL single buffer mode this bit is forced to 1'b0 by bit RVL_BMOD. Please see also <a href="#">Section 9.2.1.2, “MCU Operating Modes</a> for information regarding Result List usage in case of Stop or Wait Mode. 0 ADC Result List 0 is active. 1 ADC Result List 1 is active.
5 DBECC_ERR	<b>Double Bit ECC Error Flag</b> — This flag indicates that a double bit ECC error occurred during conversion command load or result storage and ADC ceases operation. In order to make the ADC operational again an ADC Soft-Reset must be issued. This bit is cleared if bit ADC_EN is clear. 0 No double bit ECC error occurred. 1 A double bit ECC error occurred.
3 READY	<b>Ready For Restart Event Flag</b> — This flag indicates that ADC is in its idle state and ready for a Restart Event. It can be used to verify after exit from Wait Mode if a Restart Event can be issued and processed immediately without any latency time due to an ongoing Sequence Abort Event after exit from MCU Wait Mode (see also the Note in <a href="#">Section 9.2.1.2, “MCU Operating Modes</a> ). 0 ADC not in idle state. 1 ADC is in idle state.

If measured when

- a)  $V_{HBI1}$  selected with  $BVHS = 0$

$$V_{\text{measure}} \geq V_{HBI1\_A} \text{ (rising edge) or } V_{\text{measure}} \geq V_{HBI1\_D} \text{ (falling edge)}$$

or when

- a)  $V_{HBI2}$  selected with  $BVHS = 1$

$$V_{\text{measure}} \geq V_{HBI2\_A} \text{ (rising edge) or } V_{\text{measure}} \geq V_{HBI2\_D} \text{ (falling edge)}$$

then BVHC is set. BVHC status bit indicates that a high voltage at pin VSUP is present. The High Voltage Interrupt flag (BVHIF) is set to 1 when a Voltage High Condition (BVHC) changes state. The Interrupt flag BVHIF can only be cleared by writing a 1. If the interrupt is enabled by bit BVHIE the module requests an interrupt to MCU (BATI).

### 13.3.2.10 PWM Channel Counter Registers (PWMCNTx)

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source. The counter can be read at any time without affecting the count or the operation of the PWM channel. In left aligned output mode, the counter counts from 0 to the value in the period register - 1. In center aligned output mode, the counter counts from 0 up to the value in the period register and then back down to 0.

Any value written to the counter causes the counter to reset to \$00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. The counter is also cleared at the end of the effective period (see [Section 13.4.2.5, “Left Aligned Outputs”](#) and [Section 13.4.2.6, “Center Aligned Outputs”](#) for more details). When the channel is disabled ( $PWMEx = 0$ ), the PWMCNTx register does not count. When a channel becomes enabled ( $PWMEx = 1$ ), the associated PWM counter starts at the count in the PWMCNTx register. For more detailed information on the operation of the counters, see [Section 13.4.2.4, “PWM Timer Counters”](#).

In concatenated mode, writes to the 16-bit counter by using a 16-bit access or writes to either the low or high order byte of the counter will reset the 16-bit counter. Reads of the 16-bit counter must be made by 16-bit access to maintain data coherency.

#### NOTE

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

Module Base + 0x000C = PWMCNT0, 0x000D = PWMCNT1, 0x000E = PWMCNT2, 0x000F = PWMCNT3  
Module Base + 0x0010 = PWMCNT4, 0x0011 = PWMCNT5, 0x0012 = PWMCNT6, 0x0013 = PWMCNT7

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

Figure 13-12. PWM Channel Counter Registers (PWMCNTx)

<sup>1</sup> 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 (any value written causes PWM counter to be reset to \$00).

### 13.3.2.11 PWM Channel Period Registers (PWMPERx)

There is a dedicated period register for each channel. The value in this register determines the period of the associated PWM channel.

The period registers for each channel are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends

Each channel counter can be read at anytime without affecting the count or the operation of the PWM channel.

Any value written to the counter causes the counter to reset to \$00, the counter direction to be set to up, the immediate load of both duty and period registers with values from the buffers, and the output to change according to the polarity bit. When the channel is disabled ( $PWMEx = 0$ ), the counter stops. When a channel becomes enabled ( $PWMEx = 1$ ), the associated PWM counter continues from the count in the PWCNTx register. This allows the waveform to continue where it left off when the channel is re-enabled. When the channel is disabled, writing “0” to the period register will cause the counter to reset on the next selected clock.

#### NOTE

If the user wants to start a new “clean” PWM waveform without any “history” from the old waveform, the user must write to channel counter (PWCNTx) prior to enabling the PWM channel ( $PWMEx = 1$ ).

