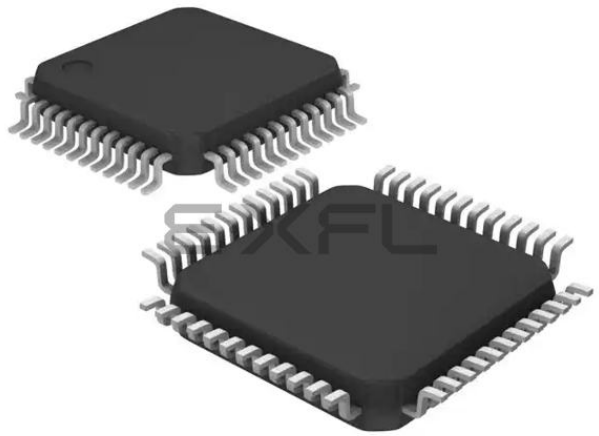


**Welcome to E-XFL.COM**

## 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, I²C, IrDA, LINbus, SCI, SPI, UART/USART |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 34 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 512 x 8 |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 5.5V ~ 18V |
| Data Converters | A/D 10x10b; D/A 1x8b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 48-LQFP |
| Supplier Device Package | 48-LQFP (7x7) |
| Purchase URL | https://www.e-xfl.com/product-detail/nxp-semiconductors/s912zvla12f0vlf |

## Appendix D
## LINPHY Electrical Specifications



## Appendix E
## NVM Electrical Parameters



## Appendix F
## BATS Electrical Specifications



## Appendix G
## DAC_8B5V Electrical Specifications



## Appendix H
## PGA Electrical Specifications



## Appendix I
## ACMP Electrical Specifications



## Appendix J
## PIM Electrical Specifications



## Appendix K
## SPI Electrical Specifications

in system wait mode. For further power consumption the peripherals can individually turn off their local clocks. Asserting $\overline{\text{RESET}}$, $\overline{\text{XIRQ}}$, $\overline{\text{IRQ}}$, or any other interrupt that is not masked, either locally or globally by a CCR bit, ends system wait mode.

- Static power modes:
  Static power (Stop) modes are entered following the CPU STOP instruction unless an NVM command is active. When no NVM commands are active, the Stop request is acknowledged and the device enters either Stop or Pseudo Stop mode.
  - Pseudo-stop: In this mode the system clocks are stopped but the oscillator is still running and the real time interrupt (RTI), watchdog (COP) and Autonomous Periodic Interrupt (API) may be enabled. Other peripherals are turned off. This mode consumes more current than system STOP mode but, as the oscillator continues to run, the full speed wake up time from this mode is significantly shorter.
  - Stop: In this mode the oscillator is stopped and clocks are switched off. The counters and dividers remain frozen. The autonomous periodic interrupt (API) may remain active but has a very low power consumption. If the BDC is enabled in Stop mode, the VREG remains in full performance mode and the CPMU continues operation as in run mode. With BDC enabled and BDCCIS bit set, then all clocks remain active to allow BDC access to internal peripherals. If the BDC is enabled and BDCCIS is clear, then the BDCSI clock remains active, but bus and core clocks are disabled.

## 1.11 Security

The MCU security mechanism prevents unauthorized access to the flash memory. It must be emphasized that part of the security must lie with the application code. An extreme example would be application code that dumps the contents of the internal memory. This would defeat the purpose of security. Also, if an application has the capability of downloading code through a serial port and then executing that code (e.g. an application containing bootloader code), then this capability could potentially be used to read the EEPROM and Flash memory contents even when the microcontroller is in the secure state. In this example, the security of the application could be enhanced by requiring a response authentication before any code can be downloaded.

Device security details are also described in the flash block description.

### 1.11.1 Features

The security features of the S12Z chip family are:

- Prevent external access of the non-volatile memories (Flash, EEPROM) content
- Restrict execution of NVM commands

### 1.11.2 Securing the Microcontroller

The chip can be secured by programming the security bits located in the options/security byte in the Flash memory array. These non-volatile bits keep the device secured through reset and power-down.

| Port | Pin | Pin Function & Priority | I/O | Description | Routing Register Bit | Pin Function after Reset |
|---|---|---|---|---|---|---|
| AD | PAD7 | AMPP (DAC) | I | DAC AMP non-inverting input (+) | — | GPIO |
| | | AN7 | I | ADC0 analog input | — | |
| | | PTADL[7]/ KWADL[7] | I/O | General-purpose; with interrupt and key-wakeup | — | |
| | PAD6 | AMP (DAC) | O | DAC AMP output | — | |
| | | AN6 | I | ADC0 analog input | — | |
| | | PTADL[6]/ KWADL[6] | I/O | General-purpose; with interrupt and key-wakeup | — | |
| | PAD5 | AN5 | I | ADC0 analog input | — | |
| | | (ETRIG0) | I | ADC0 external trigger | TRIG0RR2-0 | |
| | | PTADL[5]/ KWADL[5] | I/O | General-purpose; with interrupt and key-wakeup | — | |
| | PAD4 | ACMP_1 | I | ACMP input 1 (to mux) | — | |
| | | AN4 | I | ADC0 analog input | — | |
| | | PTADL[4]/ KWADL[4] | I/O | General-purpose; with interrupt and key-wakeup | — | |
| | PAD3 | PGA_IN1 | I | PGA input option 1 | — | |
| | | AN3 | I | ADC0 analog input | — | |
| | | PTADL[3]/ KWADL[3] | I/O | General-purpose; with interrupt and key-wakeup | — | |
| | PAD2 | PGA_IN0 | I | PGA input option 0 | — | |
| | | AN2 | I | ADC0 analog input | — | |
| | | PTADL[2]/ KWADL[2] | I/O | General-purpose; with interrupt and key-wakeup | — | |
| | PAD1 | PGA_REF | I | PGA reference input | — | |
| | | VRL | I | ADC0 voltage reference low | — | |
| | | AN1 | I | ADC0 analog input | — | |
| | | PTADL[1]/ KWADL[1] | I/O | General-purpose; with interrupt and key-wakeup | — | |
| | PAD0 | ACMP_0 | I | ACMP input 0 | — | |
| | | VRH | I | ADC0 voltage reference high | — | |
| | | AN0 | I | ADC0 analog input | — | |
| | | PTADL[0]/ KWADL[0] | I/O | General-purpose; with interrupt and key-wakeup | — | |

