MC9RS08LE4

Reference Manual

RS08 Microcontrollers

Related Documentation:

- **MC9RS08LE4 (Data Sheet)**
  Contains pin assignments and diagrams, all electrical specifications, and mechanical drawing outlines.

Find the most current versions of all documents at:
http://www.freescale.com

MC9RS08LE4RM
Rev. 3
5/2015

freescale.com

MC9RS08LE4 Features

8-Bit RS08 Central Processor Unit (CPU)

- Up to 20 MHz CPU at 2.7 V to 5.5 V across temperature range of –40°C to 85°C
- Subset of HC08 instruction set with added BGND instruction

On-Chip Memory

- 4 KB flash read/program/erase over full operating voltage and temperature
- 256-byte random-access memory (RAM)
- Security circuitry to prevent unauthorized access to flash contents

Power-Saving Modes

- Wait and stop

Clock Source Options

- Oscillator (XOSC) — Loop-control Pierce oscillator; crystal or ceramic resonator range of 31.25 kHz to 39.0625 kHz or 1 MHz to 20 MHz
- Internal Clock Source (ICS) — Internal clock source module containing a frequency-locked-loop (FLL) controlled by internal or external reference; precision trimming of internal reference allows 0.2% resolution and 2% deviation over temperature and voltage; supports bus frequencies up to 10 MHz

System Protection

- Watchdog computer operating properly (COP) reset with option to run from dedicated 1 kHz internal clock source or bus clock
- Low-voltage detection with reset or interrupt
- Illegal opcode detection with reset
- Illegal address detection with reset
- Flash protection

Development Support

- Single-wire background debug interface
- Breakpoint capability to allow single breakpoint setting during in-circuit debugging

Peripherals

- **LCD** — Up to 8 × 14 or 4 × 18 segments; compatible with 5 V or 3 V LCD glass displays using on-chip resistor bias network; functional in wait, stop modes for very low power LCD operation; frontplane and backplane pins multiplexed with GPIO functions; selectable frontplane and backplane configurations
- **ADC** — 8-channel, 10-bit resolution; 2.5 µs conversion time; automatic compare function; 1.7 mV/°C temperature sensor; internal bandgap reference channel; operation in stop; fully functional from 2.7 V to 5.5 V
- **TPM** — Two 2-channel 16-bit timer/pulse-width modulator (TPM) modules; selectable input capture, output compare, or buffered edge- or center-aligned PWM on each channel
- **SCI** — One serial communications interface module with optional 13-bit break; LIN extensions
- **KBI** — 8-pin keyborad interrupt module

Input/Output

- 26 GPIOs including 1 output-only pin and 1 input-only pin.
- Hysteresis and configurable pullup device on all input pins; configurable slew rate and drive strength on all output pins.

Package Options

- 28-pin SOIC

MC9RS08LE4 MCU Reference Manual

Covers: MC9RS08LE4

MC9RS08LE4
Rev. 3
5/2015

Revision History

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://freescale.com

The following revision history table summarizes changes contained in this document.

Revision Number	Revision Date	Description of Changes
1	9/2/2008	Initial public released.
2	3/14/2013	Updated the ICS block guide to ICSV1 for the following changes: • Removed the BDC clock • Updated the block diagram • Reserved the bit 4 in the ICSSC register • In the FEI, FEE, FBI, FBE, changed to “the FLL loop will lock the FLL frequency to 512 times the reference frequency, as selected by the RDIV bits”
3	5/2015	Updated Figure 10-2.

List of Chapters

Chapter Number | Title | Page

Contents

Section Number **Title** **Page**

Chapter 1
Device Overview

Chapter 2
Pins and Connections

Chapter 3
Modes of Operation

Chapter 4
Memory

Chapter 5
Resets, Interrupts, and System Configuration

Chapter 6
Parallel Input/Output

Chapter 7
Keyboard Interrupt (S08KBIV2)

Chapter 8
Central Processor Unit (RS08CPUV1)

Chapter 9
Analog-to-Digital Converter (ADC10V1)

Chapter 10
Internal Clock Source (S08ICSV1)

Chapter 11 Serial Communications Interface (S08SCIV4)

Chapter 12 16-Bit Timer/PWM (S08TPMV2)

Chapter 13
Liquid Crystal Display Module (S08LCDV1)

Chapter 14
Development Support

Chapter 1
Device Overview

1.1 Introduction

The MC9RS08LE4 microcontrollers are members of the low-cost RS08 family of 8-bit microcontroller units (MCUs). The device is composed of standard on-chip modules including a very small and highly efficient RS08 CPU core, on-chip RAM, nonvolatile memory, two 16-bit TPM modules, a serial communications interface (SCI), an 8-channel 10-bit analog-to-digital converter (ADC), an 8-pin keyboard interrupt module (KBI), and a liquid crystal display module (LCD).

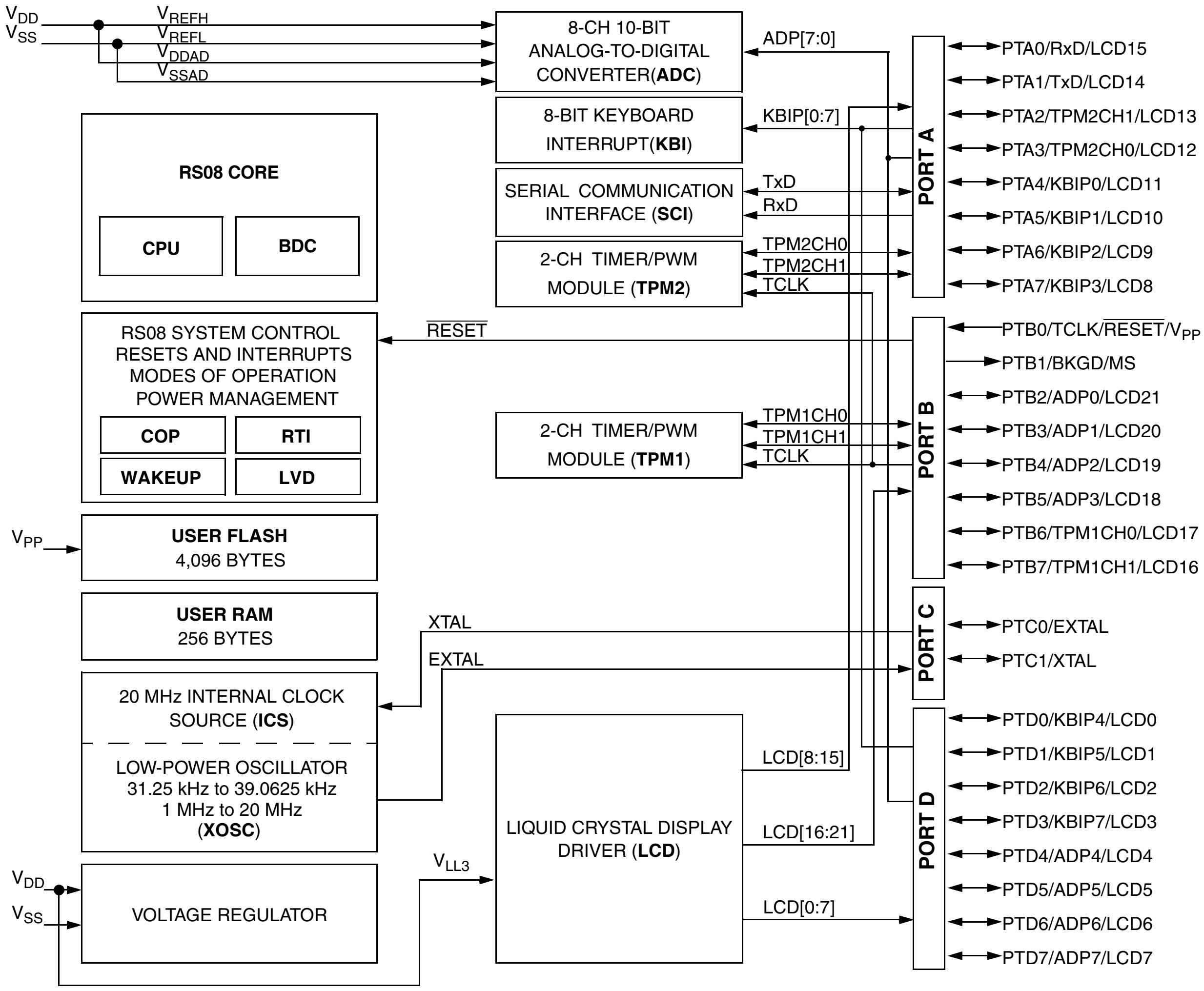
Table 1-1 summarizes the peripheral availability for the device available in the MC9RS08LE4.

Table 1-1. Devices in the MC9RS08LE4 Series

Feature	Device
	MC9RS08LE4
Package	**28-pin SOIC**
FLASH	4,096 Bytes
RAM	256 Bytes
RTI	Yes
ADC	8-ch
KBI	8-pin
SCI	Yes
TPM	2 × 2-ch
LCD	8 × 14 4 × 18
I/O pins	26
Package types	28-pin SOIC

1.2 MCU Block Diagram

The block diagram in Figure 1-1 shows the structure of the MC9RS08LE4 MCU.



NOTES:

1. PTB0/TCLK/$\overline{\text{RESET}}$/V_{PP} is an input-only pin when used as port pin
2. PTB1/BKGD/MS is an output-only pin

Figure 1-1. MC9RS08LE4 Series Block Diagram

Table 1-2 lists the functional versions of the on-chip modules.

Table 1-2. Versions of On-Chip Modules

Module		Version
Analog-to-Digital Converter	(ADC10)	1
Central Processing Unit	(CPU)	1
Keyboard Interrupt	(KBI)	2
Internal Clock Source	(ICS)	1
Oscillator	(XOSC)	1

Table 1-2. Versions of On-Chip Modules (continued)

Module		Version
Serial Communications Interface	(SCI)	4
Timer Pulse-Width Modulator	(TPM)	2
Liquid Crystal Display Module	(LCD)	1

1.3 System Clock Distribution

Figure 1-2 shows a simplified clock connection diagram. Some modules in the MCU have selectable clock inputs as shown in Figure 1-2. The clock inputs to the modules indicate the clock(s) used to drive the module function.

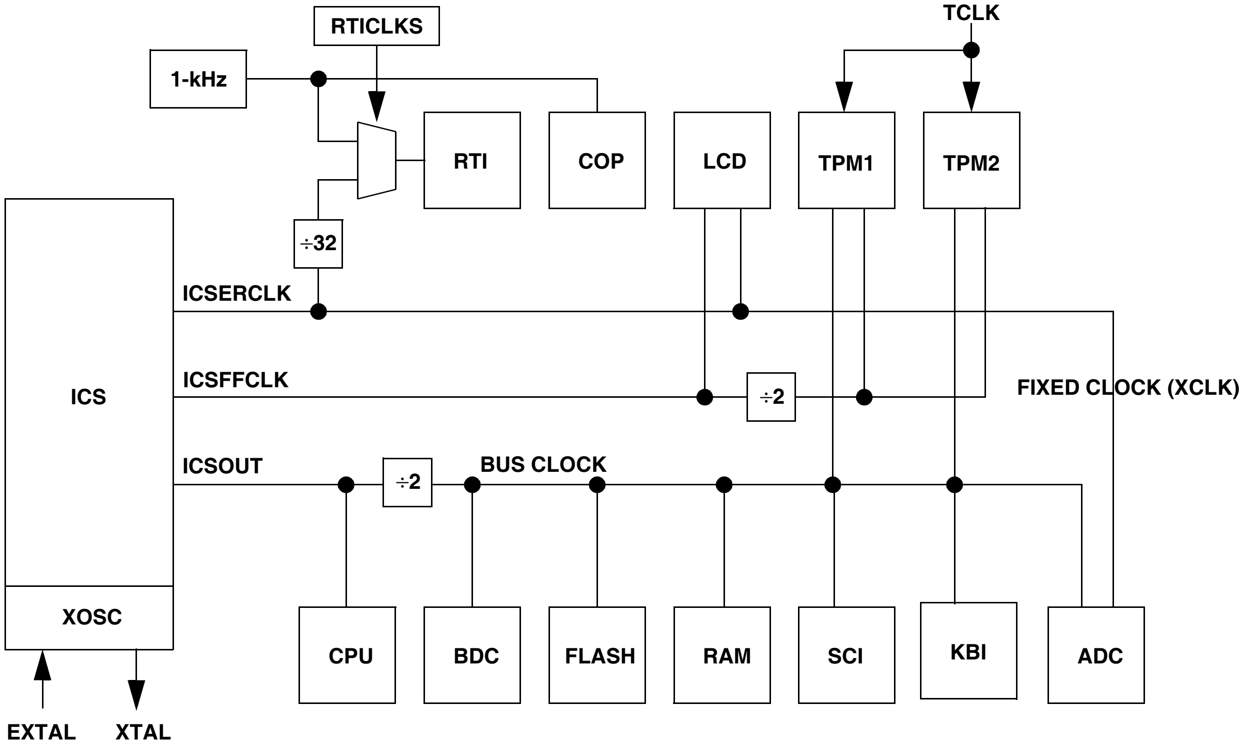


Figure 1-2. System Clock Distribution Diagram

The ICS supplies the following clock sources:

- ICSOUT — This clock source is used as the CPU clock and is divided by 2 to generate the peripheral BUS CLOCK. Control bits in the ICS control registers determine which of three clock sources is connected:
  - — Internal reference clock
  - — External reference clock
  - — Frequency-locked loop (FLL) output

  See Chapter 10, “Internal Clock Source (S08ICSV1)”,” for details on configuring the ICSOUT clock.

- ICSERCLK — This is the external reference clock and can be selected as the alternate clock for the ADC and LCD modules. Section 10.4.6, “Optional External Reference Clock” explains the ICSERCLK in more detail. See Chapter 9, “Analog-to-Digital Converter (ADC10V1)” and Chapter 13, “Liquid Crystal Display Module (S08LCDV1)” for more information regarding the use of ICSERCLK with these modules.
- ICSFFCLK — This clock can be selected as the alternate clock for LCD module. It is divided by 2 to generate the FIXED CLOCK (XCLK) . The FIXED CLOCK can be selected as clock source for the TPM modules. The frequency of the ICSFFCLK is determined by the settings of the ICS. See the Section 10.4.7, “Fixed Frequency Clock” for details.
- TCLK — TCLK is the optional external clock source for the TPM modules. The TCLK must be limited to 1/4th the frequency of the bus clock for synchronization. See Chapter 12, “16-Bit Timer/PWM (S08TPMV2)” for more details.
- 1 kHz — This clock is generated from an internal low power oscillator that is completely independent of the ICS module. The clock can be selected as the clock source to the COP module. It can also See Section 5.4, “Computer Operating Properly (COP) Watchdog”,” for details on using the1 kHz clock with this module.
- RTICLKS — This bit is used to select the clock source of RTI.

Chapter 2
Pins and Connections

2.1 Introduction

This chapter describes signals that connect to package pins. It includes a pinout diagram, a table of signal properties, and detailed discussion of signals.

2.2 Device Pin Assignment

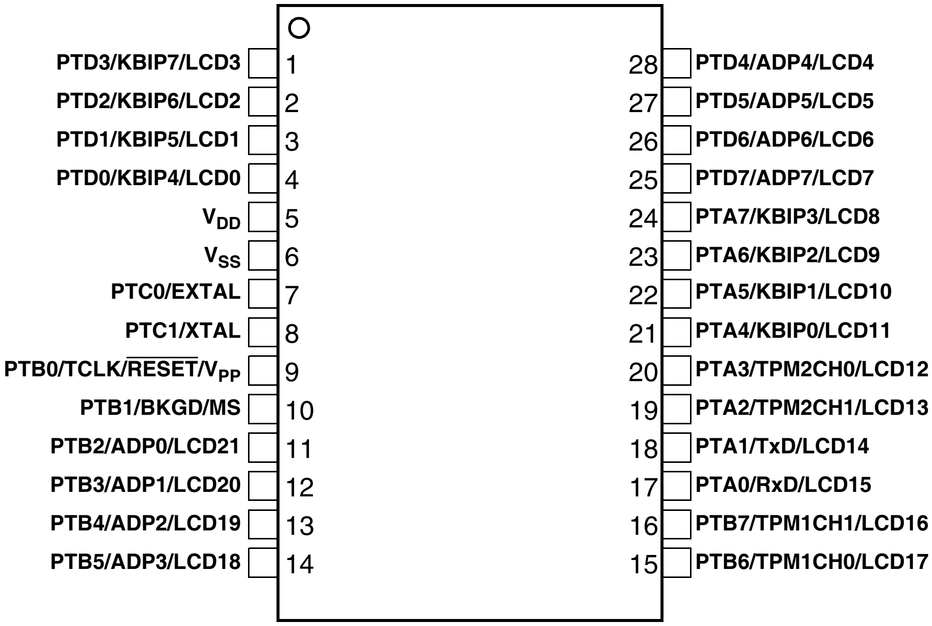
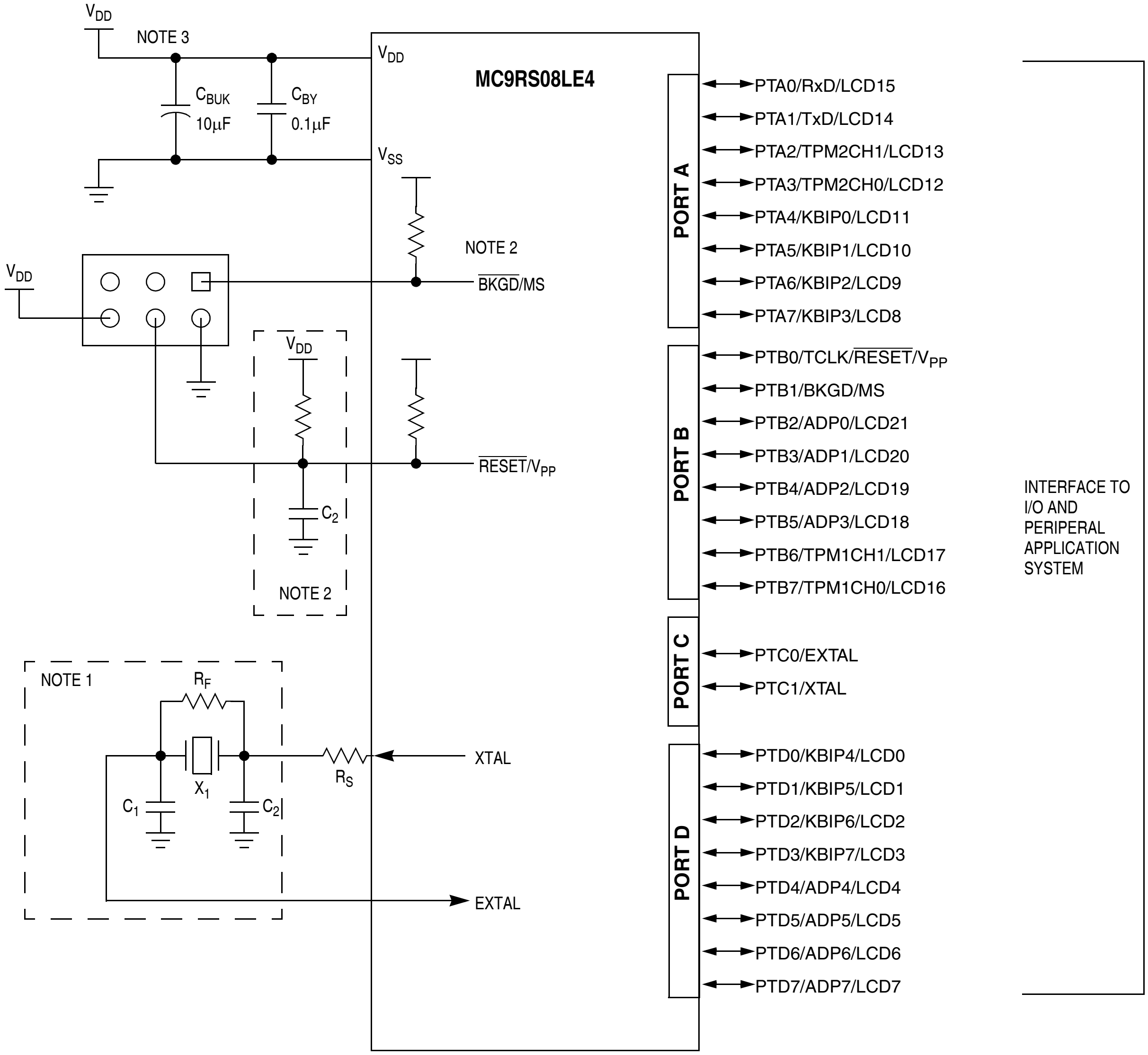


Figure 2-1. MC9RS08LE4 in 28-Pin SOIC Package

2.3 Recommended System Connections

Figure 2-2 shows pin connections that are common to almost all MC9RS08LE4 application systems.



NOTE
1. Not required if using the internal oscillator option.
2. An RC filter on RESET is recommended for EMC-sensitive applications.
3. LCD power supply uses the $V_{DD.}$

Figure 2-2. Basic System Connections

2.3.1 Power (V_{DD}, V_{SS})

V_{DD} and V_{SS} are the primary power-supply pins for the MCU. This voltage source supplies power to all I/O buffer circuitry and to an internal voltage regulator. The internal voltage regulator provides regulated lower-voltage source to the CPU and other internal circuitry of the MCU.

Typically, application systems have two separate capacitors across the power pins. In this case, a bulk electrolytic capacitor, such as a 10 µF tantalum capacitor, provide bulk charge storage for the overall system and a 0.1 µF ceramic bypass capacitor must be located as close to the MCU power pins as practical to suppress high-frequency noise.

2.3.2 Oscillator (XTAL, EXTAL)

Immediately after reset, the MCU uses an internally generated clock provided by the internal clock source (ICS) module. The internal frequency is nominally 8 MHz and the default ICS settings will provide for an 4 MHz bus out of reset.

The oscillator module (XOSC) in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator. Rather than a crystal or ceramic resonator, an external oscillator can be connected to the EXTAL input pin.

R_S (when used) and R_F must be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 must be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

R_F is used to provide a bias path to keep the EXTAL input in its linear range during crystal startup; its value is not generally critical. Typical systems use 1 MΩ to 10 MΩ. Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) could prevent startup.

C1 and C2 are typically in the 5 pF to 25 pF range and are chosen to match the requirements of a specific crystal or resonator. When selecting C1 and C2, consider printed circuit board (PCB) capacitance and MCU pin capacitance. The crystal manufacturer typically specifies a load capacitance that is the series combination of C1 and C2 (which are usually the same size). As a first-order approximation, use 10 pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

2.3.3 PTB0/TCLK/$\overline{\text{RESET}}$/V_{PP} Pin

After a power-on reset (POR) into user mode, the $\overline{\text{RESET}}$/V_{PP} pin defaults to a general-purpose input port pin, PTB0. Setting RSTPE in SOPT configures the pin to be the $\overline{\text{RESET}}$ input pin. After it is configured as $\overline{\text{RESET}}$, the pin will remain as $\overline{\text{RESET}}$ until the next reset. The $\overline{\text{RESET}}$ pin can be used to reset the MCU from an external source when the pin is driven low. When enabled as the $\overline{\text{RESET}}$ pin (RSTPE = 1), the internal pullup device is automatically enabled.

External V_{PP} voltage (typically 12 V, see “Flash Specifications” in MC9RS08LE4 Data Sheet) is required on this pin when performing flash programming or erasing. The V_{PP} connection is always connected to the internal flash module, regardless of the pin function. To avoid stressing the flash memory, external V_{PP} voltage must be removed and voltage higher than V_{DD} must be avoided when flash memory programming or erasing is not taking place.

NOTE

This pin does not contain a clamp diode to V_{DD} and must not be driven above V_{DD} when flash programming or erasing is not taking place.

2.3.4 PTB1/BKGD/MS

During a power-on-reset (POR) or background debug force reset, the PTB1/BKGD/MS pin functions as a mode selection pin. Immediately after reset rises, the pin functions as the background pin and can be used for background debug communication. When functioning as a background/mode selection pin, the pin includes an internal pullup device, input hysteresis, a standard output driver, and no output slew rate control.

The background debug communication function is enabled when BKGDPE in SOPT is set. BKGDPE is set following any reset of the MCU and must be cleared to use the PTB1/BKGD/MS pin’s alternative pin function.

If nothing is connected to this pin, the MCU enters normal operating mode at the rising edge of reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD/MS low during the rising edge of reset, which forces the MCU to active background mode.

The BKGD pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target MCU’s BDC clock-per-bit time. The target MCU’s BDC clock could be as fast as the bus clock rate, so there should never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

Although the BKGD pin is a pseudo open-drain pin, the background debug communication protocol provides brief, actively driven, high speedup pulses to ensure fast rise times. Small capacitances from cables and the absolute value of the internal pullup device play almost no role in determining rise and fall times on the BKGD pin.

2.3.5 LCD Pins

2.3.5.1 LCD Driver (LCD[0:21])

There are 22 LCD driver pins in MC9RS08LE4. All of the LCD driver pins can operate as GPIO. Immediately after reset, the LCD driver pins are high-impedance. LCD/GPIO pins are configured as GPIO by default. For detailed information about LCD driver pins see Chapter 13, “Liquid Crystal Display Module (S08LCDV1)”.

2.3.6 General-Purpose I/O and Peripheral Ports

The MC9RS08LE4 MCU supports up to 26 general-purpose I/O pins, which are shared with on-chip peripheral functions (TPM, serial I/O, ADC, keyboard interrupts, LCD, etc.).

When a port pin is configured as a general-purpose output or a peripheral uses the port pin as an output, software can select one of two drive strengths and enable or disable slew-rate control. When a port pin is configured as a general-purpose input or a peripheral uses the port pin as an input, software can enable a pullup device.

For information about controlling these pins as general-purpose I/O pins, see Chapter 6, “Parallel Input/Output”. For information about how and when on-chip peripheral systems use these pins, see the appropriate module chapter.

Immediately after reset, all pins are configured as high-impedance general-purpose inputs with internal pullup devices disabled.

Table 2-1. Pin Availability by Package Pin-Count

Pin Number	<-- Lowest **Priority** --> Highest			
28	**Port Pin**	**Alt 1**	**Alt 2**	**Alt 3**
1	PTD3		KBIP7	LCD3
2	PTD2		KBIP6	LCD2
3	PTD1		KBIP5	LCD1
4	PTD0		KBIP4	LCD0
5				V_{DD}
6				V_{SS}
7	PTC0		EXTAL	
8	PTC1		XTAL	
9	PTB0	TCLK	$\overline{\text{RESET}}$	V_{PP}
10	PTB1		BKGD	MS
11	PTB2		ADP0	LCD21
12	PTB3		ADP1	LCD20
13	PTB4		ADP2	LCD19
14	PTB5		ADP3	LCD18
15	PTB6		TPM1CH0	LCD17
16	PTB7		TPM1CH1	LCD16
17	PTA0		RxD	LCD15
18	PTA1		TxD	LCD14
19	PTA2		TPM2CH1	LCD13
20	PTA3		TPM2CH0	LCD12
21	PTA4		KBIP0	LCD11
22	PTA5		KBIP1	LCD10
23	PTA6		KBIP2	LCD9
24	PTA7		KBIP3	LCD8
25	PTD7		ADP7	LCD7
26	PTD6		ADP6	LCD6
27	PTD5		ADP5	LCD5
28	PTD4		ADP4	LCD4

NOTE

When an alternative function is enabled, it is possible to get a spurious edge to the module and user software must clear out any associated flags before interrupts are enabled. Table 2-1 illustrates the priority if multiple modules are enabled. The highest priority module has control over the pin. Selecting a higher priority pin function with a lower priority function already enabled can cause spurious edges to the lower priority module. Disable all modules that share a pin before enabling another module.

Chapter 3
Modes of Operation

3.1 Introduction

The operating modes of the MC9RS08LE4 are described in this section. Entry into each mode, exit from each mode, and functionality while in each mode are described.

3.2 Features

- Active background mode for code development
- Run mode:
  - CPU running
  - System clocks running
  - Full voltage regulation is maintained
- Wait mode:
  - CPU halts operation to conserve power
  - System clocks running
  - Full voltage regulation is maintained
- Stop modes:
  - CPU and bus clocks stopped
  - System clocks are stopped;
  - All internal circuits remain powered for fast recovery

3.3 Run Mode

This is the normal operating mode for the MC9RS08LE4 series. This mode is selected when the BKGD/MS pin is high at the rising edge of reset. In this mode, the CPU executes code from internal memory with execution beginning at the address $3FFD. A JMP instruction (opcode $BC) with operand located at $3FFE–$3FFF must be programmed for correct reset operation into the vector fetching process as in HC08/S08 families, the user program is responsible for performing a JMP instruction to relocate the program counter to the correct user program start location.