Generally, writes to the counter are done prior to enabling a channel in order to start from a known state. However, writing a counter can also be done while the PWM channel is enabled (counting). The effect is similar to writing the counter when the channel is disabled, except that the new period is started immediately with the output set according to the polarity bit.

#### NOTE

Writing to the counter while the channel is enabled can cause an irregular PWM cycle to occur.

The counter is cleared at the end of the effective period (see [Section 13.4.2.5, “Left Aligned Outputs”](#) and [Section 13.4.2.6, “Center Aligned Outputs”](#) for more details).

**Table 13-12. PWM Timer Counter Conditions**

Counter Clears (\$00)	Counter Counts	Counter Stops
When PWCNTx register written to any value	When PWM channel is enabled ( $PWMEx = 1$ ). Counts from last value in PWCNTx.	When PWM channel is disabled ( $PWMEx = 0$ )
Effective period ends		

### 13.4.2.5 Left Aligned Outputs

The PWM timer provides the choice of two types of outputs, left aligned or center aligned. They are selected with the CAEx bits in the PWMCAE register. If the CAEx bit is cleared ( $CAEx = 0$ ), the corresponding PWM output will be left aligned.

In left aligned output mode, the 8-bit counter is configured as an up counter only. It compares to two registers, a duty register and a period register as shown in the block diagram in [Figure 13-16](#). When the PWM counter matches the duty register the output flip-flop changes state causing the PWM waveform to also change state. A match between the PWM counter and the period register resets the counter and the output flip-flop, as shown in [Figure 13-16](#), as well as performing a load from the double buffer period and duty register to the associated registers, as described in [Section 13.4.2.3, “PWM Period and Duty”](#). The counter counts from 0 to the value in the period register – 1.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. [Table 14-18](#) summarizes the results of the data bit samples.

**Table 14-18. Data Bit Recovery**

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

**NOTE**

The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit (logic 0).

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. [Table 14-19](#) summarizes the results of the stop bit samples.

**Table 14-19. Stop Bit Recovery**

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

In [Figure 14-24](#), a large burst of noise is perceived as the beginning of a start bit, although the test sample at RT5 is high. The RT5 sample sets the noise flag. Although this is a worst-case misalignment of perceived bit time, the data samples RT8, RT9, and RT10 are within the bit time and data recovery is successful.

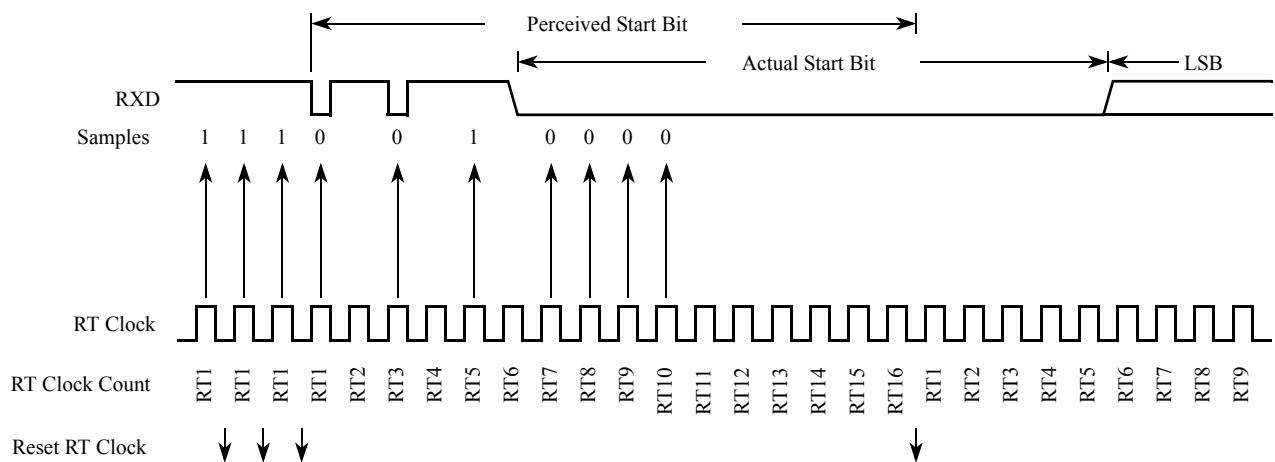


Figure 14-24. Start Bit Search Example 3

[Figure 14-25](#) shows the effect of noise early in the start bit time. Although this noise does not affect proper synchronization with the start bit time, it does set the noise flag.

