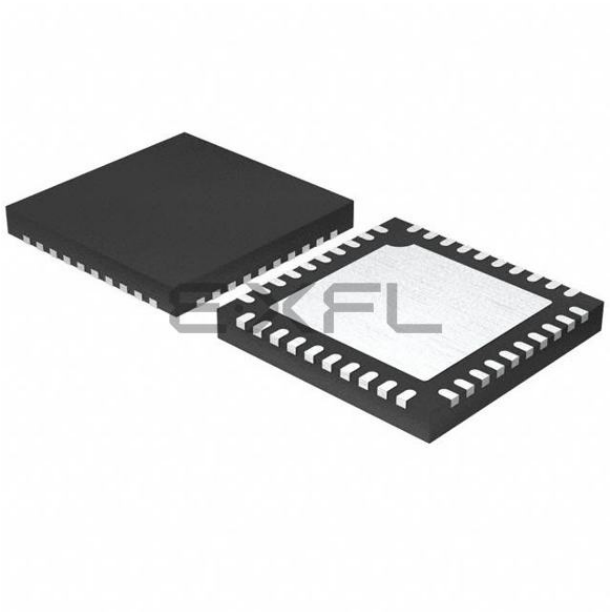

Welcome to E-XFL.COM

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	8051
Core Size	8-Bit
Speed	25MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	DMA, LCD, POR, PWM, WDT
Number of I/O	34
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8.25K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.8V
Data Converters	A/D 16x10b/12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	40-VFQFN Exposed Pad
Supplier Device Package	40-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f961-b-gm

Table 3.1. Pin Definitions for the C8051F96x (Continued)

Name	Pin Numbers			Type	Description
	DQFN76	TQFP80	QFN40		
P5.2 LCD18	B23	63		D I/O or A In A O	Port 5.2. See Port I/O Section for a complete description. LCD Segment Pin 18
P5.3 LCD19	B22	62		D I/O or A In A O	Port 5.3. See Port I/O Section for a complete description. LCD Segment Pin 19
P5.4 LCD20	D4	59		D I/O or A In A O	Port 5.4. See Port I/O Section for a complete description. LCD Segment Pin 20
P5.5 LCD21	B21	55		D I/O or A In A O	Port 5.5. See Port I/O Section for a complete description. LCD Segment Pin 21
P5.6 LCD22	B15	44		D I/O or A In A O	Port 5.6. See Port I/O Section for a complete description. LCD Segment Pin 22
P5.7 LCD23	D3	42		D I/O or A In A O	Port 5.7. See Port I/O Section for a complete description. LCD Segment Pin 23
P6.0 LCD24	B14	40		D I/O or A In A O	Port 6.0. See Port I/O Section for a complete description. LCD Segment Pin 24
P6.1 LCD25	B13	37		D I/O or A In A O	Port 6.1. See Port I/O Section for a complete description. LCD Segment Pin 25

3.1.3. Soldering Guidelines

3.1.3.1. Solder Mask Design

All metal pads are to be non-solder mask defined (NSMD). Clearance between the solder mask and the metal pad is to be 60 μm minimum, all the way around the pad.

3.1.3.2. Stencil Design

1. A stainless steel, laser-cut and electro-polished stencil with trapezoidal walls should be used to assure good solder paste release.
2. The stencil thickness should be 0.125 mm (5 mils).
3. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pads.
4. A 2x2 array of 1.25 mm square openings on 1.60 mm pitch should be used for the center ground pad.

3.1.3.3. Card Assembly

1. A No-Clean, Type-3 solder paste is recommended.
2. The recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

3.1.3.4. Inner via placement

1. Inner via placement per Figure 3.6.
2. Reccomended via hole size is 0.150 mm (6 mil) laser drilled holes.

7.3. Comparator Response Time

Comparator response time may be configured in software via the CPTnMD registers described on “CPT0MD: Comparator 0 Mode Selection” on page 109 and “CPT1MD: Comparator 1 Mode Selection” on page 111. Four response time settings are available: Mode 0 (Fastest Response Time), Mode 1, Mode 2, and Mode 3 (Lowest Power). Selecting a longer response time reduces the Comparator active supply current. The Comparators also have low power shutdown state, which is entered any time the comparator is disabled. Comparator rising edge and falling edge response times are typically not equal. See Table 4.16 on page 74 for complete comparator timing and supply current specifications.

7.4. Comparator Hysterisis

The Comparators feature software-programmable hysterisis that can be used to stabilize the comparator output while a transition is occurring on the input. Using the CPTnCN registers, the user can program both the amount of hysteresis voltage (referred to the input voltage) and the positive and negative-going symmetry of this hysteresis around the threshold voltage (i.e., the comparator negative input).

Figure 7.3 shows that when positive hysterisis is enabled, the comparator output does not transition from logic 0 to logic 1 until the comparator positive input voltage has exceeded the threshold voltage by an amount equal to the programmed hysterisis. It also shows that when negative hysterisis is enabled, the comparator output does not transition from logic 1 to logic 0 until the comparator positive input voltage has fallen below the threshold voltage by an amount equal to the programmed hysterisis.