3.4 Active Background Mode

The active background mode functions are managed through the background debug controller (BDC) in the RS08 core. The BDC provides the means for analyzing MCU operation during software development.

Active background mode is entered in any of four ways:

- When the BKGD/MS pin is low during POR or immediately after issuing a background debug force reset
- When a BACKGROUND command is received through the BKGD/MS pin
- When a BGND instruction is executed
- When a BDC breakpoint is encountered

After entering active background mode, the CPU is held in a suspended state waiting for serial background commands rather than executing instructions from the user application program.

There are two types of background commands :

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:
  - Memory access commands
  - Memory access with status commands
  - The BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode. Active background commands include commands to:
  - Read or write CPU registers
  - Trace one user program instruction at a time
  - Leave active background mode to return to the user application program (GO)

The active background mode is used to program user application program into the flash program memory before the MCU is operated in run mode for the first time. When the MC9RS08LE4 is shipped, the flash program memory is erased unless specifically noted; therefore, there is no program that can be executed in run mode until the flash memory is initially programmed.

For additional information about the active background mode, refer to Chapter 14, “Development Support”.

3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The program counter (PC) is halted at the position where the WAIT instruction is executed. When an interrupt request occurs:

1. MCU exits wait mode and resumes processing.
2. PC is incremented by one and fetches the next instruction to be processed.

The user program must poll the corresponding interrupt source that woke the MCU, because no vector fetching process is involved.

While the MCU is in wait mode, not all background debug commands can be used. Only the BACKGROUND command and memory access with status commands are available. The memory access with status commands do not allow memory access, but they report an error indicating that the MCU is in stop or wait mode. The BACKGROUND command can be used to wake the MCU from wait mode and enter active background mode.

Table 3-1 summarizes the behavior of the MCU in wait mode.

Table 3-1. Wait Mode Behavior

Mode	CPU	Regulator	ICS	I/O Pins	RTI	LCD	ADC	Digital Peripherals
Wait	Standby	On	On	States Held	Optionally On	Optionally On	Optionally On	Optionally On

3.6 Stop Modes

Stop mode is entered upon execution of a STOP instruction when the STOPE bit in the system option register is set. In stop mode, all internal clocks to the CPU and the modules are halted. If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU does not enter stop mode and an illegal opcode reset is forced.

Table 3-2 summarizes the behavior of the MCU in stop mode.

Table 3-2. Stop Mode Behavior

Mode	CPU	Regulator	ICS[1]	I/O Pins	RTI	LCD[2]	ADC[3]	Digital Peripherals
Stop	Standby	Optionally On	Optionally On	States Held	Optionally On	Optionally On	Optionally On	Standby

[1] ICS requires IREFSTEN = 1 and LVDE and LVDSE must be set to allow operations in stop.

[2] Requires the asynchronous LCD reference, LVDE, and LVDESE bits in the SPMSC1 to be set, otherwise LCD is in standby mode.

[3] Requires the asynchronous ADC reference, LVDE, and LVDESE bits in the SPMSC1 to be set, otherwise ADC is in standby mode.

Upon entering stop modes, all of the clocks in the MCU are halted. The ICS is turned off by default when the IREFSTEN bit is cleared and the voltage regulator is put in standby. The states of all of the internal registers and logic, as well as the RAM content, are maintained. The I/O pin states are held.

Exit from stop is done by asserting $\overline{\text{RESET}}$, any enabled asynchronous interrupt, or the real-time interrupt. The asynchronous interrupts include the KBI pins, LVD interrupt, or ADC interrupt.

If stop is exited by asserting the $\overline{\text{RESET}}$ pin, the MCU will be reset and program execution starts at location $3FFD. If exited by means of an asynchronous interrupt or real-time interrupt, the MCU increments the program counter (PC), which halts at the location where the STOP instruction is executed, and the next instruction is fetched and executed accordingly. The user program must probe for the corresponding interrupt source that wakes the CPU.

A separate self-clocked source (≈1 kHz) for the real-time interrupt allows a wakeup from stop mode with no external components. When RTIS = 000, the real-time interrupt function and the 1 kHz source are disabled. Power consumption is lower when the 1 kHz source is disabled, but in that case, the real-time interrupt cannot wake the MCU from stop.

The external reference clock can also be enabled for the real-time interrupt to allow a wakeup from stop mode. The external clock reference is enabled by setting the EREFSTEN bit. For the ICS to run in stop mode, the LVDE and LVDSE bits in the SPMSC1 must be set before entering the stop.

To enable ADC in STOP mode the asynchronous ADC clock and LVD must be enabled by setting LVDE and LVDSE, otherwise the ADC is in standby.

3.6.1 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in Chapter 14, “Development Support”. If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation.

Most background commands are not available in stop mode. The memory access with status commands do not allow memory access, but they report an error indicating that the MCU is in stop or wait mode. The BACKGROUND command can be used to wake the MCU from stop and enter active background mode if the ENBDM bit is set. After entering background debug mode, all BACKGROUND commands are available.

Table 3-3 summarizes the behavior of the MCU in stop mode when entry into the active background mode is enabled.

Table 3-3. BDM Enabled Stop Mode Behavior

Mode	CPU	Regulator	ICS	I/O Pins	RTI	LCD[1]	ADC[2]	Digital Peripherals
Stop	Standby	On	On	States Held	Optionally On	Optionally On	Optionally On	Standby

[1] Requires the asynchronous LCD reference, LVDE, and LVDESE bits in the SPMSC1 to be set, otherwise LCD is in standby mode.

[2] Requires the asynchronous ADC reference, LVDE, and LVDESE bits in the SPMSC1 to be set, otherwise ADC is in standby mode.

3.6.2 LVD Enabled in Stop Mode

The LVD system is capable of generating an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop mode (LVDE and LVDSE bits in SPMSC1 both set) at the time the CPU executes a STOP instruction, the voltage regulator remains active.

Table 3-4 summarizes the behavior of the MCU in stop mode when LVD reset is enabled.

Table 3-4. LVD Enabled Stop Mode Behavior

Mode	CPU	Regulator	ICS	I/O Pins	RTI	LCD[1]	ADC[2]	Digital Peripherals
Stop	Standby	On	Optionally On	States Held	Optionally On	Optionally On	Optionally On	Standby

[1] Requires the asynchronous LCD reference, LVDE, and LVDESE bits in the SPMSC1 to be set, otherwise LCD is in standby mode.

[2] Requires the asynchronous ADC reference, LVDE, and LVDESE bits in the SPMSC1 to be set, otherwise ADC is in standby mode.

Chapter 4
Memory

4.1 MC9RS08LE4 Series Memory Map

Figure 4-1 shows the memory map for the MC9RS08LE4 series. On-chip memory in the MC9RS08LE4 MCU consists of RAM, flash program memory for nonvolatile data storage, plus I/O and control/status registers.

The memory map is associated as follows:

- Fast access RAM using tiny and short instructions ($0000–$000D)
- Indirect data access D[X] ($000E)
- Index register X for D[X] ($000F)
- Frequently used peripheral registers ($0010–$001E, $0020–$004F)
- PAGESEL register ($001F)
- RAM for MC9RS08LE4 ($0050–$00BF, $0100–$017F)
- Paging window ($00C0–$00FF)
- High page registers ($0200–$027F)
- Nonvolatile memory for MC9RS08LE4 ($3000–$3FFF)

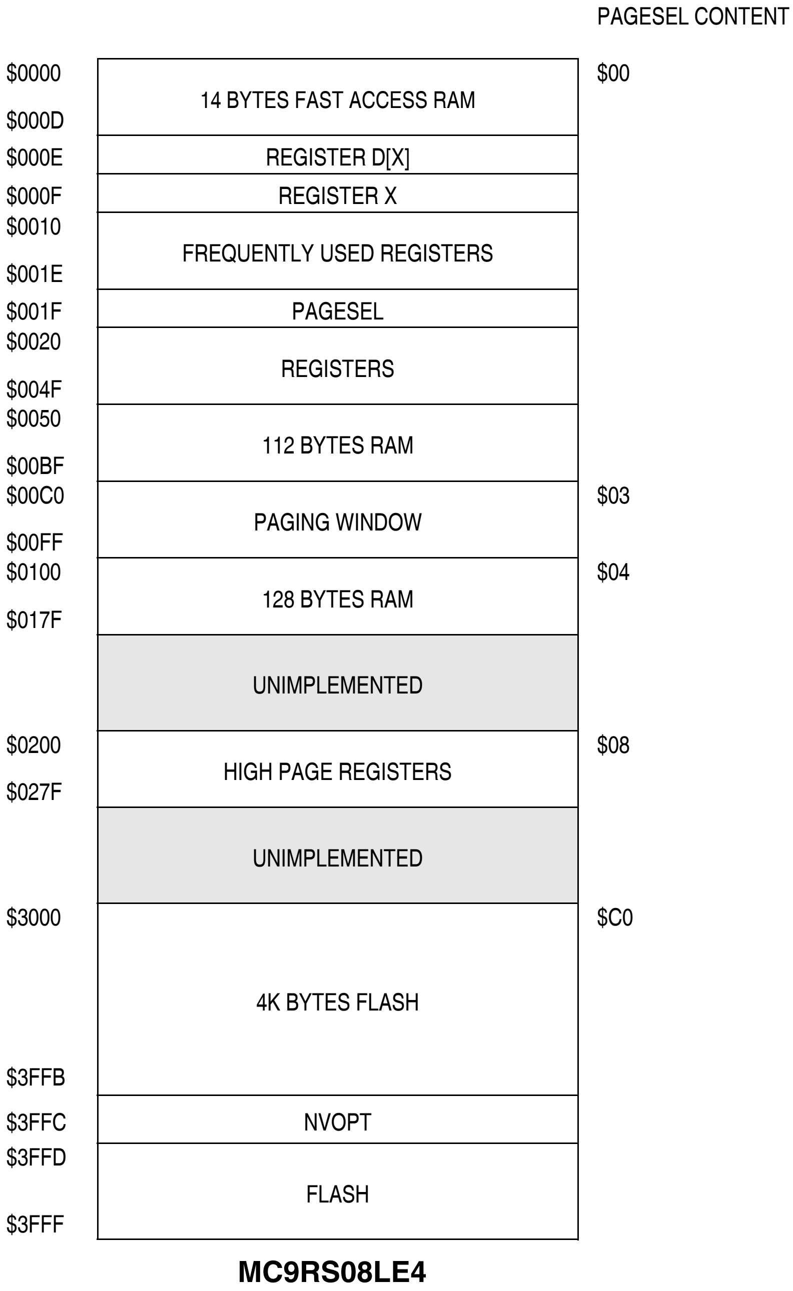


Figure 4-1. MC9RS08LE4 Series Memory Map

4.2 Unimplemented Memory

Attempting to access data or execute an instruction fetched from any unimplemented memory address causes reset.

4.3 Indexed/Indirect Addressing

Register D[X] and register X together perform the indirect data access. Register D[X] is mapped to address $000E. Register X is located in address $000F. The 8-bit register X contains the address that is used when register D[X] is accessed. Register X is cleared to zero upon reset. By programming register X, any location on the first page ($0000–$00FF) can be read/written via register D[X]. Figure 4-2 shows the relationship between D[X] and register X. For example, in HC08/S08 syntax *lda,x* is comparable to *lda D[X]* in RS08 coding when register X has been programmed with the index value.

The physical location of $000E is in RAM. Accessing the location through D[X] returns $000E RAM content when register X contains $0E. The physical location of $000F is register X, itself. Reading the location through D[X] returns register X content; writing to the location modifies register X.

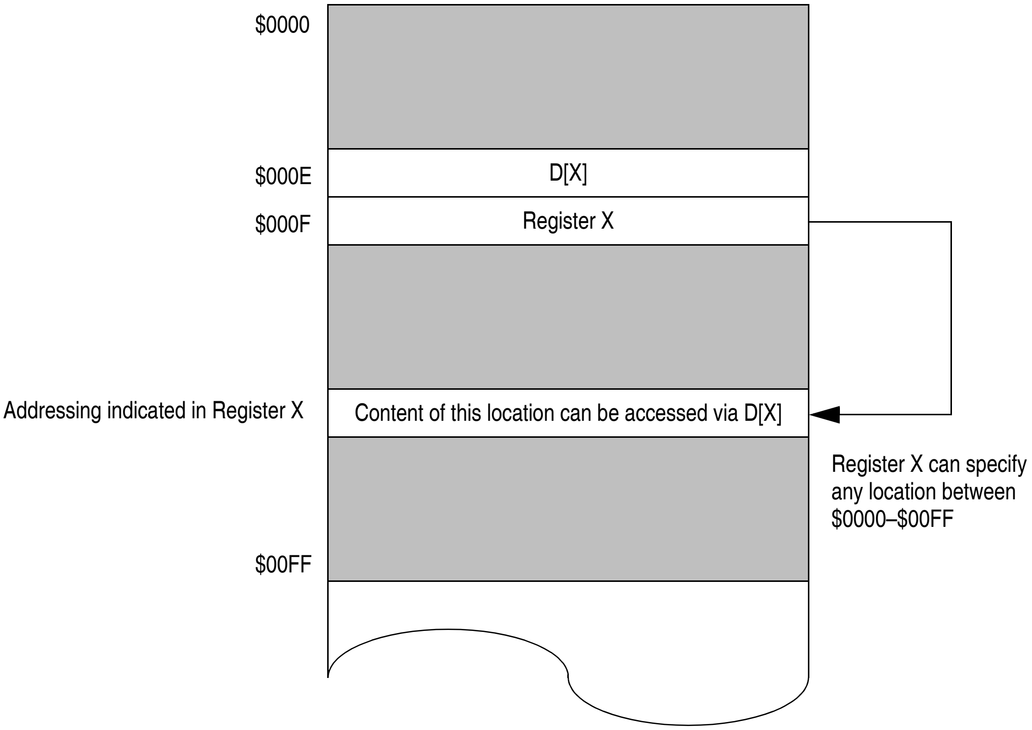

$0000
$000E
D[X]
$000F
Register X
Addressing indicated in Register X
Content of this location can be accessed via D[X]
Register X can specify any location between $0000–$00FF
$00FF

Figure 4-2. Relationship between Register D[X] and Register X

4.4 Register Addresses and Bit Assignments

The fast access RAM area can be accessed by using tiny, short, and direct addressing mode instructions. For tiny addressing mode instructions, the operand is encoded along with the opcode to a single byte.

Frequently used registers can make use of the short addressing mode instructions for faster load, store, and clear operations. For short addressing mode instructions, the operand is encoded along with the opcode to a single byte.

Table 4-1. Register Summary (Sheet 1 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$0000– $000D		Fast Access RAM							
$000E	D[X]	Bit 7	6	5	4	3	2	1	Bit 0
$000F	X	Bit 7	6	5	4	3	2	1	Bit 0
$0010	ADCSC1	COCO	AIEN	ADCO	ADCH				
$0011	ADCSC2	ADACT	ADTRG	ACFE	ACFGT	0	0	0	0
$0012	ADCRH	0	0	0	0	0	0	ADR9	ADR8
$0013	ADCRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
$0014	ADCCVH	0	0	0	0	—	—	ADCV9	ADCV8
$0015	ADCCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
$0016	ADCCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
$0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
$0018	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
$0019	SOPT	COPE	COPT	STOPE	0	0	0	BKGDPE	RSTPE
$001A	SIP1	LVD	SCIT	SCIR	SCIE	ADC	KBI	LCD	RTI
$001B	SIP2	—	TPM2CH1	TPM2CH0	TPM2	—	TPM1CH1	TPM1CH0	TPM1
$001C	KBISC	0	0	0	0	KBF	KBACK	KBIE	KBMOD
$001D	KBIPE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
$001E	KBIES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
$001F	PAGESEL	Bit 7	6	5	4	3	2	1	Bit 0
$0020	LCDWF0	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
$0021	LCDWF1	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
$0022	LCDWF2	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
$0023	LCDWF3	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
$0024	LCDWF4	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
$0025	LCDWF5	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
$0026	LCDWF6	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
$0027	LCDWF7	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
$0028	LCDWF8	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8
$0029	LCDWF9	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
$002A	LCDWF10	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
$002B	LCDWF11	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
$002C	LCDWF12	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
$002D	LCDWF13	BPHLCD13	BPGLCD13	BPFLCD13	BPELCD13	BPDLCD13	BPCLCD13	BPBLCD13	BPALCD13
$002E	LCDWF14	BPHLCD14	BPGLCD14	BPFLCD14	BPELCD14	BPDLCD14	BPCLCD14	BPBLCD14	BPALCD14
$002F	LCDWF15	BPHLCD15	BPGLCD15	BPFLCD15	BPELCD15	BPDLCD15	BPCLCD15	BPBLCD15	BPALCD15
$0030	LCDWF16	BPHLCD16	BPGLCD16	BPFLCD16	BPELCD16	BPDLCD16	BPCLCD16	BPBLCD16	BPALCD16
$0031	LCDWF17	BPHLCD17	BPGLCD17	BPFLCD17	BPELCD17	BPDLCD17	BPCLCD17	BPBLCD17	BPALCD17
$0032	LCDWF18	BPHLCD18	BPGLCD18	BPFLCD18	BPELCD18	BPDLCD18	BPCLCD18	BPBLCD18	BPALCD18
$0033	LCDWF19	BPHLCD19	BPGLCD19	BPFLCD19	BPELCD19	BPDLCD19	BPCLCD19	BPBLCD19	BPALCD19
$0034	LCDWF20	BPHLCD20	BPGLCD20	BPFLCD20	BPELCD20	BPDLCD20	BPCLCD20	BPBLCD20	BPALCD20
$0035	LCDWF21	BPHLCD21	BPGLCD21	BPFLCD21	BPELCD21	BPDLCD21	BPCLCD21	BPBLCD21	BPALCD21

Table 4-1. Register Summary (Sheet 2 of 4) (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$0036	Reserved	—	—	—	—	—	—	—	—
$0037	Reserved	—	—	—	—	—	—	—	—
$0038	Reserved	—	—	—	—	—	—	—	—
$0039	Reserved	—	—	—	—	—	—	—	—
$003A	Reserved	—	—	—	—	—	—	—	—
$003B	Reserved	—	—	—	—	—	—	—	—
$003C	Reserved	—	—	—	—	—	—	—	—
$003D	Reserved	—	—	—	—	—	—	—	—
$003E	Reserved	—	—	—	—	—	—	—	—
$003F	Reserved	—	—	—	—	—	—	—	—
$0040	LCDC0	LCDEN	SOURCE	LCLK2	LCLK1	LCLK0	DUTY2	DUTY1	DUTY0
$0041	LCDC1	LCDIEN	0	0	0	0	LCDEN	LCDWAI	LCDSTP
$0042	LCDSUPPLY	CPSEL	0	LADJ1	LADJ0	0	0	VSUPPLY1	VSUPPLY0
$0043	Reserved	—	—	—	—	—	—	—	—
$0044	LCDBCTL	BLINK	ALT	BLANK	0	BMODE	BRATE2	BRATE1	BRATE0
$0045	LCDS	LCDIF	0	0	0	0	0	0	0
$0046	PTAD	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
$0047	PTADD	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
$0048	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
$0049	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	0	0
$004A	PTCD	0	0	0	0	0	0	PTCD1	PTCD0
$004B	PTCDD	0	0	0	0	0	0	PTCDD1	PTCDD0
$004C	PTDD	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
$004D	PTDDD	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
$004E	Unimplemented	—	—	—	—	—	—	—	—
$004F		—	—	—	—	—	—	—	—
$0050– $00BF		RAM							
$00C0– $00FF		Paging Window							
$0100– $017F		RAM							
$0180– $019F	Unimplemented	— —	— —	— —	— —	— —	— —	— —	— —
$0200	PTAPE	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
$0201	PTAPUD	PTAPUD7	PTAPUD6	PTAPUD5	PTAPUD4	PTAPUD3	PTAPUD2	PTAPUD1	PTAPUD0
$0202	PTADS	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
$0203	PTASE	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
$0204	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	0	PTBPE0
$0205	PTBPUD	PTBPUD7	PTBPUD6	PTBPUD5	PTBPUD4	PTBPUD3	PTBPUD2	0	PTBPUD0
$0206	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	0
$0207	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	0

Table 4-1. Register Summary (Sheet 3 of 4) (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$0208	PTCPE	0	0	0	0	0	0	PTCPE1	PTCPE0
$0209	PTCPUD	0	0	0	0	0	0	PTCPUD1	PTCPUD0
$020A	PTCDS	0	0	0	0	0	0	PTCDS1	PTCDS0
$020B	PTCSE	0	0	0	0	0	0	PTCSE1	PTCSE0
$020C	PTDPE	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
$020D	PTDPUD	PTDPUD7	PTDPUD6	PTDPUD5	PTDPUD4	PTDPUD3	PTDPUD2	PTDPUD1	PTDPUD0
$020E	PTDDS	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
$020F	PTDSE	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
$0210	SCIBDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
$0211	SCIBDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
$0212	SCIC1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
$0213	SCIC2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
$0214	SCIS1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
$0215	SCIS2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
$0216	SCIC3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
$0217	SCID	Bit 7	6	5	4	3	2	1	Bit 0
$0218	Reserved	—	—	—	—	—	—	—	—
$0219	Reserved	—	—	—	—	—	—	—	—
$021A	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
$021B	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
$021C	SRTISC	RTIF	RTIACK	RTICLKS	RTIE	0	RTIS		
$021D	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
$021E	Reserved	—	—	—	—	—	—	—	—
$021F	Reserved	—	—	—	—	—	—	—	—
$0220	TPM1SC	TOF	TOIE	CPWMS	CLKS		PS		
$0221	TPM1CNTH	Bit 15	14	13	12	11	10	9	8
$0222	TPM1CNTL	7	6	5	4	3	2	1	Bit 0
$0223	TPM1MODH	Bit 15	14	13	12	11	10	9	8
$0224	TPM1MODL	7	6	5	4	3	2	1	Bit 0
$0225	TPM1C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
$0226	TPM1C0VH	Bit 15	14	13	12	11	10	9	8
$0227	TPM1C0VL	7	6	5	4	3	2	1	Bit 0
$0228	TPM1C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
$0229	TPM1C1VH	Bit 15	14	13	12	11	10	9	8
$022A	TPM1C1VL	7	6	5	4	3	2	1	Bit 0
$022B	Reserved	—	—	—	—	—	—	—	—
$022C	ICSC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
$022D	ICSC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
$022E	ICSTRM	TRIM							
$022F	ICSSC	0	0	0	0	CLKST		OSCINIT	FTRIM
$0230	TPM2SC	TOF	TOIE	CPWMS	CLKS		PS		
$0231	TPM2CNTH	Bit 15	14	13	12	11	10	9	8

Table 4-1. Register Summary (Sheet 4 of 4) (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$0232	TPM2CNTL	7	6	5	4	3	2	1	Bit 0
$0233	TPM2MODH	Bit 15	14	13	12	11	10	9	8
$0234	TPM2MODL	7	6	5	4	3	2	1	Bit 0
$0235	TPM2C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	—	—
$0236	TPM2C0VH	Bit 15	14	13	12	11	10	9	8
$0237	TPM2C0VL	7	6	5	4	3	2	1	Bit 0
$0238	TPM2C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	—	—
$0239	TPM2C1VH	Bit 15	14	13	12	11	10	9	8
$023A	TPM2C1VL	7	6	5	4	3	2	1	Bit 0
$023B	Reserved	—	—	—	—	—	—	—	—
$023C	FOPT	0	0	0	0	0	0	0	SECD
$023D	FLCR	0	0	0	0	HVEN	MASS	0	PGM
$023E	Unimplemented	—	—	—	—	—	—	—	—
$023F		—	—	—	—	—	—	—	—
$0240	LCDPEN0	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0
$0241	LCDPEN1	PEN15	PEN14	PEN13	PEN12	PEN11	PEN10	PEN9	PEN8
$0242	LCDPEN2	—	—	PEN21	PEN20	PEN19	PEN18	PEN17	PEN16
$0243	Reserved	—	—	—	—	—	—	—	—
$0244	Reserved	—	—	—	—	—	—	—	—
$0245	Reserved	—	—	—	—	—	—	—	—
$0246	Reserved	—	—	—	—	—	—	—	—
$0247	Reserved	—	—	—	—	—	—	—	—
$0248	LCDBPEN0	BPEN7	BPEN6	BPEN5	BPEN4	BPEN3	BPEN2	BPEN1	BPEN0
$0249	LCDBPEN1	BPEN15	BPEN14	BPEN13	BPEN12	BPEN11	BPEN10	BPEN9	BPEN8
$024A	LCDBPEN2	—	—	BPEN21	BPEN20	BPEN19	BPEN18	BPEN17	BPEN16
$024B	Reserved	—	—	—	—	—	—	—	—
$024C	Reserved	—	—	—	—	—	—	—	—
$024D	Reserved	—	—	—	—	—	—	—	—
$024E	Reserved	—	—	—	—	—	—	—	—
$024F	Reserved	—	—	—	—	—	—	—	—
$0250–$027F	Unimplemented	Unimplemented							
$3FFA[1]	Reserved	ICS Trim							
$3FFB[1]	Reserved	—	—	—	—	—	—	—	FTRIM
$3FFC	NVOPT	0	0	0	0	0	0	0	SECD

— =Unimplemented or Reserved

[1] If using the MCU untrimmed, $3FFA and $3FFB may be used by applications.

4.5 RAM (System RAM)

The MC9RS08LE4 includes three sections of static RAM. The locations from $0000 to $000D can be directly accessed using the more efficient tiny addressing mode instructions and short addressing mode instructions. Location $000E RAM can be accessed through D[X] register when register X is $0E or through the paging window location $00CE when PAGESEL register is $00. The second section of RAM starts from $0050 to $00BF, and it can be accessed using direct addressing mode instructions.

The RAM retains data when the MCU is in low-power wait and stop mode. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention.

4.6 Flash

The flash memory is primarily for program storage. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because the device does not include on-chip charge pump circuitry, external V_{PP} is required for program and erase operations.

4.6.1 Features

Features of the flash memory include:

- Flash memory size
  - MC9RS08LE4: 4,096 bytes
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Security feature for flash memory

4.6.2 Flash Programming Procedure

Flash memory is programed on a row basis. A row consists of 64 consecutive bytes starting from addresses $XX00, $XX40, $XX80, or $XXC0. Use this procedure to program a row of flash memory.

1. Apply external V_{PP}.
2. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
3. Write any data to the flash memory, via the high page accessing window $00C0–$00FF, within the address range of the row to be programmed. (Prior to the data writing operation, the PAGE register must be configured correctly to map the high page accessing window to the corresponding flash memroy.)
4. Wait for a time, t_{nvs} (minimum 5 μs).
5. Set the HVEN bit.
6. Wait for a time, t_{pgs} (minimum 10 μs).
7. Write data to the flash memory location to be programmed.
8. Wait for time, t_{prog} (20 μs to 40 μs).
9. Repeat steps 6 and 7 until all bytes within the row are programmed.

10. Clear the PGM bit.
11. Wait for time, t_{nvh} (minimum 5 μs).
12. Clear the HVEN bit.
13. After time, t_{rcv} (1 μs), the memory can be accessed in read mode again.
14. Remove external V_{PP}.

This process is repeated throughout the memory until all data is programmed.

NOTE

Flash memory cannot be programmed or erased by software code executed in flash memory. To program or erase flash memory, commands must be executed from RAM or BDC commands. User code should not enter wait or stop during erase or program sequence.

These operations must be performed in the order shown, other unrelated operations may occur between the steps.

4.6.3 Flash Memory Mass Erase Operation

Use this procedure to mass erase the entire flash memory.

1. Apply external V_{PP}.
2. Set the MASS bit in the flash memory control register.
3. Write any data to any flash memory, via the high page accessing window $00C0–$00FF. (Prior to the data writing operation, the PAGE register must be configured correctly to map the high page accessing window to the any FLASH locations).
4. Wait for a time, t_{nvs}.
5. Set the HVEN bit.
6. Wait for a time t_{merase}.
7. Clear the MASS bit.
8. Wait for a time, t_{nvhl}.
9. Clear the HVEN bit.
10. After time, t_{rcv}, the memory can be accessed in read mode again.
11. Remove external V_{PP}.

NOTE

Flash memory cannot be programmed or erased by software code executed from flash memory. To program or erase flash memory, commands must be executed from RAM or BDC commands. User code should not enter wait or stop mode during erase or program sequence.

These operations must be performed in the order shown, but other unrelated operations may occur between the steps.