## 2.3.1 Register Map

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0200 | MODRR0 | R | 0 | 0 | CAN0RR | IIC0RR | SCI1RR | S0L0RR2-0 | | |
| | | W | | | | | | | | |
| 0x0201 | MODRR1 | R | PWM7RR | PWM6RR | PWM5RR | PWM4RR | 0 | PWM2RR | 0 | PWM0RR |
| | | W | | | | | | | | |
| 0x0202 | MODRR2 | R | T1C1RR | T1C0RR | T0C5RR | T0C4RR | T0C3RR | T0C2RR | 0 | 0 |
| | | W | | | | | | | | |
| 0x0203 | MODRR3 | R | 0 | 0 | 0 | 0 | TRIG0RR2 | TRIG0NEG | TRIG0RR1 | TRIG0RR0 |
| | | W | | | | | | | | |
| 0x0204 | MODRR4 | R | 0 | 0 | 0 | 0 | 0 | 0 | T0IC3RR1-0 | |
| | | W | | | | | | | | |
| 0x0206–0x0207 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0208 | ECLKCTL | R | NECLK | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0209 | IRQCR | R | IRQE | IRQEN | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x020A–0x020C | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x020D | Reserved | R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | | W | | | | | | | | |
| 0x020E | Reserved | R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | | W | | | | | | | | |
| 0x020F | Reserved | R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | | W | | | | | | | | |
| 0x0210–0x025F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0260 | PTE | R | 0 | 0 | 0 | 0 | 0 | 0 | PTE1 | PTE0 |
| | | W | | | | | | | | |
| 0x0261 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

Table 7-22 shows the effect for RWE and RW on the comparison conditions. These bits are ignored if INST is set, as matches based on instructions reaching the execution stage are data independent.

**Table 7-22. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 0 | x | 0 | RW not used in comparison |
| 0 | x | 1 | RW not used in comparison |
| 1 | 0 | 0 | Write match |
| 1 | 0 | 1 | No match |
| 1 | 1 | 0 | No match |
| 1 | 1 | 1 | Read match |

### 7.3.2.13 Debug Comparator B Address Register (DBGBAH, DBGBAM, DBGBAL)

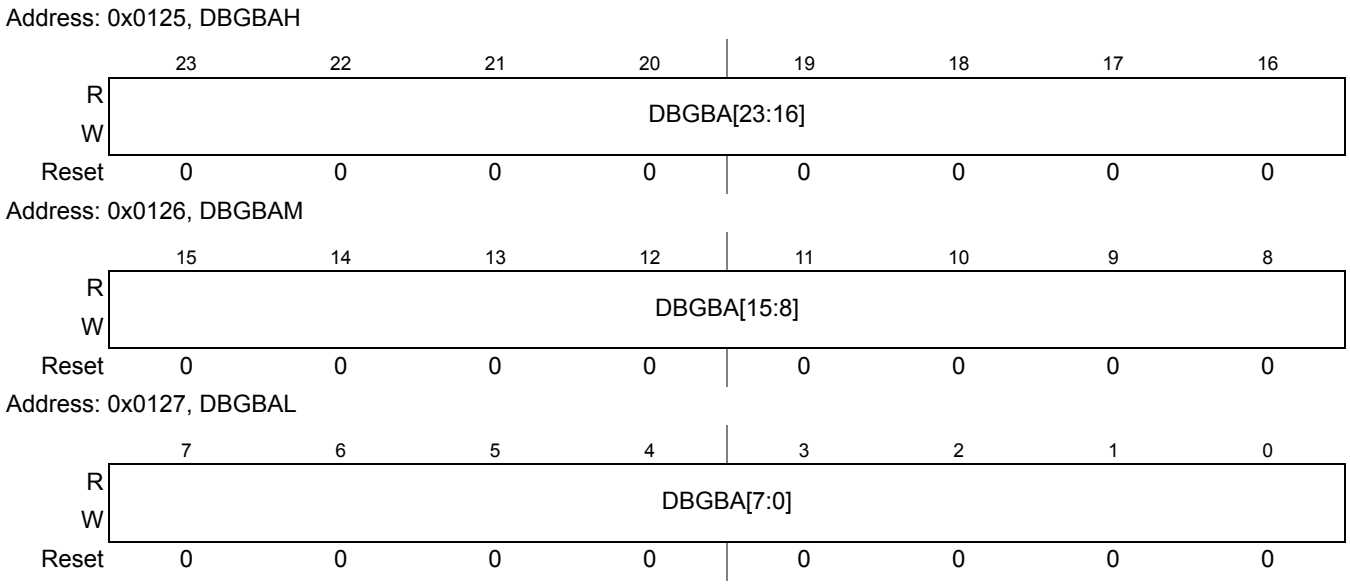



**Figure 7-16. Debug Comparator B Address Register**

Read: Anytime.

Write: If DBG not armed.

**Table 7-23. DBGBAH, DBGBAM, DBGBAL Field Descriptions**

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

### 7.3.2.14 Debug Comparator D Control Register (DBGDCTL)

Address: 0x0140

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | INST | 0 | RW | RWE | reserved | COMPE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 7-17. Debug Comparator D Control Register**

Read: Anytime.

Write: If DBG not armed.

**Table 7-24. DBGDCTL Field Descriptions**

| Field[1] | Description |
|---|---|
| 5<br>INST | **Instruction Select** — This bit configures the comparator to compare PC or data access addresses.<br>0 Comparator compares addresses of data accesses<br>1 Comparator compares PC address |
| 3<br>RW | **Read/Write Comparator Value Bit** — The RW bit controls whether read or write is used in compare for the associated comparator. The RW bit is ignored if RWE is clear or INST is set.<br>0 Write cycle is matched<br>1 Read cycle is matched |
| 2<br>RWE | **Read/Write Enable Bit** — The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is ignored if INST is set.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |
| 0<br>COMPE | **Enable Bit** — Determines if comparator is enabled<br>0 The comparator is not enabled<br>1 The comparator is enabled |

1 If the CDCM field selects range mode comparisons, then DBGCCTL bits configure the comparison, DBGDCTL is ignored.

Table 7-25 shows the effect for RWE and RW on the comparison conditions. These bits are ignored if INST is set, because matches based on opcodes reaching the execution stage are data independent.

