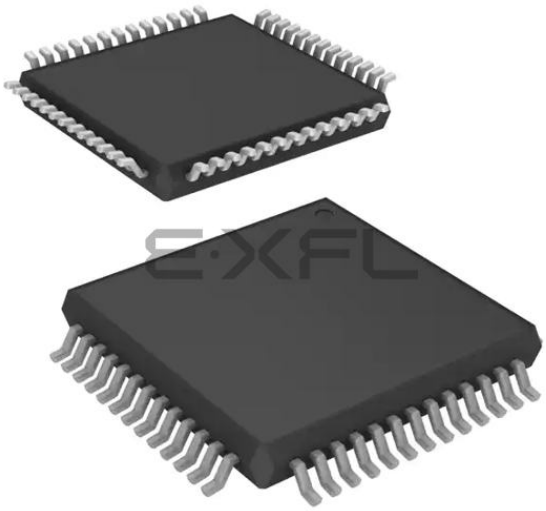


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	-
Peripherals	LCD, LVD, POR, PWM
Number of I/O	40
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	52-LQFP
Supplier Device Package	52-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc908lv8cpbe

Table 1-1. Pin Functions (Continued)

Pin Name	Pin Description	Input/Output	Voltage Level
PTC0/FP19 PTC1/FP20 PTC2/FP21 PTC3/FP22 PTC4/FP23 PTC5/FP24 PTC6 PTC7	8-bit general-purpose I/O port	Input/output	V_{DD}
	PTC0–PTC5 as LCD frontplane drivers, FP19–FP24	Output	V_{DD}
PTD0/FP11 PTD1/FP12 PTD2/FP13 PTD3/FP14 PTD4/FP15 PTD5/FP16 PTD6/FP17 PTD7/FP18	8-bit general-purpose I/O port	Input/output	V_{DD}
	PTD0–PTD7 as LCD frontplane drivers, FP11–FP18	Output	V_{DD}
PTE0/FP3 PTE1/FP4 PTE2/FP5 PTE3/FP6 PTE4/FP7 PTE5/FP8 PTE6/FP9 PTE7/FP10	8-bit general-purpose I/O port	Input/output	V_{DD}
	PTE0–PTE7 as LCD frontplane drivers, FP3–FP10	Output	V_{DD}

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE04	Interrupt Status Register 1 (INT1)	Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE05	Interrupt Status Register 2 (INT2)	Read:	0	0	0	0	0	IF9	IF8	IF7
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE06	Interrupt Status Register 3 (INT3)	Read:	0	0	0	0	0	IF17	IF16	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 4-2. SIM I/O Register Summary (Continued)

4.2 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in Figure 4-3. This clock can come from either the oscillator module or from the on-chip PLL. (See Chapter 5 Clock Generator Module (CGM).)

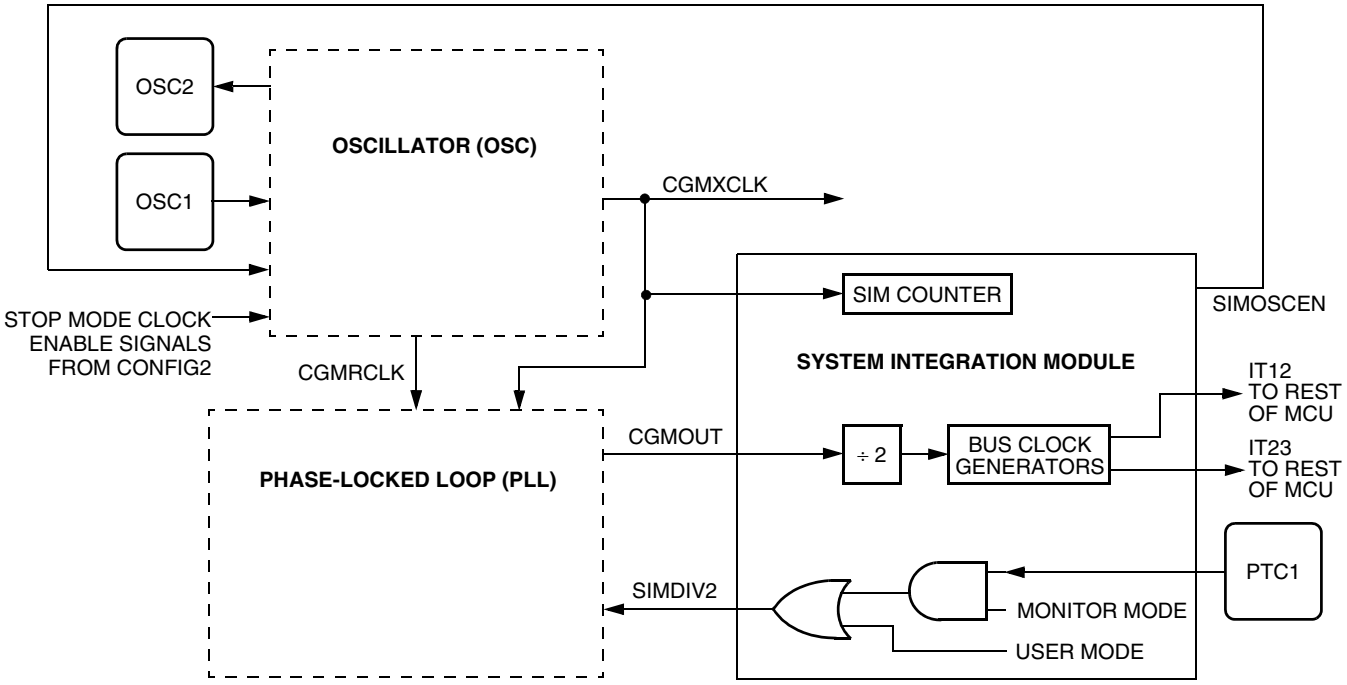


Figure 4-3. CGM Clock Signals

4.2.1 Bus Timing

In user mode, the internal bus frequency is either the oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four.

- The SIM enables CGMOUT.
- Internal clocks to the CPU and modules are held inactive for 4096 CGMXCLK cycles to allow stabilization of the oscillator.
- The $\overline{RST}$ pin is driven low during the oscillator stabilization time.
- The POR bit of the SIM reset status register (SRSR) is set and all other bits in the register are cleared.