4.6.4 Flash Memory Security

The MC9RS08LE4 includes circuitry to prevent unauthorized access to the contents of flash memory. When security is engaged, flash memory is considered a secure resource. The RAM, direct-page registers, and background debug controller are considered unsecured resources. The attempts to access a secure memory are blocked (reads return all 0s) when it is through the background debug interface or when BKGDPE is set.

Security is engaged or disengaged based on the state of a nonvolatile register bit (SECD) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from flash memory into the working FOPT register in high-page register space. Engage security by programming the NVOPT location, which can be executed at the same time the flah memory is programmed (SECD = 0). The next time the device is reset via POR, internal reset, or external reset, security is engaged. To disengage security, a mass erase must be performed via BDM commands and followed by pin reset or POR.

The separate background debug controller can also be used for registers and RAM access. Flash memory mass erase is possible by writing to the flash memory control register that follows the flash memory mass erase procedure listed in Section 4.6.3, “Flash Memory Mass Erase Operation” via BDM commands.

Security can always be disengaged through the background debug interface by these steps:

1. Mass erase flash memory via background BDM commands or RAM loaded program.
2. Perform reset and the device boots up with security disengaged.

NOTE

When the device boots up to normal operating mode, where MS pin is high during reset, with SECD programmed (SECD = 0), flash memory security is engaged. BKGDPE is reset to 0, all BDM communication is blocked, and background debug is not allowed.

4.6.5 Flash Memory Registers and Control Bits

The flash memory module has a nonvolatile register NVOPT ($3FFC) in flash memory that is copied into the corresponding control register, FOPT ($023C), at reset.

4.6.6 Flash Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from flash memory to FOPT. Bits 7 through 1 are not used and always read 0. This register may be read at any time, but writing to it has no meaning or effect.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	SECD
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	This register is loaded from nonvolatile location NVOPT during reset.							

▨ = Unimplemented or Reserved

Figure 4-3. Flash Options Register (FOPT)

Table 4-2. FOPT Register Field Descriptions

Field	Description
0 SECD	**Security State Code** — This bit field determines the security state of the MCU. When the MCU is secured, the contents flash memory cannot be accessed by instructions from any unsecured source including the background debug interface. For more detailed information about security, refer to Section 4.6.4, “Flash Memory Security” 0 Security engaged 1 Security disengaged

4.6.7 Flash Control Register (FLCR)

	7	6	5	4	3	2	1	0
R	0	0	0	0	HVEN	MASS	0	PGM
W	▨	▨	▨	▨			▨	
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 4-4. FLASH Control Register (FLCR)

Table 4-3. FLCR Register Field Descriptions

Field	Description
3 HVEN	**High Voltage Enable** — This read/write bit enables high voltages to the flash array for program and erase operation. HVEN can be set only if PGM = 1 or MASS = 1 and the proper sequence for program or erase is followed. 0 High voltage disabled to array. 1 High voltage enabled to array.
2 MASS	**Mass Erase Control Bit** — This read/write bit configures the memory for mass erase operation. 0 Mass erase operation not selected. 1 Mass erase operation selected.
0 PGM[1]	**Program Control Bit** — This read/write bit configures the memory for program operation. PGM is interlocked with the MASS bit so both bits cannot be equal to 1 or set to 1 at the same time. 0 Program operation not selected. 1 Program operation selected.

[1] When flash memory security is engaged, writing to PGM bit has no effect. As a result, flash memory programming is not allowed.

4.6.8 Page Select Register (PAGESEL)

There is a 64-byte window ($00C0–$00FF) in the direct-page reserved for paging access. Programming the page select register determines the corresponding 64-byte block on the memory map for direct-page access. For example, when the PAGESEL register is programmed with value $08, the high-page register ($0200–$013F) can be accessed through the paging window ($00C0–$00FF) via direct addressing mode instructions.

	7	6	5	4	3	2	1	0
R W	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6
Reset	0	0	0	0	1	0	0	0

Figure 4-5. Page Select Register (PAGESEL)

Table 4-4. PAGESEL Field Descriptions

Field	Description
7:0 AD[13:6]	**Page Selector** — These bits define the address line bit 6 to bit 13, which determines the 64-byte block boundary of the memory block accessed via the direct paging window.

Start address of memory block selected									0	0	0	0	0	0
	AD[13:6]								▨	▨	▨	▨	▨	▨

Figure 4-6. Memory Block Boundary Selector

Table 4-5 shows the memory block to be accessed through paging window ($00C0–$00FF).

Table 4-5. Paging Window

Page	Memory	Contents
$00	$0000–$003F	RAM, D[X], X, PAGESEL, and registers
$01	$0040–$007F	RAM and registers
$02	$0080–$00BF	RAM
$03	$00C0–$00FF	Paging window itself
$06–$07	$0180–$01FF	Not available
$08–$09	$0200–$027F	High page registers
$0A–$7F	$0280–$1FFF	Not available
$80–$FF	$2000–$3FFF	Flash

Chapter 5
Resets, Interrupts, and System Configuration

5.1 Introduction

This chapter describes basic reset and interrupt mechanisms and the various sources of reset and interrupt in the MC9RS08LE4 series. Some interrupt sources from peripheral modules are discussed in detail in other chapters. This chapter contains basic information about all reset and interrupt sources. A few reset and interrupt sources, including the computer operating properly (COP) watchdog, are not part of on-chip peripheral systems but are part of the system control logic.

5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configurations and reliable operations
- System reset status register (SRS) to indicate source of most recent reset
- System interrupt pending registers (SIP1 and SIP2) to indicate status of pending system interrupts
  - — KBI interrupt with enable
  - — ADC interrupt with enable
  - — SCI interrupt with enable
  - — TPM1 and TPM2 interrupt with enable
  - — LCD interrupt with enable
  - — LVD interrupt with enable
  - — RTI interrupt with enable

5.3 MCU Reset

Resetting the MCU provides a way to start processing from known set of initial conditions. During reset, most control and status registers are forced to initial values and the program counter is started from location $3FFD. A JMP instruction (opcode $BC) with operand located at $3FFE–$3FFF must be programmed into the user application for correct reset operation. The operand defines the location at which the user program will start. On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pullup/pulldown devices disabled.

The MC9RS08LE4 has nine sources for reset:

- External pin reset (PIN) — enabled using RSTPE in SOPT
- Power-on reset (POR)
- Low-voltage detect (LVD)

- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Background debug forced reset via BDC command BDC_RESET

Each of these sources, except the background debug forced reset, has an associated bit in the system reset status (SRS).

5.4 Computer Operating Properly (COP) Watchdog

The COP watchdog is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP counter periodically. If the application program gets lost and fails to reset the COP counter before it times out, a system reset is generated to force the system back to a known starting point.

After any reset, the COPE becomes set in SOPT, which enables the COP watchdog (see Section 5.8.2, “System Options Register (SOPT)”, for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPE. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

There is an associated short and long time-out controlled by COPT in SOPT. Table 5-1 summaries the control functions of the COPT bit. The COP watchdog operates from the 1 kHz clock source and defaults to the associated long time-out (2^8 cycles).

Table 5-1. COP Configuration Options

COPT	1-kHz COP Overflow Count[1]
0	2^5 cycles (32 ms)
1	2^8 cycles (256 ms)

[1] Values shown in this column are based on $t_{RTI} \approx 1$ ms.

Even if the application will use the reset default setting of COPE and COPT, the user should write to the write-once SOPT registers during reset initialization to lock in the settings. In this way, they cannot be changed accidentally if the application program gets lost. The initial write to SOPT will reset the COP counter.

In background debug mode, the COP counter does not increment.

When the MCU enters stop mode, the COP counter is re-initialized to zero upon entry to stop mode. The COP counter begins from zero as soon as the MCU exits stop mode.

5.5 Interrupts

The MC9RS08LE4 does not include an interrupt controller with vector table lookup mechanism as used in the HC08 and HCS08 devices. However, the interrupt sources from the modules such as LVD and KBI are still available to wake the CPU from wait or stop mode. It needs the user application to poll the corresponding module to determine the source of wakeup.

Each wakeup source of the module is associated with a corresponding interrupt enable bit. If the bit is disabled, the interrupt source is gated, and that particular source cannot wake the CPU from wait or stop mode. However, the corresponding interrupt flag is set to indicate that an external wakeup event has occurred.

The system interrupt pending registers (SIP1 and SIP2) indicate the status of the system pending interrupt. When the read-only bit of SIPs is enabled, it shows there is a pending interrupt to be serviced from the indicated module. Writing to the register bit has no effect. The pending interrupt flag is cleared automatically when all the corresponding interrupt flags from the indicated module are cleared.

5.6 Low-Voltage Detect (LVD) System

The MC9RS08LE4 adopts a system to protect memory contents and control MCU system states against low-voltage conditions during supply voltage variations. The system comprises of a power-on reset (POR) circuit and an LVD circuit with a predefined trip voltage. The LVD circuit is enabled with LVDE in SPMSC1. The LVD is disabled upon entering stop mode unless LVDSE is set in SPMSC1. If LVDSE and LVDE are set, the current consumption in stop with the LVD enabled will be greater.

5.6.1 Power-On Reset Operation

When power is initially applied to the MCU, or when the supply voltage drops below the V_{POR} level, the POR circuit causes a reset. As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply has risen above the V_{LVD} level. The POR bit and the LVD bit in SRS are set following a POR.

5.6.2 LVD Reset Operation

The LVD can be configured to generate a reset upon detection of a low-voltage condition by setting LVDRE to 1. After an LVD reset has occurred, the LVD system holds the MCU in reset until the supply voltage has risen above the level V_{LVD}. The LVD bit in the SRS register is set following either an LVD reset or POR.

5.6.3 LVD Interrupt Operation

When a low-voltage condition is detected and the LVD circuit is configured using SPMSC1 for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), LVDF in SPMSC1 is set and an LVD interrupt request occurs.

5.7 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts. The RTI is driven from the 1 kHz internal clock reference or the external clock reference (ICSERCLK) from the ICS module. The external clock reference is divided by 32 to produce a low-frequency clock by the RTI logic for applications requiring more accurate real-time interrupts. The RTICLKS bit in SRTISC is used to select the RTI clock source. Both the 1 kHz and the external clock sources for the RTI can be used when the MCU is in run, wait, or stop mode. For the external clock source to run in stop, the LVDE and LVDSE bits in the SPMSC1 must be set before entering stop.

The SRTISC register includes a read-only status flag, a write-only acknowledge bit, and a 3-bit control value (RTIS) used to select one of seven wakeup periods or disable RTI. The RTI has a local interrupt enable, RTIE, to allow masking of the real-time interrupt. The RTI can be disabled by writing each bit of RTIS to zeros, and no interrupts will be generated.

5.8 Reset, Interrupt, and System Control Registers and Control Bits

Refer to the direct-page register summary in Chapter 4, “Memory” for the absolute address assignments of all registers. This section refers to registers and control bits only by their names. A header file available from Freescale Semiconductor can be used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT register are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in detail in Chapter 3, “Modes of Operation”.

5.8.1 System Reset Status Register (SRS)

This register includes six read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by the BDC_RESET command, all status bits in SRS will be cleared. Writing any value to this register address clears the COP watchdog timer without affecting the contents of this register. The reset state of these bits depends on what caused the MCU to reset.

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	0	LVD	0
W	Writing any value to SRS address clears COP watchdog timer.							
POR	1	0	0	0	0	0	1	0
LVR:	U	0	0	0	0	0	1	0
Any other reset:	0	(1)	(1)	(1)	(1)	0	0	0

U = Unaffected by reset

[1] Any of these reset sources that are active at the time of reset cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset are cleared.

Figure 5-1. System Reset Status (SRS)

Table 5-2. SRS Register Field Descriptions

Field	Description
7 POR	**Power-On Reset** — Reset caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVR) status bit is also set to indicate that the reset occurred while the internal supply was below the LVR threshold. 0 Reset not caused by POR. 1 Reset caused by POR.
6 PIN	**External Reset Pin** — Reset caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
5 COP	**Computer Operating Properly (COP) Watchdog** — Reset caused by the COP watchdog timer timing out. This reset source may be blocked by COPE = 0. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.
4 ILOP	**Illegal Opcode** — Reset caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by STOPE = 0 in the SOPT register. The BGND instruction is considered illegal if active background mode is disabled by ENBDM = 0 in the BDCSC register. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.
3 ILAD	**Illegal Address** — Reset caused by an attempt to access either data or an instruction at an unimplemented memory address. 0 Reset not caused by an illegal address. 1 An illegal address caused reset.
1 LVD	**Low Voltage Detect** — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.

5.8.2 System Options Register (SOPT)

This register is a write-once register so only the first write after reset is effective. It can be read at any time. Any subsequent attempt to write to SOPT (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT must be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

	7	6	5	4	3	2	1	0
R	COPE	COPT	STOPE	0	0	0	BKGDPE	RSTPE
W				▨	▨	▨		
Reset	1	1	0	0	0	0	1[1]	u
POR	1	1	0	0	0	0	1	0

▨ = Unimplemented or Reserved

u = Unaffected

Figure 5-2. System Options Register (SOPT)

[1] When the device is reset into normal operating mode (MS is high during reset), BKGDPE is reset to 1 if flash memory security is disengaged (SECD = 1); BKGDPE is reset to 0 if flash memory security is engaged (SECD = 0). When the device is reset into active BDM mode (MS is low during reset), BKGDPE is always reset to 1 so that BDM communication is allowed.

Table 5-3. SOPT Register Field Descriptions

Field	Description
7 COPE	**COP Watchdog Enable** — These write-once bit selects whether the COP watchdog is enabled. 0 COP watchdog timer disabled. 1 COP watchdog timer enabled (force reset on timeout).
6 COPT	**COP Watchdog Timeout** — This write-once bit selects the timeout period of the COP. 0 Short timeout period (2^5 cycles) selected. 1 Long timeout period (2^8 cycles) selected.
5 STOPE	**Stop Mode Enable** — This write-once bit is used to enable stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.
1 BKGDPE[1]	**Background Debug Mode Pin Enable** — When set, this write-once bit enables the PTB1/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output only alternative functions. this pin defaults to the BKGDMS function following any MCU reset. 0 PTB1/BKGD/MS pin functions as PTB1. 1 PTB1/BKGD/MS pin functions as BKGD/MS.
0 RSTPE	**$\overline{\text{RESET}}$ Pin Enable** — When set, this write-once bit enables the PTB0/$\overline{\text{RESET}}$/V_{PP} pin to function as $\overline{\text{RESET}}$. When clear, the pin functions as one of its input-only alternative functions. This pin is input-only port function following an MCU POR. When RSTPE is set, an internal pullup device is enabled on $\overline{\text{RESET}}$. 0 PTB0/$\overline{\text{RESET}}$/V_{PP} pin function as PTB0/V_{PP}. 1 PTB0/$\overline{\text{RESET}}$/V_{PP} pin function as $\overline{\text{RESET}}$/V_{PP}.

[1] BKGDPE can write once only from value 1 to 0. Writing from value 0 to 1 by user software is not allowed. BKGDPE can be changed back to 1 by a POR or reset with proper condition only.

5.8.3 System Device Identification Register (SDIDH, SDIDL)

This read-only registers are included so host development systems can identify the RS08 derivative and revision number. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target MCU.

	7	6	5	4	3	2	1	0
R	▨	▨	▨	▨	ID11	ID10	ID9	ID8
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	1	0	0	0

▨ = Unimplemented or Reserved

Figure 5-3. System Device Identification Register — High (SDIDH)

Table 5-4. SDIDH Register Field Descriptions

Field	Description
3:0 ID[11:8]	**Part Identification Number** — Each derivative in the RS08 family has a unique identification number. The MC9RS08LE4 is hard coded to the value 0x0805. See also ID bits in Figure 5-4.

	7	6	5	4	3	2	1	0
R	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	1	0	1

▨ = Unimplemented or Reserved

Figure 5-4. System Device Identification Register — Low (SDIDL)

Table 5-5. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	**Part Identification Number** — Each derivative in the RS08 family has a unique identification number. The MC9RS08LE4 is hard coded to the value 0x0804. See also ID bits in , “”.

5.8.4 System PMC Real-Time Interrupt Status and Control (SRTISC)

SRTISC contains the status and control bits associated with the PMC real-time interrupt function.

	7	6	5	4	3	2	1	0
R	RTIF	0	RTICLKS	RTIE	0	RTIS		
W	▨	RTIACK			▨			
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 5-5. PMC Real-Time Interrupt Status and Control (SRTISC)

Table 5-6. SRTISC Register Field Descriptions

Field	Description
7 RTIF	**Real-Time Flag** — This read-only status bit indicates the periodic wakeup timer has timed out. 0 Periodic wakeup timer not timed out. 1 Periodic wakeup timer timed out.
6 RTIACK	**Real-Time Interrupt Acknowledge** — This write-only bit is used to acknowledge real-time interrupt request (write 1 to clear RTIF). Writing 0 has no meaning or effect. Reads always return 0.
5 RTICLKS	**Real-Time Interrupt Clock Select** — This read/write bit selects the clock source for the real-time interrupt. 0 Real-time interrupt request clock source is internal 1-kHz oscillator. 1 Real-time interrupt request clock source is external and is divided by 32 in RTI logic to produce a trimmed 1-kHz clock source for RTI counter.
4 RTIE	**Real-Time Interrupt Enable** — This read-write bit enables real-time interrupts. 0 Real-time interrupts disabled. 1 Real-time interrupts enabled.
2:0 RTIS[2:0]	**Bandgap Buffer Enable** — These read/write bits select the period for the RTI.

Table 5-7. Real-Time Interrupt Period

RTIS	1-kHz RTI Timeout[1]	T_{extclk} RTI Timeout[2]
000	Disable RTI	Disable RTI
001	8 ms	256 × T_{extclk}
010	32 ms	1,024 × T_{extclk}
011	64 ms	2,048 × T_{extclk}
100	128 ms	4,096 × T_{extclk}
101	256 ms	8,192 × T_{extclk}
110	512 ms	16,284 × T_{extclk}
111	1.024 s	32,768 × T_{extclk}

[1] Timeout values are based on RTI clock source of 1 ms period. Consult MC9RS08LE4 Data Sheet for tolerances of internal 1 kHz source, t_{RTI}.

[2] T_{extclk} is the period of external clock source connected to RTI.

5.8.5 System Power Management Status and Control 1 Register (SPMSC1)

This register contains status and control bits to support the low-voltage detection function, and to enable the bandgap voltage reference by the ADC module.

	7	6	5	4	3	2	1	0
R	LVDF[1]	0	LVDIE	LVDRE[2]	LVDSE	LVDE[2]	0	BGBE
W	▨	LVDACK					▨	
Reset:	0	0	0	1	1	1	0	0

▨ = Unimplemented or Reserved

[1] LVDF will be set in the case when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW}.

[2] This bit can be written only once after reset. Additional writes are ignored.

Figure 5-6. System Power Management Status and Control 1 Register (SPMSC1)

Table 5-8. SPMSC1 Register Field Descriptions

Field	Description
7 LVDF	**Low-Voltage Detect Flag** — The LVDF bit indicates the low-voltage detect status if LVDE is set. 0 Low-voltage is being or has been detected. 1 Low-voltage has not been detected.
6 LVDACK	**Low-Voltage Detect Acknowledge** — Writing a 1 to LVDACK clears the LVD interrupt request and clears LVDF if a low voltage is not detected.
5 LVDIE	**Low-Voltage Detect Interrupt Enable** — The LVDIE bit controls the LVD interrupt if LVDE is set. This bit has no effect if LVDE is 0. 0 LVD interrupt disabled. 1 LVD interrupt enabled.
4 LVDRE	**Low-Voltage Detect Reset Enable** — The LVDRE bit controls the LVD reset if LVDE is set. This bit has no effect if the LVDE is 0. LVD reset has priority over LVD interrupt, if both are enabled. In all test modes, the LVDRE bit value is ignored and functions as if the LVDRE is 0. 0 LVD reset disabled. 1 LVD reset enabled.
3 LVDSE	**Low-Voltage Detect Stop Enable** — The LVDSE bit controls the behavior of the LVD when the MCU stop mode is entered if LVDE is set. This bit has no effect if the LVDE is 0. 0 LVD disabled in MCU stop mode. 1 LVD enabled in MCU stop mode.
2 LVDE	**Low-Voltage Detect Enable** — The LVDE bit controls whether the LVD is enabled. 0 LVD is disabled. 1 LVD is enabled.
0 BGBE	**Bandgap Buffer Enable** — The BGBE bit is used to enable the bandgap buffered output. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

5.8.6 System Interrupt Pending Register 1 (SIP1)

This register contains status of the pending interrupt from the modules.

	7	6	5	4	3	2	1	0
R	LVD	SCIT	SCIR	SCIE	ADC	KBI	LCD	RTI
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 5-7. System Interrupt Pending Register 1 (SIP1)

Table 5-9. SIP1 Register Field Descriptions

Field	Description
7 LVD	**LVD Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt from LVD module. Clearing the corresponding flag in LVD module clears this bit. Reset also clears this bit. 0 There is no pending LVD interrupt. 1 There is a pending LVD interrupt.
6 SCIT	**SCI Transmit Interrupt Pending** — This read-only bit indicates whether there is a pending transmit interrupt from SCI module. This flag can be trigged by TDRE and TC. Clearing the corresponding flag in SCI module clears this bit. Reset also clears this bit. 0 There is no pending SCI transmit interrupt. 1 There is a pending SCI transmit interrupt.
5 SCIR	**SCI Receive Interrupt Pending** — This read-only bit indicates whether there is a pending receive interrupt from SCI module. This flag can be trigged by IDLE, RDRF, LBKDIF, or RXEDGIF. Clearing the corresponding flag in SCI module clears this bit. Reset also clears this bit. 0 There is no receive pending SCI interrupt. 1 There is a receive pending SCI interrupt.
4 SCIE	**SCI Error Interrupt Pending —** This read-only bit indicates whether there is a pending error interrupt from SCI module. This flag can be trigged by OR, NF, FE, PF. Clearing the corresponding flag in SCI module clears this bit. Reset also clears this bit. 0 There is no pending SCI error interrupt. 1 There is a pending SCI error interrupt.
3 ADC	**ADC Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt from ADC module. Clearing the corresponding flag in ADC module clears this bit. Reset also clears this bit. 0 There is no pending ADC interrupt. 1 There is a pending ADC interrupt.
2 KBI	**KBI Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt from KBI module. Clearing the corresponding flag in KBI module clears this bit. Reset also clears this bit. 0 There is no pending KBI interrupt. 1 There is a pending KBI interrupt.
1 LCD	**LCD Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt from LCD module. Clearing the corresponding flag in LCD module clears this bit. Reset also clears this bit. 0 There is no pending LCD interrupt. 1 There is a pending LCD interrupt.
0 RTI	**RTI Interrupt Pending** — This read-only bit indicates whether there is a pending interrupt from RTI module. Clearing the corresponding flag in RTI module clears this bit. Reset also clears this bit. 0 There is no pending RTI interrupt. 1 There is a pending RTI interrupt.

5.8.7 System Interrupt Pending Register 2 (SIP2)

This register contains status of the pending interrupt from the modules.

	7	6	5	4	3	2	1	0
R	▨	TPM2CH1	TPM2CH0	TPM2	▨	TPM1CH1	TPM1CH0	TPM1
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 5-8. System Interrupt Pending Register 2 (SIP2)

Table 5-10. SIP2 Register Field Descriptions

Field	Description
6 TPM2CH1	**TPM2 Channel 1 Interrupt Pending** — This read-only bit indicates whether there is a pending TPM2 channel 1 overflow (CH1F) interrupt from TPM2 module. Clearing the corresponding flag in TPM2 module clears this bit. Reset also clears this bit. 0 There is no pending TPM2 channel 1 overflow interrupt. 1 There is a pending TPM2 channel 1 overflow interrupt.
5 TPM2CH0	**TPM2 Channel 0 Interrupt Pending** — This read-only bit indicates whether there is a pending TPM2 channel 0 overflow (CH0F) interrupt from TPM2 module. Clearing the corresponding flag in TPM2 module clears this bit. Reset also clears this bit. 0 There is no pending TPM2 channel 0 overflow interrupt. 1 There is a pending TPM2 channel 0 overflow interrupt.
4 TPM2	**TPM2 Interrupt Pending** — This read-only bit indicates whether there is a pending TPM2 overflow (TOF) interrupt from TPM2 module. Clearing the corresponding flag in TPM2 module clears this bit. Reset also clears this bit. 0 There is no pending TPM2 overflow interrupt. 1 There is a pending TPM2 overflow interrupt.
2 TPM1CH1	**TPM1 Channel 1 Interrupt Pending** — This read-only bit indicates whether there is a pending TPM1 channel 1 overflow (CH1F) interrupt from TPM1 module. Clearing the corresponding flag in TPM1 module clears this bit. Reset also clears this bit. 0 There is no pending TPM1 channel 1 overflow interrupt. 1 There is a pending TPM1 channel 1 overflow interrupt.
1 TPM1CH0	**TPM1 Channel 0 Interrupt Pending** — This read-only bit indicates whether there is a pending TPM1 channel 0 overflow (CH0F) interrupt from TPM1 module. Clearing the corresponding flag in TPM1 module clears this bit. Reset also clears this bit. 0 There is no pending TPM1 channel 0 overflow interrupt. 1 There is a pending TPM1 channel 0 overflow interrupt.
0 TPM1	**TPM1 Interrupt Pending** — This read-only bit indicates whether there is a pending TPM1 overflow (TOF) interrupt from TPM1 module. Clearing the corresponding flag in TPM1 module clears this bit. Reset also clears this bit. 0 There is no pending TPM1 overflow interrupt. 1 There is a pending TPM1 overflow interrupt.

Chapter 6
Parallel Input/Output

6.1 Introduction

This chapter explains software controls related to parallel input/output (I/O). The MC9RS08LE4 has four I/O ports which include a total of 26 general-purpose I/O pins. Refer to Chapter 2, “Pins and Connections” for more information about pin assignments and external hardware considerations of these pins.

All of these I/O pins are shared with on-chip peripheral functions as shown in Table 2-1. The peripheral modules have priority over the I/Os so that when a peripheral is enabled, the I/O functions associated with the shared pins are disabled. After reset, the shared peripheral functions are disabled so that the pins are controlled by the I/O. All of the I/Os are configured as inputs (PTxDDn = 0) with pullup/pulldown devices disabled (PTxPEn = 0), except for output-only pin PTB1, which defaults to the BKGD/MS function.

Reading and writing of parallel I/Os is performed through the port data registers. The direction, either input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in Figure 6-1.

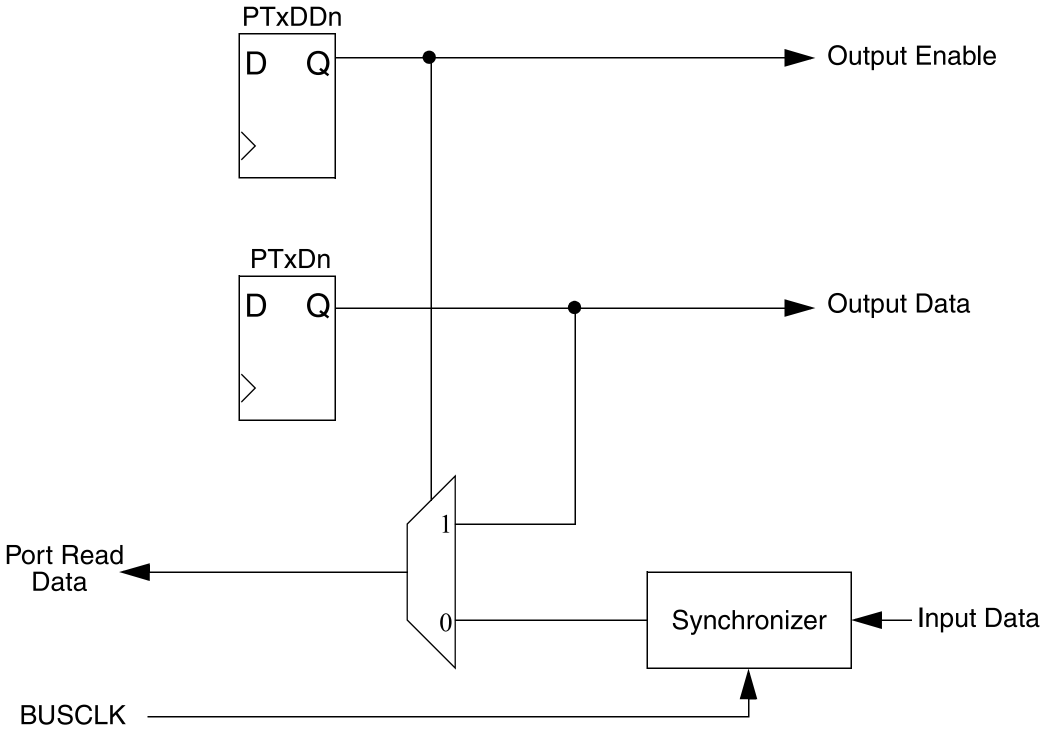


Figure 6-1. Parallel I/O Block Diagram

The data direction control bit (PTxDDn) determines whether the output buffer for the associated pin is enabled. It also controls the source to read port data register. The input buffer for the associated pin is always enabled unless the pin is enabled for analog function or is an output-only pin.