**Table 7-25. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 0 | x | 0 | RW not used in comparison |
| 0 | x | 1 | RW not used in comparison |
| 1 | 0 | 0 | Write match |
| 1 | 0 | 1 | No match |
| 1 | 1 | 0 | No match |
| 1 | 1 | 1 | Read match |

| Address Offset | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x000E | RESERVED CPMUTEST1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x000F | CPMU ARMCOP | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x0010 | CPMU HTCTL | R | 0 | 0 | VSEL | 0 | HTE | HTDS | HTIE | HTIF |
| | | W | | | | | | | | |
| 0x0011 | CPMU LVCTL | R | 0 | 0 | 0 | 0 | 0 | LVDS | LVIE | LVIF |
| | | W | | | | | | | | |
| 0x0012 | CPMU APICTL | R | APICLK | 0 | 0 | APIES | APIEA | APIFE | APIE | APIF |
| | | W | | | | | | | | |
| 0x0013 | CPMUACLKTR | R | ACLKTR5 | ACLKTR4 | ACLKTR3 | ACLKTR2 | ACLKTR1 | ACLKTR0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0014 | CPMUAPIRH | R | APIR15 | APIR14 | APIR13 | APIR12 | APIR11 | APIR10 | APIR9 | APIR8 |
| | | W | | | | | | | | |
| 0x0015 | CPMUAPIRL | R | APIR7 | APIR6 | APIR5 | APIR4 | APIR3 | APIR2 | APIR1 | APIR0 |
| | | W | | | | | | | | |
| 0x0016 | RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0017 | CPMUHTTR | R | HTOE | 0 | 0 | 0 | HTTR3 | HTTR2 | HTTR1 | HTTR0 |
| | | W | | | | | | | | |
| 0x0018 | CPMU IRCTRIMH | R | TCTRIM[4:0] | | | | | 0 | IRCTRIM[9:8] | |
| | | W | | | | | | | | |
| 0x0019 | CPMU IRCTRIML | R | IRCTRIM[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x001A | CPMUOSC | R | OSCE | 0 | Reserved | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x001B | CPMUPROT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PROT |
| | | W | | | | | | | | |
| 0x001C | RESERVED CPMUTEST2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x001D | CPMU VREGCTL | R | VREG5VEN | 0 | 0 | 0 | 0 | 0 | EXTXON | INTXON |
| | | W | | | | | | | | |
| 0x001E | CPMUOSC2 | R | 0 | 0 | 0 | 0 | 0 | 0 | OMRE | OSCMOD |
| | | W | | | | | | | | |
| 0x001F | RESERVED | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |

= Unimplemented or Reserved

**Figure 9-3. CPMU Register Summary**

Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTIF | 0 | 0 | LOCKIF | LOCK | 0 | OSCIF | UPOSC |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-10. S12CPMU_UHV Flags Register (CPMUIFLG)**

Read: Anytime

Write: Refer to each bit for individual write conditions

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

| Field | Description |
|---|---|
| 7<br>RTIF | **Real Time Interrupt Flag —** RTIF is set to 1 at the end of the RTI period. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (RTIE=1), RTIF causes an interrupt request.<br>0 RTI time-out has not yet occurred.<br>1 RTI time-out has occurred. |
| 4<br>LOCKIF | **PLL Lock Interrupt Flag** — LOCKIF is set to 1 when LOCK status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LOCKIE=1), LOCKIF causes an interrupt request.<br>0 No change in LOCK bit.<br>1 LOCK bit has changed. |
| 3<br>LOCK | **Lock Status Bit** — LOCK reflects the current state of PLL lock condition. Writes have no effect. While PLL is unlocked (LOCK=0) $f_{PLL}$ is $f_{VCO}$ / 4 to protect the system from high core clock frequencies during the PLL stabilization time $t_{lock}$.<br>0 VCOCLK is not within the desired tolerance of the target frequency.<br>$f_{PLL} = f_{VCO}/4$.<br>1 VCOCLK is within the desired tolerance of the target frequency.<br>$f_{PLL} = f_{VCO}/(POSTDIV+1)$. |
| 1<br>OSCIF | **Oscillator Interrupt Flag** — OSCIF is set to 1 when UPOSC status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (OSCIE=1), OSCIF causes an interrupt request.<br>0 No change in UPOSC bit.<br>1 UPOSC bit has changed. |
| 0<br>UPOSC | **Oscillator Status Bit —** UPOSC reflects the status of the oscillator. Writes have no effect. Entering Full Stop Mode UPOSC is cleared.<br>0 The oscillator is off or oscillation is not qualified by the PLL.<br>1 The oscillator is qualified by the PLL. |