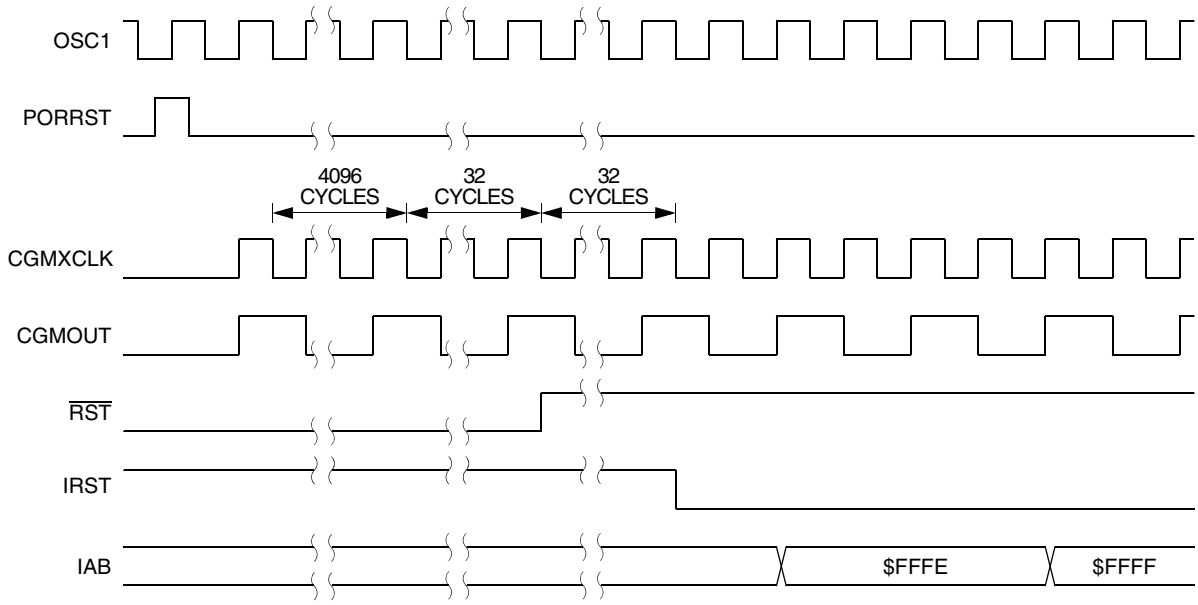


Figure 4-7. POR Recovery

4.3.2.2 Computer Operating Properly (COP) Reset

An input to the SIM is reserved for the COP reset signal. The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR). The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

To prevent a COP module timeout, write any value to location $FFFF. Writing to location $FFFF clears the COP counter and bits 12 through 5 of the SIM counter. The SIM counter output, which occurs at least every $2^{13} - 2^4$ CGMXCLK cycles, drives the COP counter. The COP should be serviced as soon as possible out of reset to guarantee the maximum amount of time before the first timeout.

The COP module is disabled if the $\overline{RST}$ pin or the $\overline{IRQ}$ pin is held at V_{TST} while the MCU is in monitor mode. The COP module can be disabled only through combinational logic conditioned with the high voltage signal on the $\overline{RST}$ or the $\overline{IRQ}$ pin. This prevents the COP from becoming disabled as a result of external noise. During a break state, V_{TST} on the $\overline{RST}$ pin disables the COP module.

4.3.2.3 Illegal Opcode Reset

The SIM decodes signals from the CPU to detect illegal instructions. An illegal instruction sets the ILOP bit in the SIM reset status register (SRSR) and causes a reset.

If the stop enable bit, STOP, in the mask option register is logic 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

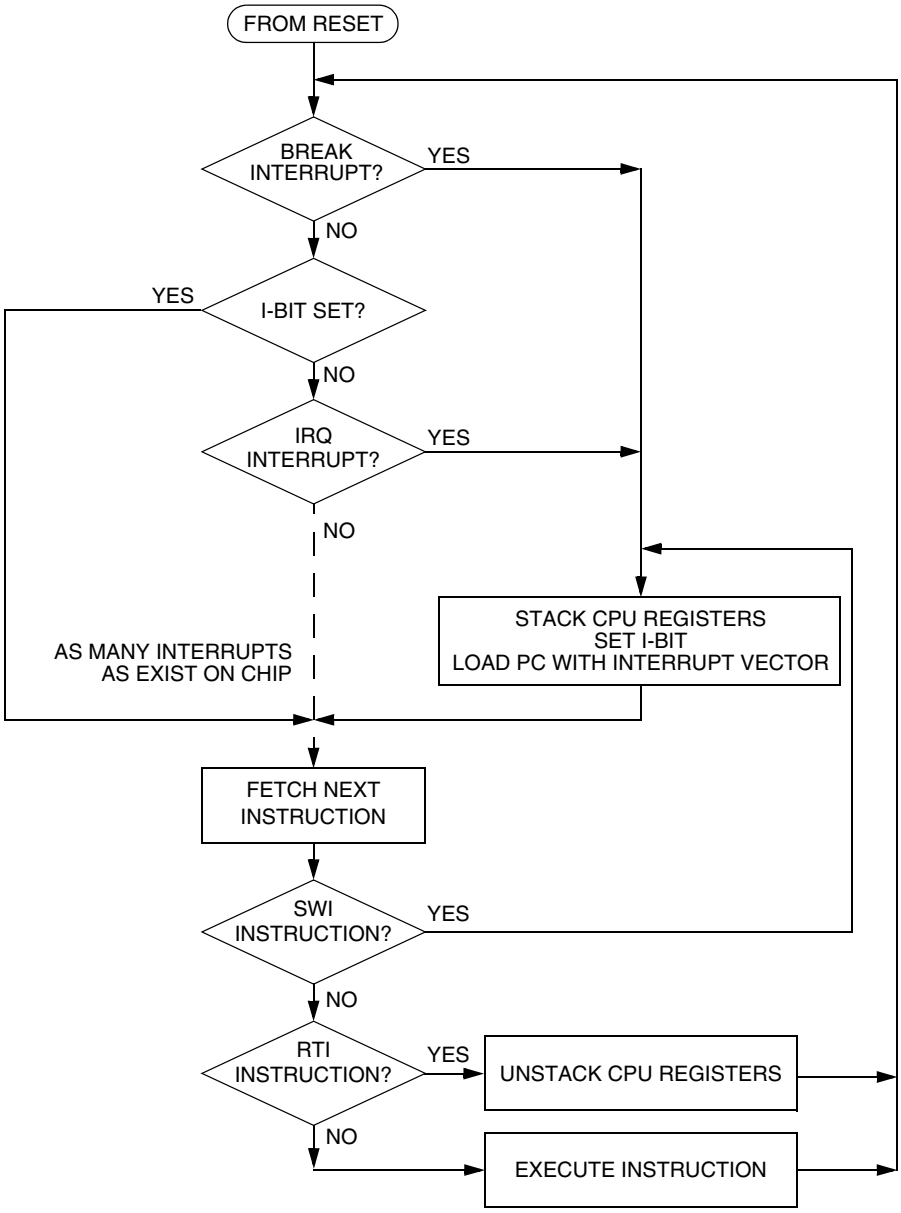


Figure 4-10. Interrupt Processing

4.5.1.1 Hardware Interrupts

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register) and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. Figure 4-11 demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

4.5.3 Reset

All reset sources always have equal and highest priority and cannot be arbitrated.

4.5.4 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See Chapter 16 Development Support.) The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

4.5.5 Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select whether flags are protected from being cleared by properly initialize the break clear flag enable bit (BCFE) in the SIM break flag control register (SBFCR).

Protecting flags in break mode ensures that set flags will not be cleared while in break mode. This protection allows registers to be freely read and written during break mode without losing status flag information.

Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a 2-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

4.6 Low-Power Modes

Executing the WAIT or STOP instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described in the following subsections. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

4.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 4-15 shows the timing for wait mode entry.

A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

Wait mode also can be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is logic 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.

5.5.1 PLL Control Register

The PLL control register (PCTL) contains the interrupt enable and flag bits, the on/off switch, the base clock selector bit, the prescaler bits, and the VCO power-of-two range selector bits.