When a shared digital function is enabled for a pin, the output buffer is controlled by the shared function. However, the data direction register bit will continue to control the source to read the port data register. When a shared analog function is enabled for a pin, both the input and output buffers are disabled. A value of 0 is read for any port data bit where the bit is an input (PTxDDn = 0) and the input buffer is disabled. In general, whenever a pin is shared with both an alternative digital function and an analog function, the analog function has priority so that if both the digital and analog functions are enabled, the analog function controls the pin.

Write to the port data register before changing the direction of a port pin to output. This ensures that the pin will not be driven temporarily with an old data value that happened to be in the port data register.

A set of registers associated with the parallel I/O ports are located in the high-page register space that operates independently of the parallel I/O registers. These registers are used to control pullup/pulldown and slew rate for the pins.

6.2 Pin Behavior in Low-Power Modes

In wait and stop modes, all pin states are maintained because internal logic stays powered up. Upon recovery, all pin functions are the same as before entering stop.

6.3 Parallel I/O and Pin Control Registers

This section provides information about the registers associated with the parallel I/O ports and pin control functions. These data registers are located in page one of the memory map and the other registers are located in the high-page register section of memory.

Refer to tables in Chapter 4, “Memory” for the absolute address assignments for all parallel I/O and pin control registers. This section refers to registers and control bits only by their names. A header file available from Freescale Semiconductor can be used to translate these names into the appropriate absolute addresses.

6.3.1 Port A I/O Registers (PTAD and PTADD)

Port A parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
Reset	0	0	0	0	0	0	0	0

Figure 6-2. Port A Data Register (PTAD)

Table 6-1. PTAD Register Field Descriptions

Field	Description
7:0 PTAD[7:0]	**Port A Data Register Bits** — For port A pins that are configured as inputs, reads return the logic level to the pin. For port A pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port A pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTAD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
Reset	0	0	0	0	0	0	0	0

Figure 6-3. Data Direction for Port A Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

Field	Description
7:0 PTADD[7:0]	**Data Direction for Port A Bits** — These read/write bits control the direction of port A pins and what is read for PTAD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port A bit n and PTAD reads return the contents of PTADn.

6.3.2 Port A Pin Control Registers (PTAPE, PTASE, PTAPUD, PTADS)

In addition to the I/O control, port A pins are controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-4. Internal Pulling Enable for Port A (PTAPE)

Table 6-3. PTADD Register Field Descriptions

Field	Description
[7:0] PTAPE[7:0]	**Internal Pullup Enable for Port A Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs, these bits have no effect and the internal pulling devices are disabled. 0 Internal pulling device disabled for port A bit n. 1 Internal pulling device enabled for port A bit n.

	7	6	5	4	3	2	1	0
R W	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
Reset	1	1	1	1	1	1	1	1

Figure 6-5. Output Slew Rate Control Enable for Port A (PTASE)

Table 6-4. PTASE Register Field Descriptions

Field	Description
7:0 PTASE[7:0]	**Output Slew Rate Control Enable for Port A Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port A bit n. 1 Output slew rate control enabled for port A bit n.

	7	6	5	4	3	2	1	0
R W	PTAPUD7	PTAPUD6	PTAPUD5	PTAPUD4	PTAPUD3	PTAPUD2	PTAPUD1	PTAPUD0
Reset	0	0	0	0	0	0	0	0

Figure 6-6. Pullup/Pulldown Device Control for Port A (PTAPUD)

Table 6-5. PTAPUD Register Field Descriptions

Field	Description
7:6 PTAPUD[7:6]	**Pullup/Pulldown Device Control for Port A Bits** — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTA pin. The actual pullup/pulldown device is only enabled by enabling the associated PTAPE bit. 0 Internal pullup device is selected for port A bit n. 1 Internal pulldown device is selected for port A bit n.

	7	6	5	4	3	2	1	0
R W	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
Reset	0	0	0	0	0	0	0	0

Figure 6-7. Drive Strength Selection for Port A (PTADS)

Table 6-6. PTADS Register Field Descriptions

Field	Description
7:0 PTADS[7:0]	**Output Drive Strength Selection for Port A Bits** — Each of these control bits selects between low and high output driver for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port A bit n. 1 High output drive strength selected for port A bit n.

6.3.3 Port B I/O Registers (PTBD and PTBDD)

Port B parallel I/O function is controlled by the registers listed.

	7	6	5	4	3	2	1	0
R W	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
Reset	0	0	0	0	0	0	0	0

Figure 6-8. Port B Data Register (PTBD)

Table 6-7. PTBD Register Field Descriptions

Field	Description
7:0 PTBD[7:0]	**Port B Data Register Bits** — For port B pins that are configured as inputs, reads return the logic level on the pin. For port B pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port B pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTBD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	0	0
Reset	0	0	0	0	0	0	0	0

Figure 6-9. Data Direction for Port B (PTBDD)

Table 6-8. PTBDD Register Field Descriptions

Field	Description
7:2 PTBDD[7:2]	**Data Direction for Port B Bits** — These read/write bits control the direction of port B pins and what is read for PTBD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port B bit n and PTBD reads return the contents of PTBDn.

6.3.4 Port B Pin Control Registers (PTBPE, PTBSE, PTBPUE, PTBDS)

In addition to the I/O control, port B pins are controlled by the registers listed.

	7	6	5	4	3	2	1	0
R W	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	0	PTBPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-10. Internal Pulling Enable for Port B (PTBPE)

Table 6-9. PTBPE Register Field Descriptions

Field	Description
7:2, 0 PTBPE[7:2, 0]	**Internal Pullup Enable for Port B Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs, these bits have no effect and the internal pulling devices are disabled. 0 Internal pulling device disabled for port B bit n. 1 Internal pulling device enabled for port B bit n.

	7	6	5	4	3	2	1	0
R W	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	0
Reset	0	0	0	0	0	1	1	1

Figure 6-11. Output Slew Rate Control Enable (PTBSE)

Table 6-10. PTBSE Register Field Descriptions

Field	Description
7:1 PTBSE[7:1]	**Output Slew Rate Control Enable for Port B Bits**— Each of these control bits determines whether output slew rate control is enabled for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port B bit n. 1 Output slew rate control enabled for port B bit n.

	7	6	5	4	3	2	1	0
R W	PTBPUD7	PTBPUD6	PTBPUD5	PTBPUD4	PTBPUD3	PTBPUD2	0	PTBPUD0
Reset	0	0	0	0	0	0	0	0

Figure 6-12. Pullup/Pulldown Device Control for Port B (PTBPUD)

Table 6-11. PTBPUD Register Field Descriptions

Field	Description
7:2, 0 PTBPUD[7:2, 0]	**Pullup/Pulldown Device Control for Port B Bits** — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTB pin. The actual pullup/pulldown device is only enabled by enabling the associated PTBPE bit. 0 Internal pullup device is selected for port B bit n. 1 Internal pulldown device is selected for port B bit n.

	7	6	5	4	3	2	1	0
R	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-13. Drive Strength Selection for Port B (PTBDS)

Table 6-12. PTBDS Register Field Descriptions

Field	Description
7:1 PTBDS[7:1]	**Output Drive Strength Selection for Port B Bits** — Each of these control bits selects between low and high output driver for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port B bit n. 1 High output drive strength selected for port B bit n.

6.3.5 Port C I/O Registers (PTCD and PTCDD)

Port C parallel I/O function is controlled by the registers listed.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTCD1	PTCD0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-14. Port C Data Register (PTCD)

Table 6-13. PTCD Register Field Descriptions

Field	Description
1:0 PTCD [1:0]	**Port C Data Register Bits** — For port C pins that are configured as inputs, reads return the logic level on the pin. For port C pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port C pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTCD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTCDD1	PTCDD0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-15. Data Direction for Port C (PTCDD)

Table 6-14. PTCDD Register Field Descriptions

Field	Description
1:0 PTCDD[1:0]	**Data Direction for Port C Bits** — These read/write bits control the direction of port C pins and what is read for PTCD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port C bit n and PTCD reads return the contents of PTCDn.

6.3.6 Port C Pin Control Registers (PTCPE, PTCSE, PTCPUD, PTCDS)

In addition to the I/O control, port C pins are controlled by the registers listed.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTCPE1	PTCPE0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-16. Internal Pulling Enable for Port C (PTCPE)

Table 6-15. PTCPE Register Field Descriptions

Field	Description
1:0 PTCPE [1:0]	**Internal Pullup Enable for Port C Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs, these bits have no effect and the internal pulling devices are disabled. 0 Internal pulling device disabled for port C bit n. 1 Internal pulling device enabled for port C bit n.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTCSE1	PTCSE0
W								
Reset	1	1	0	0	1	1	1	1

Figure 6-17. Output Slew Rate Control Enable for Port C (PTCSE)

Table 6-16. PTCSE Register Field Descriptions

Field	Description
1:0 PTCSE [1:0]	**Output Slew Rate Control Enable for Port C Bits** — Each of these control bits determines whether output slew rate control is enabled for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port C bit n. 1 Output slew rate control enabled for port C bit n.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTCPUD1	PTCPUD0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-18. Pullup/Pulldown Device Control for Port C (PTCPUD)

Table 6-17. PTCPUD Register Field Descriptions

Field	Description
1:0 PTCPUD [1:0]	**Pullup/Pulldown Device Control for Port C Bits** — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTC pin. The actual pullup/pulldown device is only enabled by enabling the associated PTCPE bit. 0 Internal pullup device is selected for port C bit n. 1 Internal pulldown device is selected for port C bit n.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	PTCDS1	PTCDS0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-19. Drive Strength Selection for Port C (PTCDS)

Table 6-18. PTCDS Register Field Descriptions

Field	Description
1:0 PTCDS[1:0]	**Output Drive Strength Selection for Port C Bits** — Each of these control bits selects between low and high output driver for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port C bit n. 1 High output drive strength selected for port C bit n.

6.3.7 Port D I/O Registers (PTDD and PTDDD)

Port D parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
Reset	0	0	0	0	0	0	0	0

Figure 6-20. Port D Data Register (PTDD)

Table 6-19. PTDD Register Field Descriptions

Field	Description
7:0 PTDD[7:0]	**Port D Data Register Bits** — For port D pins that are configured as inputs, reads return the logic level on the pin. For port D pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port D pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTDD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
Reset	0	0	0	0	0	0	0	0

Figure 6-21. Data Direction for Port D (PTDDD)

Table 6-20. PTDDD Register Field Descriptions

Field	Description
7:0 PTDDD[7:0]	**Data Direction for Port D Bits** — These read/write bits control the direction of port D pins and what is read for PTDD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port D bit n and PTDD reads return the contents of PTDDn.

6.3.8 Port D Pin Control Registers (PTDPE, PTDSE, PTDPUD, PTDDS)

In addition to the I/O control, port D pins are controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-22. Internal Pulling Enable for Port D (PTDPE)

Table 6-21. PTDPE Register Field Descriptions

Field	Description
7:0 PTDPE[7:0]	**Internal Pullup Enable for Port D Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTD pin. For port D pins that are configured as outputs, these bits have no effect and the internal pulling devices are disabled. 0 Internal pulling device disabled for port D bit n. 1 Internal pulling device enabled for port D bit n.

	7	6	5	4	3	2	1	0
R W	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
Reset	1	1	1	1	1	1	1	1

Figure 6-23. Output Slew Rate Control Enable for Port D (PTDSE)

Table 6-22. PTDSE Register Field Descriptions

Field	Description
7:0 PTDSE[7:0]	**Output Slew Rate Control Enable for Port D Bits** — Each of these control bits determines whether output slew rate control is enabled for the associated PTD pin. For port D pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port D bit n. 1 Output slew rate control enabled for port D bit n.

Offset

	7	6	5	4	3	2	1	0
R W	PTDPUD7	PTDPUD6	PTDPUD5	PTDPUD4	PTDPUD3	PTDPUD2	PTDPUD1	PTDPUD0
Reset	0	0	0	0	0	0	0	0

Figure 6-24. Pullup/Pulldown Device Control for Port D (PTDPUD)

Table 6-23. PTDPUD Register Field Descriptions

Field	Description
7:0 PTDPUD[7:0]	**Pullup/Pulldown Device Control for Port D Bits** — Each of these control bits determines whether the internal pullup or pulldown device is selected for the associated PTD pin. The actual pullup/pulldown device is only enabled by enabling the associated PTDPE bit. 0 Internal pullup device is selected for port D bit n. 1 Internal pulldown device is selected for port D bit n.

	7	6	5	4	3	2	1	0
R W	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
Reset	0	0	0	0	0	0	0	0

Figure 6-25. Drive Strength Selection for Port D (PTDDS)

Table 6-24. PTDDS Register Field Descriptions

Field	Description
7:0 PTDDS[7:0]	**Output Drive Strength Selection for Port D Bits** — Each of these control bits selects between low and high output driver for the associated PTD pin. For port D pins that are configured as inputs, these bits have no effect. 0 Low output drive strength selected for port D bit n. 1 High output drive strength selected for port D bit n.

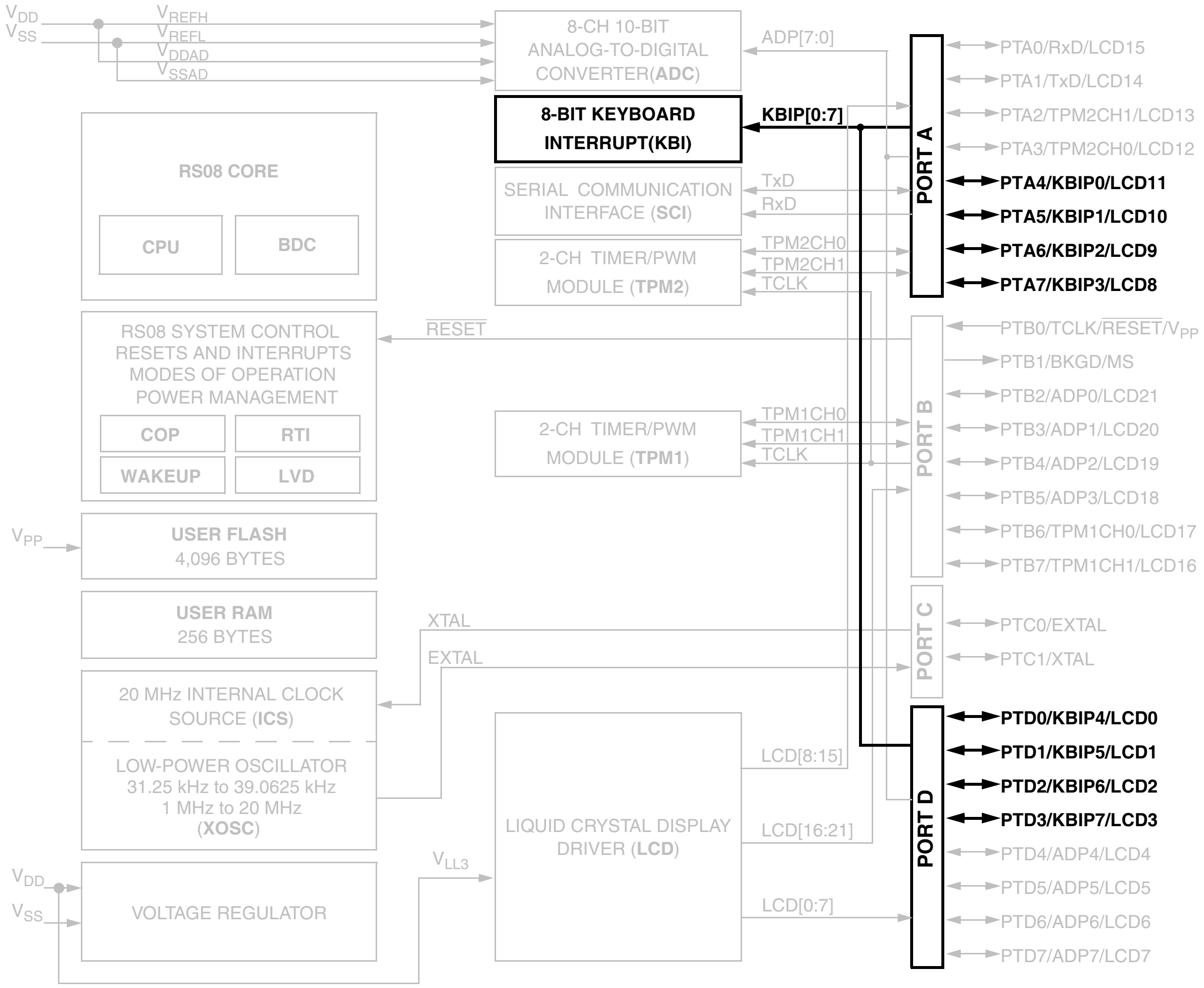
Chapter 7
Keyboard Interrupt (S08KBIV2)

7.1 Introduction

The keyboard interrupt (KBI) module provides up to four independently enabled external interrupt sources.

All KBI pins share KBI functionality with LCD pins. KBI functionality must be disabled for those used as LCD pins.

Figure 7-1 shows the MC9RS08LE4 block diagram, highlighting the KBI block and pins.



NOTES:

1. PTB0/TCLK/RESET/V_{PP} is an input-only pin when used as port pin
2. PTB1/BKGD/MS is an output-only pin

Figure 7-1. MC9RS08LE4 Series Block Diagram Highlighting KBI Block and Pins

7.1.1 Features

The KBI features include:

- Up to eight keyboard interrupt pins with individual pin enable bits.
- Each keyboard interrupt pin is programmable as falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity.
- One software enabled keyboard interrupt.
- Exit from low-power modes.

7.1.2 Modes of Operation

This section defines the KBI operation in wait, stop, and background debug modes.

7.1.2.1 KBI in Wait Mode

The KBI continues to operate in wait mode if enabled before executing the WAIT instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of wait mode if the KBI interrupt is enabled (KBIE = 1).

7.1.2.2 KBI in Stop Modes

The KBI operates asynchronously in stop3 mode if enabled before executing the STOP instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of stop3 mode if the KBI interrupt is enabled (KBIE = 1).

During either stop1 or stop2 mode, the KBI is disabled. In some systems, the pins associated with the KBI may be sources of wakeup from stop1 or stop2, see the stop modes section in the Modes of Operation chapter. Upon wake-up from stop1 or stop2 mode, the KBI module will be in the reset state.

7.1.2.3 KBI in Active Background Mode

When the microcontroller is in active background mode, the KBI will continue to operate normally.

7.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown Figure 7-2.

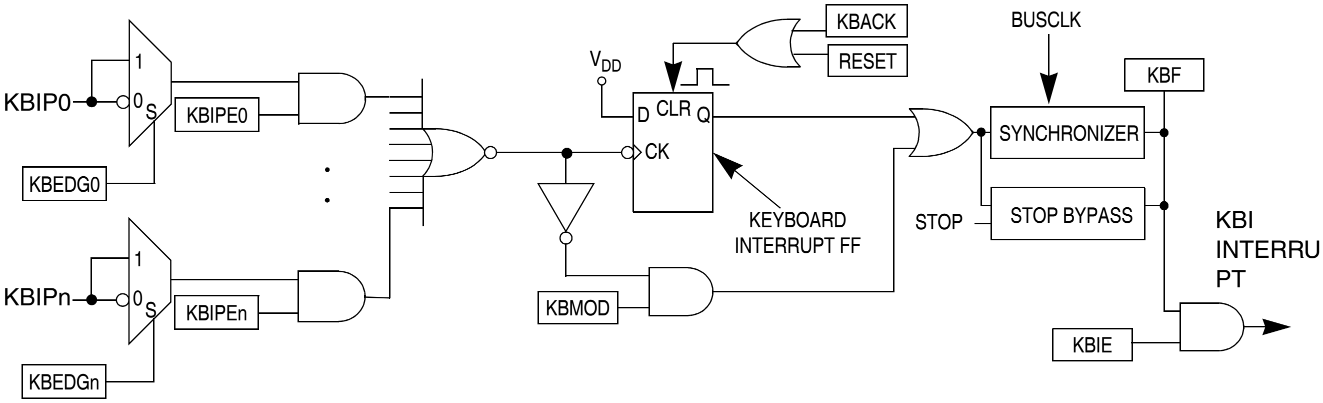


Figure 7-2. KBI Block Diagram

7.2 External Signal Description

The KBI input pins can be used to detect either falling edges, or both falling edge and low level interrupt requests. The KBI input pins can also be used to detect either rising edges, or both rising edge and high level interrupt requests.

The signal properties of KBI are shown in Table 7-1.

Table 7-1. Signal Properties

Signal	Function	I/O
KBIPn	Keyboard interrupt pins	I

7.3 Register Definition

7.3.1 Register Descriptions

The KBI includes three registers:

- An 8-bit pin status and control register.
- An 8-bit pin enable register.
- An 8-bit edge select register.

Refer to the direct-page register summary in the Memory chapter for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names.

Some MCUs may have more than one KBI, so register names include placeholder characters to identify which KBI is being referenced.

7.3.1.1 KBI Status and Control Register (KBISC)

KBISC contains the status flag and control bits, which are used to configure the KBI.

	7	6	5	4	3	2	1	0
R	0	0	0	0	KBF	0	KBIE	KBMOD
W	▨	▨	▨	▨	▨	KBACK		
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 7-3. KBI Status and Control Register

Table 7-2. KBISC Register Field Descriptions

Field	Description
7:4	Unused register bits, always read 0.
3 KBF	**Keyboard Interrupt Flag** — KBF indicates when a keyboard interrupt is detected. Writes have no effect on KBF. 0 No keyboard interrupt detected. 1 Keyboard interrupt detected.
2 KBACK	**Keyboard Acknowledge** — Writing a 1 to KBACK is part of the flag clearing mechanism. KBACK always reads as 0.
1 KBIE	**Keyboard Interrupt Enable** — KBIE determines whether a keyboard interrupt is requested. 0 Keyboard interrupt request not enabled. 1 Keyboard interrupt request enabled.
0 KBMOD	**Keyboard Detection Mode** — KBMOD (along with the KBEDG bits) controls the detection mode of the keyboard interrupt pins.0Keyboard detects edges only. 1 Keyboard detects both edges and levels.

7.3.1.2 KBI Pin Enable Register (KBIPE)

KBIPE contains the pin enable control bits.

	7	6	5	4	3	2	1	0
R W	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
Reset:	0	0	0	0	0	0	0	0

Figure 7-4. KBI Pin Enable Register

Table 7-3. KBIPE Register Field Descriptions

Field	Description
7:0 KBIPEn	**Keyboard Pin Enables** — Each of the KBIPEn bits enable the corresponding keyboard interrupt pin. 0 Pin not enabled as keyboard interrupt. 1 Pin enabled as keyboard interrupt.

7.3.1.3 KBI Edge Select Register (KBIES)

KBIES contains the edge select control bits.

	7	6	5	4	3	2	1	0
R W	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
Reset:	0	0	0	0	0	0	0	0

Figure 7-5. KBI Edge Select Register

Table 7-4. KBIES Register Field Descriptions

Field	Description
7:0 KBEDGn	**Keyboard Edge Selects** — Each of the KBEDGn bits selects the falling edge/low level or rising edge/high level function of the corresponding pin). 0 Falling edge/low level. 1 Rising edge/high level.

7.4 Functional Description

This on-chip peripheral module is called a keyboard interrupt (KBI) module because originally it was designed to simplify the connection and use of row-column matrices of keyboard switches. However, these inputs are also useful as extra external interrupt inputs and as an external means of waking the MCU from stop or wait low-power modes.

The KBI module allows up to eight pins to act as additional interrupt sources. Writing to the KBIPEn bits in the keyboard interrupt pin enable register (KBIPE) independently enables or disables each KBI pin. Each KBI pin can be configured as edge sensitive or edge and level sensitive based on the KBMOD bit in the keyboard interrupt status and control register (KBISC). Edge sensitive can be software programmed to be either falling or rising; the level can be either low or high. The polarity of the edge or edge and level sensitivity is selected using the KBEDGn bits in the keyboard interrupt edge select register (KBIES).

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled keyboard inputs must be at the deasserted logic level. A falling edge is detected when an enabled keyboard input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

7.4.1 Edge Only Sensitivity

A valid edge on an enabled KBI pin will set KBF in KBISC. If KBIE in KBISC is set, an interrupt request will be presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in KBISC.NOTE:

If the keyboard interrupt is edge-sensitive only, a falling (or rising) edge on a keyboard pin does not latch an interrupt request if another keyboard pin is already asserted. To prevent losing an interrupt request on one pin because another pin is still asserted, software can disable the asserted pin interrupt while having the unasserted pin interrupt enabled. The asserted status of a pin is reflected by its associated I/O general purpose data register.

7.4.2 Edge and Level Sensitivity

A valid edge or level on an enabled KBI pin will set KBF in KBISC. If KBIE in KBISC is set, an interrupt request will be presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in KBISC provided all enabled keyboard inputs are at their deasserted levels. KBF will remain set if any enabled KBI pin is asserted while attempting to clear by writing a 1 to KBACK.

7.4.3 KBI Pullup/Pulldown Resistors

The KBI pins can be configured to use an internal pullup/pulldown resistor using the associated I/O port pullup enable register. If an internal resistor is enabled, the KBIES register is used to select whether the resistor is a pullup (KBEDGn = 0) or a pulldown (KBEDGn = 1).

7.4.4 KBI Initialization

When a keyboard interrupt pin is first enabled it is possible to get a false keyboard interrupt flag. To prevent a false interrupt request during keyboard initialization, the user should do the following:

1. Mask keyboard interrupts by clearing KBIE in KBISC.
2. Enable the KBI polarity by setting the appropriate KBEDGn bits in KBIES.
3. If using internal pullup/pulldown device, configure the associated pullup enable bits in PTxPE.
4. Enable the KBI pins by setting the appropriate KBIPEn bits in KBIPE.
5. Write to KBACK in KBISC to clear any false interrupts.
6. Set KBIE in KBISC to enable interrupts.

Chapter 8
Central Processor Unit (RS08CPUV1)

8.1 Introduction

This chapter is a summary of information about the registers, addressing modes, and instruction set of the RS08 Family CPU. For a more detailed discussion, refer to the RS08 Core Reference Manual, volume 1, Freescale Semiconductor document order number RS08RMv1.

The RS08 CPU has been developed to target extremely low-cost embedded applications using a process-independent design methodology, allowing it to keep pace with rapid developments in silicon processing technology.

The main features of the RS08 core are:

- Streamlined programmer's model
- Subset of HCS08 instruction set with minor instruction extensions
- Minimal instruction set for cost-sensitive embedded applications
- New instructions for shadow program counter manipulation, SHA and SLA
- New short and tiny addressing modes for code size optimization
- 16K bytes accessible memory space
- Reset will fetch the first instruction from $3FFD
- Low-power modes supported through the execution of the STOP and WAIT instructions
- Debug and FLASH programming support using the background debug controller module
- Illegal address and opcode detection with reset

8.2 Programmer's Model and CPU Registers

Figure 8-1 shows the programmer's model for the RS08 CPU. These registers are not located in the memory map of the microcontroller. They are built directly inside the CPU logic.

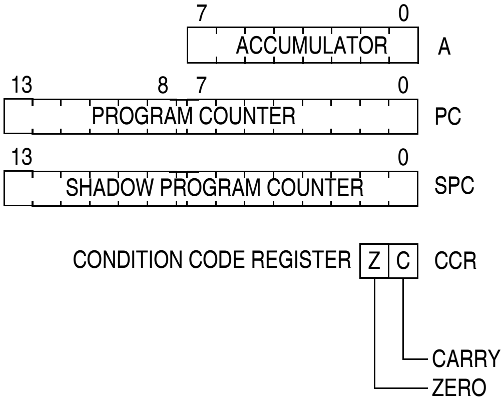


Figure 8-1. CPU Registers

In addition to the CPU registers, there are three memory mapped registers that are tightly coupled with the core address generation during data read and write operations. They are the indexed data register (D[X]), the index register (X), and the page select register (PAGESEL). These registers are located at $000E, $000F, and $001F, respectively.