The amount of positive hysterisis is determined by the settings of the CPnHYP bits in the CPTnCN register and the amount of negative hysteresis voltage is determined by the settings of the CPnHYN bits in the same register. Settings of 20 mV, 10 mV, 5 mV, or 0 mV can be programmed for both positive and negative hysterisis. See Section “Table 4.16. Comparator Electrical Characteristics” on page 74 for complete comparator hysterisis specifications.

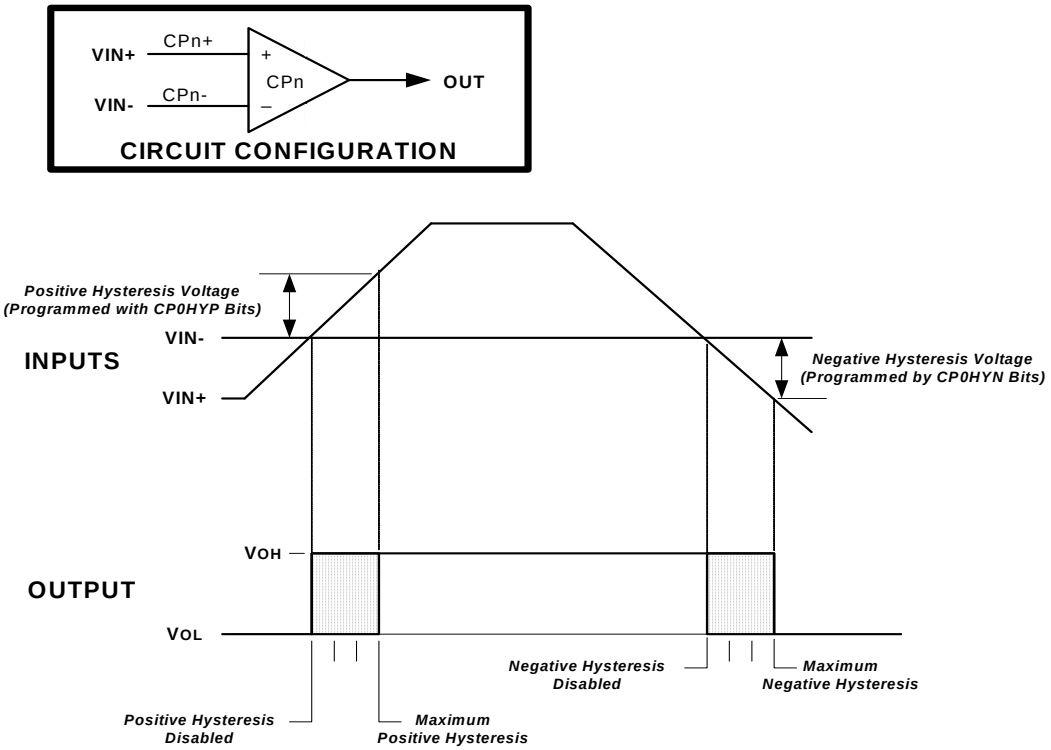

Figure 7.3. Comparator Hysteresis Plot

Table 8.1. CIP-51 Instruction Set Summary

Mnemonic	Description	Bytes	Clock Cycles
Arithmetic Operations			
ADD A, Rn	Add register to A	1	1
ADD A, direct	Add direct byte to A	2	2
ADD A, @Ri	Add indirect RAM to A	1	2
ADD A, #data	Add immediate to A	2	2
ADDC A, Rn	Add register to A with carry	1	1
ADDC A, direct	Add direct byte to A with carry	2	2
ADDC A, @Ri	Add indirect RAM to A with carry	1	2
ADDC A, #data	Add immediate to A with carry	2	2
SUBB A, Rn	Subtract register from A with borrow	1	1
SUBB A, direct	Subtract direct byte from A with borrow	2	2
SUBB A, @Ri	Subtract indirect RAM from A with borrow	1	2
SUBB A, #data	Subtract immediate from A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	2
INC @Ri	Increment indirect RAM	1	2
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	2
DEC @Ri	Decrement indirect RAM	1	2
INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1
Logical Operations			
ANL A, Rn	AND Register to A	1	1
ANL A, direct	AND direct byte to A	2	2
ANL A, @Ri	AND indirect RAM to A	1	2
ANL A, #data	AND immediate to A	2	2
ANL direct, A	AND A to direct byte	2	2
ANL direct, #data	AND immediate to direct byte	3	3
ORL A, Rn	OR Register to A	1	1
ORL A, direct	OR direct byte to A	2	2
ORL A, @Ri	OR indirect RAM to A	1	2
ORL A, #data	OR immediate to A	2	2
ORL direct, A	OR A to direct byte	2	2
ORL direct, #data	OR immediate to direct byte	3	3
XRL A, Rn	Exclusive-OR Register to A	1	1
XRL A, direct	Exclusive-OR direct byte to A	2	2
XRL A, @Ri	Exclusive-OR indirect RAM to A	1	2
XRL A, #data	Exclusive-OR immediate to A	2	2
XRL direct, A	Exclusive-OR A to direct byte	2	2
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3