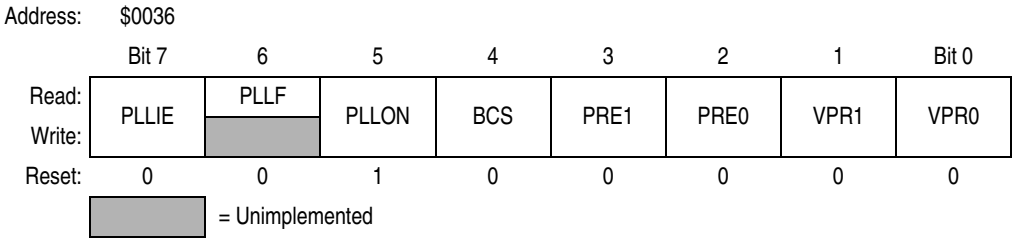


Figure 5-4. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

This read/write bit enables the PLL to generate an interrupt request when the LOCK bit toggles, setting the PLL flag, PLLF. When the AUTO bit in the PLL bandwidth control register (PBWC) is clear, PLLIE cannot be written and reads as logic 0. Reset clears the PLLIE bit.

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL Interrupt Flag Bit

This read-only bit is set whenever the LOCK bit toggles. PLLF generates an interrupt request if the PLLIE bit also is set. PLLF always reads as logic 0 when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. Clear the PLLF bit by reading the PLL control register. Reset clears the PLLF bit.

1 = Change in lock condition
0 = No change in lock condition

NOTE

Do not inadvertently clear the PLLF bit. Any read or read-modify-write operation on the PLL control register clears the PLLF bit.

PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). (See 5.3.8 Base Clock Selector Circuit.) Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. (See 5.3.8 Base Clock Selector Circuit.) Reset clears the BCS bit.

1 = CGMVCLK divided by two drives CGMOUT
0 = CGMXCLK divided by two drives CGMOUT

NOTE

Before enabling a TIM channel register for input capture operation, make sure that the TCHx pin is stable for at least two bus clocks.

TOVx — Toggle On Overflow Bit

When channel x is an output compare channel, this read/write bit controls the behavior of the channel x output when the TIM counter overflows. When channel x is an input capture channel, TOVx has no effect.

Reset clears the TOVx bit.

1 = Channel x pin toggles on TIM counter overflow

0 = Channel x pin does not toggle on TIM counter overflow

NOTE

When TOVx is set, a TIM counter overflow takes precedence over a channel x output compare if both occur at the same time.

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 1, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 6-11 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

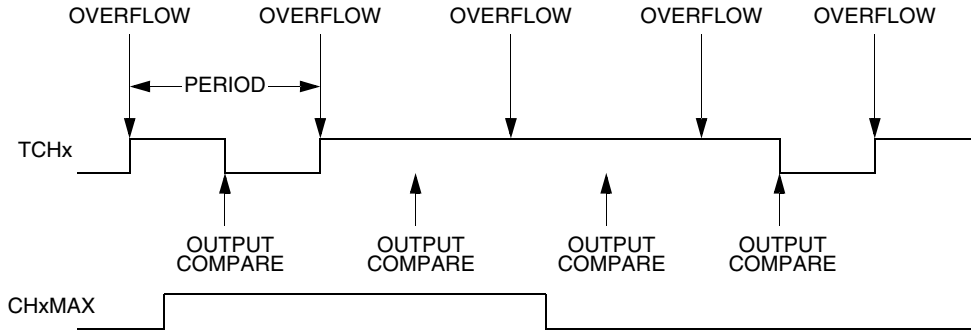


Figure 6-11. CHxMAX Latency

6.9.5 TIM Channel Registers

These read/write registers contain the captured TIM counter value of the input capture function or the output compare value of the output compare function. The state of the TIM channel registers after reset is unknown.

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the TIM channel x registers (TCHxH) inhibits input captures until the low byte (TCHxL) is read.

In output compare mode (MSxB:MSxA ≠ 0:0), writing to the high byte of the TIM channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.

Chapter 7
Programmable Periodic Interrupt (PPI)

7.1 Introduction

This section describes the programmable periodic interrupt (PPI) module. The PPI will generate periodic interrupts at user selectable rates using a counter clocked by the selected clock.

7.2 Features

Features of the PPI include:

- Seven user selectable periodic interrupts
- User selectable clock source:
  - Internal 32kHz
  - CGMXCLK output from CGM module
  - External clock from PPIECK pin

7.3 Functional Description

The PPI module generates periodic interrupt requests to the CPU. The interrupt request is treated as a regular keyboard interrupt request, with the difference that instead of a pin, the interrupt signal is generated by internal logic.

When PPI counter reaches the defined count, it generates an interrupt request. The latched status of interrupt generation of the PPI can be read directly from the PPI1L bit. This is a read-only status bit which is occupies a bit position in the register. The latch can be cleared by writing to the ACKK bit in the KBSCR register.

The PPI counter can count and generate interrupts even when the MCU is in stop mode if the corresponding clock source is enabled.

Figure 7-1 is a block diagram of the PPI.

9.4.2 LCD Voltages (V_{LCD}, V_{LCD1}, V_{LCD2}, V_{LCD3})

The voltage V_{LCD} is from the V_{LCD} pin and must not exceed V_{DD}. V_{LCD1}, V_{LCD2}, and V_{LCD3} are internal bias voltages for the LCD driver waveforms. They are derived from V_{LCD} using a resistor ladder (see Figure 9-3).

The relative potential of the LCD voltages are:

- $V_{LCD} = V_{DD}$
- $V_{LCD1} = 2/3 \times (V_{LCD} - V_{bias})$
- $V_{LCD2} = 1/3 \times (V_{LCD} - V_{bias})$
- $V_{LCD3} = V_{bias}$

The V_{LCD3} bias voltage, V_{bias}, is controlled by the LCD contrast control bits, LCCON[2:0].

9.4.3 LCD Cycle Frame

The LCD driver module uses the CGMXCLK (see Chapter 5 Clock Generator Module (CGM)) as the input reference clock. This clock is divided to produce the LCD waveform base clock, LCDCLK, by configuring the LCLK[2:0] bits in the LCD clock register. The LCDCLK clocks the backplane and the frontplane output waveforms.

The LCD cycle frame is determined by the equation:

$$\text{LCD CYCLE FRAME} = \frac{1}{\text{LCD WAVEFORM BASE CLOCK} \times \text{DUTY}}$$

For example, for 1/3 duty and 256Hz waveform base clock:

$$\text{LCD CYCLE FRAME} = \frac{1}{256 \times (1/3)}$$

$$= 11.72 \text{ ms}$$

9.4.4 Fast Charge and Low Current

The default value for each of the bias resistors (see Figure 9-3), R_{LCD}, in the resistor ladder is approximately 37kΩ at V_{LCD} = 3V. The relatively high current drain through the 37kΩ resistor ladder may not be suitable for some LCD panel connections. Lowering this current is possible by setting the LC bit in the LCD control register, switching the R_{LCD} value to 146kΩ.

Although the lower current drain is desirable, but in some LCD panel connections, the higher current is required to drive the capacitive load of the LCD panel. In most cases, the higher current is only required when the LCD waveforms change state (the rising and falling edges in the LCD output waveforms). The fast charge option is designed to have the high current for the switching and the low current for the steady state. Setting the FC bit in the LCD control register selects the fast charge option. The R_{LCD} value is set to 37kΩ (for high current) for a fraction of time for each LCD waveform switching edge, and then back to 146kΩ for the steady state period. The duration of the fast charge time is set by configuring the FCCTL[1:0] bits in the LCD clock register, and can be LCDCLK/32, LCDCLK/64, or LCDCLK/128. Figure 9-4 shows the fast charge clock relative to the BP0 waveform.