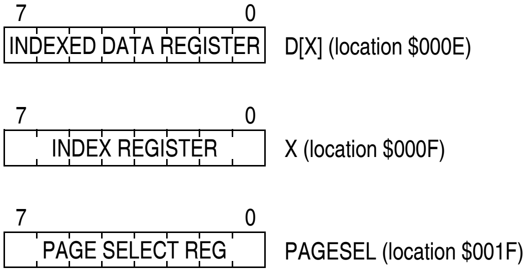


Figure 8-2. Memory Mapped Registers

8.2.1 Accumulator (A)

This general-purpose 8-bit register is the primary data register for RS08 MCUs. Data can be read from memory into A with a load accumulator (LDA) instruction. The data in A can be written into memory with a store accumulator (STA) instruction. Various addressing mode variations allow a great deal of flexibility in specifying the memory location involved in a load or store instruction. Exchange instructions allow values to be exchanged between A and SPC high (SHA) and also between A and SPC low (SLA).

Arithmetic, shift, and logical operations can be performed on the value in A as in ADD, SUB, RORA, INCA, DECA, AND, ORA, EOR, etc. In some of these instructions, such as INCA and LSLA, the value in A is the only input operand and the result replaces the value in A. In other cases, such as ADD and AND, there are two operands: the value in A and a second value from memory. The result of the arithmetic or logical operation replaces the value in A.

Some instructions, such as memory-to-memory move instructions (MOV), do not use the accumulator. DBNZ also relieves A because it allows a loop counter to be implemented in a memory variable rather than the accumulator.

During reset, the accumulator is loaded with $00.

8.2.2 Program Counter (PC)

The program counter is a 14-bit register that contains the address of the next instruction or operand to be fetched.

During normal execution, the program counter automatically increments to the next sequential memory location each time an instruction or operand is fetched. Jump, branch, and return operations load the program counter with an address other than that of the next sequential location. This is called a change-of-flow.

During reset, the program counter is loaded with $3FFD and the program will start execution from this specific location.

8.2.3 Shadow Program Counter (SPC)

The shadow program counter is a 14-bit register. During a subroutine call using either a JSR or a BSR instruction, the return address will be saved into the SPC. Upon completion of the subroutine, the RTS instruction will restore the content of the program counter from the shadow program counter.

During reset, the shadow program counter is loaded with $3FFD.

8.2.4 Condition Code Register (CCR)

The 2-bit condition code register contains two status flags. The content of the CCR in the RS08 is not directly readable. The CCR bits can be tested using conditional branch instructions such as BCC and BEQ. These two register bits are directly accessible through the BDC interface. The following paragraphs provide detailed information about the CCR bits and how they are used. Figure 8-3 identifies the CCR bits and their bit positions.

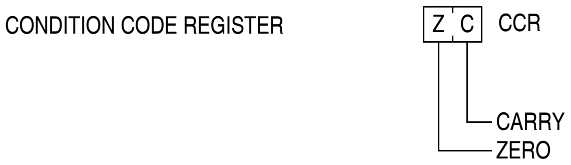


Figure 8-3. Condition Code Register (CCR)

The status bits (Z and C) are cleared to 0 after reset.

The two status bits indicate the results of arithmetic and other instructions. Conditional branch instructions will either branch to a new program location or allow the program to continue to the next instruction after the branch, depending on the values in the CCR status bit. Conditional branch instructions, such as BCC, BCS, and BNE, cause a branch depending on the state of a single CCR bit.

Often, the conditional branch immediately follows the instruction that caused the CCR bit(s) to be updated, as in this sequence:

```
         cmp     #5        ;compare accumulator A to 5
         blo     lower     ;branch if A smaller 5
more:    deca             ;do this if A not higher than or same as 5
lower:
```

Other instructions may be executed between the test and the conditional branch as long as the only instructions used are those which do not disturb the CCR bits that affect the conditional branch. For instance, a test is performed in a subroutine or function and the conditional branch is not executed until the subroutine has returned to the main program. This is a form of parameter passing (that is, information is returned to the calling program in the condition code bits).

Z — Zero Flag

The Z bit is set to indicate the result of an operation was $00.

Branch if equal (BEQ) and branch if not equal (BNE) are simple branches that branch based solely on the value in the Z bit. All load, store, move, arithmetic, logical, shift, and rotate instructions cause the Z bit to be updated.

C — Carry

After an addition operation, the C bit is set if the source operands were both greater than or equal to $80 or if one of the operands was greater than or equal to $80 and the result was less than $80. This is equivalent to an unsigned overflow. A subtract or compare performs a subtraction of a memory operand from the contents of a CPU register so after a subtract operation, the C bit is set if the unsigned value of the memory operand was greater than the unsigned value of the CPU register. This is equivalent to an unsigned borrow or underflow.

Branch if carry clear (BCC) and branch if carry set (BCS) are branches that branch based solely on the value in the C bit. The C bit is also used by the unsigned branches BLO and BHS. Add, subtract, shift, and rotate instructions cause the C bit to be updated. The branch if bit set (BRSET) and branch if bit clear (BRCLR) instructions copy the tested bit into the C bit to facilitate efficient serial-to-parallel conversion algorithms. Set carry (SEC) and clear carry (CLC) allow the carry bit to be set or cleared directly. This is useful in combination with the shift and rotate instructions and for routines that pass status information back to a main program, from a subroutine, in the C bit.

The C bit is included in shift and rotate operations so those operations can easily be extended to multi-byte operands. The shift and rotate operations can be considered 9-bit shifts that include an 8-bit operand or CPU register and the carry bit of the CCR. After a logical shift, C holds the bit that was shifted out of the 8-bit operand. If a rotate instruction is used next, this C bit is shifted into the operand for the rotate, and the bit that gets shifted out the other end of the operand replaces the value in C so it can be used in subsequent rotate instructions.

8.2.5 Indexed Data Register (D[X])

This 8-bit indexed data register allows the user to access the data in the direct page address space indexed by X. This register resides at the memory mapped location $000E. For details on the D[X] register, please refer to Section 8.3.8, “Indexed Addressing Mode (IX, Implemented by Pseudo Instructions).”

8.2.6 Index Register (X)

This 8-bit index register allows the user to index or address any location in the direct page address space. This register resides at the memory mapped location $000F. For details on the X register, please refer to Section 8.3.8, “Indexed Addressing Mode (IX, Implemented by Pseudo Instructions).”

8.2.7 Page Select Register (PAGESEL)

This 8-bit page select register allows the user to access all memory locations in the entire 16K-byte address space through a page window located from $00C0 to $00FF. This register resides at the memory mapped location $001F. For details on the PAGESEL register, please refer to the RS08 Core Reference Manual.

8.3 Addressing Modes

Whenever the MCU reads information from memory or writes information into memory, an addressing mode is used to determine the exact address where the information is read from or written to. This section explains several addressing modes and how each is useful in different programming situations.

Every opcode tells the CPU to perform a certain operation in a certain way. Many instructions, such as load accumulator (LDA), allow several different ways to specify the memory location to be operated on, and each addressing mode variation requires a separate opcode. All of these variations use the same instruction mnemonic, and the assembler knows which opcode to use based on the syntax and location of the operand field. In some cases, special characters are used to indicate a specific addressing mode (such as the # [pound] symbol, which indicates immediate addressing mode). In other cases, the value of the operand tells the assembler which addressing mode to use. For example, the assembler chooses short addressing mode instead of direct addressing mode if the operand address is from $0000 to $001F. Besides allowing the assembler to choose the addressing mode based on the operand address, assembler directives can also be used to force direct or tiny/short addressing mode by using the “>” or “<” prefix before the operand, respectively.

Some instructions use more than one addressing mode. For example, the move instructions use one addressing mode to access the source value from memory and a second addressing mode to access the destination memory location. For these move instructions, both addressing modes are listed in the documentation. All branch instructions use relative (REL) addressing mode to determine the destination for the branch, but BRCLR, BRSET, CBEQ, and DBNZ also must access a memory operand. These instructions are classified by the addressing mode used for the memory operand, and the relative addressing mode for the branch offset is assumed.

The discussion in the following paragraphs includes how each addressing mode works and the syntax clues that instruct the assembler to use a specific addressing mode.

8.3.1 Inherent Addressing Mode (INH)

This addressing mode is used when the CPU inherently knows everything it needs to complete the instruction and no addressing information is supplied in the source code. Usually, the operands that the CPU needs are located in the CPU’s internal registers, as in LSLA, CLRA, INCA, SLA, RTS, and others. A few inherent instructions, including no operation (NOP) and background (BGND), have no operands.

8.3.2 Relative Addressing Mode (REL)

Relative addressing mode is used to specify the offset address for branch instructions relative to the program counter. Typically, the programmer specifies the destination with a program label or an expression in the operand field of the branch instruction; the assembler calculates the difference between the location counter (which points at the next address after the branch instruction at the time) and the address represented by the label or expression in the operand field. This difference is called the offset and is an 8-bit two's complement number. The assembler stores this offset in the object code for the branch instruction.

During execution, the CPU evaluates the condition that controls the branch. If the branch condition is true, the CPU sign-extends the offset to a 14-bit value, adds the offset to the current PC, and uses this as the address where it will fetch the next instruction and continue execution rather than continuing execution with the next instruction after the branch. Because the offset is an 8-bit two's complement value, the destination must be within the range –128 to +127 locations from the address that follows the last byte of object code for the branch instruction.

A common method to create a simple infinite loop is to use a branch instruction that branches to itself. This is sometimes used to end short code segments during debug. Typically, to get out of this infinite loop, use the debug host (through background commands) to stop the program, examine registers and memory, or to start execution from a new location. This construct is not used in normal application programs except in the case where the program has detected an error and wants to force the COP watchdog timer to timeout. (The branch in the infinite loop executes repeatedly until the watchdog timer eventually causes a reset.)

8.3.3 Immediate Addressing Mode (IMM)

In this addressing mode, the operand is located immediately after the opcode in the instruction stream. This addressing mode is used when the programmer wants to use an explicit value that is known at the time the program is written. A # (pound) symbol is used to tell the assembler to use the operand as a data value rather than an address where the desired value should be accessed.

The size of the immediate operand is always 8 bits. The assembler automatically will truncate or extend the operand as needed to match the size needed for the instruction. Most assemblers generate a warning if a 16-bit operand is provided.

It is the programmer's responsibility to use the # symbol to tell the assembler when immediate addressing should be used. The assembler does not consider it an error to leave off the # symbol because the resulting statement is still a valid instruction (although it may mean something different than the programmer intended).

8.3.4 Tiny Addressing Mode (TNY)

TNY addressing mode is capable of addressing only the first 16 bytes in the address map, from $0000 to $000F. This addressing mode is available for INC, DEC, ADD, and SUB instructions. A system can be optimized by placing the most computation-intensive data in this area of memory.

Because the 4-bit address is embedded in the opcode, only the least significant four bits of the address must be included in the instruction; this saves program space and execution time. During execution, the CPU adds 10 high-order 0s to the 4-bit operand address and uses the combined 14-bit address ($000x) to access the intended operand.

8.3.5 Short Addressing Mode (SRT)

SRT addressing mode is capable of addressing only the first 32 bytes in the address map, from $0000 to $001F. This addressing mode is available for CLR, LDA, and STA instructions. A system can be optimized by placing the most computation-intensive data in this area of memory.

Because the 5-bit address is embedded in the opcode, only the least significant five bits of the address must be included in the instruction; this saves program space and execution time. During execution, the CPU adds nine high-order 0s to the 5-bit operand address and uses the combined 14-bit address ($000x or $001x) to access the intended operand.

8.3.6 Direct Addressing Mode (DIR)

DIR addressing mode is used to access operands located in direct address space ($0000 through $00FF).

During execution, the CPU adds six high-order 0s to the low byte of the direct address operand that follows the opcode. The CPU uses the combined 14-bit address ($00xx) to access the intended operand.

8.3.7 Extended Addressing Mode (EXT)

In the extended addressing mode, the 14-bit address of the operand is included in the object code in the low-order 14 bits of the next two bytes after the opcode. This addressing mode is only used in JSR and JMP instructions for jump destination address in RS08 MCUs.

8.3.8 Indexed Addressing Mode (IX, Implemented by Pseudo Instructions)

Indexed addressing mode is sometimes called indirect addressing mode because an index register is used as a reference to access the intended operand.

An important feature of indexed addressing mode is that the operand address is computed during execution based on the current contents of the X index register located in $000F of the memory map rather than being a constant address location that was determined during program assembly. This allows writing of a program that accesses different operand locations depending on the results of earlier program instructions (rather than accessing a location that was determined when the program was written).

The index addressing mode supported by the RS08 Family uses the register X located at $000F as an index and D[X] register located at $000E as the indexed data register. By programming the index register X, any location in the direct page can be read/written via the indexed data register D[X].

These pseudo instructions can be used with all instructions supporting direct, short, and tiny addressing modes by using the D[X] as the operand.

8.4 Special Operations

Most of what the CPU does is described by the instruction set, but a few special operations must be considered, such as how the CPU starts at the beginning of an application program after power is first applied. After the program begins running, the current instruction normally determines what the CPU will do next. Two exceptional events can cause the CPU to temporarily suspend normal program execution:

- Reset events force the CPU to start over at the beginning of the application program, which forces execution to start at $3FFD.
- A host development system can cause the CPU to go to active background mode rather than continuing to the next instruction in the application program.

8.4.1 Reset Sequence

Processing begins at the trailing edge of a reset event. The number of things that can cause reset events can vary slightly from one RS08 derivative to another; however, the most common sources are: power-on reset, the external $\overline{\text{RESET}}$ pin, low-voltage reset, COP watchdog timeout, illegal opcode detect, and illegal address access. For more information about how the MCU recognizes reset events and determines the difference between internal and external causes, refer to the Resets and Interrupts chapter.

Reset events force the MCU to immediately stop what it is doing and begin responding to reset. Any instruction that was in process will be aborted immediately without completing any remaining clock cycles. A short sequence of activities is completed to decide whether the source of reset was internal or external and to record the cause of reset. For the remainder of the time, the reset source remains active and the internal clocks are stopped to save power. At the trailing edge of the reset event, the clocks resume and the CPU exits from the reset condition. The program counter is reset to $3FFD and an instruction fetch will be started after the release of reset.

For the device to execute code from the on-chip memory starting from $3FFD after reset, care should be taken to not force the BKDG pin low on the end of reset because this will force the device into active background mode where the CPU will wait for a command from the background communication interface.

8.4.2 Interrupts

The interrupt mechanism in RS08 is not used to interrupt the normal flow of instructions; it is used to wake up the RS08 from wait and stop modes. In run mode, interrupt events must be polled by the CPU. The interrupt feature is not compatible with Freescale's HC05, HC08, or HCS08 Families.

8.4.3 Wait and Stop Mode

Wait and stop modes are entered by executing a WAIT or STOP instruction, respectively. In these modes, the clocks to the CPU are shut down to save power and CPU activity is suspended. The CPU remains in this low-power state until an interrupt or reset event wakes it up. Please refer to the Resets and Interrupts chapter for the effects of wait and stop on other device peripherals.

8.4.4 Active Background Mode

Active background mode refers to the condition in which the CPU has stopped executing user program instructions and is waiting for serial commands from the background debug system. Refer to the Development Support chapter for detailed information on active background mode.

The arithmetic left shift pseudo instruction is also available because its operation is identical to logical shift left.

8.5 Summary Instruction Table

Instruction Set Summary Nomenclature

The nomenclature listed here is used in the instruction descriptions in Table 8-1 through Table 8-2.

Operators

( ) = Contents of register or memory location shown inside parentheses
← = Is loaded with (read: “gets”)
⇔ = Exchange with
& = Boolean AND
| = Boolean OR
⊕ = Boolean exclusive-OR
: = Concatenate
+ = Add

CPU registers

A = Accumulator
CCR = Condition code register
PC = Program counter
PCH = Program counter, higher order (most significant) six bits
PCL = Program counter, lower order (least significant) eight bits
SPC = Shadow program counter
SPCH = Shadow program counter, higher order (most significant) six bits
SPCL = Shadow program counter, lower order (least significant) eight bits

Memory and addressing

M = A memory location or absolute data, depending on addressing mode
rel = The relative offset, which is the two’s complement number stored in the last byte of machine code corresponding to a branch instruction
X = Pseudo index register, memory location $000F
,X or D[X] = Memory location $000E pointing to the memory location defined by the pseudo index register (location $000F)

Condition code register (CCR) bits

Z = Zero indicator
C = Carry/borrow

CCR activity notation

– = Bit not affected
0 = Bit forced to 0
1 = Bit forced to 1
Þ = Bit set or cleared according to results of operation

U	=	Undefined after the operation

Machine coding notation

dd	=	Low-order eight bits of a direct address $0000–$00FF (high byte assumed to be $00)
ii	=	One byte of immediate data
hh	=	High-order 6-bit of 14-bit extended address prefixed with 2-bit of 0
ll	=	Low-order byte of 14-bit extended address
rr	=	Relative offset

Source form

Everything in the source forms columns, *except expressions in italic characters,* is literal information which must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic is always a literal expression. All commas, pound signs (#), parentheses, and plus signs (+) are literal characters.

n — Any label or expression that evaluates to a single integer in the range 0–7.

x — Any label or expression that evaluates to a single hexadecimal integer in the range $0–$F.

opr8i — Any label or expression that evaluates to an 8-bit immediate value.

opr4a — Any label or expression that evaluates to a Tiny address (4-bit value). The instruction treats this 4-bit value as the low order four bits of an address in the 16-Kbyte address space ($0000–$000F). This 4-bit value is embedded in the low order four bits in the opcode.

opr5a — Any label or expression that evaluates to a Short address (5-bit value). The instruction treats this 5-bit value as the low order five bits of an address in the 16-Kbyte address space ($0000–$001F). This 5-bit value is embedded in the low order 5 bits in the opcode.

opr8a — Any label or expression that evaluates to an 8-bit value. The instruction treats this 8-bit value as the low order eight bits of an address in the 16-Kbyte address space ($0000–$00FF).

opr16a — Any label or expression that evaluates to a 14-bit value. On the RS08 core, the upper two bits are always 0s. The instruction treats this value as an address in the 16-Kbyte address space.

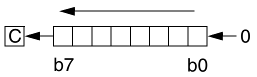
rel — Any label or expression that refers to an address that is within –128 to +127 locations from the next address after the last byte of object code for the current instruction. The assembler will calculate the 8-bit signed offset and include it in the object code for this instruction.

Address modes

INH	=	Inherent (no operands)
IMD	=	Immediate to Direct (in MOV instruction)
IMM	=	Immediate
DD	=	Direct to Direct (in MOV instruction)
DIR	=	Direct
SRT	=	Short
TNY	=	Tiny

EXT = Extended
REL = 8-bit relative offset

Table 8-1. Instruction Set Summary (Sheet 1 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
ADC *#opr8i* ADC *opr8a* ADC ,X [1] ADC X	Add with Carry	A ← (A) + (M) + (C) A ← (A) + (X) + (C)	↕	↕	IMM DIR IX DIR	A9 B9 B9 B9	ii dd 0E 0F	2 3 3 3
ADD *#opr8i* ADD *opr8a* ADD *opr4a* ADD ,X [1] ADD X	Add without Carry	A ← (A) + (M)	↕	↕	IMM DIR TNY IX DIR	AB BB 6*x* 6E 6F	ii dd	2 3 3 3 3
AND *#opr8i* AND *opr8a* AND ,X [1] AND X	Logical AND	A ← (A) & (M) A ← (A) & (X)	↕	–	IMM DIR IX DIR	A4 B4 B4 B4	ii dd 0E 0F	2 3 3 3
ASLA[1]	Arithmetic Shift Left		↕	↕	INH	48		1
BCC *rel*	Branch if Carry Bit Clear	PC ← (PC) + $0002 + *rel*, if (C) = 0	–	–	REL	34	rr	3
BCLR *n,opr8a* BCLR *n*,D[X] BCLR *n*,X	Clear Bit n in Memory	Mn ← 0	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7) IX (b0) IX (b1) IX (b2) IX (b3) IX (b4) IX (b5) IX (b6) IX (b7) DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F 11 13 15 17 19 1B 1D 1F 11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd 0E 0E 0E 0E 0E 0E 0E 0E 0F 0F 0F 0F 0F 0F 0F 0F	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + $0002 + *rel*, if (C) = 1	–	–	REL	35	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + $0002 + *rel*, if (Z) = 1	–	–	REL	37	rr	3

1. This is a pseudo instruction supported by the normal RS08 instruction set.

2. This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

Table 8-1. Instruction Set Summary (Sheet 2 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
BGND	Background	Enter Background Debug Mode	–	–	INH	BF		5+
BHS *rel*[1]	Branch if Higher or Same (Same as BCC)	PC ← (PC) + $0002 + *rel*, if (C) = 0	–	–	REL	34	rr	3
BLO *rel*[1]	Branch if Lower (Same as BCS)	PC ← (PC) + $0002 + *rel*, if (C) = 1	–	–	REL	35	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + $0002 + *rel*, if (Z) = 0	–	–	REL	36	rr	3
BRA *rel*	Branch Always	PC ← (PC) + $0002 + *rel*	–	–	REL	30	rr	3
BRN *rel*[1]	Branch Never	PC ← (PC) + $0002	–	–	REL	30	00	3
BRCLR *n*,*opr8a*,*rel*	Branch if Bit *n* in Memory Clear	PC ← (PC) + $0003 + *rel*, if (Mn) = 0	–	↕	DIR (b0)	01	dd rr	5
					DIR (b1)	03	dd rr	5
					DIR (b2)	05	dd rr	5
					DIR (b3)	07	dd rr	5
					DIR (b4)	09	dd rr	5
					DIR (b5)	0B	dd rr	5
					DIR (b6)	0D	dd rr	5
					DIR (b7)	0F	dd rr	5
BRCLR *n*,D[X],*rel*					IX (b0)	01	0E rr	5
					IX (b1)	03	0E rr	5
					IX (b2)	05	0E rr	5
					IX (b3)	07	0E rr	5
					IX (b4)	09	0E rr	5
					IX (b5)	0B	0E rr	5
					IX (b6)	0D	0E rr	5
					IX (b7)	0F	0E rr	5
BRCLR *n*,X,*rel*					DIR (b0)	01	0F rr	5
					DIR (b1)	03	0F rr	5
					DIR (b2)	05	0F rr	5
					DIR (b3)	07	0F rr	5
					DIR (b4)	09	0F rr	5
					DIR (b5)	0B	0F rr	5
					DIR (b6)	0D	0F rr	5
					DIR (b7)	0F	0F rr	5
BRSET *n*,*opr8a*,*rel*	Branch if Bit *n* in Memory Set	PC ← (PC) + $0003 + *rel*, if (Mn) = 1	–	↕	DIR (b0)	00	dd rr	5
					DIR (b1)	02	dd rr	5
					DIR (b2)	04	dd rr	5
					DIR (b3)	06	dd rr	5
					DIR (b4)	08	dd rr	5
					DIR (b5)	0A	dd rr	5
					DIR (b6)	0C	dd rr	5
					DIR (b7)	0E	dd rr	5
BRSET *n*,D[X],*rel*					IX (b0)	00	0E rr	5
					IX (b1)	02	0E rr	5
					IX (b2)	04	0E rr	5
					IX (b3)	06	0E rr	5
					IX (b4)	08	0E rr	5
					IX (b5)	0A	0E rr	5
					IX (b6)	0C	0E rr	5
					IX (b7)	0E	0E rr	5
BRSET *n*,X,*rel*					DIR (b0)	00	0F rr	5
					DIR (b1)	02	0F rr	5
					DIR (b2)	04	0F rr	5
					DIR (b3)	06	0F rr	5
					DIR (b4)	08	0F rr	5
					DIR (b5)	0A	0F rr	5
					DIR (b6)	0C	0F rr	5
					DIR (b7)	0E	0F rr	5

[1] This is a pseudo instruction supported by the normal RS08 instruction set.

[2] This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

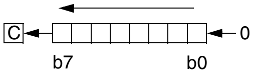
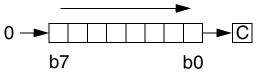
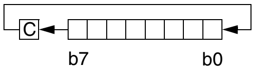
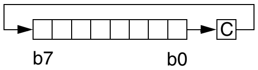
Table 8-1. Instruction Set Summary (Sheet 3 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
BSET *n,opr8a* BSET *n*,D[X] BSET *n*,X	Set Bit *n* in Memory	Mn ← 1	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7) IX (b0) IX (b1) IX (b2) IX (b3) IX (b4) IX (b5) IX (b6) IX (b7) DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E 10 12 14 16 18 1A 1C 1E 10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd 0E 0E 0E 0E 0E 0E 0E 0E 0F 0F 0F 0F 0F 0F 0F 0F	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
BSR *rel*	Branch Subroutine	PC ← (PC) + 2 Push PC to shadow PC PC ← (PC) + *rel*	–	–	REL	AD	rr	3
CBEQA #*opr8i,rel* CBEQ *opr8a,rel* CBEQ ,X,*rel* [1],[2] CBEQ X,*rel* [1]	Compare and Branch if Equal	PC ← (PC) + $0003 + *rel*, if (A) – (M) = $00 PC ← (PC) + $0003 + *rel*, if (A) – (M) = $00 PC ← (PC) + $0003 + *rel*, if (A) – (X) = $00	–	–	IMM DIR IX DIR	41 31 31 31	ii rr dd rr 0E rr 0F rr	4 5 5 5
CLC	Clear Carry Bit	C ← 0	–	0	INH	38		1
CLR *opr8a* CLR *opr5a* CLR ,X [1] CLRA CLRX [1]	Clear	M ← $00 A ← $00 X ← $00	1	–	DIR SRT IX INH INH	3F 8*x* / 9*x* 8E 4F 8F	dd	3 2 2 1 2
CMP #*opr8i* CMP *opr8a* CMP ,X [1] CMP X [1]	Compare Accumulator with Memory	(A) – (M) (A) – (X)	↕	↕	IMM DIR IX INH	A1 B1 B1 B1	ii dd 0E 0F	2 3 3 3
COMA	Complement (One's Complement)	$A \leftarrow (\overline{A})$	↕	1	INH	43		1
DBNZ *opr8a,rel* DBNZ ,X,rel [1] DBNZA *rel* DBNZX rel [1]	Decrement and Branch if Not Zero	A ← (A) – $01 or M ← (M) - $01 PC ← (PC) + $0003 + *rel* if (result) ≠ 0 for DBNZ direct PC ← (PC) + $0002 + *rel* if (result) ≠ 0 for DBNZA X ← (X) – $01 PC ← (PC) + $0003 + *rel* if (result) ≠ 0	–	–	DIR IX INH INH	3B 3B 4B 3B	dd rr 0E rr rr 0F rr	7 7 4 7
DEC *opr8a* DEC *opr4a* DEC ,X [1] DECA DEC X	Decrement	M ← (M) – $01 A ← (A) – $01 X ← (X) – $01	↕	–	DIR TNY IX INH DIR	3A 5*x* 5E 4A 5F	dd	5 4 4 1 4

[1] This is a pseudo instruction supported by the normal RS08 instruction set.