SFR Definition 11.3. DMA0MINT: DMA0 Mid-Point Interrupt

Bit	7	6	5	4	3	2	1	0
Name		CH6_MINT	CH5_MINT	CH4_MINT	CH3_MINT	CH2_MINT	CH1_MINT	CH0_MINT
Type	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x2; SFR Address = 0xD4

Bit	Name	Function
7	Unused	Read = 0b, Write = Don't Care
6	CH6_MINT	**Channel 6 Mid-Point Interrupt Flag.** 0: Mid-Point interrupt has not occured on channel 6. 1: Mid-Point interrupt has not occured on channel 6.
5	CH5_MINT	**Channel 5 Mid-Point Interrupt Flag.** 0: Mid-Point interrupt has not occured on channel 5. 1: Mid-Point interrupt has not occured on channel 5.
4	CH4_MINT	**Channel 4 Mid-Point Interrupt Flag.** 0: Mid-Point interrupt has not occured on channel 4. 1: Mid-Point interrupt has not occured on channel 4**.**
3	CH3_MINT	**Channel 3 Mid-Point Interrupt Flag.** 0: Mid-Point interrupt has not occured on channel 3. 1: Mid-Point interrupt has not occured on channel 3.
2	CH2_MINT	**Channel 2 Mid-Point Interrupt Flag.** 0: Mid-Point interrupt has not occured on channel 2. 1: Mid-Point interrupt has not occured on channel 2.
1	CH1_MINT	**Channel 1 Mid-Point Interrupt Flag.** 0: Mid-Point interrupt has not occured on channel 1. 1: Mid-Point interrupt has not occured on channel 1.
0	CH0_MINT	**Channel 0 Mid-Point Interrupt Flag.** 0: Mid-Point interrupt has not occured on channel 0. 1: Mid-Point interrupt has not occured on channel 0.

Note: Mid-point Interrupt flag is set when the offset address DMA0NAOH/L equals to half of data transfer size DMA0NSZH/L if the transfer size is an even number or half of data transfer size DMA0NSZH/L plus one if the transfer size is an odd number. This flag must be cleared by software or system reset.The mid-point interrupt is enabled by setting bit 6 of DMA0NCF with DMA0SEL configured for the corresponding channel.

12.2. 32-bit CRC Algorithm

The C8051F41x CRC unit calculates the 32-bit CRC using a poly of 0x04C11DB7. The CRC-32 algorithm is "reflected", meaning that all of the input bytes and the final 32-bit output are bit-reversed in the processing engine. The following is a description of a simplified CRC algorithm that produces results identical to the hardware:

Step 1. XOR the least-significant byte of the current CRC result with the input byte. If this is the first iteration of the CRC unit, the current CRC result will be the set initial value (0x00000000 or 0xFFFFFFFF).
Step 2. Right-shift the CRC result.
Step 3. If the LSB of the CRC result is set, XOR the CRC result with the reflected polynomial (0xEDB88320).
Step 4. Repeat at Step 2 for the number of input bits (8).

For example, the 32-bit 'F41x CRC algorithm can be described by the following code:

```
unsigned long UpdateCRC (unsigned long CRC_acc, unsigned char CRC_input)
{
   unsigned char i; // loop counter
   #define POLY 0xEDB88320 // bit-reversed version of the poly 0x04C11DB7
   // Create the CRC "dividend" for polynomial arithmetic (binary arithmetic
   // with no carries)

   CRC_acc = CRC_acc ^ CRC_input;

   // "Divide" the poly into the dividend using CRC XOR subtraction
   // CRC_acc holds the "remainder" of each divide
   //
   // Only complete this division for 8 bits since input is 1 byte
   for (i = 0; i < 8; i++)
   {
      // Check if the MSB is set (if MSB is 1, then the POLY can "divide"
      // into the "dividend")
      if ((CRC_acc & 0x00000001) == 0x00000001)
      {
         // if so, shift the CRC value, and XOR "subtract" the poly
         CRC_acc = CRC_acc >> 1;
         CRC_acc ^= POLY;
      }
      else
      {
         // if not, just shift the CRC value
         CRC_acc = CRC_acc >> 1;
      }
   }
   // Return the final remainder (CRC value)
   return CRC_acc;
}
```

The following table lists several input values and the associated outputs using the 32-bit 'F41x CRC algorithm (an initial value of 0xFFFFFFFF is used):