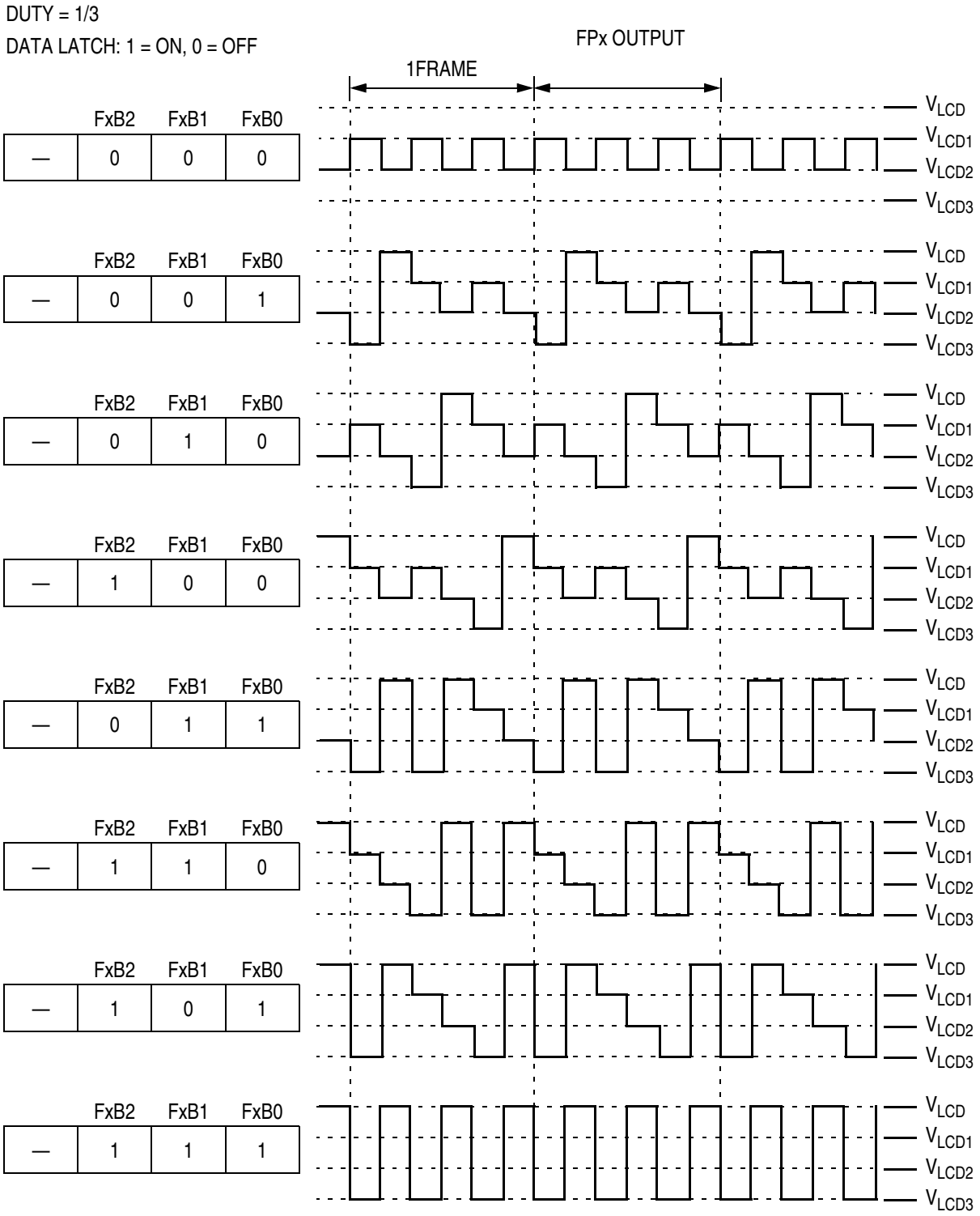


Figure 9-9. 1/3 Duty LCD Frontplane Driver Waveforms

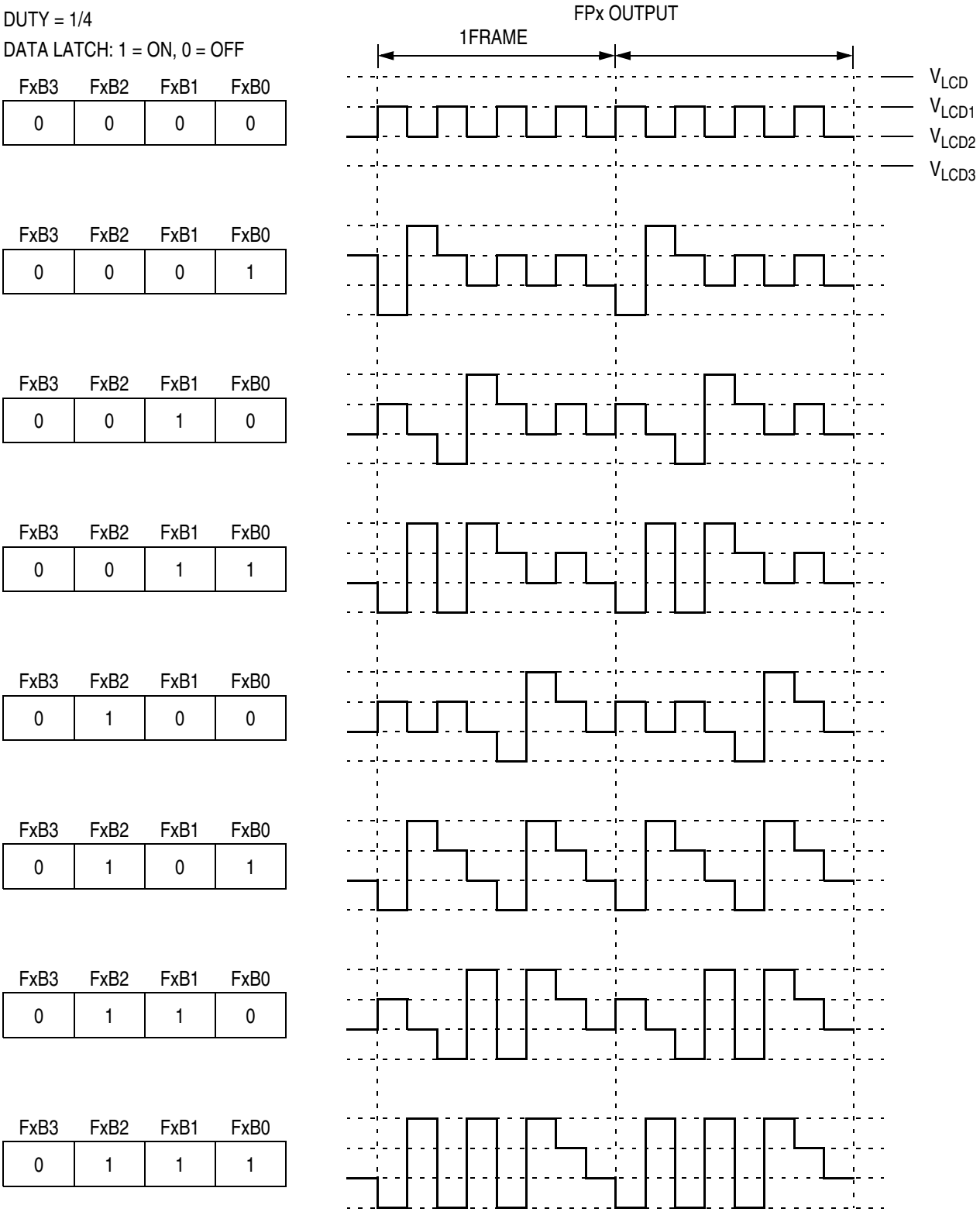


Figure 9-10. 1/4 Duty LCD Frontplane Driver Waveforms

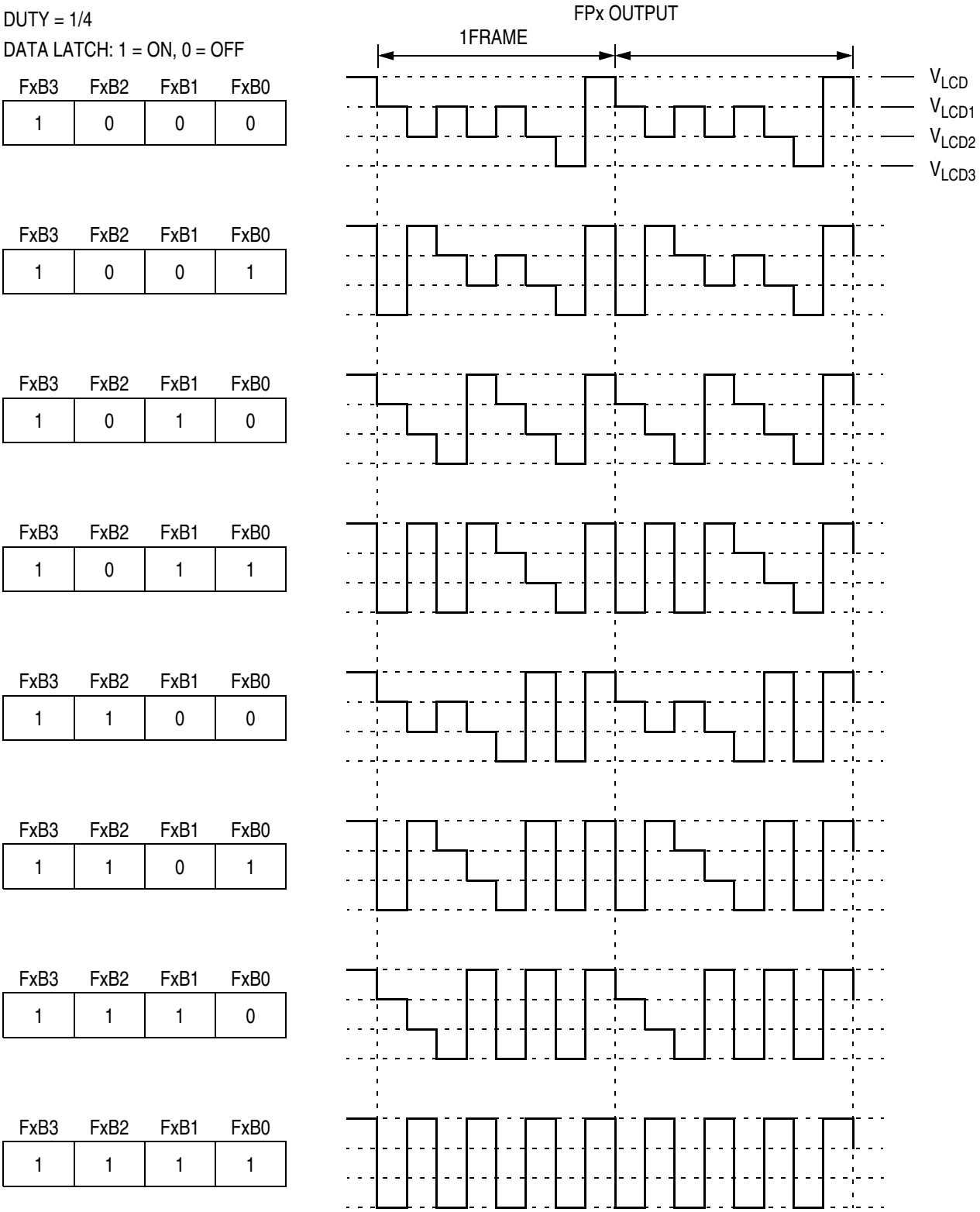


Figure 9-11. 1/4 Duty LCD Frontplane Driver Waveforms (continued)

$0059	LCD Data Register 8 (LDAT8)	Read: Write:	F15B3	F15B2	F15B1	F15B0	F14B3	F14B2	F14B1	F14B0
		Reset:	U	U	U	U	U	U	U	U
$005A	LCD Data Register 9 (LDAT9)	Read: Write:	F17B3	F17B2	F17B1	F17B0	F16B3	F16B2	F16B1	F16B0
		Reset:	U	U	U	U	U	U	U	U
$005B	LCD Data Register 10 (LDAT10)	Read: Write:	F19B3	F19B2	F19B1	F19B0	F18B3	F18B2	F18B1	F18B0
		Reset:	U	U	U	U	U	U	U	U
$005C	LCD Data Register 11 (LDAT11)	Read: Write:	F21B3	F21B2	F21B1	F21B0	F20B3	F20B2	F20B1	F20B0
		Reset:	U	U	U	U	U	U	U	U
$005D	LCD Data Register 12 (LDAT12)	Read: Write:	F23B3	F23B2	F23B1	F23B0	F22B3	F22B2	F22B1	F22B0
		Reset:	U	U	U	U	U	U	U	U
$005E	LCD Data Register 13 (LDAT13)	Read:	0	0	0	0	F24B3	F24B2	F24B1	F24B0
		Write:								
		Reset:	0	0	0	0	U	U	U	U

U = Unaffected = Unimplemented

Figure 9-18. LCD Data Registers 1–13 (LDAT1–LDAT13)

The vector fetch or software clear may occur before or after the interrupt pin returns to logic one. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

When set, the IMASK bit in the INTSCR mask all external interrupt requests. A latched interrupt request is not presented to the interrupt priority logic unless the IMASK bit is clear.

NOTE

The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. (See 4.5 Exception Control.)