[2] This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

Table 8-1. Instruction Set Summary (Sheet 4 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
EOR #*opr8i* EOR *opr8a* EOR ,X [1] EOR X	Exclusive OR Memory with Accumulator	A ← (A ⊕ M) A ← (A ⊕ X)	↕	–	IMM DIR IX DIR	A8 B8 B8 B8	ii dd 0E 0F	2 3 3 3
INC *opr8a* INC *opr4a* INC ,X [1] INCA INCX [1]	Increment	M ← (M) + $01 A ← (A) + $01 X ← (X) + $01	↕	–	DIR TNY IX INH INH	3C 2*x* 2E 4C 2F	dd	5 4 4 1 4
JMP *opr16a*	Jump	PC ← Effective Address	–	–	EXT	BC	hh ll	4
JSR *opr16a*	Jump to Subroutine	PC ← (PC) + 3 Push PC to shadow PC PC ← Effective Address	–	–	EXT	BD	hh ll	4
LDA #*opr8i* LDA *opr8a* LDA *opr5a* LDA ,X [1]	Load Accumulator from Memory	A ← (M)	↕	–	IMM DIR SRT IX	A6 B6 C*x*/D*x* CE	ii dd	2 3 3 3
LDX #*opr8i* [1] LDX *opr8a* [1] LDX ,X [1]	Load Index Register from Memory	$0F ← (M)	↕	–	IMD DIR IX	3E 4E 4E	ii 0F dd 0F 0E 0E	4 5 5
LSLA	Logical Shift Left		↕	↕	INH	48		1
LSRA	Logical Shift Right		↕	↕	INH	44		1
MOV *opr8a,opr8a* MOV #*opr8i,opr8a* MOV D[X],*opr8a* MOV *opr8a*,D[X] MOV #*opr8i*,D[X]	Move	$(M)_{destination} \leftarrow (M)_{source}$	↕	–	DD IMD IX/DIR DIR/IX IMM/IX	4E 3E 4E 4E 3E	dd dd ii dd 0E dd dd 0E ii 0E	5 4 5 5 4
NOP	No Operation	None	–	–	INH	AC		1
ORA #*opr8i* ORA *opr8a* ORA ,X [1] ORA X	Inclusive OR Accumulator and Memory	A ← (A) ǀ (M) A ← (A) ǀ (X)	↕	–	IMM DIR IX DIR	AA BA BA BA	ii dd 0E 0F	2 3 3 3
ROLA	Rotate Left through Carry		↕	↕	INH	49		1
RORA	Rotate Right through Carry		↕	↕	INH	46		1
RTS	Return from Subroutine	Pull PC from shadow PC	–	–	INH	BE		3
SBC #*opr8i* SBC *opr8a* SBC ,X [1] SBC X	Subtract with Carry	A ← (A) – (M) – (C) A ← (A) – (X) – (C)	↕	↕	IMM DIR IX DIR	A2 B2 B2 B2	ii dd 0E 0F	2 3 3 3

1. This is a pseudo instruction supported by the normal RS08 instruction set.

2. This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

Table 8-1. Instruction Set Summary (Sheet 5 of 5)

Source Form	Description	Operation	Effect on CCR		Address Mode	Opcode	Operand	Cycles
			Z	C				
SEC	Set Carry Bit	C ← 1	–	1	INH	39		1
SHA	Swap Shadow PC High with A	A ⇔ SPCH	–	–	INH	45		1
SLA	Swap Shadow PC Low with A	A ⇔ SPCL	–	–	INH	42		1
STA *opr8a* STA *opr5a* STA ,X [1] STA X	Store Accumulator in Memory	M ← (A)	↕	–	DIR SRT IX SRT	B7 E*x* / F*x* EE EF	dd	3 2 2 2
STX *opr8a* [1]	Store Index Register in Memory	M ← (X)	↕	–	DIR	4E	0F dd	5
STOP	Put MCU into stop mode		–	–	INH	AE		2+
SUB #*opr8i* SUB *opr8a* SUB *opr4a* SUB ,X [1] SUB X	Subtract	A ← (A) – (M) A ← (A) – (X)	↕	↕	IMM DIR TNY IX DIR	A0 B0 7*x* 7E 7F	ii dd	2 3 3 3 3
TAX[1]	Transfer A to X	X ← (A)	↕	–	INH	EF		2
TST opr8a [1] TSTA [1] TST ,X [1] TSTX [1]	Test for Zero	(M) – $00 (A) – $00 (X) – $00	↕	–	DD INH IX INH	4E AA 4E 4E	dd dd 00 0E 0E 0F 0F	5 2 5 5
TXA[1]	Transfer X to A	A ← (X)	↕	–	INH	CF		3
WAIT	Put MCU into WAIT mode		–	–	INH	AF		2+

[1] This is a pseudo instruction supported by the normal RS08 instruction set.

[2] This instruction is different from that of the HC08 and HCS08 in that the RS08 does not auto-increment the index register.

Table 8-2. Opcode Map

	DIR	DIR	TNY	DIR/REL	INH	TNY	TNY	TNY	SRT	SRT	IMM/INH	DIR/EXT	SRT	SRT	SRT	SRT
HIGH LOW	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	5 BRSET0 3 DIR	5 BSET0 2 DIR	4 INC 1 TNY	3 BRA 2 REL	▨	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 SUB 2 IMM	3 SUB 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
1	5 BRCLR0 3 DIR	5 BCLR0 2 DIR	4 INC 1 TNY	5 CBEQ 3 DIR	4 CBEQA 3 IMM	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 CMP 2 IMM	3 CMP 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
2	5 BRSET1 3 DIR	5 BSET1 2 DIR	4 INC 1 TNY	▨	1 SLA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 SBC 2 IMM	3 SBC 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
3	5 BRCLR1 3 DIR	5 BCLR1 2 DIR	4 INC 1 TNY	▨	1 COMA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	▨	▨	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
4	5 BRSET2 3 DIR	5 BSET2 2 DIR	4 INC 1 TNY	3 BCC 2 REL	1 LSRA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 AND 2 IMM	3 AND 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
5	5 BRCLR2 3 DIR	5 BCLR2 2 DIR	4 INC 1 TNY	3 BCS 2 REL	1 SHA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	▨	▨	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
6	5 BRSET3 3 DIR	5 BSET3 2 DIR	4 INC 1 TNY	3 BNE 2 REL	1 RORA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 LDA 2 IMM	3 LDA 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
7	5 BRCLR3 3 DIR	5 BCLR3 2 DIR	4 INC 1 TNY	3 BEQ 2 REL	▨	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	▨	3 STA 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
8	5 BRSET4 3 DIR	5 BSET4 2 DIR	4 INC 1 TNY	1 CLC 1 INH	1 LSLA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 EOR 2 IMM	3 EOR 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
9	5 BRCLR4 3 DIR	5 BCLR4 2 DIR	4 INC 1 TNY	1 SEC 1 INH	1 ROLA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 ADC 2 IMM	3 ADC 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
A	5 BRSET5 3 DIR	5 BSET5 2 DIR	4 INC 1 TNY	5 DEC 2 DIR	1 DECA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 ORA 2 IMM	3 ORA 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
B	5 BRCLR5 3 DIR	5 BCLR5 2 DIR	4 INC 1 TNY	6 DBNZ 3 DIR	4 DBNZA 2 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2 ADD 2 IMM	3 ADD 2 DIR	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
C	5 BRSET6 3 DIR	5 BSET6 2 DIR	4 INC 1 TNY	5 INC 2 DIR	1 INCA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	1 NOP 1 INH	4 JMP 3 EXT	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
D	5 BRCLR6 3 DIR	5 BCLR6 2 DIR	4 INC 1 TNY	▨	▨	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	3 BSR 2 REL	4 JSR 3 EXT	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
E	5 BRSET7 3 DIR	5 BSET7 2 DIR	4 INC 1 TNY	4 MOV 3 IMD	5 MOV 3 DD	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2+ STOP 1 INH	3 RTS 1 INH	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT
F	5 BRCLR7 3 DIR	5 BCLR7 2 DIR	4 INC 1 TNY	3 CLR 2 DIR	1 CLRA 1 INH	4 DEC 1 TNY	3 ADD 1 TNY	3 SUB 1 TNY	2 CLR 1 SRT	2 CLR 1 SRT	2+ WAIT 1 INH	5+ BGND 1 INH	3 LDA 1 SRT	3 LDA 1 SRT	2 STA 1 SRT	2 STA 1 SRT

INH	Inherent	REL	Relative
IMM	Immediate	SRT	Short
DIR	Direct	TNY	Tiny
EXT	Extended		
DD	Direct-Direct	IMD	Immediate-Direct

▨	Gray box is decoded as illegal instruction

High Byte of Opcode in Hexadecimal B

Low Byte of Opcode in Hexadecimal	0	3 SUB 2 DIR	RS08 Cycles Opcode Mnemonic Number of Bytes / Addressing Mode

Chapter 9
Analog-to-Digital Converter (ADC10V1)

9.1 Introduction

The 10-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system chip.

All ADC pins share with LCD pins. ADC functionality must be disabled for those used as LCD pins.

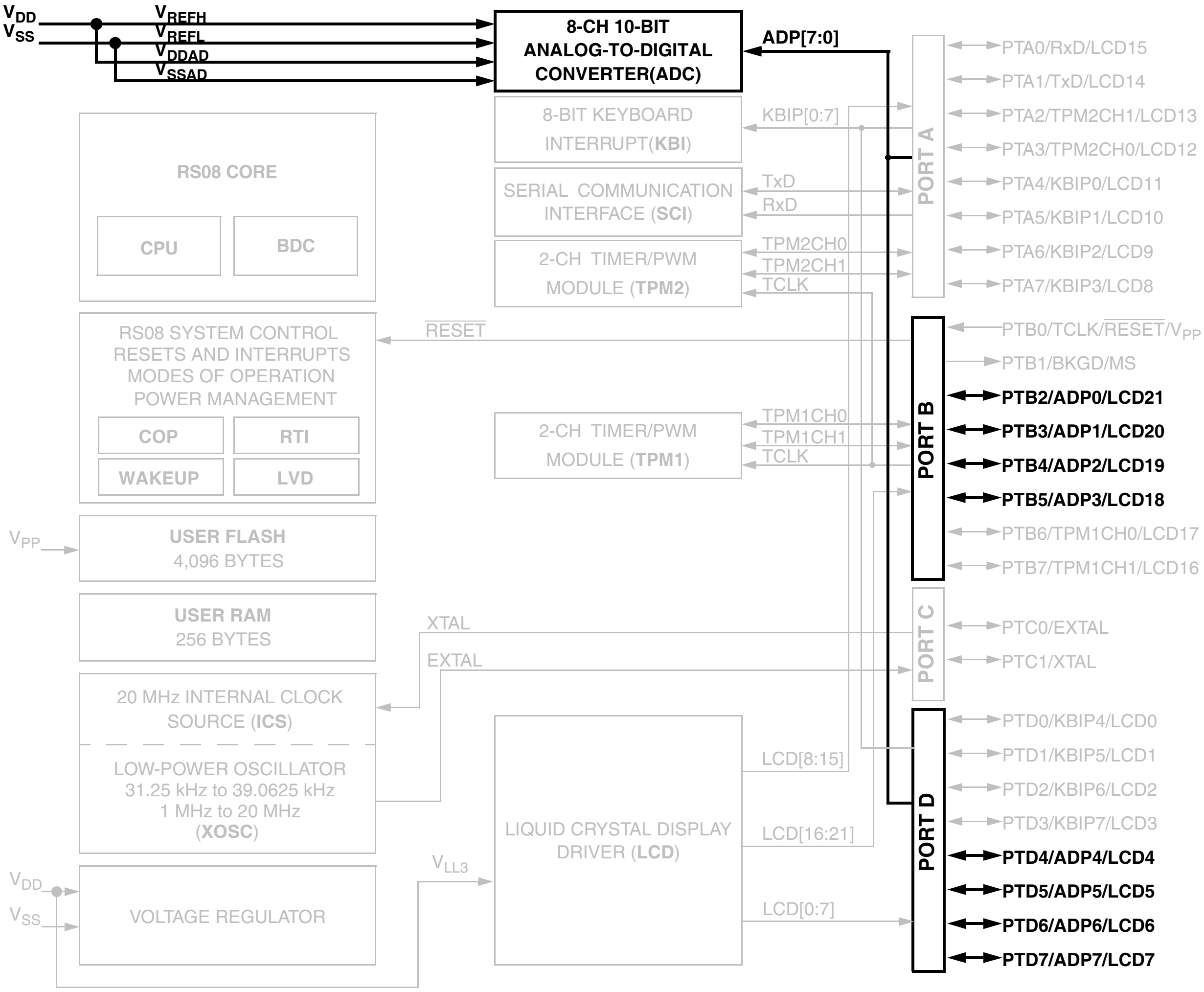
9.1.1 ADC Channel Assignments

Figure 9-1 shows the ADC channel assignments. Reserved channels convert to an unknown value.

Figure 9-1. ADC Channel Assignment

ADCH	Input Select
00000	AD0
00001	AD1
00010	AD2
00011	AD3
00100	AD4
00101	AD5
00110	AD6
00111	AD7
01000	V_{REFL}
01001	V_{REFL}
01010	V_{REFL}
01011	V_{REFL}
01100	V_{REFL}
01101	V_{REFL}
01110	V_{REFL}
01111	V_{REFL}
10000	V_{REFL}
10001	V_{REFL}
10010	V_{REFL}
10011	V_{REFL}
10100	V_{REFL}
10101	V_{LL2}
10110	V_{LL1}
10111	Reserved
11000	Reserved
11001	Reserved
11010	Temperature Sensor
11011	Reserved
11100	V_{REFH}
11101	V_{REFH}
11110	V_{REFL}
11111	Module disabled

Figure 9-2 shows the MC9RS08LE4 block diagram, highlighting the ADC block and pins.



NOTES:
1. PTB0/TCLK/RESET/V_{PP} is an input-only pin when used as port pin
2. PTB1/BKGD/MS is an output-only pin

Figure 9-2. MC9RS08LE4 Series Block Diagram Highlighting ADC Block and Pins

9.1.2 Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) within the module, or the alternate clock (ALTCLK). The ALTCLK on the MC9RS08LE4 is connected to the ICSERCLK. Refer to Chapter 10, “Internal Clock Source (S08ICSV1)” for more information.

9.1.3 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 9-1 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \qquad \textit{Eqn. 9-1}$$

where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.
- m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and m values in the data sheet.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP}, and compares it to V_{TEMP25}. If V_{TEMP} is greater than V_{TEMP25} the cold slope value is applied in Equation 9-1. If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in Equation 9-1.

9.1.4 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 10 bits resolution.
- Up to 28 analog inputs.
- Output formatted in 10- or 8-bit right-justified format.
- Single or continuous conversion (automatic return to idle after single conversion).
- Configurable sample time and conversion speed/power.
- Conversion complete flag and interrupt.
- Input clock selectable from up to four sources.
- Operation in wait or stop modes for lower noise operation.
- Asynchronous clock source for lower noise operation.
- Selectable asynchronous hardware conversion trigger.
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value.

9.1.5 Block Diagram

Figure 9-3 provides a block diagram of the ADC module

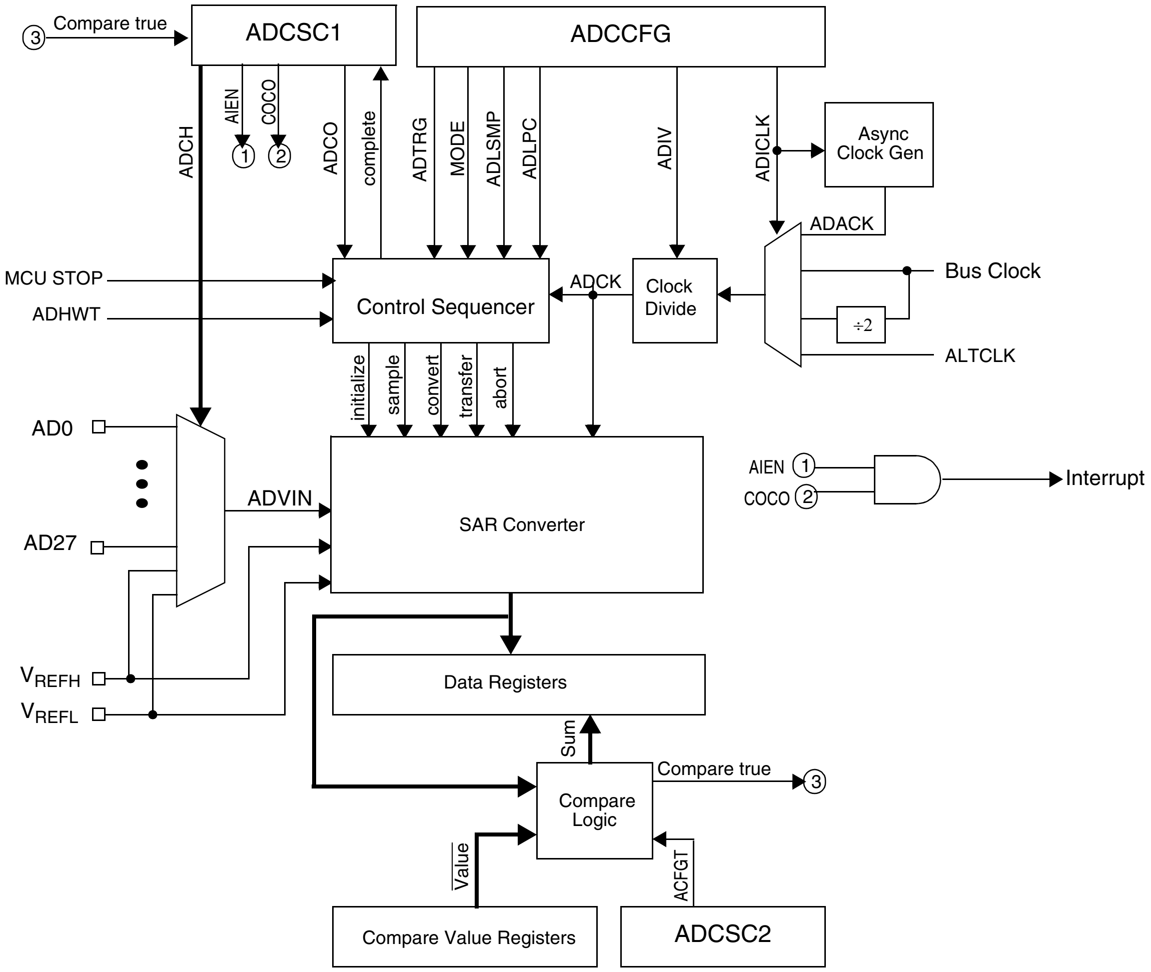


Figure 9-3. ADC Block Diagram

9.2 External Signal Description

The ADC module supports up to 28 separate analog inputs. It also requires four supply/reference/ground connections.

Table 9-1. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V_{REFH}	High reference voltage
V_{REFL}	Low reference voltage
V_{DDAD}	Analog power supply
V_{SSAD}	Analog ground

9.2.1 Analog Power (V_{DDAD})

The ADC analog portion uses V_{DDAD} as its power connection. In some packages, V_{DDAD} is connected internally to V_{DD}. If externally available, connect the V_{DDAD} pin to the same voltage potential as V_{DD}. External filtering might be necessary to ensure clean V_{DDAD} for good results.

9.2.2 Analog Ground (V_{SSAD})

The ADC analog portion uses V_{SSAD} as its ground connection. In some packages, V_{SSAD} is connected internally to V_{SS}. If externally available, connect the V_{SSAD} pin to the same voltage potential as V_{SS}.

9.2.3 Voltage Reference High (V_{REFH})

V_{REFH} is the high reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDAD}. If externally available, V_{REFH} can be connected to the same potential as V_{DDAD}, or can be driven by an external source that is between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}).

9.2.4 Voltage Reference Low (V_{REFL})

V_{REFL} is the low reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSAD}. If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSAD}.

9.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

9.3 Register Definition

These memory mapped registers control and monitor operation of the ADC:

- Status and control register, ADCSC1and ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin enable registers, APCTL1, APCTL2, APCTL3

9.3.1 Status and Control Register 1 (ADCSC1)

This section describes the function of the ADC status and control register 1 (ADCSC1). Writing ADCSC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

	7	6	5	4	3	2	1	0
R	COCO	AIEN	ADCO	ADCH				
W	▨							
Reset:	0	0	0	1	1	1	1	1

▨ = Unimplemented or Reserved

Figure 9-4. Status and Control Register (ADCSC1)

Table 9-2. ADCSC1 Register Field Descriptions

Field	Description
7 COCO	**Conversion Complete Flag** — The COCO flag is a read-only bit which is set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1) the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared whenever ADCSC1 is written or whenever ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	**Interrupt Enable** — AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	**Continuous Conversion Enable** — ADCO is used to enable continuous conversions. 0 One conversion following a triggered operation[1] 1 Continuous conversions initiated following a triggered operation is selected.
4:0 ADCH	**Input Channel Select** — The ADCH bits form a 5-bit field which selects one of the input channels. The input channels are detailed in Figure 9-5. The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

[1] See Table 9-5 for how to specify a hardware or software trigger type(ADTRG).

Figure 9-5. Input Channel Select

ADCH	Input Select
00000	AD0
00001	AD1
00010	AD2
00011	AD3
00100	AD4
00101	AD5
00110	AD6
00111	AD7
10000	AD16
10001	AD17
10010	AD18
10011	AD19
10100	AD20
10101	AD21
10110	AD22
10111	AD23

Figure 9-5. Input Channel Select (continued)

ADCH	Input Select	ADCH	Input Select
01000	AD8	11000	AD24
01001	AD9	11001	AD25
01010	AD10	11010	AD26
01011	AD11	11011	AD27
01100	AD12	11100	Reserved
01101	AD13	11101	V_{REFH}
01110	AD14	11110	V_{REFL}
01111	AD15	11111	Module disabled

9.3.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register is used to control the compare function, conversion trigger and conversion active of the ADC module.

	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	0	0	R[1]	R[1]
W	▨				▨	▨		
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

[1] Bits 1 and 0 are reserved bits that must always be written to 0.

Figure 9-6. Status and Control Register 2 (ADCSC2)

Table 9-3. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	**Conversion Active** — ADACT indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	**Conversion Trigger Select** — ADTRG selects the type of trigger to be used for initiating a conversion. 0 Software trigger selected.(initiates a conversion following a write to ADCSC1). 1 Hardware trigger selected.(initiates a conversion following the assertion of the ADHWT input).
5 ACFE	**Compare Function Enable** — ACFE is used to enable the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	**Compare Function Greater Than Enable** — ACFGT configures the compare function to trigger upon conversion of the input being monitored. 0 Compare triggered when input is less than compare level 1 Compare triggered when input is greater than or equal to compare level

9.3.3 Data Result High Register (ADCRH)

ADCRH contains the upper two bits of the result of a 10-bit conversion. When configured for 8-bit conversions both ADR8 and ADR9 are equal to zero. ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. In 10-bit MODE, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, then the intermediate conversion result will be lost. In 8-bit mode there is no interlocking with ADCRL. In the case that the MODE bits are changed, any data in ADCRH becomes invalid.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	ADR9	ADR8
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-7. Data Result High Register (ADCRH)

9.3.4 Data Result Low Register (ADCRL)

ADCRL contains the lower eight bits of the result of a 10-bit conversion, and all eight bits of an 8-bit conversion. This register updates each time a conversion completes except when an automatic compare is enabled and the compare condition is not met. In 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion completes, then the intermediate conversion results are lost. In 8-bit mode, there is no interlocking with ADCRH. When MODE bits are changed, data in ADCRL becomes invalid.

	7	6	5	4	3	2	1	0
R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-8. Data Result Low Register (ADCRL)

9.3.5 Compare Value High Register (ADCCVH)

This register holds the upper two bits of the 10-bit compare value. These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled.In 8-bit operation, ADCCVH is not used during compare.

	7	6	5	4	3	2	1	0
R	0	0	0	0	▨	▨	ADCV9	ADCV8
W	▨	▨	▨	▨				
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 9-9. Compare Value High Register (ADCCVH)

9.3.6 Compare Value Low Register (ADCCVL)

This register holds the lower 8 bits of the 10-bit compare value, or all 8 bits of the 8-bit compare value. Bits ADCV7:ADCV0 are compared to the lower 8 bits of the result following a conversion in 10-bit or 8-bit mode.

	7	6	5	4	3	2	1	0
R W	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
Reset:	0	0	0	0	0	0	0	0

Figure 9-10. Compare Value Low Register(ADCCVL)

9.3.7 Configuration Register (ADCCFG)

ADCCFG is used to select the mode of operation, clock source, clock divide, and configure for low power or long sample time.

	7	6	5	4	3	2	1	0
R W	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
Reset:	0	0	0	0	0	0	0	0

Figure 9-11. Configuration Register (ADCCFG)

Table 9-4. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	**Low Power Configuration** — ADLPC controls the speed and power configuration of the successive approximation converter. This optimize power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: {FC31} power is reduced at the expense of maximum clock speed.
6:5 ADIV	**Clock Divide Select** — ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. Table 9-5 shows the available clock configurations.

Table 9-4. ADCCFG Register Field Descriptions (continued)

Field	Description
4 ADLSMP	**Long Sample Time Configuration** — ADLSMP selects between long and short sample periods. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time
3:2 MODE	**Conversion Mode Selection** — MODE bits are used to select between 10- or 8-bit operation. See Table 9-6.
1:0 ADICLK	**Input Clock Select** — ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 9-7.

Table 9-5. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 9-6. Conversion Modes

MODE	Mode Description
00	8-bit conversion (N=8)
01	Reserved
10	10-bit conversion (N=10)
11	Reserved

Table 9-7. Input Clock Select

ADICLK	Selected Clock Source
00	Bus clock
01	Bus clock divided by 2
10	Alternate clock (ALTCLK)
11	Asynchronous clock (ADACK)

9.3.8 Pin Control 1 Register (APCTL1)

The pin control registers disable the I/O port control of MCU pins used as analog inputs. APCTL1 controls the pins associated with channels 0–7 of the ADC module.

	7	6	5	4	3	2	1	0
R W	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
Reset:	0	0	0	0	0	0	0	0

Figure 9-12. Pin Control 1 Register (APCTL1)

Table 9-8. APCTL1 Register Field Descriptions

Field	Description
7 ADPC7	**ADC Pin Control 7** — ADPC7 is used to control the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	**ADC Pin Control 6** — ADPC6 is used to control the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	**ADC Pin Control 5** — ADPC5 is used to control the pin associated with channel AD5. 0 AD5 pin I/O control enabled 1 AD5 pin I/O control disabled
4 ADPC4	**ADC Pin Control 4** — ADPC4 is used to control the pin associated with channel AD4. 0 AD4 pin I/O control enabled 1 AD4 pin I/O control disabled
3 ADPC3	**ADC Pin Control 3** — ADPC3 is used to control the pin associated with channel AD3. 0 AD3 pin I/O control enabled 1 AD3 pin I/O control disabled
2 ADPC2	**ADC Pin Control 2** — ADPC2 is used to control the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled
1 ADPC1	**ADC Pin Control 1** — ADPC1 is used to control the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	**ADC Pin Control 0** — ADPC0 is used to control the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

9.3.9 Pin Control 2 Register (APCTL2)

APCTL2 is used to control channels 8–15 of the ADC module.

	7	6	5	4	3	2	1	0
R W	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
Reset:	0	0	0	0	0	0	0	0

Figure 9-13. Pin Control 2 Register (APCTL2)

Table 9-9. APCTL2 Register Field Descriptions

Field	Description
7 ADPC15	**ADC Pin Control 15** — ADPC15 is used to control the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	**ADC Pin Control 14** — ADPC14 is used to control the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	**ADC Pin Control 13** — ADPC13 is used to control the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	**ADC Pin Control 12** — ADPC12 is used to control the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	**ADC Pin Control 11** — ADPC11 is used to control the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	**ADC Pin Control 10** — ADPC10 is used to control the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled
1 ADPC9	**ADC Pin Control 9** — ADPC9 is used to control the pin associated with channel AD9. 0 AD9 pin I/O control enabled 1 AD9 pin I/O control disabled
0 ADPC8	**ADC Pin Control 8** — ADPC8 is used to control the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

9.3.10 Pin Control 3 Register (APCTL3)

APCTL3 is used to control channels 16–23 of the ADC module.

	7	6	5	4	3	2	1	0
R W	ADPC23	ADPC22	ADPC21	ADPC20	ADPC19	ADPC18	ADPC17	ADPC16
Reset:	0	0	0	0	0	0	0	0

Figure 9-14. Pin Control 3 Register (APCTL3)

Table 9-10. APCTL3 Register Field Descriptions

Field	Description
7 ADPC23	**ADC Pin Control 23** — ADPC23 is used to control the pin associated with channel AD23. 0 AD23 pin I/O control enabled 1 AD23 pin I/O control disabled
6 ADPC22	**ADC Pin Control 22** — ADPC22 is used to control the pin associated with channel AD22. 0 AD22 pin I/O control enabled 1 AD22 pin I/O control disabled
5 ADPC21	**ADC Pin Control 21** — ADPC21 is used to control the pin associated with channel AD21. 0 AD21 pin I/O control enabled 1 AD21 pin I/O control disabled
4 ADPC20	**ADC Pin Control 20** — ADPC20 is used to control the pin associated with channel AD20. 0 AD20 pin I/O control enabled 1 AD20 pin I/O control disabled
3 ADPC19	**ADC Pin Control 19** — ADPC19 is used to control the pin associated with channel AD19. 0 AD19 pin I/O control enabled 1 AD19 pin I/O control disabled
2 ADPC18	**ADC Pin Control 18** — ADPC18 is used to control the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled
1 ADPC17	**ADC Pin Control 17** — ADPC17 is used to control the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	**ADC Pin Control 16** — ADPC16 is used to control the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

9.4 Functional Description

The ADC module is disabled during reset or when the ADCH bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle, the module is in its lowest power state.

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. The selected channel voltage is converted by a successive approximation algorithm into an 11-bit digital result. In 8-bit mode, the selected channel voltage is converted into a 9-bit digital result by a successive approximation algorithm.

When the conversion completes, the result is placed in the data registers (ADCRH and ADCRL).In 10-bit mode, the result is rounded to 10 bits and placed in ADCRH and ADCRL. In 8-bit mode, the result is rounded to 8 bits and placed in ADCRL. The conversion complete flag (COCO) is then set, and an interrupt is generated if the conversion complete interrupt has been enabled (AIEN = 1).