14.1. Hardware Description

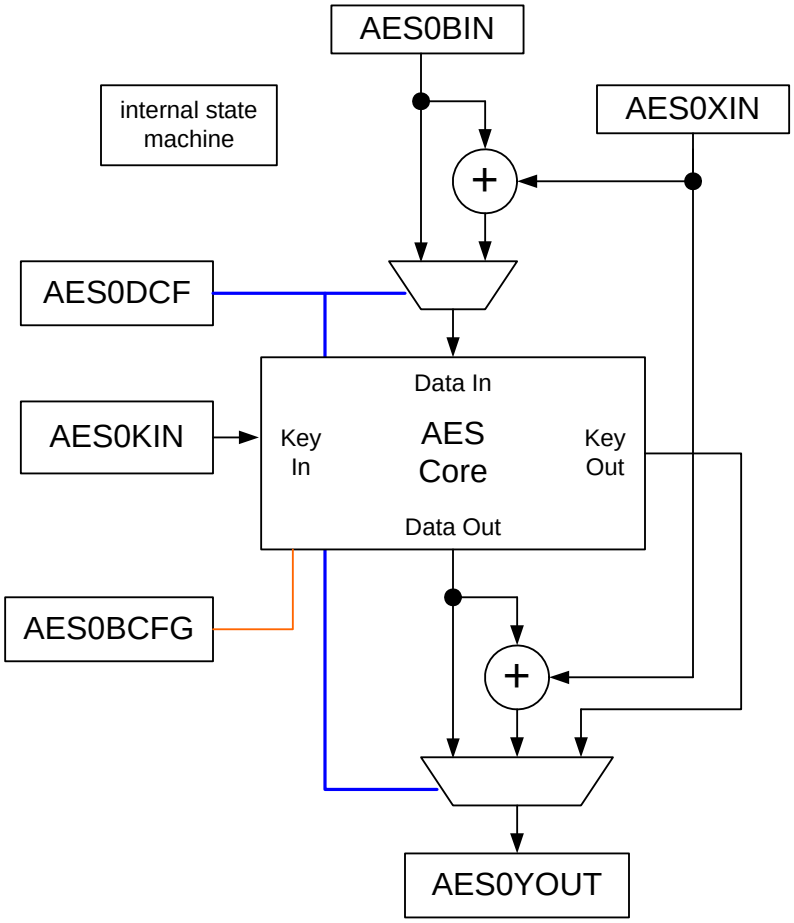

Figure 14.1. AES Peripheral Block Diagram

The AES Encryption module consists of these elements.

- AES Encryption/Decryption Core
- Configuration sfrs
- Key input sfr
- Data sfrs
- Input Multiplexer
- Output Multiplexer
- Input Exclusive OR block
- Output Exclusive OR block
- Internal State Machine

14.6.2. CBC Encryption Initialization Vector Location

The first block to be encrypted uses the initialization vector for the AES0XIN data. Subsequent blocks will use the encrypted ciphertext from the previous block. The DMA is capable of encrypting multiple blocks. If the initialization is located at an arbitrary location in xram, the DMA base address location will need to be changed to the start of the encrypted ciphertext after encrypting the first block. However, if the initialization vector explicitly located in xram immediately before the encrypted ciphertext, the pointer will be advanced to the start of the encrypted ciphertext naturally and multiple blocks can be encrypted autonomously.

14.6.3. CBC Encryption using DMA

Normally, the AES block is used with the DMA. This provides the best performance and lowest power consumption. Code examples are provided in 8051 compiler independent C code using the DMA. It is highly recommended to use with the code examples. The steps are documented in the datasheet for completeness.

- Prepare encryption Key, initialization vector, and data to be encrypted in xram.

(The initialization vector should be located immediately before the data to be encrypted to encrypt multiple blocks.)

- Reset AES module by clearing bit 2 of AES0BCFG.
- Disable the first four DMA channels by clearing bits 0 to 3 in the DMA0EN sfr.
- Configure the first DMA channel for the AES0KIN sfr
  - Select the first DMA channel by writing 0x00 to the DMA0SEL sfr
  - Configure the first DMA channel to move xram to AES0KIN sfr by writing 0x05 to the DMA0NCF sfr
  - Write 0x01 to DMA0NMD to enable wrapping
  - Write the xram location of encryption key to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the key length in bytes to DMA0NSZL sfr
  - Clear the DMA0NSZH sfr
  - Clear the DMA0NAOH and DMA0NAOL sfrs
- Configure the second DMA channel for the AES0BIN sfr.
  - Select the second DMA channel by writing 0x01 to the DMA0SEL sfr.
  - Configure the second DMA channel to move xram to AES0BIN sfr by writing 0x06 to the DMA0NCF sfr.
  - Clear DMA0NMD to disable wrapping.
  - Write the xram address of the data to be encrypted to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the number of bytes to be encrypted in multiples of 16 bytes to the DMA0NSZH and DMA0NSZL sfrs.
  - Clear the DMA0NAOH and DMA0NAOL sfrs.
- Configure the third DMA channel for the AES0XIN sfr.
  - Select the third DMA channel by writing 0x02 to the DMA0SEL sfr.
  - Configure the third DMA channel to move xram to AES0XIN sfr by writing 0x07 to the DMA0NCF sfr.
  - Clear DMA0NMD to disable wrapping.
  - Write the xram address of initialization vector to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the number of bytes to be encrypted in multiples of 16 bytes to the DMA0NSZH and DMA0NSZL sfrs.
  - Clear the DMA0NAOH and DMA0NAOL sfrs.
- * Configure the fourth DMA channel for the AES0YOUT sfr
  - Select the fourth channel by writing 0x03 to the DMA0SEL sfr
  - Configure the fourth DMA channel to move the contents of the AES0YOUT sfr to xram by writing 0x08 to the DMA0NCF sfr
  - Enable transfer complete interrupt by setting bit 7 of DMA0NCF sfr
  - Clear DMA0NMD to disable wrapping
  - Write the xram address for encrypted data to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the number of bytes to be encrypted in multiples of 16 bytes to the DMA0NSZH and DMA0NSZL sfrs.
  - Clear the DMA0NAOH and DMA0NAOL sfrs.
- Clear first four DMA interrupts by clearing bits 0 to 2 in the DMA0INT sfr.