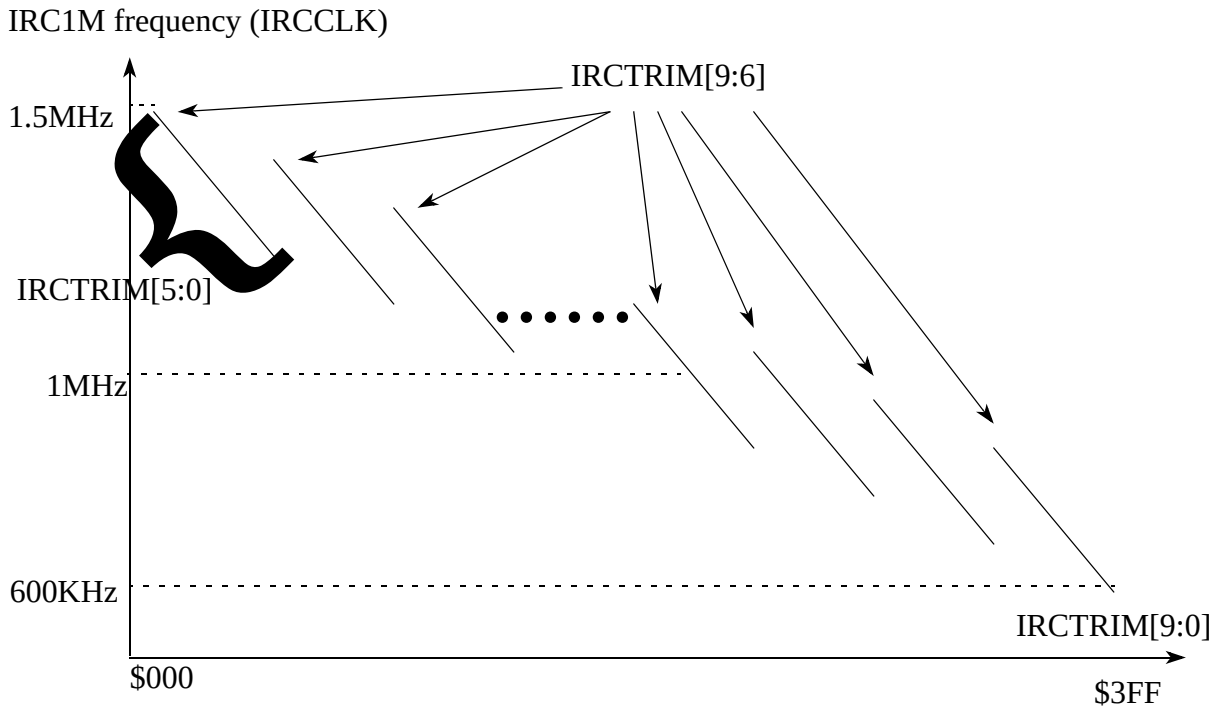



**Figure 9-31. IRC1M Frequency Trimming Diagram**

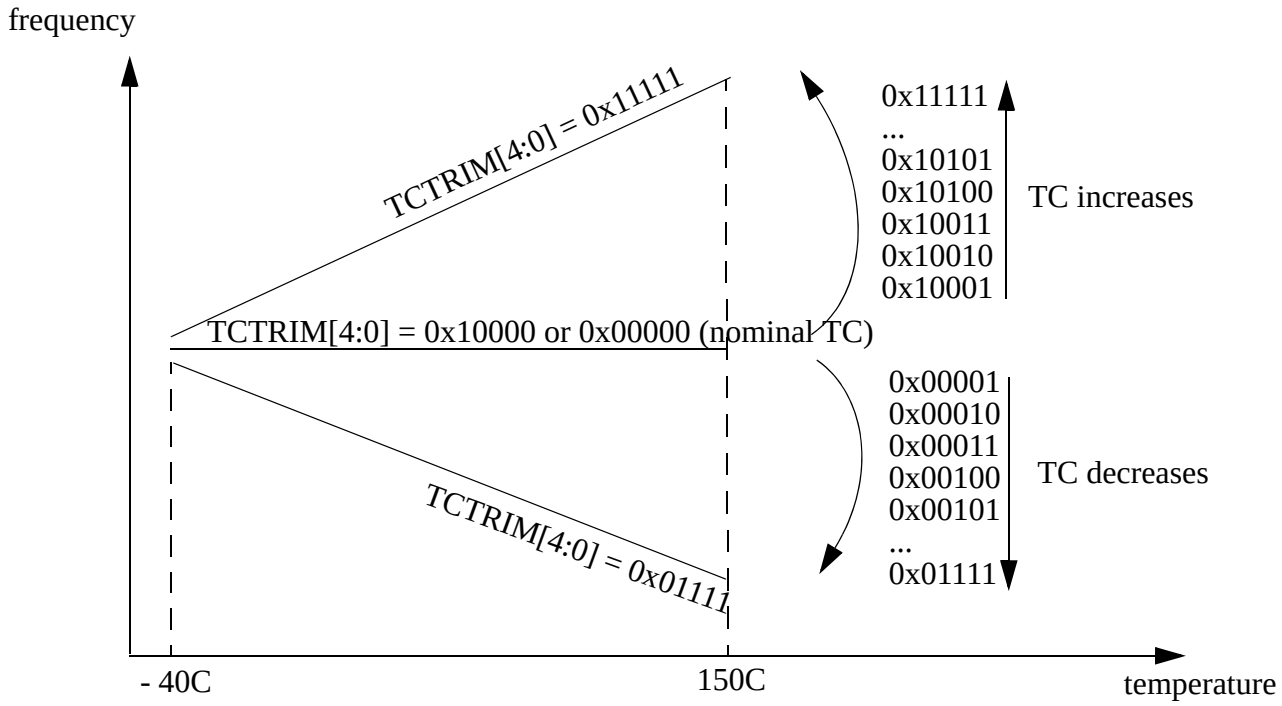



**Figure 9-32. Influence of TCTRIM[4:0] on the Temperature Coefficient**

### NOTE

The frequency is not necessarily linear with the temperature (in most cases it will not be). The above diagram is meant only to give the direction (positive or negative) of the variation of the TC, relative to the nominal TC.

Setting TCTRIM[4:0] at 0x00000 or 0x10000 does not mean that the temperature coefficient will be zero. These two combinations basically switch off the TC compensation module, which results in the nominal TC of the IRC1M.

Upon detection of a status change (UPOSC) the OSCIF flag is set. Going into Full Stop Mode or disabling the oscillator can also cause a status change of UPOSC.

Any change in PLL configuration or any other event which causes the PLL lock status to be cleared leads to a loss of the oscillator status information as well (UPOSC=0).

Oscillator status change interrupts are locally enabled with the OSCIE bit.

**NOTE**

Loosing the oscillator status (UPOSC=0) affects the clock configuration of the system[1]. This needs to be dealt with in application software.

#### 9.6.1.4 Low-Voltage Interrupt (LVI)

In FPM the input voltage VDDA is monitored. Whenever VDDA drops below level $V_{LVIA,}$ the status bit LVDS is set to 1. When VDDA rises above level $V_{LVID}$ the status bit LVDS is cleared to 0. An interrupt, indicated by flag LVIF = 1, is triggered by any change of the status bit LVDS if interrupt enable bit LVIE = 1.

#### 9.6.1.5 HTI - High Temperature Interrupt

In FPM the junction temperature $T_J$ is monitored. Whenever $T_J$ exceeds level $T_{HTIA}$ the status bit HTDS is set to 1. Vice versa, HTDS is reset to 0 when $T_J$ get below level $T_{HTID}$. An interrupt, indicated by flag HTIF = 1, is triggered by any change of the status bit HTDS, if interrupt enable bit HTIE = 1.

#### 9.6.1.6 Autonomous Periodical Interrupt (API)

The API sub-block can generate periodical interrupts independent of the clock source of the MCU. To enable the timer, the bit APIFE needs to be set.

The API timer is either clocked by the Autonomous Clock (ACLK - trimmable internal RC oscillator) or the Bus Clock. Timer operation will freeze when MCU clock source is selected and Bus Clock is turned off. The clock source can be selected with bit APICLK. APICLK can only be written when APIFE is not set.

The APIR[15:0] bits determine the interrupt period. APIR[15:0] can only be written when APIFE is cleared. As soon as APIFE is set, the timer starts running for the period selected by APIR[15:0] bits. When the configured time has elapsed, the flag APIF is set. An interrupt, indicated by flag APIF = 1, is triggered if interrupt enable bit APIE = 1. The timer is re-started automatically again after it has set APIF.