The ADC module can automatically compare a conversion result with the contents of its compare registers. Set the ACFE bit to enble the compare function and operate in conjunction with any of the conversion modes and configurations.

9.4.1 Clock Select and Divide Control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADICLK bits.

- The bus clock, equal to the frequency at which the software is executed. This is the default selection following reset.
- The bus clock divided by 2. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK) – This clock is generated from a clock source within the ADC module. When selected as the clock source this clock remains active while the MCU is in wait or stop mode and allows conversions in these modes for lower noise operation.

The selected clock's frequency must fall within the range specified for ADCK. If the available clocks are too slow, the ADC will not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divided by 1, 2, 4, or 8.

9.4.2 Input Select and Pin Control

The pin control registers (APCTL3, APCTL2, and APCTL1) disable the I/O port control of the pins used as analog inputs.When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

9.4.3 Hardware Trigger

The ADC module enables ADHWT, a selectable asynchronous hardware conversion trigger, when the ADTRG bit is set. This source is not available on all MCUs. Consult the module introduction for information on the ADHWT source specific to this MCU.

When ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion initiates on the rising edge of ADHWT. If a conversion is in progress when a rising edge occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed. The hardware trigger function operates with any of the conversion modes and configurations.

9.4.4 Conversion Control

Conversions can be performed in 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

9.4.4.1 Initiating Conversions

A conversion is initiated:

- A write to ADCSC1 (with ADCH bits not all 1s) if software triggered operation is selected.
- A hardware trigger (ADHWT) event if hardware triggered operation is selected.
- The transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC1 is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

9.4.4.2 Completing Conversions

A conversion completes when the result transferred into the data-result registers, ADCRH and ADCRL. The setting of COCO. An interrupt is generated if AIEN is high when COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADCRH and ADCRL if the previous data is in the process of being read while in 10-bit MODE (the ADCRH register has been read; the ADCRL register has not). When blocking is active, the data transfer is blocked, COCO is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation terminates. In all other cases, when a data transfer is blocked, another conversion initiates regardless of the state of ADCO (single or continuous conversions enabled).

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this, the data registers must not be read after initiating a single conversion until the conversion completes.

9.4.4.3 Aborting Conversions

Any conversion in progress aborts when:

- A write to ADCSC1 occurs (the current conversion is aborted and a new conversion is initiated, if ADCH are not all 1s).
- A write to ADCSC2, ADCCFG, ADCCVH, or ADCCVL occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset.
- The MCU enters stop mode with ADACK not enabled.

When a conversion is aborted, the contents of ADCRH and ADCRL, data registers is not altered but continues to be the values transferred after the completion of the last successful conversion. If a reset aborts the conversion, ADCRH and ADCRL return to their reset states.

9.4.4.4 Power Control

The ADC module remains in idle until a conversion is initiated. If ADACK is selected as the conversion clock source, the ADACK clock generator is also enabled.

Setting ADLPC can reduce power consumption. resulting in a lower maximum value for f_{ADCK} (see the electrical specifications).

9.4.4.5 Total Conversion Time

The total conversion time depends on the sample time (as determined by ADLSMP), the MCU bus frequency, the conversion mode (8-bit or 10-bit), and the frequency of the conversion clock (f_{ADCK}). After the module becomes active, sampling of the input begins. ADLSMP is used to select between short and long sample times.When sampling completes, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The conversion result transferrs to ADCRH and ADCRL upon completion of the conversion algorithm.

- If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADLSMP=0).
- If the bus frequency is less than 1/11th of the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when long sample is enabled (ADLSMP=1).

The maximum total conversion time for different conditions is summarized in Table 9-11.

Table 9-11. Total Conversion Time vs. Control Conditions

Conversion Type	ADICLK	ADLSMP	Max Total Conversion Time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 μs + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	0	5 μs + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 μs + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit	11	1	5 μs + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

The chosen clock source and the divide ratio determine the maximum total conversion time. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits.

For example,you could use the following equation conversion time for a single conversion in 10-bit mode:

$$\text{Conversion time} = \frac{\text{23 ADCK cyc}}{\text{8 MHz/1}} + \frac{\text{5 bus cyc}}{\text{8 MHz}} = 3.5\ \mu\text{s}$$

$$\text{Number of bus cycles} = 3.5\ \mu\text{s x 8 MHz} = \text{28 cycles}$$

where:

- **Mode:** 10-Bit
- **Input Clock Source:** Bus Clock
- **Input Clock Ratio:** Divide by 1
- **Bus Frequency:** 8 MHz
- **Conversion Time:** 3.5 μs

NOTE

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

9.4.5 Automatic Compare Function

The compare function can be configured to check for an upper or lower limit. After the input is sampled and converted, the result is added to the two's complement of the compare value (ADCCVH and ADCCVL). When comparing to an upper limit (ACFGT = 1), if the result is greater-than or equal-to the compare value, COCO is set. When comparing to a lower limit (ACFGT = 0), if the result is less than the compare value, COCO is set. The value generated by the addition of the conversion result and the two's complement of the compare value is transferred to ADCRH and ADCRL.

Following a conversion where the compare function is enabled, and the compare condition is not true, COCO is not set and no data is transferred to the result registers. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

NOTE

The compare function can be used to monitor the voltage on a channel while the MCU is in wait or stop mode. The ADC interrupt wakes the MCU when the compare condition is met.

9.4.6 MCU Wait Mode Operation

The WAIT instruction puts the MCU in a lower power-consumption standby mode. Recovery is very fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

While in wait mode, the bus clock, bus clock divided by two, and ADACK are available as conversion clock sources. Using ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for this MCU. Consult the module introduction for information on ALTCLK specific to this MCU.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from wait mode if the ADC interrupt is enabled (AIEN = 1).

9.4.7 MCU Stop Mode Operation

The STOP instruction puts the MCU in a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

9.4.7.1 Stop Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not the conversion clock, executing a STOP instruction aborts the current conversion and places the ADC in its idle state. The contents of ADCRH and ADCRL are unaffected by stop mode.After exiting from stop mode, a software or hardware trigger is required to resume conversions.

9.4.7.2 Stop Mode With ADACK Enabled

If ADACK is the conversion clock, the ADC continues operation during stop mode. For guaranteed ADC operation, the MCU’s voltage regulator must remain active during stop mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop mode, it continues until completion. Conversions can be initiated while the MCU is in stop mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop mode if the ADC interrupt is enabled (AIEN = 1).

NOTE

The ADC module to wake the system from low power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this, your application should ensure that the data transfer blocking mechanism (discussed in Section 9.4.4.2, “Completing Conversions) is cleared when entering stop and continuing ADC conversions.

9.5 Initialization Information

This section gives basic direction on how to initialize and configure the ADC module. Among other options, you can configure the module for:

- 8-bit or 10-bit resolution
- single or continuous conversion

- polled or interrupt approach

Refer to Table 9-5, Table 9-6, andTable 9-7 for information used in the following example.

NOTE

> Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

9.5.1 ADC Module Initialization Example

Before the ADC module completes conversions, an initialization must be performed.

9.5.1.1 Initialization Sequence

A typical initialization sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. Also used for selecting sample time and low-power configuration.
2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select continuous or once-only conversions and to enable or disable conversion complete interrupts. Also, use this register to select the input channel on which conversions are to be performed.

9.5.1.2 Pseudo-Code Example

In this example, the ADC module is configured with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

ADCCFG = 0x98 (%10011000)

Bit 7	ADLPC	1	Configures for low power (lowers maximum clock speed)
Bit 6:5	ADIV	00	Sets ADCK to input clock ÷ 1
Bit 4	ADLSMP	1	Configures for long sample time
Bit 3:2	MODE	10	Sets mode at 10-bit conversions
Bit 1:0	ADICLK	00	Selects bus clock as input clock source

ADCSC2 = 0x00 (%00000000)

Bit 7	ADACT	0	Indicates if a conversion is in progress
Bit 6	ADTRG	0	Selects software trigger
Bit 5	ACFE	0	Disables compare function
Bit 4	ACFGT	0	Not used in this example
Bit 3:2		00	Unimplemented or reserved, always reads zero
Bit 1:0		00	Reserved for Freescale internal use; always write zero

ADCSC1 = 0x41 (%01000001)

Bit 7	COCO	0	Indicates when a conversion completes
Bit 6	AIEN	1	Enable conversion complete interrupt
Bit 5	ADCO	0	Specifies one conversion only (continuous conversions disabled)
Bit 4:0	ADCH	00001	Selects input channel 1 selected as ADC input channel

ADCRH/L = 0xxx

Holds results of conversion. Read high byte (ADCRH) before low byte (ADCRL) so that conversion data cannot be overwritten with data from the next conversion.

ADCCVH/L = 0xxx

Holds compare value when compare function enabled

APCTL1=0x02

AD1 pin I/O control disabled. All other AD pins remain general purpose I/O pins

APCTL2=0x00

All other AD pins remain general purpose I/O pins

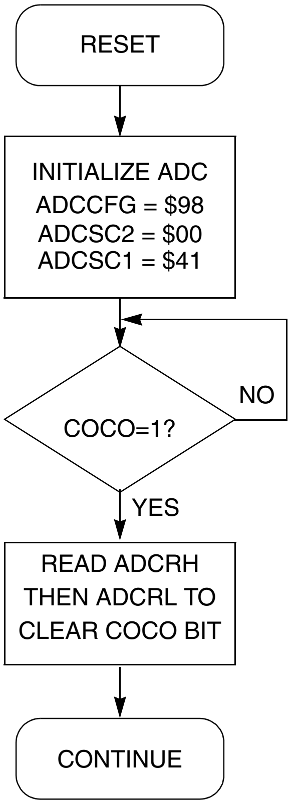


Figure 9-15. Initialization Flowchart Example

9.6 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

9.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how to use them for best results.

9.6.1.1 Analog Supply Pins

The ADC module has analog power and ground supplies (V_{DDAD} and V_{SSAD}) available as separate pins on some devices. On other devices, V_{SSAD} is shared on the same pin as the MCU digital V_{SS}; on others, both V_{SSAD} and V_{DDAD} are shared with the MCU digital supply pins. In these cases, separate pads for the analog supplies are bonded to the same pin as the corresponding digital supply so some degree of isolation between the supplies is maintained.

When available on a separate pin, V_{DDAD} and V_{SSAD} must be connected to the same voltage potential as their corresponding MCU digital supply (V_{DD} and V_{SS}) and must be routed carefully for maximum noise immunity with bypass capacitors placed as near as possible to the package.

In cases where separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSAD} pin. Typically this should be the only ground connection between these supplies if possible. The V_{SSAD} pin makes a good, single-point ground location.

9.6.1.2 Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is V_{REFH}, which can be shared on the same pin as V_{DDAD} on some devices. The low reference is V_{REFL}, which can be shared on the same pin as V_{SSAD} on some devices.

When available on a separate pin, V_{REFH} can be connected to the same potential as V_{DDAD}, or can be driven by an external source that is between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}). When available on a separate pin, V_{REFL} must be connected to the same voltage potential as V_{SSAD}. Both V_{REFH} and V_{REFL} must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current will cause a voltage drop which could result in conversion errors. Inductance in this path must be minimum (parasitic only).

9.6.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer will be in its high impedance state and the pullup is disabled. Also, the input buffer draws dc current when its input is not at either V_{DD} or V_{SS}. Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to V_{SSA}.

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL}. If the input is equal to or exceeds V_{REFH}, the converter circuit converts the signal to $3FF (full scale 10-bit representation) or $FF (full scale 8-bit representation). If the input is equal to or less than V_{REFL}, the converter circuit converts it to $000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There will be a brief current associated with V_{REFL} when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

9.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

9.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 10-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is below 5 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

9.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause a conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than V_{DDAD} / (2^N*I_{LEAK}) for less than 1/4LSB leakage error (N = 8 in 8-bit mode or 10 in 10-bit mode).

9.6.2.3 Noise-Induced Errors

System noise during the sample or conversion process can affect the concersion accuracy . The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 μF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 μF low-ESR capacitor from V_{DDAD} to V_{SSAD}.
- If inductive isolation is used from the primary supply, an additional 1 μF capacitor is placed from V_{DDAD} to V_{SSAD}.

- V_{SSAD} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - — For software triggered conversions, immediately follow the write to the ADCSC1 with a WAIT instruction or STOP instruction.
  - — For stop mode operation, select ADACK as the clock source. Operation in stop reduces V_{DD} noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

In some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop or I/O activity can not be halted, try the following recommended actions to reduce the effect of noise on the accuracy:

- Place a 0.01 μF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSAD} (this improves noise issues, but can affect sample rate based upon the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

9.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 1024 steps (in 10-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8 or 10), defined as 1LSB, is:

$$1\text{LSB} = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 9-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions, the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2LSB in 8- or 10-bit mode. As a consequence, the code width of the first ($000) conversion is only 1/2LSB and the code width of the last ($FF or $3FF) is 1.5LSB.

9.6.2.5 Linearity Errors

The ADC might also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the application should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — The difference between the actual code width of the first conversion and the ideal code width (1/2LSB). Note, if the first conversion is $001, then the difference between the actual $001 code width and its ideal (1LSB) is used.
- Full-scale error (E_{FS}) — The difference between the actual code width of the last conversion and the ideal code width (1.5LSB). Note, if the last conversion is $3FE, then the difference between the actual $3FE code width and its ideal (1LSB) is used.

- Differential non-linearity (DNL) — The worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — The highest-value (the absolute value) of the running sum DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — The difference between the actual transfer function and the ideal straight-line transfer function, and therefore includes all forms of error.

9.6.2.6 Code Jitter, Non-Monotonicity and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error;

Code jitter— is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 lsb and increases with noise. Repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 9.6.2.3 reduce this error.

Non-monotonicity —is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage.

Missing codes —are values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is monotonic and has no missing codes.

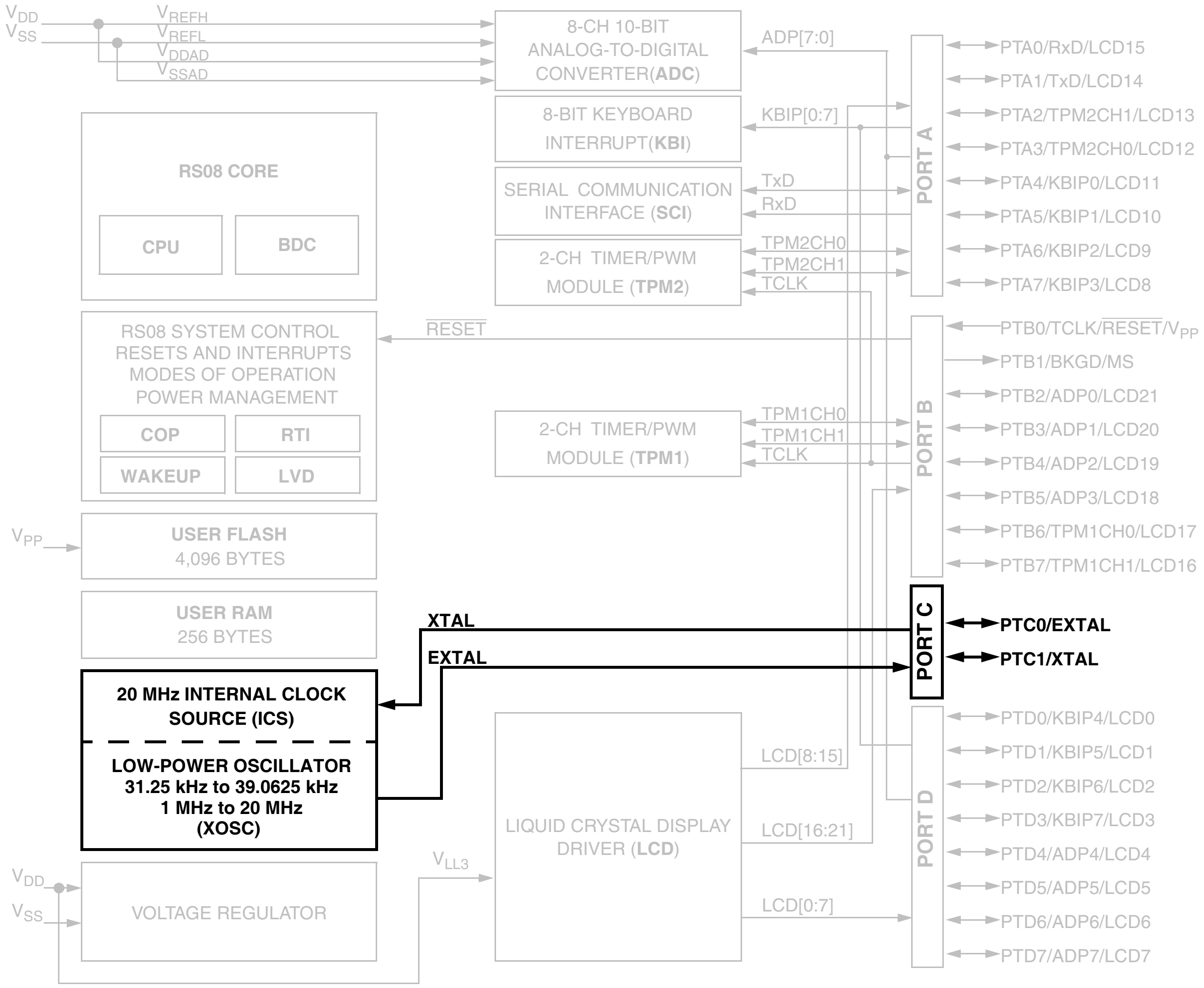
Chapter 10
Internal Clock Source (S08ICSV1)

10.1 Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by an internal reference clock. The module can provide this FLL clock or the internal reference clock as a source for the MCU system clock, ICSOUT.

No matter which clock source is chosen, ICSOUT is passed through a bus clock divider (BDIV), which allows a lower final output clock frequency to be derived. ICSOUT is two times the bus frequency.

Figure 10-1 shows the MC9RS08LE4 block diagram highlighting the ICS block and pins.



NOTES:

1. PTB0/TCLK/RESET/V_{PP} is an input-only pin when used as port pin
2. PTB1/BKGD/MS is an output-only pin

Figure 10-1. MC9RS08LE4 Block Diagram Highlighting ICS Block and Pins

10.1.1 Features

Key features of the ICS module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
  - 0.2% resolution using internal 32 kHz reference
  - 2% deviation over voltage and temperature using internal 32 kHz reference
- Internal or external reference clocks up to 5 MHz can be used to control the FLL
  - 3 bit select for reference divider is provided
- Internal reference clock has 9 trim bits available
- Internal or external reference clocks can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
  - 2 bit select for clock divider is provided
    - Allowable dividers are: 1, 2, 4, 8
- Control signals for a low power oscillator as the external reference clock are provided
  - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL Engaged Internal mode is automatically selected out of reset

10.1.2 Modes of Operation

There are seven modes of operation for the ICS: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

10.1.2.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL which is controlled by the internal reference clock.

10.1.2.2 FLL Engaged External (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from the FLL which is controlled by an external reference clock.

10.1.2.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The ICS supplies a clock derived from the internal reference clock.

10.1.2.4 FLL Bypassed Internal Low Power (FBILP)

In FLL bypassed internal low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock.

10.1.2.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source.

10.1.2.6 FLL Bypassed External Low Power (FBELP)

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source.

10.1.2.7 Stop (STOP)

In stop mode the FLL is disabled and the internal or external reference clocks can be selected to be enabled or disabled. The ICS does not provide an MCU clock source.

10.1.3 Block Diagram

Figure 10-2 is the ICS block diagram.

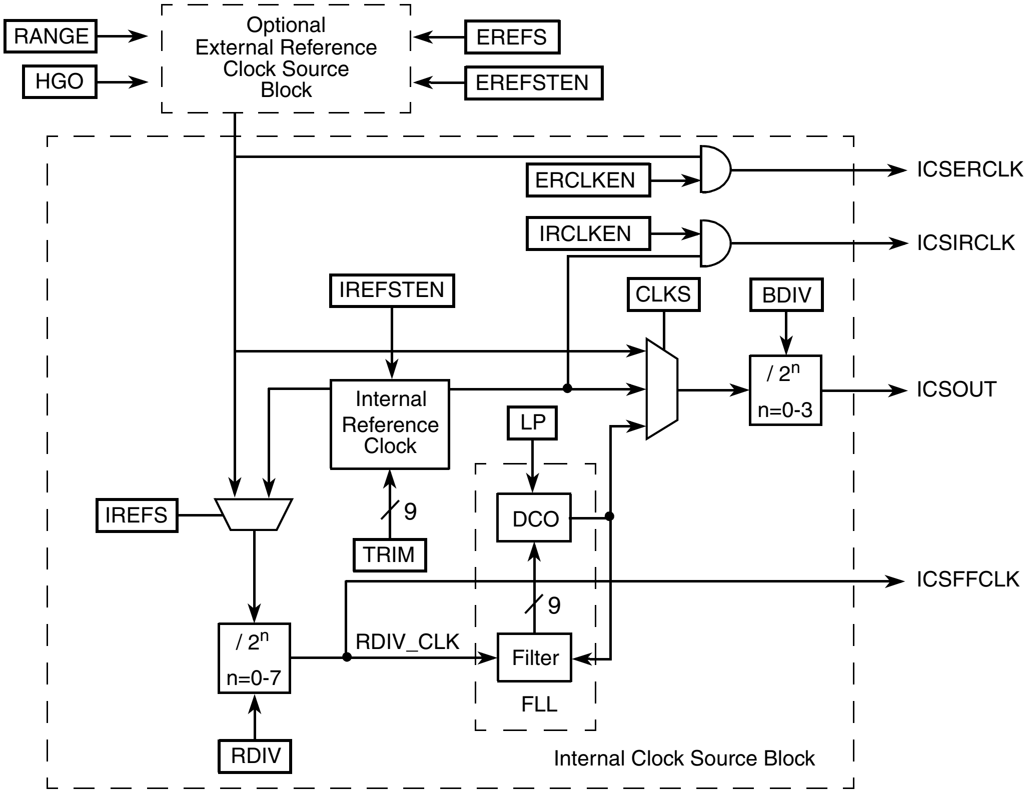


Figure 10-2. Internal Clock Source (ICS) Block Diagram

10.2 External Signal Description

There are no ICS signals that connect off chip.

10.3 Register Definition

Figure 10-1 is a summary of ICS registers.

Table 10-1. ICS Register Summary

Name		7	6	5	4	3	2	1	0
ICSC1	R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
	W								
ICSC2	R	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
	W								
ICSTRM	R	TRIM							
	W								
ICSSC	R	0	0	0	0	CLKST		OSCINIT	FTRIM
	W								

10.3.1 ICS Control Register 1 (ICSC1)

	7	6	5	4	3	2	1	0
R W	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
Reset:	0	0	0	0	0	1	0	0

Figure 10-3. ICS Control Register 1 (ICSC1)

Table 10-2. ICS Control Register 1 Field Descriptions

Field	Description
7:6 CLKS	**Clock Source Select** — Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits. 00 Output of FLL is selected. 01 Internal reference clock is selected. 10 External reference clock is selected. 11 Reserved, defaults to 00.
5:3 RDIV	**Reference Divider** — Selects the amount to divide down the FLL reference clock selected by the IREFS bits. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. 000 Encoding 0 — Divides reference clock by 1 (reset default) 001 Encoding 1 — Divides reference clock by 2 010 Encoding 2 — Divides reference clock by 4 011 Encoding 3 — Divides reference clock by 8 100 Encoding 4 — Divides reference clock by 16 101 Encoding 5 — Divides reference clock by 32 110 Encoding 6 — Divides reference clock by 64 111 Encoding 7 — Divides reference clock by 128
2 IREFS	**Internal Reference Select** — The IREFS bit selects the reference clock source for the FLL. 1 Internal reference clock selected 0 External reference clock selected
1 IRCLKEN	**Internal Reference Clock Enable** — The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK. 1 ICSIRCLK active 0 ICSIRCLK inactive
0 IREFSTEN	**Internal Reference Stop Enable** — The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set or if ICS is in FEI, FBI, or FBILP mode before entering stop 0 Internal reference clock is disabled in stop

10.3.2 ICS Control Register 2 (ICSC2)

	7	6	5	4	3	2	1	0
R W	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
Reset:	0	1	0	0	0	0	0	0

Figure 10-4. ICS Control Register 2 (ICSC2)

Table 10-3. ICS Control Register 2 Field Descriptions

Field	Description
7:6 BDIV	**Bus Frequency Divider** — Selects the amount to divide down the clock source selected by the CLKS bits. This controls the bus frequency. 00 Encoding 0 — Divides selected clock by 1 01 Encoding 1 — Divides selected clock by 2 (reset default) 10 Encoding 2 — Divides selected clock by 4 11 Encoding 3 — Divides selected clock by 8
5 RANGE	**Frequency Range Select** — Selects the frequency range for the external oscillator. 1 High frequency range selected for the external oscillator 0 Low frequency range selected for the external oscillator
4 HGO	**High Gain Oscillator Select** — The HGO bit controls the external oscillator mode of operation. 1 Configure external oscillator for high gain operation 0 Configure external oscillator for low power operation
3 LP	**Low Power Select** — The LP bit controls whether the FLL is disabled in FLL bypassed modes. 1 FLL is disabled in bypass modes 0 FLL is not disabled in bypass mode
2 EREFS	**External Reference Select** — The EREFS bit selects the source for the external reference clock. 1 Oscillator requested 0 External Clock Source requested
1 ERCLKEN	**External Reference Enable** — The ERCLKEN bit enables the external reference clock for use as ICSERCLK. 1 ICSERCLK active 0 ICSERCLK inactive
0 EREFSTEN	**External Reference Stop Enable** — The EREFSTEN bit controls whether or not the external reference clock remains enabled when the ICS enters stop mode. 1 External reference clock stays enabled in stop if ERCLKEN is set or if ICS is in FEE, FBE, or FBELP mode before entering stop 0 External reference clock is disabled in stop

10.3.3 ICS Trim Register (ICSTRM)

	7	6	5	4	3	2	1	0
R W	TRIM							
POR:	1	0	0	0	0	0	0	0
Reset:	U	U	U	U	U	U	U	U

Figure 10-5. ICS Trim Register (ICSTRM)

Table 10-4. ICS Trim Register Field Descriptions

Field	Description
7:0 TRIM	**ICS Trim Setting** — The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' effect are binary weighted (i.e., bit 1 will adjust twice as much as bit 0). Increasing the binary value in TRIM will increase the period, and decreasing the value will decrease the period. An additional fine trim bit is available in ICSSC as the FTRIM bit.

10.3.4 ICS Status and Control (ICSSC)

	7	6	5	4	3	2	1	0
R	0	0	0	0	CLKST		OSCINIT	FTRIM
W								
POR:	0	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	U

Figure 10-6. ICS Status and Control Register (ICSSC)

Table 10-5. ICS Status and Control Register Field Descriptions

Field	Description
7:2	Reserved, must be cleared.
1	**OSC Initialization** — If the external reference clock is selected by ERCLKEN or by the ICS being in FEE, FBE, or FBELP mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either ERCLKEN or EREFS are cleared.
0	**ICS Fine Trim** — The FTRIM bit controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible.
3-2 CLKST	Clock Mode Status — The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Output of FLL is selected. 01 FLL Bypassed, Internal reference clock is selected. 10 FLL Bypassed, External reference clock is selected. 11 Reserved.