SFR Definition 14.2. AES0DCFG: AES Data Configuration

Bit	7	6	5	4	3	2	1	0
Name						OUTSEL[1:0]		XORIN
Type	R	R	R	R	R	R/W		R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xEA; SFR page = 0x2; Not bit-Addressable

Bit	Name	Function
2:1	OUTSEL[1:0]	**DATA Select.** These bits select the output data source for the AES0YOUT sfr. 00: Direct AES Data 01: AES Data XOR with AES0XIN 10: Inverse Key 11: reserved
0	XORIN	**XOR Input Enable.** Setting this bit with enable the XOR data path on the AES input. If enabled, AES0BIN will be XORed with the AES0XIN and the results will feed into the AES data input. Clearing this bit to 0 will disable the XOR gate on the input. The contents of AES0BIN will go directly into the AES data input.

SFR Definition 17.1. IE: Interrupt Enable

Bit	7	6	5	4	3	2	1	0
Name	EA	ESPI0	ET2	ES0	ET1	EX1	ET0	EX0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Page = All Pages; SFR Address = 0xA8; Bit-Addressable

Bit	Name	Function
7	EA	**Enable All Interrupts.** Globally enables/disables all interrupts. It overrides individual interrupt mask settings. 0: Disable all interrupt sources. 1: Enable each interrupt according to its individual mask setting.
6	ESPI0	**Enable Serial Peripheral Interface (SPI0) Interrupt.** This bit sets the masking of the SPI0 interrupts. 0: Disable all SPI0 interrupts. 1: Enable interrupt requests generated by SPI0.
5	ET2	**Enable Timer 2 Interrupt.** This bit sets the masking of the Timer 2 interrupt. 0: Disable Timer 2 interrupt. 1: Enable interrupt requests generated by the TF2L or TF2H flags.
4	ES0	**Enable UART0 Interrupt.** This bit sets the masking of the UART0 interrupt. 0: Disable UART0 interrupt. 1: Enable UART0 interrupt.
3	ET1	**Enable Timer 1 Interrupt.** This bit sets the masking of the Timer 1 interrupt. 0: Disable all Timer 1 interrupt. 1: Enable interrupt requests generated by the TF1 flag.
2	EX1	**Enable External Interrupt 1.** This bit sets the masking of External Interrupt 1. 0: Disable external interrupt 1. 1: Enable interrupt requests generated by the $\overline{\text{INT1}}$ input.
1	ET0	**Enable Timer 0 Interrupt.** This bit sets the masking of the Timer 0 interrupt. 0: Disable all Timer 0 interrupt. 1: Enable interrupt requests generated by the TF0 flag.
0	EX0	**Enable External Interrupt 0.** This bit sets the masking of External Interrupt 0. 0: Disable external interrupt 0. 1: Enable interrupt requests generated by the $\overline{\text{INT0}}$ input.

SFR Definition 23.2. OSCICN: Internal Oscillator Control

Bit	7	6	5	4	3	2	1	0
Name	IOSCEN	IFRDY						
Type	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	Varies	Varies	Varies	Varies	Varies	Varies

SFR Page = 0x0; SFR Address = 0xB2

Bit	Name	Function
7	IOSCEN	**Internal Oscillator Enable.** 0: Internal oscillator disabled. 1: Internal oscillator enabled.
6	IFRDY	**Internal Oscillator Frequency Ready Flag.** 0: Internal oscillator is not running at its programmed frequency. 1: Internal oscillator is running at its programmed frequency.
5:0	Reserved	Must perform read-modify-write.

Notes:

1. Read-modify-write operations such as ORL and ANL must be used to set or clear the enable bit of this register.
2. OSCBIAS (REG0CN.4) must be set to 1 before enabling the precision internal oscillator.

ness. As shown in Figure 24.2, duty cycles less than 65% indicate a robust oscillation. As the duty cycle approaches 68%, oscillation becomes less reliable and the risk of clock failure increases. Increasing the bias current (by disabling AGC) will always improve oscillation robustness and will reduce the output clock's duty cycle. This test should be performed at the worst case system conditions, as results at very low temperatures or high supply voltage will vary from results taken at room temperature or low supply voltage.