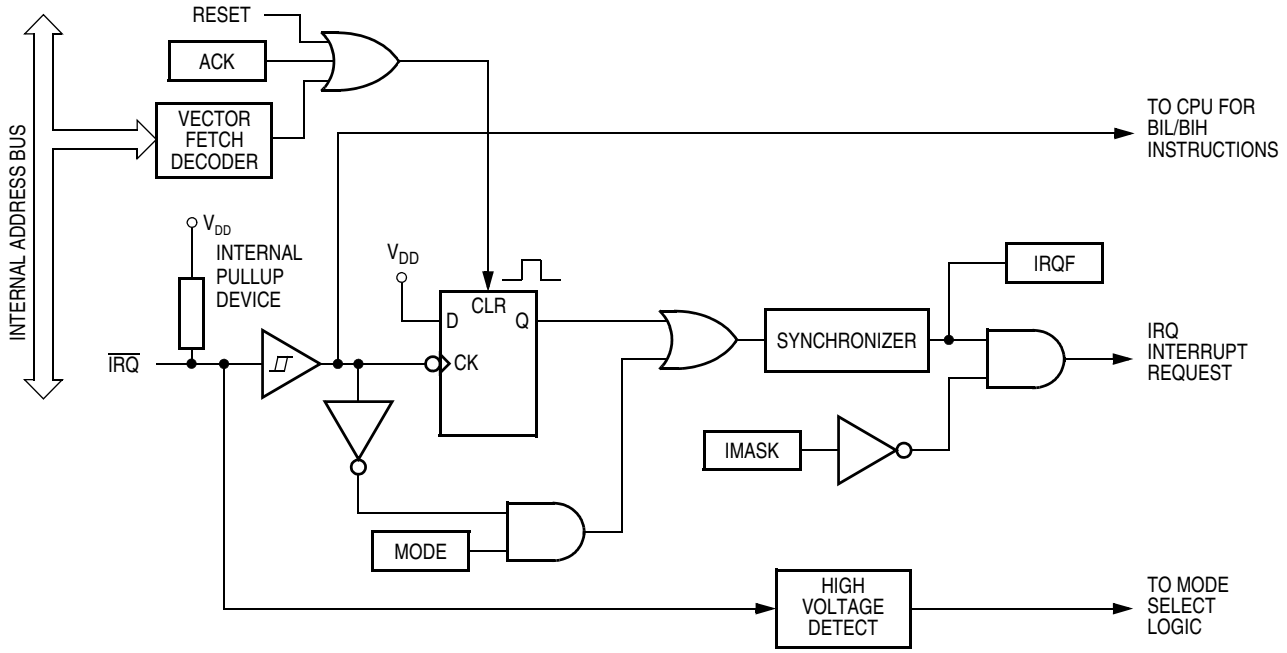


Figure 11-1. IRQ Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001E	IRQ Status and Control Register (INTSCR)	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-2. IRQ I/O Register Summary

11.5 IRQ Status and Control Register (INTSCR)

The IRQ status and control register (INTSCR) controls and monitors operation of the IRQ module. The INTSCR has the following functions:

- Shows the state of the IRQ flag
- Clears the IRQ latch
- Masks IRQ and interrupt request
- Controls triggering sensitivity of the $\overline{\text{IRQ}}$ interrupt pin

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	IRQF	0	IMASK	MODE
Write:						ACK		
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-3. IRQ Status and Control Register (INTSCR)

IRQF — IRQ Flag Bit

This read-only status bit is high when the IRQ interrupt is pending.

1 = $\overline{\text{IRQ}}$ interrupt pending
0 = $\overline{\text{IRQ}}$ interrupt not pending

ACK — IRQ Interrupt Request Acknowledge Bit

Writing a logic one to this write-only bit clears the IRQ latch. ACK always reads as logic zero. Reset clears ACK.

IMASK — IRQ Interrupt Mask Bit

Writing a logic one to this read/write bit disables IRQ interrupt requests. Reset clears IMASK.

1 = IRQ interrupt requests disabled
0 = IRQ interrupt requests enabled

MODE — IRQ Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ}}$ pin. Reset clears MODE.

1 = $\overline{\text{IRQ}}$ interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ}}$ interrupt requests on falling edges only

15.3.3 Stack Pointer

The stack pointer is a 16-bit register that contains the address of the next location on the stack. During a reset, the stack pointer is preset to $00FF. The reset stack pointer (RSP) instruction sets the least significant byte to $FF and does not affect the most significant byte. The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

In the stack pointer 8-bit offset and 16-bit offset addressing modes, the stack pointer can function as an index register to access data on the stack. The CPU uses the contents of the stack pointer to determine the conditional address of the operand.

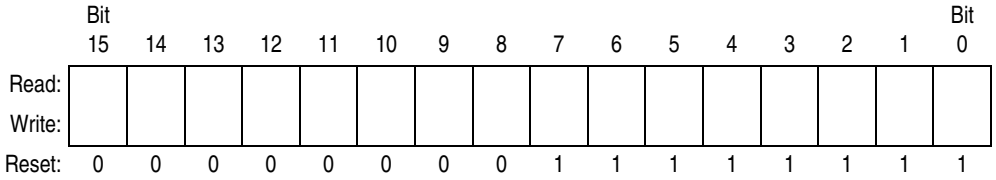


Figure 15-4. Stack Pointer (SP)

NOTE

The location of the stack is arbitrary and may be relocated anywhere in random-access memory (RAM). Moving the SP out of page 0 ($0000 to $00FF) frees direct address (page 0) space. For correct operation, the stack pointer must point only to RAM locations.

15.3.4 Program Counter

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched.

Normally, the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, and interrupt operations load the program counter with an address other than that of the next sequential location.

During reset, the program counter is loaded with the reset vector address located at $FFFE and $FFFF. The vector address is the address of the first instruction to be executed after exiting the reset state.

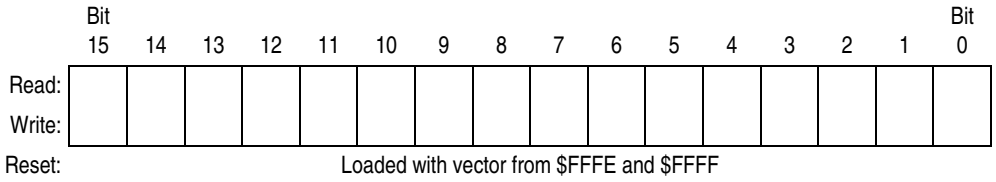


Figure 15-5. Program Counter (PC)