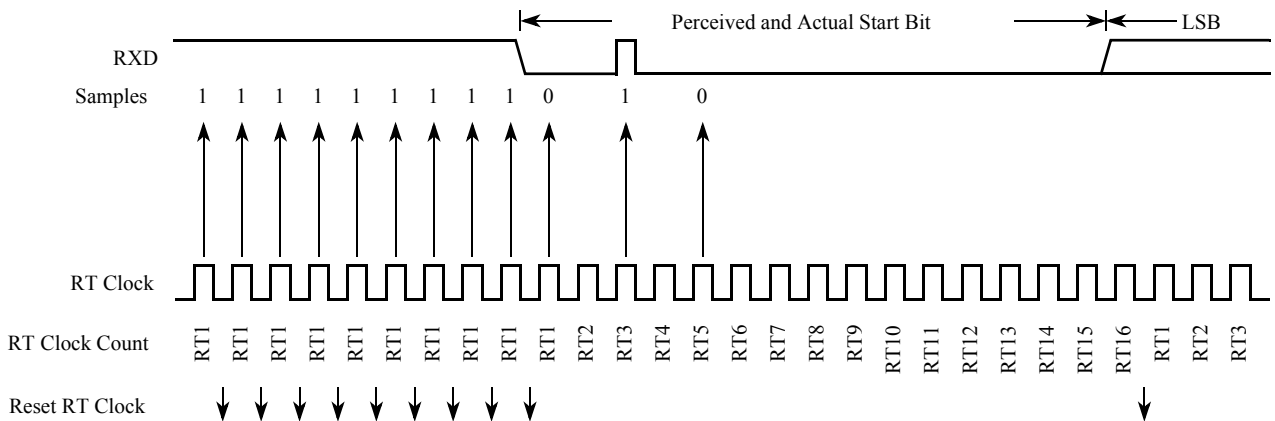
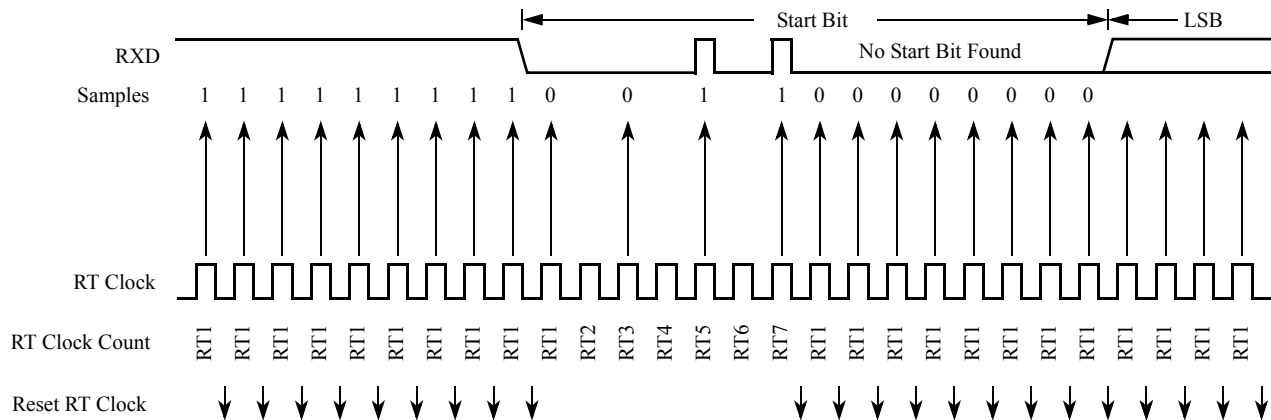


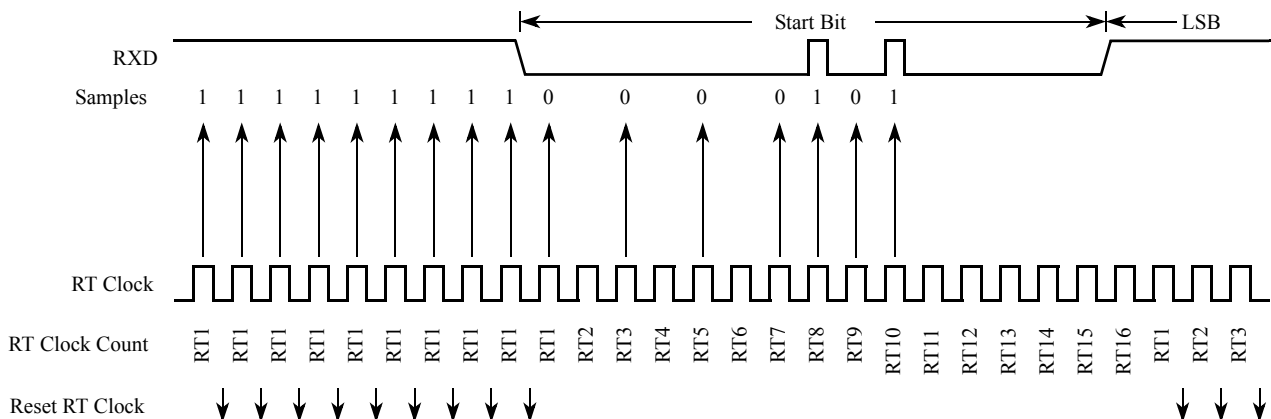
Figure 14-25. Start Bit Search Example 4