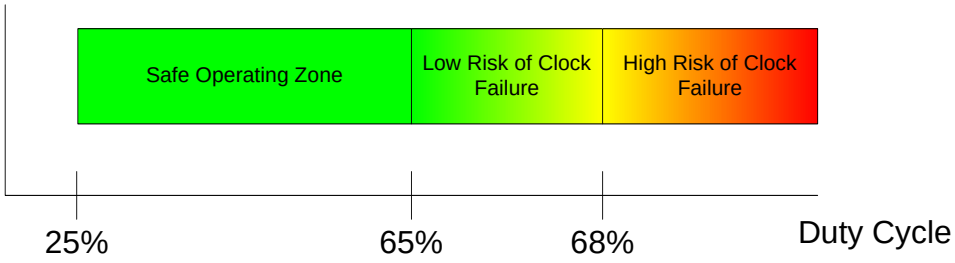

Figure 24.2. Interpreting Oscillation Robustness (Duty Cycle) Test Results

As an alternative to performing the oscillation robustness test, Automatic Gain Control may be disabled at the cost of increased power consumption (approximately 200 nA). Disabling Automatic Gain Control will provide the crystal oscillator with higher immunity against external factors which may lead to clock failure. Automatic Gain Control must be disabled if using the SmaRTClock oscillator in self-oscillate mode.

Table 24.3 shows a summary of the oscillator bias settings. The SmaRTClock Bias Doubling feature allows the self-oscillation frequency to be increased (almost doubled) and allows a higher crystal drive strength in crystal mode. High crystal drive strength is recommended when the crystal is exposed to poor environmental conditions such as excessive moisture. SmaRTClock Bias Doubling is enabled by setting BIASX2 (RTC0XCN.5) to 1.

.

Table 24.3. SmaRTClock Bias Settings

Mode	Setting	Power Consumption
Crystal	Bias Double Off, AGC On	Lowest
	Bias Double Off, AGC Off	Low
	Bias Double On, AGC On	High
	Bias Double On, AGC Off	Highest
Self-Oscillate	Bias Double Off	Low
	Bias Double On	High

25.10.2. Flutter Detection

The flutter detection can be used with either quadrature counter mode or dual counter mode when the two inputs are expected to be in step. Flutter refers to the case where one input continues toggling while the other input stops toggling. This may indicate a broken reed switch or a pressure oscillation when the wheel magnet stops at just the right distance from the reed switch. If a pressure oscillation causes a slight rotational oscillation in the wheel, it could cause a number of pulses on one of the inputs, but not on the other. All four edges are checked by the flutter detection feature (PC1 positive, PC1 negative, PC0 positive, and PC0 negative).When enabled, Flutter detection may be used as an interrupt or wake-up source.

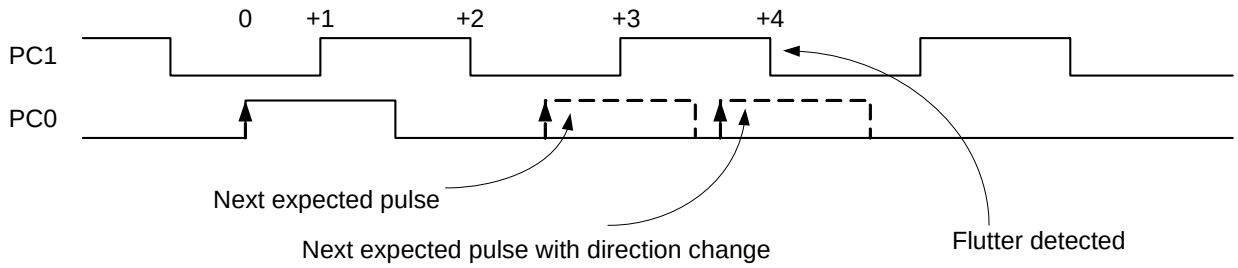

Figure 25.5. Flutter Example

For example, flutter detected on the PC0 positive edge means that 4 edges (positive or negative) were detected on PC1 since the last PC0 positive edge. Each PC0 positive edge resets the flutter detection counter while either PC1 edge increments the counter. There are similar counters for all four edges.

The flutter detection circuit provides interrupts or wake-up sources, but firmware must also read the Pulse Counter registers to determine what corrective action, if any, must be taken.

On the start of flutter event, the firmware should save both counter values and the PC0HIST register. Once the end of flutter event occurs the firmware should also save both counter values and the PC0HIST register. The stop count on flutter, STPCNTFLTR (PCMD[2]), be used to stop the counters when flutter is occurring (quadrature mode only). For quadrature mode, the opposite counter should be decremented by one. In other words, if the direction was clock-wise, the counter clock-wise counter (counter 1) should be decremented by one to correct for one increment before flutter was detected. For dual mode, two reed switches can be used to get a redundant count. If flutter starts during dual mode, both counters should be saved by firmware. After flutter stops, both counters should be read again. The counter that incremented the most was the one that picked up the flutter. There is also a mode to switch from quadrature to dual (PC0MD[1]) when flutter occurs. This changes the counter style from quadrature (count on any edge of PC1 or PC0) to dual to allow all counts to be recorded. Once flutter ends, this mode switches the counters back to quadrature mode. STPCNTFLTR does not function when PC0MD[1] is set.