Table 15-2. Opcode Map

	Bit Manipulation		Branch	Read-Modify-Write						Control		Register/Memory							
	DIR	DIR	REL	DIR	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX
MSB / LSB	0	1	2	3	4	5	6	9E6	7	8	9	A	B	C	D	9ED	E	9EE	F
0	5 BRSET0 3 DIR	4 BSET0 2 DIR	3 BRA 2 REL	4 NEG 2 DIR	1 NEGA 1 INH	1 NEGX 1 INH	4 NEG 2 IX1	5 NEG 3 SP1	3 NEG 1 IX	7 RTI 1 INH	3 BGE 2 REL	2 SUB 2 IMM	3 SUB 2 DIR	4 SUB 3 EXT	4 SUB 3 IX2	5 SUB 4 SP2	3 SUB 2 IX1	4 SUB 3 SP1	2 SUB 1 IX
1	5 BRCLR0 3 DIR	4 BCLR0 2 DIR	3 BRN 2 REL	5 CBEQ 3 DIR	4 CBEQA 3 IMM	4 CBEQX 3 IMM	5 CBEQ 3 IX1+	6 CBEQ 4 SP1	4 CBEQ 2 IX+	4 RTS 1 INH	3 BLT 2 REL	2 CMP 2 IMM	3 CMP 2 DIR	4 CMP 3 EXT	4 CMP 3 IX2	5 CMP 4 SP2	3 CMP 2 IX1	4 CMP 3 SP1	2 CMP 1 IX
2	5 BRSET1 3 DIR	4 BSET1 2 DIR	3 BHI 2 REL		5 MUL 1 INH	7 DIV 1 INH	3 NSA 1 INH		2 DAA 1 INH		3 BGT 2 REL	2 SBC 2 IMM	3 SBC 2 DIR	4 SBC 3 EXT	4 SBC 3 IX2	5 SBC 4 SP2	3 SBC 2 IX1	4 SBC 3 SP1	2 SBC 1 IX
3	5 BRCLR1 3 DIR	4 BCLR1 2 DIR	3 BLS 2 REL	4 COM 2 DIR	1 COMA 1 INH	1 COMX 1 INH	4 COM 2 IX1	5 COM 3 SP1	3 COM 1 IX	9 SWI 1 INH	3 BLE 2 REL	2 CPX 2 IMM	3 CPX 2 DIR	4 CPX 3 EXT	4 CPX 3 IX2	5 CPX 4 SP2	3 CPX 2 IX1	4 CPX 3 SP1	2 CPX 1 IX
4	5 BRSET2 3 DIR	4 BSET2 2 DIR	3 BCC 2 REL	4 LSR 2 DIR	1 LSRA 1 INH	1 LSRX 1 INH	4 LSR 2 IX1	5 LSR 3 SP1	3 LSR 1 IX	2 TAP 1 INH	2 TXS 1 INH	2 AND 2 IMM	3 AND 2 DIR	4 AND 3 EXT	4 AND 3 IX2	5 AND 4 SP2	3 AND 2 IX1	4 AND 3 SP1	2 AND 1 IX
5	5 BRCLR2 3 DIR	4 BCLR2 2 DIR	3 BCS 2 REL	4 STHX 2 DIR	3 LDHX 3 IMM	4 LDHX 2 DIR	3 CPHX 3 IMM		4 CPHX 2 DIR	1 TPA 1 INH	2 TSX 1 INH	2 BIT 2 IMM	3 BIT 2 DIR	4 BIT 3 EXT	4 BIT 3 IX2	5 BIT 4 SP2	3 BIT 2 IX1	4 BIT 3 SP1	2 BIT 1 IX
6	5 BRSET3 3 DIR	4 BSET3 2 DIR	3 BNE 2 REL	4 ROR 2 DIR	1 RORA 1 INH	1 RORX 1 INH	4 ROR 2 IX1	5 ROR 3 SP1	3 ROR 1 IX	2 PULA 1 INH		2 LDA 2 IMM	3 LDA 2 DIR	4 LDA 3 EXT	4 LDA 3 IX2	5 LDA 4 SP2	3 LDA 2 IX1	4 LDA 3 SP1	2 LDA 1 IX
7	5 BRCLR3 3 DIR	4 BCLR3 2 DIR	3 BEQ 2 REL	4 ASR 2 DIR	1 ASRA 1 INH	1 ASRX 1 INH	4 ASR 2 IX1	5 ASR 3 SP1	3 ASR 1 IX	2 PSHA 1 INH	1 TAX 1 INH	2 AIS 2 IMM	3 STA 2 DIR	4 STA 3 EXT	4 STA 3 IX2	5 STA 4 SP2	3 STA 2 IX1	4 STA 3 SP1	2 STA 1 IX
8	5 BRSET4 3 DIR	4 BSET4 2 DIR	3 BHCC 2 REL	4 LSL 2 DIR	1 LSLA 1 INH	1 LSLX 1 INH	4 LSL 2 IX1	5 LSL 3 SP1	3 LSL 1 IX	2 PULX 1 INH	1 CLC 1 INH	2 EOR 2 IMM	3 EOR 2 DIR	4 EOR 3 EXT	4 EOR 3 IX2	5 EOR 4 SP2	3 EOR 2 IX1	4 EOR 3 SP1	2 EOR 1 IX
9	5 BRCLR4 3 DIR	4 BCLR4 2 DIR	3 BHCS 2 REL	4 ROL 2 DIR	1 ROLA 1 INH	1 ROLX 1 INH	4 ROL 2 IX1	5 ROL 3 SP1	3 ROL 1 IX	2 PSHX 1 INH	1 SEC 1 INH	2 ADC 2 IMM	3 ADC 2 DIR	4 ADC 3 EXT	4 ADC 3 IX2	5 ADC 4 SP2	3 ADC 2 IX1	4 ADC 3 SP1	2 ADC 1 IX
A	5 BRSET5 3 DIR	4 BSET5 2 DIR	3 BPL 2 REL	4 DEC 2 DIR	1 DECA 1 INH	1 DECX 1 INH	4 DEC 2 IX1	5 DEC 3 SP1	3 DEC 1 IX	2 PULH 1 INH	2 CLI 1 INH	2 ORA 2 IMM	3 ORA 2 DIR	4 ORA 3 EXT	4 ORA 3 IX2	5 ORA 4 SP2	3 ORA 2 IX1	4 ORA 3 SP1	2 ORA 1 IX
B	5 BRCLR5 3 DIR	4 BCLR5 2 DIR	3 BMI 2 REL	5 DBNZ 3 DIR	3 DBNZA 2 INH	3 DBNZX 2 INH	5 DBNZ 3 IX1	6 DBNZ 4 SP1	4 DBNZ 2 IX	2 PSHH 1 INH	2 SEI 1 INH	2 ADD 2 IMM	3 ADD 2 DIR	4 ADD 3 EXT	4 ADD 3 IX2	5 ADD 4 SP2	3 ADD 2 IX1	4 ADD 3 SP1	2 ADD 1 IX
C	5 BRSET6 3 DIR	4 BSET6 2 DIR	3 BMC 2 REL	4 INC 2 DIR	1 INCA 1 INH	1 INCX 1 INH	4 INC 2 IX1	5 INC 3 SP1	3 INC 1 IX	1 CLRH 1 INH	1 RSP 1 INH		2 JMP 2 DIR	3 JMP 3 EXT	4 JMP 3 IX2		3 JMP 2 IX1		2 JMP 1 IX
D	5 BRCLR6 3 DIR	4 BCLR6 2 DIR	3 BMS 2 REL	3 TST 2 DIR	1 TSTA 1 INH	1 TSTX 1 INH	3 TST 2 IX1	4 TST 3 SP1	2 TST 1 IX		1 NOP 1 INH	4 BSR 2 REL	4 JSR 2 DIR	5 JSR 3 EXT	6 JSR 3 IX2		5 JSR 2 IX1		4 JSR 1 IX
E	5 BRSET7 3 DIR	4 BSET7 2 DIR	3 BIL 2 REL		5 MOV 3 DD	4 MOV 2 DIX+	4 MOV 3 IMD		4 MOV 2 IX+D	1 STOP 1 INH	*	2 LDX 2 IMM	3 LDX 2 DIR	4 LDX 3 EXT	4 LDX 3 IX2	5 LDX 4 SP2	3 LDX 2 IX1	4 LDX 3 SP1	2 LDX 1 IX
F	5 BRCLR7 3 DIR	4 BCLR7 2 DIR	3 BIH 2 REL	3 CLR 2 DIR	1 CLRA 1 INH	1 CLRX 1 INH	3 CLR 2 IX1	4 CLR 3 SP1	2 CLR 1 IX	1 WAIT 1 INH	1 TXA 1 INH	2 AIX 2 IMM	3 STX 2 DIR	4 STX 3 EXT	4 STX 3 IX2	5 STX 4 SP2	3 STX 2 IX1	4 STX 3 SP1	2 STX 1 IX

INH Inherent
IMM Immediate
DIR Direct
EXT Extended
DD Direct-Direct
IX+D Indexed-Direct
REL Relative
IX Indexed, No Offset
IX1 Indexed, 8-Bit Offset
IX2 Indexed, 16-Bit Offset
IMD Immediate-Direct
DIX+ Direct-Indexed
SP1 Stack Pointer, 8-Bit Offset
SP2 Stack Pointer, 16-Bit Offset
IX+ Indexed, No Offset with Post Increment
IX1+ Indexed, 1-Byte Offset with Post Increment
*Pre-byte for stack pointer indexed instructions

MSB / LSB	0
0	5 BRSET0 3 DIR

Low Byte of Opcode in Hexadecimal
High Byte of Opcode in Hexadecimal
Cycles
Opcode Mnemonic
Number of Bytes / Addressing Mode

rProgram can be used to program a range less than 32 bytes. Example 16-7 shows how to program $55 and $AA at location $E004 and $E005, respectively. In this example, a RAM block start address is $115.