Figure 14-26 shows a burst of noise near the beginning of the start bit that resets the RT clock. The sample after the reset is low but is not preceded by three high samples that would qualify as a falling edge. Depending on the timing of the start bit search and on the data, the frame may be missed entirely or it may set the framing error flag.



**Figure 14-26. Start Bit Search Example 5**

In [Figure 14-27](#), a noise burst makes the majority of data samples RT8, RT9, and RT10 high. This sets the noise flag but does not reset the RT clock. In start bits only, the RT8, RT9, and RT10 data samples are ignored.



**Figure 14-27. Start Bit Search Example 6**

#### 14.4.6.4 Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming frame, it sets the framing error flag, FE, in SCI status register 1 (SCISR1). A break character also sets the FE flag because a break character has no stop bit. The FE flag is set at the same time that the RDRF flag is set.

## Chapter 15

# Serial Peripheral Interface (S12SPIV5)

### 15.1 Introduction

The SPI module allows a duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or the SPI operation can be interrupt driven.

#### 15.1.1 Glossary of Terms

SPI	Serial Peripheral Interface
$\overline{SS}$	Slave Select
SCK	Serial Clock
MOSI	Master Output, Slave Input
MISO	Master Input, Slave Output
MOMI	Master Output, Master Input
SISO	Slave Input, Slave Output

#### 15.1.2 Features

The SPI includes these distinctive features:

- Master mode and slave mode
- Selectable 8 or 16-bit transfer width
- Bidirectional mode
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Double-buffered data register
- Serial clock with programmable polarity and phase
- Control of SPI operation during wait mode

#### 15.1.3 Modes of Operation

The SPI functions in three modes: run, wait, and stop.

- Run mode  
This is the basic mode of operation.
- Wait mode



### 15.2.3 $\overline{SS}$ — Slave Select Pin

This pin is used to output the select signal from the SPI module to another peripheral with which a data transfer is to take place when it is configured as a master and it is used as an input to receive the slave select signal when the SPI is configured as slave.

### 15.2.4 SCK — Serial Clock Pin

In master mode, this is the synchronous output clock. In slave mode, this is the synchronous input clock.

## 15.3 Memory Map and Register Definition

This section provides a detailed description of address space and registers used by the SPI.

### 15.3.1 Module Memory Map

The memory map for the SPI is given in [Figure 15-2](#). The address listed for each register is the sum of a base address and an address offset. The base address is defined at the SoC level and the address offset is defined at the module level. Reads from the reserved bits return zeros and writes to the reserved bits have no effect.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 SPICR1	R	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
	W								
0x0001 SPICR2	R	0	XFRW	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
	W								
0x0002 SPIBR	R	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
	W								
0x0003 SPISR	R	SPIF	0	SPTEF	MODF	0	0	0	0
	W								
0x0004 SPIDRH	R	R15	R14	R13	R12	R11	R10	R9	R8
	W	T15	T14	T13	T12	T11	T10	T9	T8
0x0005 SPIDRL	R	R7	R6	R5	R4	R3	R2	R1	R0
	W	T7	T6	T5	T4	T3	T2	T1	T0
0x0006 Reserved	R								
	W								
0x0007 Reserved	R								
	W								


 = Unimplemented or Reserved

Figure 15-2. SPI Register Summary

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must be generated first. The following is an example showing how a STOP signal is generated by a master receiver.