The procedure to change APICLK or APIR[15:0] is first to clear APIFE, then write to APICLK or APIR[15:0], and afterwards set APIFE.

The API Trimming bits ACLKTR[5:0] must be set so the minimum period equals 0.2 ms if stable frequency is desired.

See Table 9-21 for the trimming effect of ACLKTR[5:0].

1. For details please refer to “<st-blue>9.4.6 System Clock Configurations”

### 10.5.2.12 ADC Conversion Interrupt Flag Register (ADCCONIF)

After being set any of these bits can be cleared by writing a value of 1’b1. All bits are cleared if bit ADC_EN is clear or via ADC soft-reset (bit ADC_SR set). Writing any flag with value 1’b0 does not clear the flag. Writing any flag with value 1’b1 does not set the flag.

Module Base + 0x000C

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CON_IF[15:1] | | | | | | | | | | | | | | | EOL_IF |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 10-15. ADC Conversion Interrupt Flag Register (ADCCONIF)**

Read: Anytime

Write: Anytime

**Table 10-17. ADCCONIF Field Descriptions**

| Field | Description |
|---|---|
| 15-1<br>CON_IF[15:1] | **Conversion Interrupt Flags** — These bits could be set by the binary coded interrupt select bits INTFLG_SEL[3:0] when the corresponding conversion command has been processed and related data has been stored to RAM.<br>See also notes below. |
| 0<br>EOL_IF | **End Of List Interrupt Flag** — This bit is set by the binary coded conversion command type select bits CMD_SEL[1:0] for “end of list” type of commands and after such a command has been processed and the related data has been stored RAM.<br>See also second note below |

**NOTE**

These bits can be used to indicate if a certain packet of conversion results is available. Clearing a flag indicates that conversion results have been retrieved by software and the flag can be used again (see also Section 10.9.6, “RVL swapping in RVL double buffer mode and related registers ADCIMDRI and ADCEOLRI.

**NOTE**

Overrun situation of a flag CON_IF[15:1] and EOL_IF are indicated by flag CONIF_OIF.

### 11.3.4 AMPM Input Pin

This analog pin is used as input for the operational amplifier negative input pin, if the according mode is selected, see register bit DACM[2:0].

## 11.4 Memory Map and Register Definition

This sections provides the detailed information of all registers for the DAC_8B5V module.

### 11.4.1 Register Summary

Figure 11-2 shows the summary of all implemented registers inside the DAC_8B5V module.

**NOTE**

Register Address = Module Base Address + Address Offset, where the Module Base Address is defined at the MCU level and the Address Offset is defined at the module level.

| Address Offset / Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 DACCTL | R | FVR | DRIVE | 0 | 0 | 0 | DACM[2:0] | | |
| | W | | | | | | | | |
| 0x0001 Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0002 DACVOL | R | VOLTAGE[7:0] | | | | | | | |
| | W | | | | | | | | |
| 0x0003 - 0x0006 Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | |
| 0x0007 Reserved | R | 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| | W | | | | | | | | |

= Unimplemented

**Figure 11-2. DAC_8B5V Register Summary**

### 11.4.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

**Table 17-10. 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<br>CON67 | Concatenate Channels 6 and 7<br>0 Channels 6 and 7 are separate 8-bit PWMs.<br>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<br>CON45 | Concatenate Channels 4 and 5<br>0 Channels 4 and 5 are separate 8-bit PWMs.<br>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<br>CON23 | Concatenate Channels 2 and 3<br>0 Channels 2 and 3 are separate 8-bit PWMs.<br>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<br>CON01 | Concatenate Channels 0 and 1<br>0 Channels 0 and 1 are separate 8-bit PWMs.<br>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<br>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.<br>0 Allow the clock to the prescaler to continue while in wait mode.<br>1 Stop the input clock to the prescaler whenever the MCU is in wait mode. |
| 2<br>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.<br>0 Allow PWM to continue while in freeze mode.<br>1 Disable PWM input clock to the prescaler whenever the part is in freeze mode. This is useful for emulation. |

### 17.3.2.7 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.

## 17.4.2 PWM Channel Timers

The main part of the PWM module are the actual timers. Each of the timer channels has a counter, a period register and a duty register (each are 8-bit). The waveform output period is controlled by a match between the period register and the value in the counter. The duty is controlled by a match between the duty register and the counter value and causes the state of the output to change during the period. The starting polarity of the output is also selectable on a per channel basis. Shown below in Figure 17-16 is the block diagram for the PWM timer.