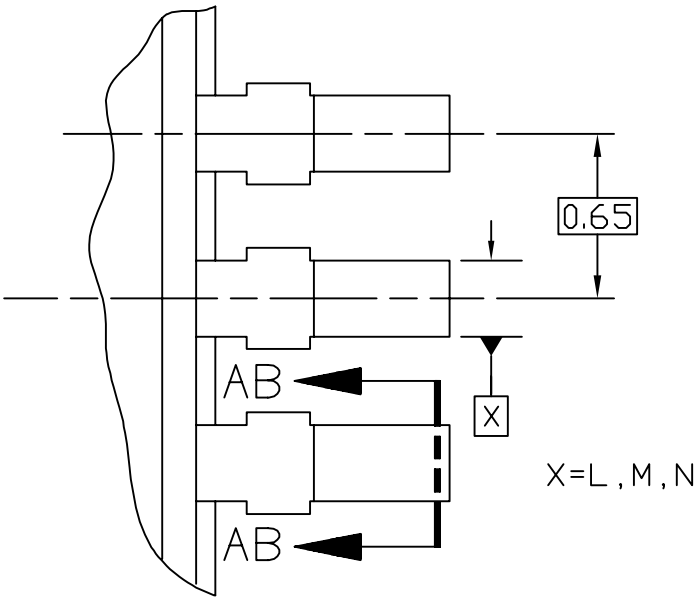
Example 16-7. Programming a Range Smaller than a Row

```
rProgram:       equ     $FF8B           ;rProgram jump address

RAMblock:       equ     $0115           ;In this example, RAM block start address $0115
CPUSPD:         equ     RAMblock        ;CPUSPD location
DATASIZE:       equ     RAMblock+1      ;DATASIZE location
ADDR:           equ     RAMblock+2      ;ADDR location (2 bytes)
DATA:           equ     RAMblock+4      ;DATA array start address

                ldhx    #$55AA          ;Load data to DATA array
                sthx    DATA

                lda     #$18            ;fop = 6.0MHz in this example
                sta     CPUSPD

                lda     #2              ;Load 2 bytes to DATASIZE
                sta     DATASIZE

                ldhx    #$E005          ;Load last address to ADDR
                sthx    ADDR

                ldhx    #RAMblock       ;Load RAM block start address to H:X

                jsr     rProgram        ;Call rProgram routine
```

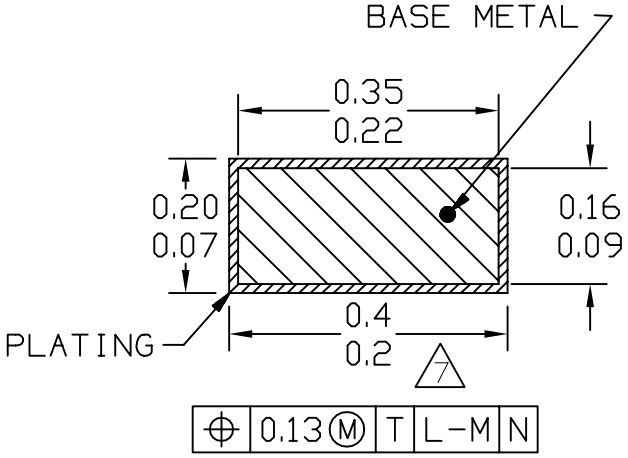
rProgram can also program a range beyond a page. Example 16-8 shows how to program 70-byte data to FLASH. In this example, the RAM block start address is $0100. The data is programmed to locations $C0F0–$C135.

Example 16-8. Programming a Range Bigger than a Page

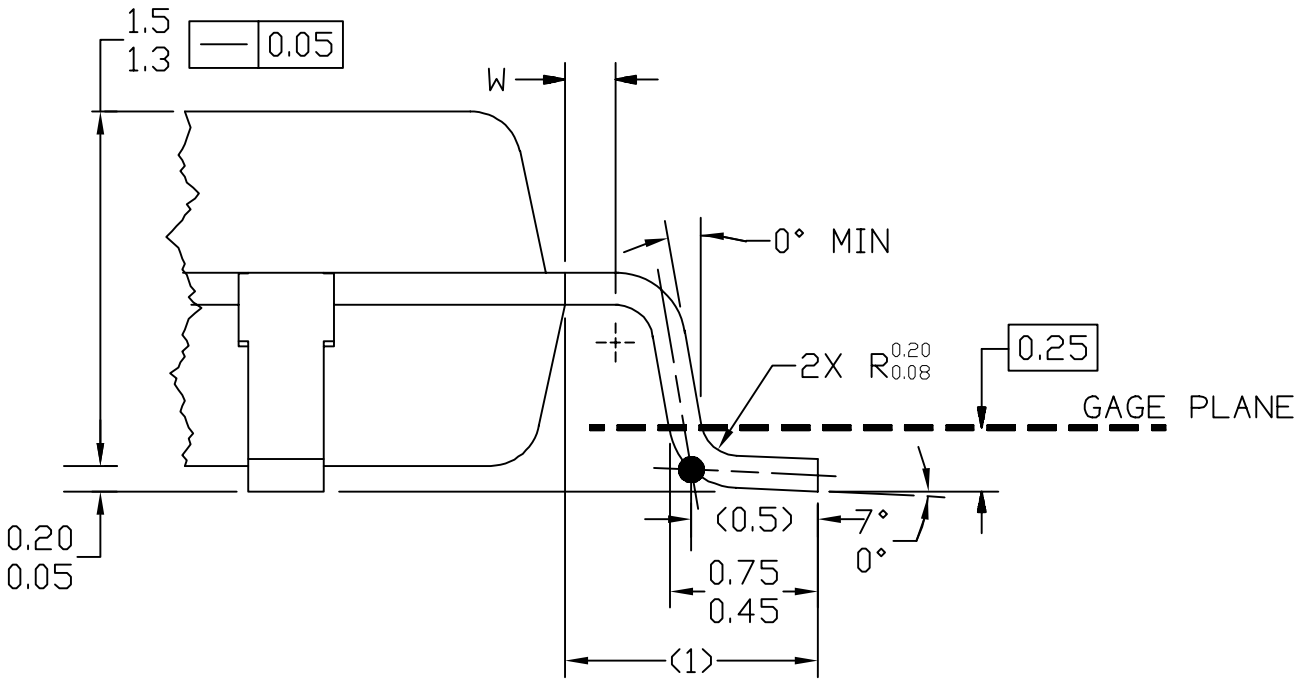
```
rProgram:       equ     $FF8B           ;rProgram jump address

RAMblock:       equ     $0100           ;In this example, RAM block start address $0100
CPUSPD:         equ     RAMblock        ;CPUSPD location
DATASIZE:       equ     RAMblock+1      ;DATASIZE location
ADDR:           equ     RAMblock+2      ;ADDR location (2 bytes)
DATA:           equ     RAMblock+4      ;DATA array start address

                ldhx    #$0000          ;Index offset into DATA array
                lda     #$AA            ;Initial data value (inverted)
Data_load:
                coma                    ;Alternate between $55 and $AA. Fill DATA array,
                sta     DATA,x          ;70 bytes data, values to program into FLASH
                aix     #1              ;(ie. 55, AA, 55, AA....)
                cphx    #$46
                bne     Data_load
```



VIEW Y



SECTION AB-AB
ROTATED 90° CLOCKWISE



VIEW AA

MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE:	DOCUMENT NO: 98ASS23228W	REV: F
52LD TQFP 10 X 10 PKG, 0.65 PITCH, 1.4 THICK	CASE NUMBER: 848D-03	05 MAY 2005
	STANDARD: NON-JEDEC	