SFR Definition 26.3. LCD0CNTRST: LCD0 Contrast Adjustment

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	Reserved	CNTRST				
Type	R/W	R/W	R/W	R/W				
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x2; SFR Address = 0x9C

Bit	Name	Function
7:5	Reserved	Read = 000. Write = Must write 000.
4:0	CNTRST	**Contrast Setpoint.** Determines the setpoint for the VLCD voltage necessary to achieve the desired contrast. 00000: 1.90 00001: 1.96 00010: 2.02 00011: 2.08 00100: 2.13 00101: 2.19 00110: 2.25 00111: 2.31 01000: 2.37 01001: 2.43 01010: 2.49 01011: 2.55 01100: 2.60 01101: 2.66 01110: 2.72 01111: 2.78 10000: 2.84 10001: 2.90 10010: 2.96 10011: 3.02 10100: 3.07 10101: 3.13 10110: 3.19 10111: 3.25 11000: 3.31 11001: 3.37 11010: 3.43 11011: 3.49 11100: 3.54 11101: 3.60 11110: 3.66 11111: 3.72

SFR Definition 27.28. P4MDIN: Port4 Input Mode

Bit	7	6	5	4	3	2	1	0
Name	P4MDIN[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = 0xF; SFR Address = 0xF2

Bit	Name	Function
7:0	P4MDIN[3:0]	**Analog Configuration Bits for P4.7–P4.0 (respectively).** Port pins configured for analog mode have their weak pullup and digital receiver disabled. The digital driver is not explicitly disabled. 0: Corresponding P4.n pin is configured for analog mode. 1: Corresponding P4.n pin is not configured for analog mode.

SFR Definition 27.29. P4MDOUT: Port4 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P4MDOUT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0xF; SFR Address = 0xF9

Bit	Name	Function
7:0	P4MDOUT[7:0]	**Output Configuration Bits for P4.7–P4.0 (respectively).** These bits control the digital driver even when the corresponding bit in register P4MDIN is logic 0. 0: Corresponding P4.n Output is open-drain. 1: Corresponding P4.n Output is push-pull.

wire slave mode), and the serial input data synchronously with the slave's system clock. If the master issues SCK, NSS, and the serial input data asynchronously, the maximum data transfer rate (bits/sec) must be less than 1/10 the system clock frequency. In the special case where the master only wants to transmit data to the slave and does not need to receive data from the slave (i.e. half-duplex operation), the SPI slave can receive data at a maximum data transfer rate (bits/sec) of 1/4 the system clock frequency. This is provided that the master issues SCK, NSS, and the serial input data synchronously with the slave's system clock.