MASR	DEC	RXCNT	;DECREASE THE RXCNT
	BEQ	ENMASR	;LAST BYTE TO BE READ
	MOVB	RXCNT,D1	;CHECK SECOND LAST BYTE
	DEC	D1	;TO BE READ
	BNE	NXMAR	;NOT LAST OR SECOND LAST
LAMAR	BSET	IBCR,#\$08	;SECOND LAST, DISABLE ACK
			;TRANSMITTING
	BRA	NXMAR	
ENMASR	BCLR	IBCR,#\$20	;LAST ONE, GENERATE 'STOP' SIGNAL
NXMAR	MOVB	IBDR,RXBUF	;READ DATA AND STORE
	RTI		

### 16.7.1.5 Generation of Repeated START

At the end of data transfer, if the master continues to want to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

RESTART	BSET	IBCR,#\$04	;ANOTHER START (RESTART)
	MOVB	CALLING,IBDR	;TRANSMIT THE CALLING ADDRESS;D0=R/W

### 16.7.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears the IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred, interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR, for slave transmits, or dummy reading from IBDR, in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

### 16.7.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices which lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL continues to be generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission while the bus is being engaged by another master, the hardware will inhibit the transmission; switch the MS/SL bit from 1 to 0

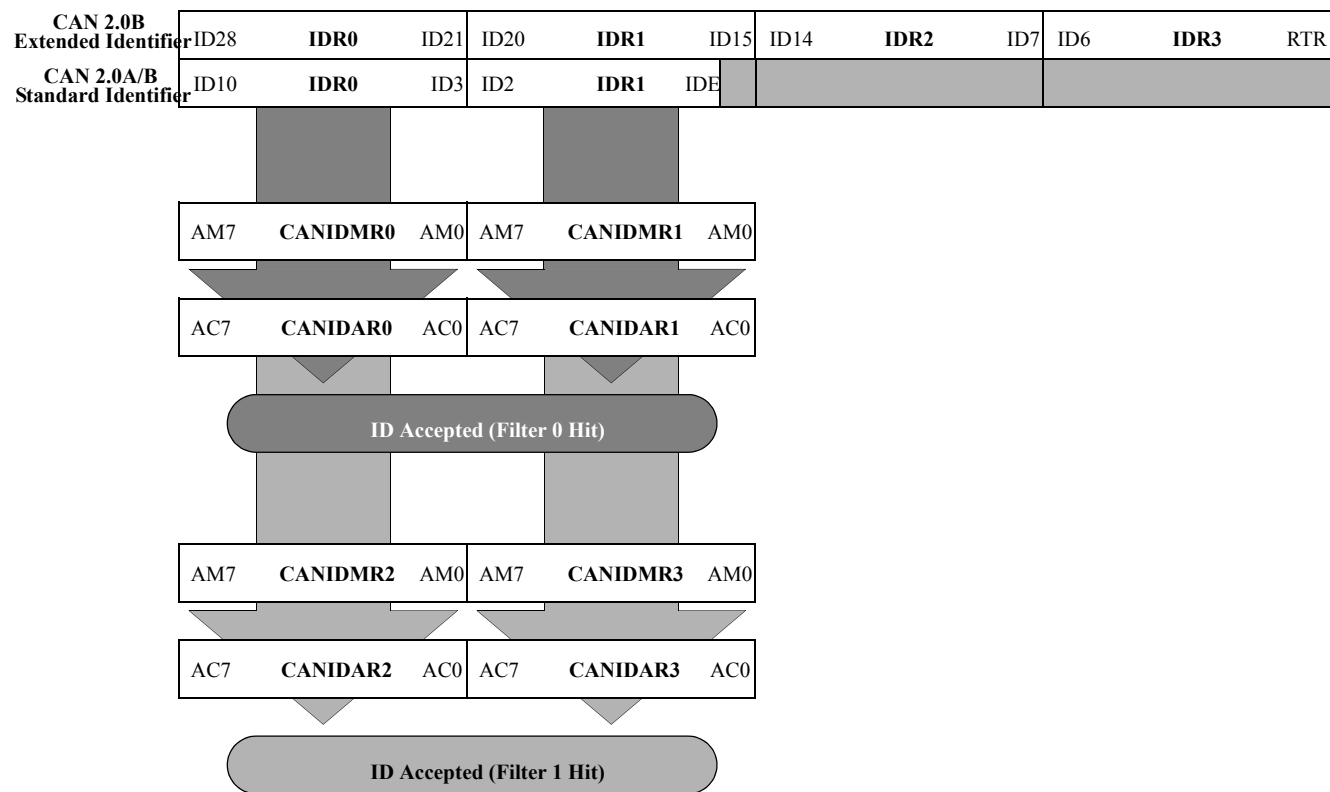


Figure 18-41. 16-bit Maskable Identifier Acceptance Filters

<sup>1</sup> % deviation from target frequency

<sup>2</sup>  $f_{\text{REF}} = 1\text{MHz}$ ,  $f_{\text{BUS}} = 32\text{MHz}$

<sup>2</sup>  $T_A$ : Ambient Temperature

### Input Offset and Hysteresis