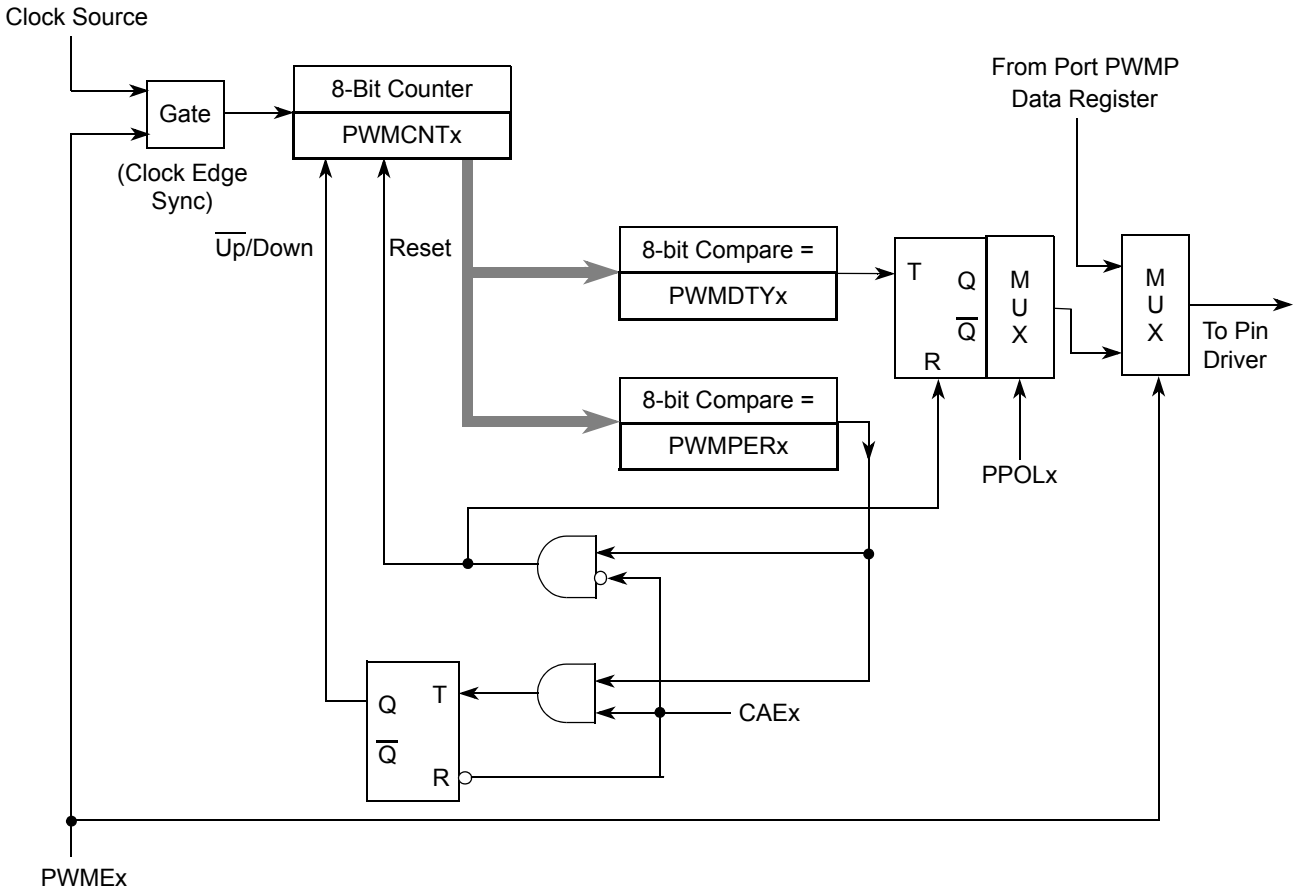



**Figure 17-16. PWM Timer Channel Block Diagram**

### 17.4.2.1 PWM Enable

Each PWM channel has an enable bit (PWMEx) to start its waveform output. When any of the PWMEx bits are set (PWMEx = 1), the associated PWM output signal is enabled immediately. However, the actual PWM waveform is not available on the associated PWM output until its clock source begins its next cycle due to the synchronization of PWMEx and the clock source. An exception to this is when channels are concatenated. Refer to Section 17.4.2.7, “PWM 16-Bit Functions” for more detail.

**NOTE**

The first PWM cycle after enabling the channel can be irregular.

Clock Source = bus clock, where bus clock= 10 MHz (100 ns period)

PPOLx = 0

PWMPERx = 4

PWMDTYx = 1

PWMx Frequency = 10 MHz/8 = 1.25 MHz

PWMx Period = 800 ns

PWMx Duty Cycle = 3/4 *100% = 75%

Shown in Figure 17-20 is the output waveform generated.

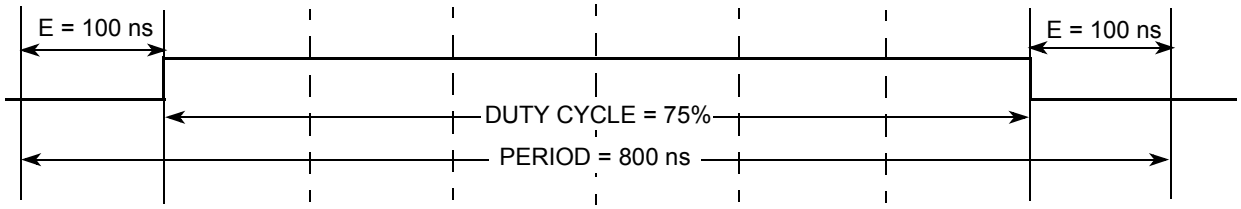



**Figure 17-20. PWM Center Aligned Output Example Waveform**

### 17.4.2.7 PWM 16-Bit Functions

The scalable PWM timer also has the option of generating up to 8-channels of 8-bits or 4-channels of 16-bits for greater PWM resolution. This 16-bit channel option is achieved through the concatenation of two 8-bit channels.

The PWMCTL register contains four control bits, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. Channels 6 and 7 are concatenated with the CON67 bit, channels 4 and 5 are concatenated with the CON45 bit, channels 2 and 3 are concatenated with the CON23 bit, and channels 0 and 1 are concatenated with the CON01 bit.

**NOTE**

Change these bits only when both corresponding channels are disabled.

When channels 6 and 7 are concatenated, channel 6 registers become the high order bytes of the double byte channel, as shown in Figure 17-21. Similarly, when channels 4 and 5 are concatenated, channel 4 registers become the high order bytes of the double byte channel. When channels 2 and 3 are concatenated, channel 2 registers become the high order bytes of the double byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high order bytes of the double byte channel.

When using the 16-bit concatenated mode, the clock source is determined by the low order 8-bit channel clock select control bits. That is channel 7 when channels 6 and 7 are concatenated, channel 5 when channels 4 and 5 are concatenated, channel 3 when channels 2 and 3 are concatenated, and channel 1 when channels 0 and 1 are concatenated. The resulting PWM is output to the pins of the corresponding low order 8-bit channel as also shown in Figure 17-21. The polarity of the resulting PWM output is controlled by the PPOLx bit of the corresponding low order 8-bit channel as well.

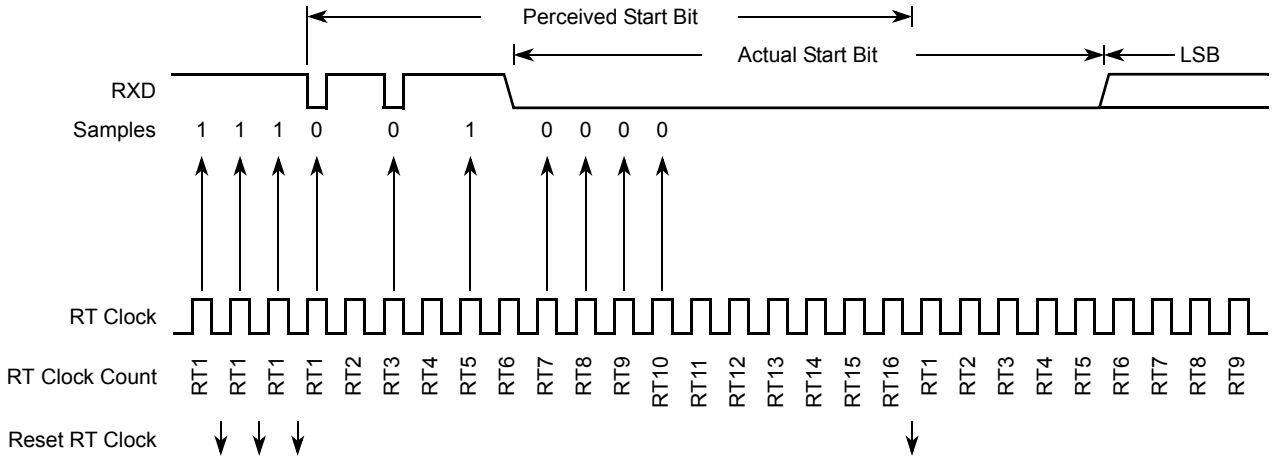



**Figure 18-24. Start Bit Search Example 3**

Figure 18-25 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.