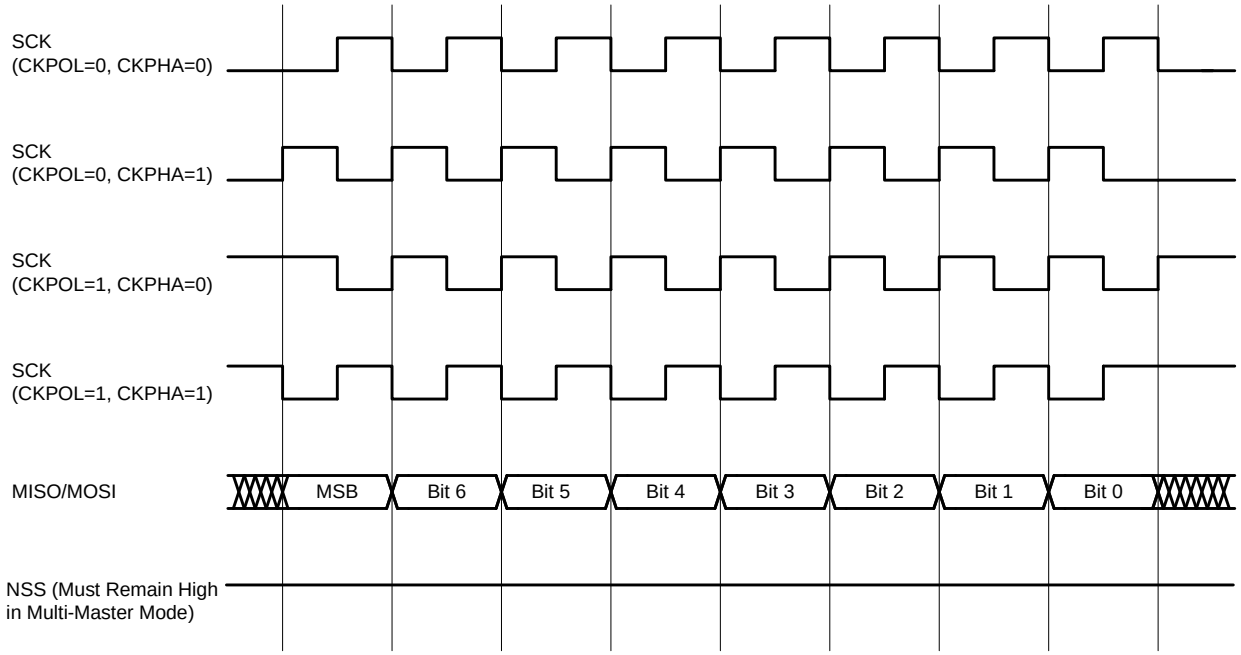

Figure 30.5. Master Mode Data/Clock Timing

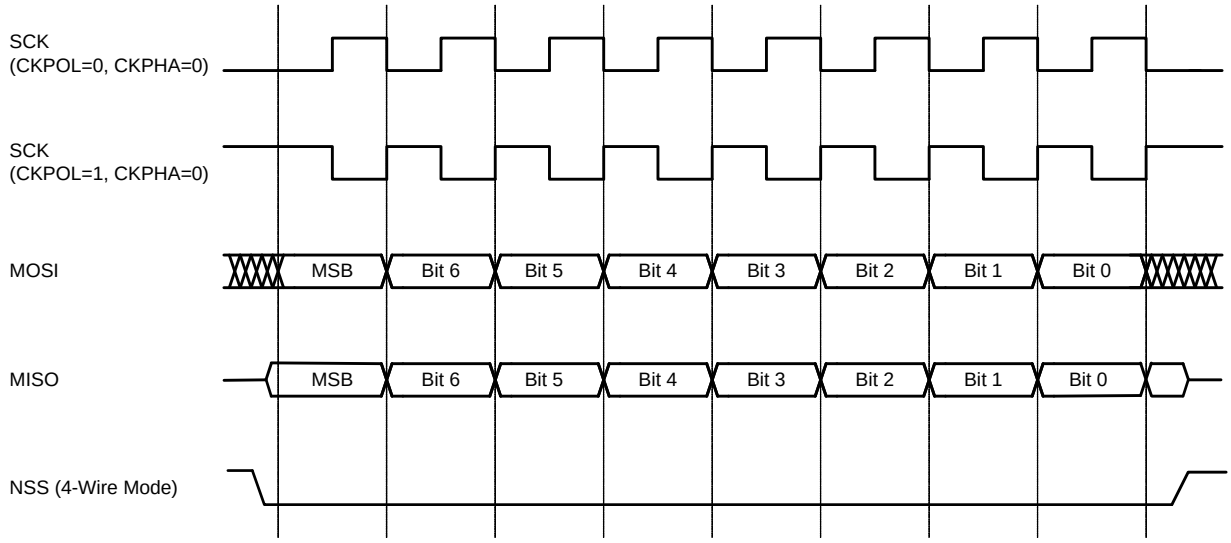

Figure 30.6. Slave Mode Data/Clock Timing (CKPHA = 0)

To to initiate a fixed-length SPI Slave mode bidirectional data transfer:

1. Configure the SPI1 SFRs normally for Slave mode.
   a. Enable Slave mode by clearing bit 6 in SPI1CFG.
   b. Configure the clock polarity CKPOL and clock phase CKPHA as desired in SPI1CFG.
   c. Configure SPI1CKR for the desired SPI clock rate.
   d. Configure SPI1CN for 4-wire slave mode.
   e. Enable the SPI by setting bit 0 of SPI1CN.
2. Configure the first DMA channel for the XRAM-to-SPI1DATA transfer:
   a. Disable the first DMA channel by clearing the corresponding bit in DMA0EN.
   b. Select the first DMA channel by writing to DMA0SEL.
   c. Configure the selected DMA channel to use the XRAM-to-SPI1DAT peripheral request by writing 0x03 to DMA0NCF.
   d. Write 0 to DMA0NMD to disable wrapping.
   e. Write the address of the first byte of the slave output (MISO) data to DMA0NBAH:L.
   f. Write the size of the SPI transfer in bytes to DMA0NSZH:L.
   g. Clear the address offset SFRs DMA0A0H:L.
3. Configure the second DMA channel for the SPI1DAT-to-XRAM transfer:
   a. Disable the second DMA channel by clearing the corresponding bit in DMA0EN.
   b. Select the second DMA channel by writing to DMA0SEL.
   c. Configure the selected DMA channel to use the SPI1DAT-to-XRAM peripheral request by writing 0x04 to DMA0NCF.
   d. Enable DMA interrupts for the second channel by setting bit 7 of DMA0NCF.
   e. Write 0 to DMA0NMD to disable wrapping.
   f. Write the address for the first byte of the slave input (MOSI) data to DMA0NBAH:L.
   g. Write the size of the SPI transfer in bytes to DMA0NSZH:L.
   h. Clear the address offset SFRs DMA0A0H:L.
   i. Enable the interrupt on the second channel by setting the corresponding bit in DMA0INT.
   j. Enable DMA interrupts by setting bit 5 of EIE2.
4. Clear the interrupt bits in DMA0INT for both channels.
5. Enable both channels by setting the corresponding bits in the DMA0EN SFR to initiate the SPI transfer operation.
6. Wait on the DMA interrupt.
7. Clear the DMA enables in the DMA0EN SFR.
8. Clear the DMA interrupts in the DMA0INT SFR.

SFR Definition 32.9. TMR2RLL: Timer 2 Reload Register Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLL[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xCA

Bit	Name	Function
7:0	TMR2RLL[7:0]	**Timer 2 Reload Register Low Byte.** TMR2RLL holds the low byte of the reload value for Timer 2.

SFR Definition 32.10. TMR2RLH: Timer 2 Reload Register High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLH[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xCB

Bit	Name	Function
7:0	TMR2RLH[7:0]	**Timer 2 Reload Register High Byte.** TMR2RLH holds the high byte of the reload value for Timer 2.