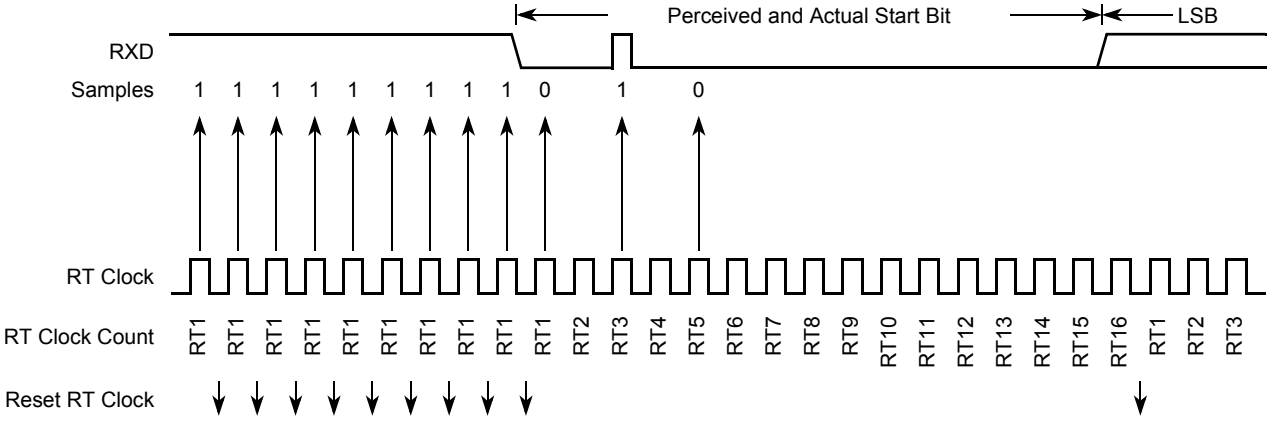



**Figure 18-25. Start Bit Search Example 4**

Figure 18-26 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.

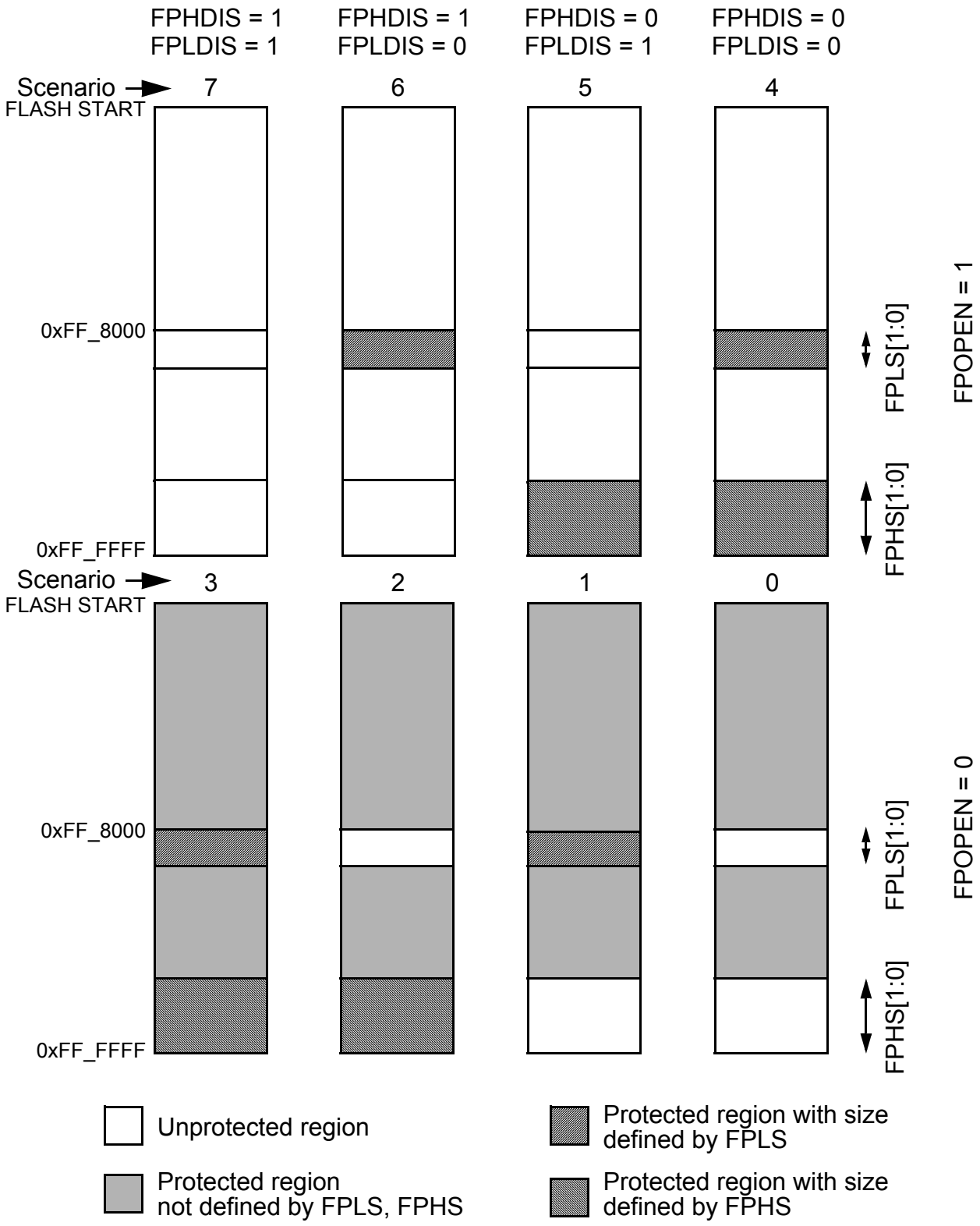
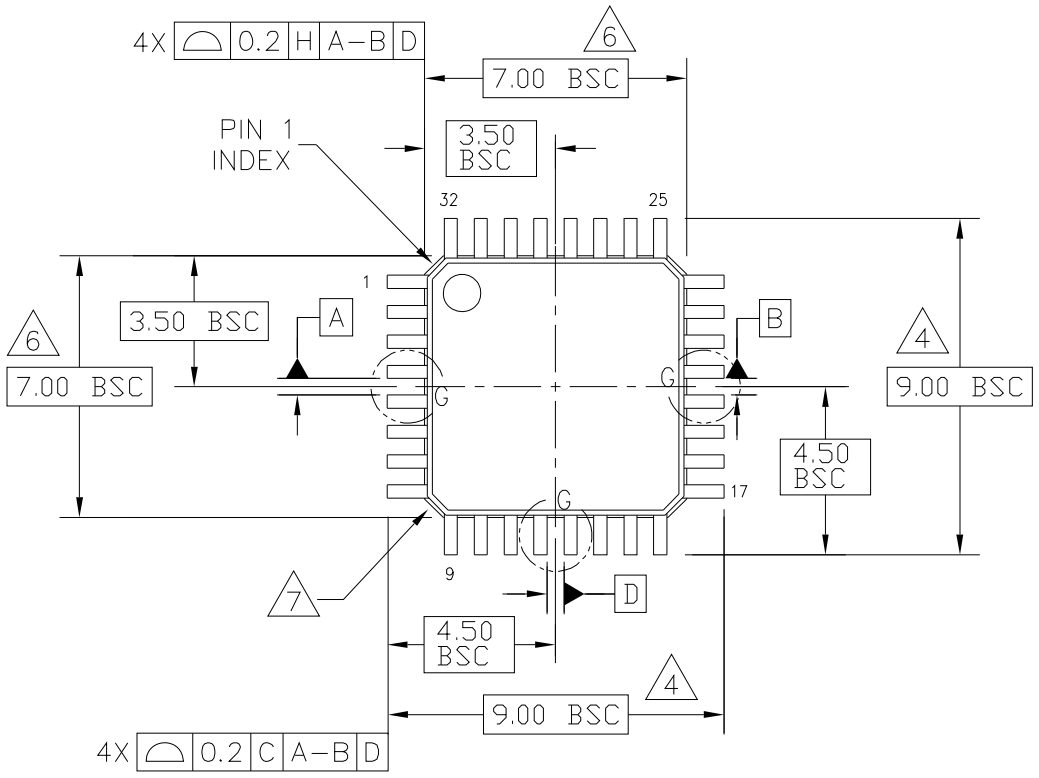
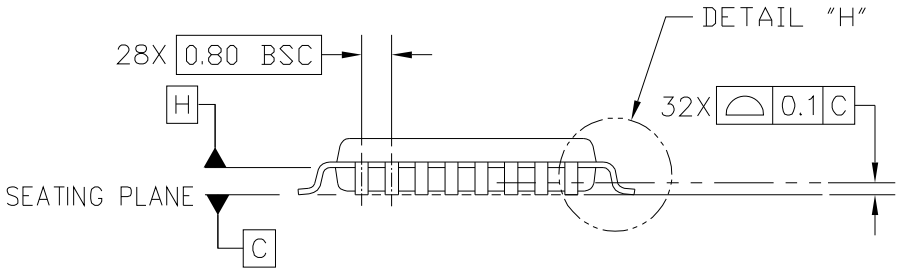



**Figure 22-14. P-Flash Protection Scenarios**




| © FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED. | MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|---|
| TITLE: LOW PROFILE QUAD FLAT PACK (LQFP) 32 LEAD, 0.8 PITCH (7 X 7 X 1.4) | DOCUMENT NO: 98ASH70029A | REV: D |
| | CASE NUMBER: 873A-03 | 19 MAY 2005 |
| | STANDARD: JEDEC MS-026 BBA | |

## O.11 0x05C0-0x05FF TIM0 (continued)

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x05D7 | TIM0TC3L | R<br>W | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x05D8 | TIM0TC4H | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x05D9 | TIM0TC4L | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x05DA | TIM0TC5H | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x05DB | TIM0TC5L | R<br>W | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| 0x05DC–<br>0x05EB | Reserved | R<br>W | | | | | | | | |
| 0x05EC | TIM0OCPD | R<br>W | RESERVED | RESERVED | OCPD5 | OCPD4 | OCPD3 | OCPD2 | OCPD1 | OCPD0 |
| 0x05ED | Reserved | R<br>W | | | | | | | | |
| 0x05EE | TIM0PTPSR | R<br>W | PTPS7 | PTPS6 | PTPS5 | PTPS4 | PTPS3 | PTPS2 | PTPS1 | PTPS0 |
| 0x05EF | Reserved | R<br>W | | | | | | | | |

## O.12 0x0600-0x063F ADC0

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0600 | ADC0CTL_0 | R<br>W | ADC_EN | ADC_SR | FRZ_MOD | SWAI | ACC_CFG[1:0] | | STR_SEQA | MOD_CFG |
| 0x0601 | ADC0CTL_1 | R<br>W | CSL_BMOD | RVL_BMOD | SMOD_ACC | AUT_RSTA | 0 | 0 | 0 | 0 |
| 0x0602 | ADC0STS | R<br>W | CSL_SEL | RVL_SEL | DBECC_ERR | Reserved | READY | 0 | 0 | 0 |
| 0x0603 | ADC0TIM | R<br>W | 0 | PRS[6:0] | | | | | | |
| 0x0604 | ADC0FMT | R<br>W | DJM | 0 | 0 | 0 | 0 | SRES[2:0] | | |
| 0x0605 | ADC0FLWCTL | R<br>W | SEQA | TRIG | RSTA | LDOK | 0 | 0 | 0 | 0 |
| 0x0606 | ADC0EIE | R<br>W | IA_EIE | CMD_EIE | EOL_EIE | Reserved | TRIG_EIE | RSTAR_EIE | LDOK_EIE | 0 |
| 0x0607 | ADC0IE | R<br>W | SEQAD_IE | CONIF_OIE | Reserved | 0 | 0 | 0 | 0 | 0 |