10.4 Functional Description

10.4.1 Operational Modes

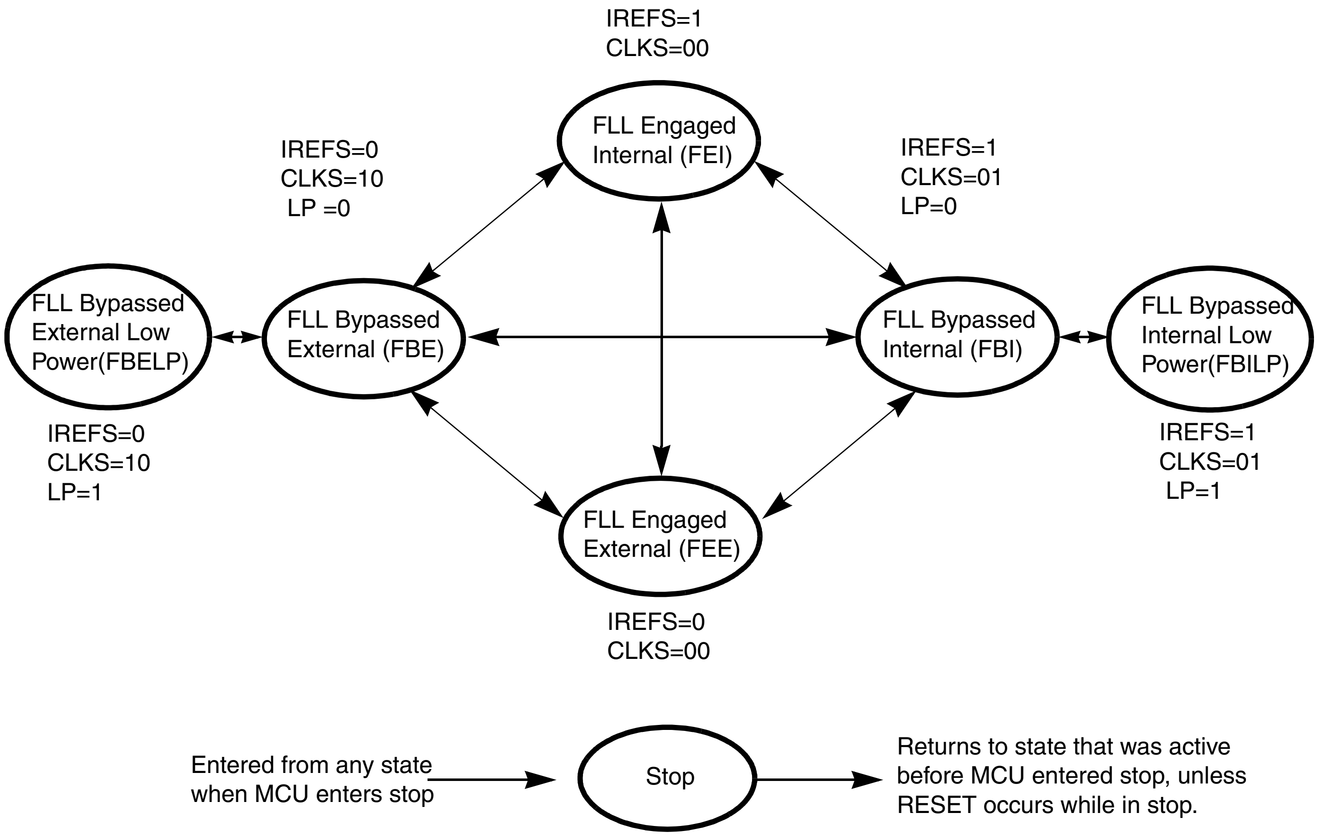


Figure 10-7. Clock Switching Modes

The seven states of the ICS are shown as a state diagram and are described below. The arrows indicate the allowed movements between the states.

10.4.1.1 FLL Engaged Internal (FEI)

FLL engaged internal (FEI) is the default mode of operation and is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL engaged internal mode, the ICSOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL loop will lock the frequency to 512 times the filter frequency, as selected by the RDIV bits. and the internal reference clock is enabled.

10.4.1.2 FLL Engaged External (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 0
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz

In FLL engaged external mode, the ICSOUT clock is derived from the FLL clock which is controlled by the external reference clock.The FLL loop will lock the frequency to 512 times the filter frequency, as selected by the RDIV bits. and the external reference clock is enabled.

10.4.1.3 FLL Bypassed Internal (FBI)

The FLL bypassed internal (FBI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1.
- LP bit is written to 0

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL loop will lock the FLL frequency to 512 times the Filter frequency, as selected by the RDIV bits. and the internal reference clock is enabled.

10.4.1.4 FLL Bypassed Internal Low Power (FBILP)

The FLL bypassed internal low power (FBILP) mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1.
- LP bit is written to 1

In FLL bypassed internal low power mode, the ICSOUT clock is derived from the internal reference clock and the FLL is disabled. and the internal reference clock is enabled.

10.4.1.5 FLL Bypassed External (FBE)

The FLL bypassed external (FBE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- LP bit is written to 0.

In FLL bypassed external mode, the ICSOUT clock is derived from the external reference clock. The FLL clock is controlled by the external reference clock, and the FLL loop will lock the FLL frequency to 512 times the filter frequency, as selected by the RDIV bits, and the external reference clock is enabled.

10.4.1.6 FLL Bypassed External Low Power (FBELP)

The FLL bypassed external low power (FBELP) mode is entered when all the following conditions occur:

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- LP bit is written to 1.

In FLL bypassed external low power mode, the ICSOUT clock is derived from the external reference clock and the FLL is disabled. The external reference clock is enabled.

10.4.1.7 Stop

Stop mode is entered whenever the MCU enters a STOP state. In this mode, all ICS clock signals are static except in the following cases:

ICSIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN bit is written to 1
- IREFSTEN bit is written to 1

ICSERCLK will be active in stop mode when all the following conditions occur:

- ERCLKEN bit is written to 1
- EREFSTEN bit is written to 1

10.4.2 Mode Switching

When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes the IREFS bit can be changed at anytime, but the RDIV bits must be changed simultaneously so that the resulting frequency stays in the range of 31.25 kHz to 39.0625 kHz. After a change in the IREFS value the FLL will begin locking again after a few full cycles of the resulting divided reference frequency.

The CLKS bits can also be changed at anytime, but the RDIV bits must be changed simultaneously so that the resulting frequency stays in the range of 31.25 kHz to 39.0625 kHz. The actual switch to the newly selected clock will not occur until after a few full cycles of the new clock. If the newly selected clock is not available, the previous clock will remain selected.

10.4.3 Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency will occur immediately.

10.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL to be disabled and thus conserve power when it is not being used. However, in some applications it may be desirable to enable the FLL and allow it to lock for maximum accuracy before switching to an FLL engaged mode. Do this by writing the LP bit to 0.

10.4.5 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal will be presented as ICSIRCLK, which can be used as an additional clock source. The ICSIRCLK frequency can be re-targeted by trimming the period of the internal reference clock. This can be done by writing a new value to the TRIM bits in the ICSTRM register. Writing a larger value will slow down the ICSIRCLK frequency, and writing a smaller value to the ICSTRM register will speed up the ICSIRCLK frequency. The TRIM bits will effect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low power (FBILP) mode. The TRIM and FTRIM value will not be affected by a reset.

Until ICSIRCLK is trimmed, programming low reference divider (RDIV) factors may result in ICSOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the Device Overview chapter).

If IREFSTEN is set and the IRCLKEN bit is written to 1, the ICSIRCLK will keep running during stop mode in order to provide a fast recovery upon exiting stop.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value can be copied to the ICSTRM register during reset initialization. The factory trim value does not include the FTRIM bit. For finer precision, the user can trim the internal oscillator in the application and set the FTRIM bit accordingly.

10.4.6 Optional External Reference Clock

The ICS module can support an external reference clock with frequencies between 31.25 kHz to 5 MHz in all modes. When the ERCLKEN is set, the external reference clock signal will be presented as ICSERCLK, which can be used as an additional clock source. When IREFS = 1, the external reference clock will not be used by the FLL and will only be used as ICSERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support (see the Device Overview chapter).

If EREFSTEN is set and the ERCLKEN bit is written to 1, the ICSERCLK will keep running during stop mode in order to provide a fast recovery upon exiting stop.

10.4.7 Fixed Frequency Clock

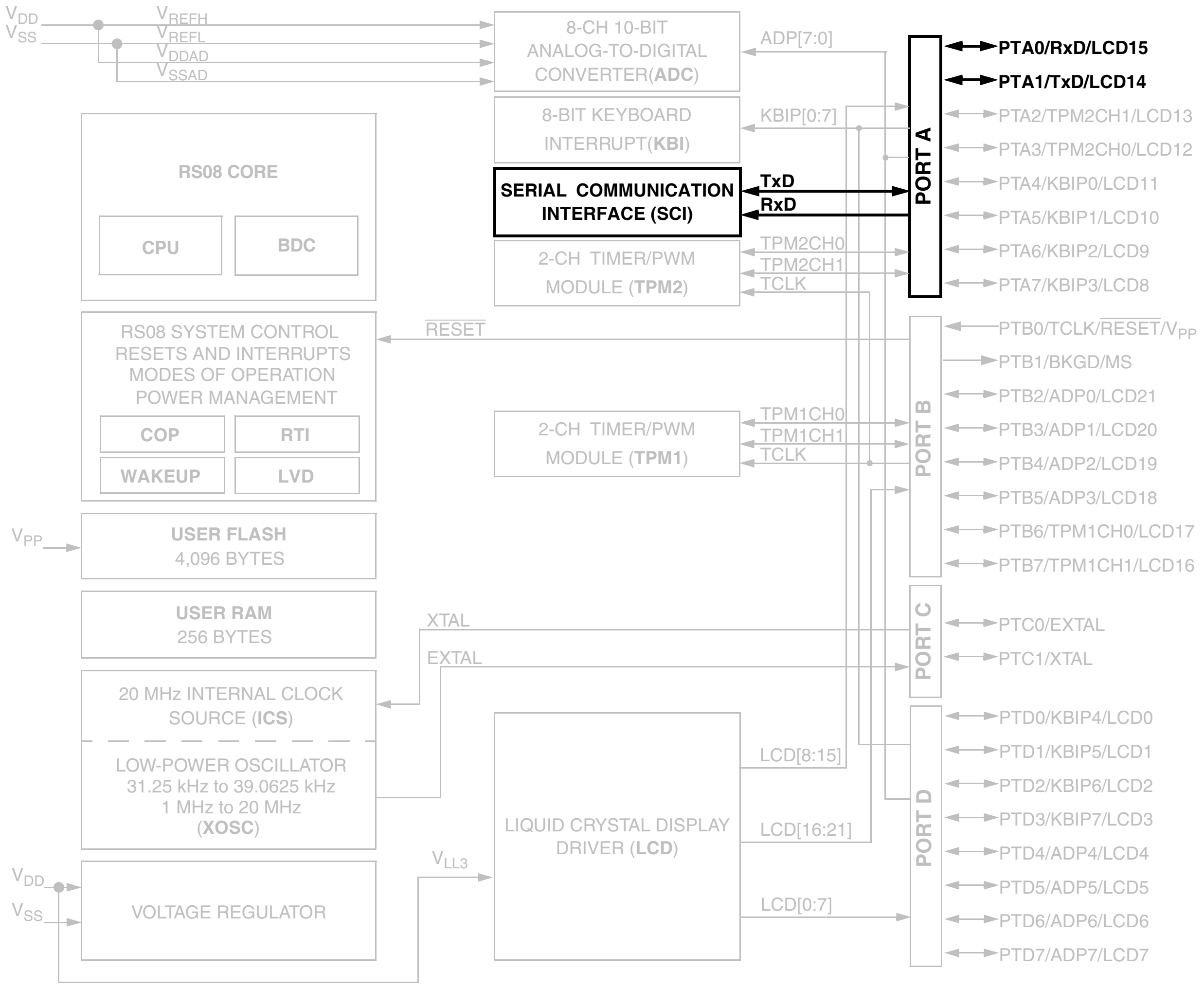
The ICS presents the divided FLL reference clock as ICSFFCLK for use as an additional clock source for peripheral modules. The ICS provides an output signal (ICSFFE) which indicates when the ICS is providing ICSOUT frequencies four times or greater than the divided FLL reference clock (ICSFFCLK). In FLL Engaged mode (FEI and FEE) this is always true and ICSFFE is always high. In ICS Bypass modes, ICSFFE will get asserted for the following combinations of BDIV and RDIV values:

- BDIV=00 (divide by 1), RDIV ≥ 010
- BDIV=01 (divide by 2), RDIV ≥ 011
- BDIV=10 (divide by 4), RDIV ≥ 100
- BDIV=11 (divide by 8), RDIV ≥ 101

Chapter 11
Serial Communications Interface (S08SCIV4)

11.1 Introduction

Figure 11-1 shows the MC9RS08LE4 block diagram, highlighting the SCI block and pins.



NOTES:
1. PTB0/TCLK/RESET/V_{PP} is an input-only pin when used as port pin
2. PTB1/BKGD/MS is an output-only pin

Figure 11-1. MC9RS08LE4 Series Block Diagram Highlighting SCI Block and Pins

11.1.1 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

11.1.2 Modes of Operation

See Section 11.3, “Functional Description,” For details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

11.1.3 Block Diagram

Figure 11-2 shows the transmitter portion of the SCI.

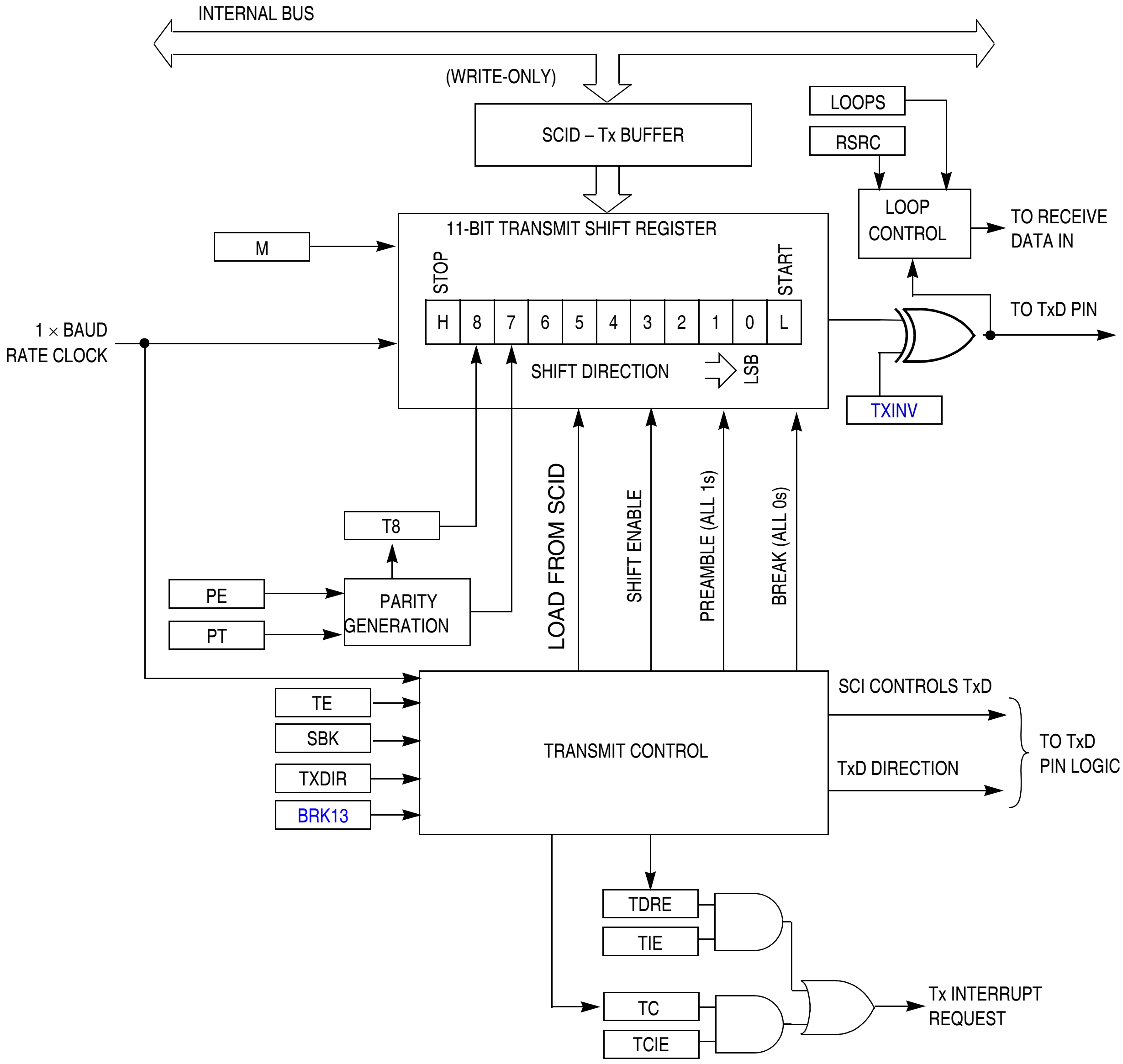


Figure 11-2. SCI Transmitter Block Diagram

Figure 11-3 shows the receiver portion of the SCI.

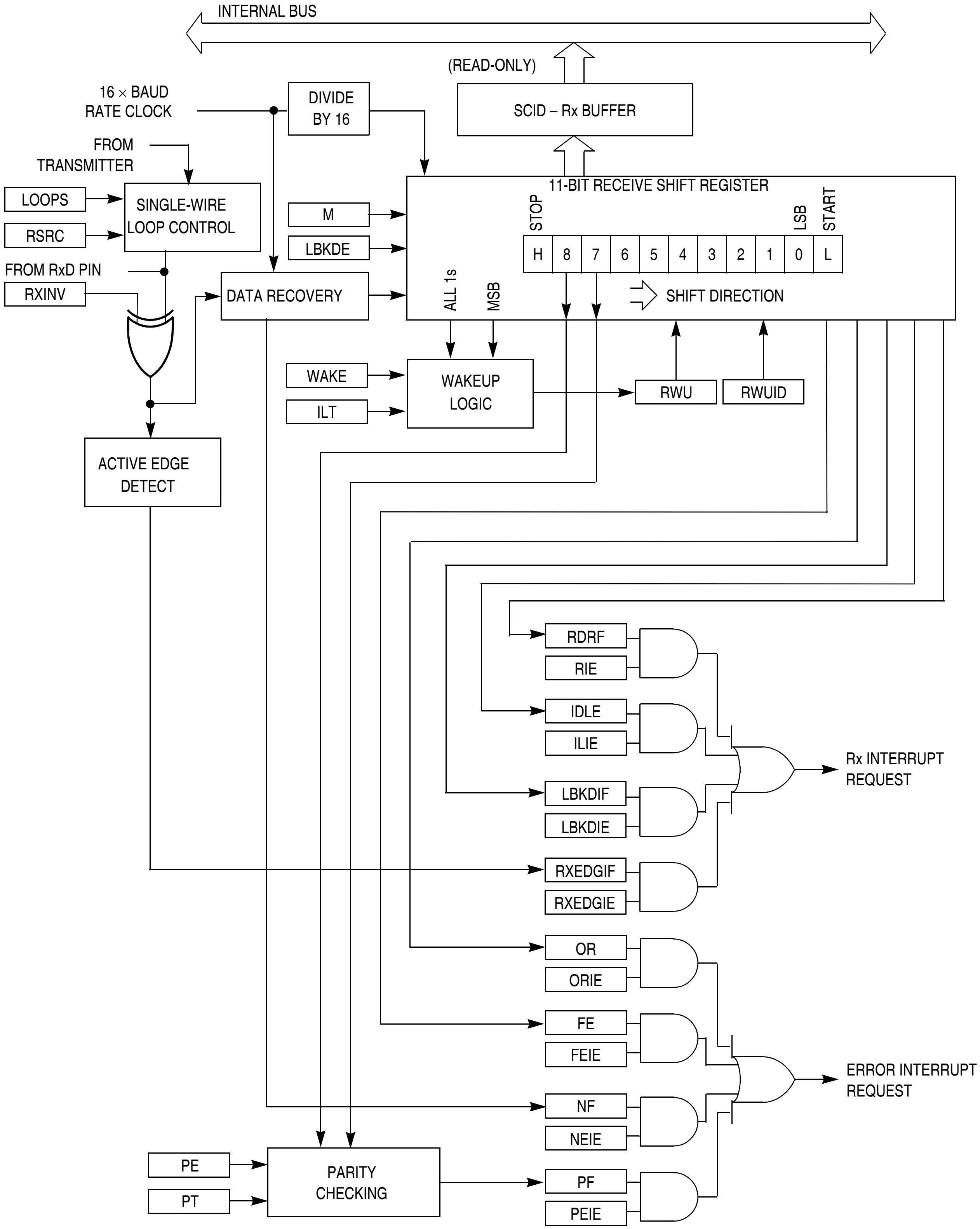


Figure 11-3. SCI Receiver Block Diagram

11.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

11.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIBDH to buffer the high half of the new value and then write to SCIBDL. The working value in SCIBDH does not change until SCIBDL is written.

SCIBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIC2 are written to 1).

	7	6	5	4	3	2	1	0
R	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
W			▨					
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-4. SCI Baud Rate Register (SCIBDH)

Table 11-1. SCIBDH Field Descriptions

Field	Description
7 LBKDIE	**LIN Break Detect Interrupt Enable (for LBKDIF)** 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	**RxD Input Active Edge Interrupt Enable (for RXEDGIF)** 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4:0 SBR[12:8]	**Baud Rate Modulo Divisor** — The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 11-2.

	7	6	5	4	3	2	1	0
R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
W								
Reset	0	0	0	0	0	1	0	0

Figure 11-5. SCI Baud Rate Register (SCIBDL)

Table 11-2. SCIBDL Field Descriptions

Field	Description
7:0 SBR[7:0]	**Baud Rate Modulo Divisor** — These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 11-1.

11.2.2 SCI Control Register 1 (SCIC1)

This read/write register is used to control various optional features of the SCI system.

	7	6	5	4	3	2	1	0
R W	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

Figure 11-6. SCI Control Register 1 (SCIC1)

Table 11-3. SCIC1 Field Descriptions

Field	Description
7 LOOPS	**Loop Mode Select** — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by SCI.
6 SCISWAI	**SCI Stops in Wait Mode** 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	**Receiver Source Select** — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	**9-Bit or 8-Bit Mode Select** 0 Normal — start + 8 data bits (LSB first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (LSB first) + 9th data bit + stop.
3 WAKE	**Receiver Wakeup Method Select** — Refer to Section 11.3.3.2, “Receiver Wakeup Operation” for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	**Idle Line Type Select** — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to Section 11.3.3.2.1, “Idle-Line Wakeup” for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.

Table 11-3. SCIC1 Field Descriptions (continued)

Field	Description
1 PE	**Parity Enable** — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	**Parity Type** — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

11.2.3 SCI Control Register 2 (SCIC2)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

Figure 11-7. SCI Control Register 2 (SCIC2)

Table 11-4. SCIC2 Field Descriptions

Field	Description
7 TIE	**Transmit Interrupt Enable (for TDRE)** 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	**Transmission Complete Interrupt Enable (for TC)** 0 Hardware interrupts from TC disabled (use polling). 1 Hardware interrupt requested when TC flag is 1.
5 RIE	**Receiver Interrupt Enable (for RDRF)** 0 Hardware interrupts from RDRF disabled (use polling). 1 Hardware interrupt requested when RDRF flag is 1.
4 ILIE	**Idle Line Interrupt Enable (for IDLE)** 0 Hardware interrupts from IDLE disabled (use polling). 1 Hardware interrupt requested when IDLE flag is 1.
3 TE	**Transmitter Enable** 0 Transmitter off. 1 Transmitter on. TE must be 1 in order to use the SCI transmitter. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system. When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin). TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to Section 11.3.2.1, “Send Break and Queued Idle” for more details. When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.

Table 11-4. SCIC2 Field Descriptions (continued)

Field	Description
2 RE	**Receiver Enable** — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1 the RxD pin reverts to being a general-purpose I/O pin even if RE = 1. 0 Receiver off. 1 Receiver on.
1 RWU	**Receiver Wakeup Control** — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 11.3.3.2, “Receiver Wakeup Operation” for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	**Send Break** — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 11.3.2.1, “Send Break and Queued Idle” for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

11.2.4 SCI Status Register 1 (SCIS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) are used to clear these status flags.

	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-8. SCI Status Register 1 (SCIS1)

Table 11-5. SCIS1 Field Descriptions

Field	Description
7 TDRE	**Transmit Data Register Empty Flag** — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIS1 with TDRE = 1 and then write to the SCI data register (SCID). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	**Transmission Complete Flag** — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete). TC is cleared automatically by reading SCIS1 with TC = 1 and then doing one of the following three things: • Write to the SCI data register (SCID) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to SBK in SCIC2
5 RDRF	**Receive Data Register Full Flag** — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCID). To clear RDRF, read SCIS1 with RDRF = 1 and then read the SCI data register (SCID). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	**Idle Line Flag** — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, read SCIS1 with IDLE = 1 and then read the SCI data register (SCID). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	**Receiver Overrun Flag** — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCID yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCID. To clear OR, read SCIS1 with OR = 1 and then read the SCI data register (SCID). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	**Noise Flag** — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIS1 and then read the SCI data register (SCID). 0 No noise detected. 1 Noise detected in the received character in SCID.

Table 11-5. SCIS1 Field Descriptions (continued)

Field	Description
1 FE	**Framing Error Flag** — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIS1 with FE = 1 and then read the SCI data register (SCID). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	**Parity Error Flag** — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIS1 and then read the SCI data register (SCID). 0 No parity error. 1 Parity error.

11.2.5 SCI Status Register 2 (SCIS2)

This register has one read-only status flag.

	7	6	5	4	3	2	1	0
R	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
W			▨					▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-9. SCI Status Register 2 (SCIS2)

Table 11-6. SCIS2 Field Descriptions

Field	Description
7 LBKDIF	**LIN Break Detect Interrupt Flag** — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	**RxD Pin Active Edge Interrupt Flag** — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV[1]	**Receive Data Inversion** — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	**Receive Wake Up Idle Detect**— RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	**Break Character Generation Length** — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

Table 11-6. SCIS2 Field Descriptions (continued)

Field	Description
1 LBKDE	**LIN Break Detection Enable**— LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	**Receiver Active Flag** — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

[1] Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold by one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave which is running 14% faster than the master. This would trigger normal break detection circuitry which is designed to detect a 10 bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

11.2.6 SCI Control Register 3 (SCIC3)

	7	6	5	4	3	2	1	0
R	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
W	▨							
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 11-10. SCI Control Register 3 (SCIC3)

Table 11-7. SCIC3 Field Descriptions

Field	Description
7 R8	**Ninth Data Bit for Receiver** — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCID register. When reading 9-bit data, read R8 before reading SCID because reading SCID completes automatic flag clearing sequences which could allow R8 and SCID to be overwritten with new data.
6 T8	**Ninth Data Bit for Transmitter** — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCID register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCID is written so T8 should be written (if it needs to change from its previous value) before SCID is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCID is written.
5 TXDIR	**TxD Pin Direction in Single-Wire Mode** — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.

Table 11-7. SCIC3 Field Descriptions (continued)

Field	Description
4 TXINV[1]	**Transmit Data Inversion** — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	**Overrun Interrupt Enable** — This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR = 1.
2 NEIE	**Noise Error Interrupt Enable** — This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF = 1.
1 FEIE	**Framing Error Interrupt Enable** — This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE = 1.
0 PEIE	**Parity Error Interrupt Enable** — This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF = 1.

[1] Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

11.2.7 SCI Data Register (SCID)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figure 11-11. SCI Data Register (SCID)

11.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

11.3.1 Baud Rate Generation

As shown in Figure 11-12, the clock source for the SCI baud rate generator is the bus-rate clock.

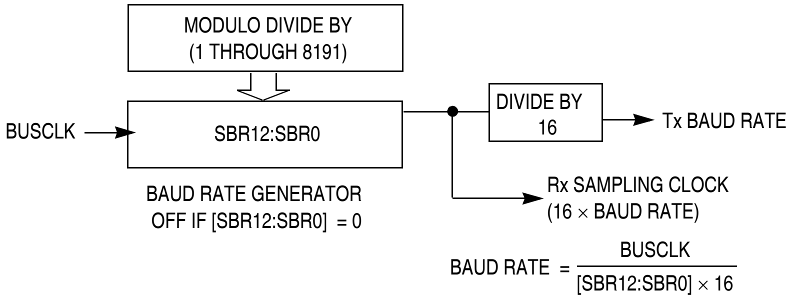


Figure 11-12. SCI Baud Rate Generation

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ±4.5 percent for 8-bit data format and about ±4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

11.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in Figure 11-2.

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in SCIC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCID).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume M = 0, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCID.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

11.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

Table 11-8. Break Character Length

BRK13	M	Break Character Length
0	0	10 bit times
0	1	11 bit times
1	0	13 bit times
1	1	14 bit times

11.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 11-3) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 11.3.5.1, “8- and 9-Bit Data Modes.” For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to Section 11.3.4, "Interrupts and Status Flags" for more details about flag clearing.

11.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

11.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

11.3.3.2.1 Idle-Line Wakeup

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message which will set the RDRF flag and generate an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

11.3.3.2.2 Address-Mark Wakeup

When WAKE = 1, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit in M = 0 mode and ninth bit in M = 1 mode).

Address-mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case the character with the MSB set is received even though the receiver was sleeping during most of this character time.

11.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCID. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1.

Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared by reading SCIS1 while RDRF = 1 and then reading SCID.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIS1 while IDLE = 1 and then reading SCID. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled (RE = 1).

11.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

11.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIC3. For the receiver, the ninth bit is held in R8 in SCIC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCID.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCID to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

11.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit is still active in stop3 mode, but not in stop2. . An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

11.3.5.3 Loop Mode

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

11.3.5.4 Single-Wire Operation

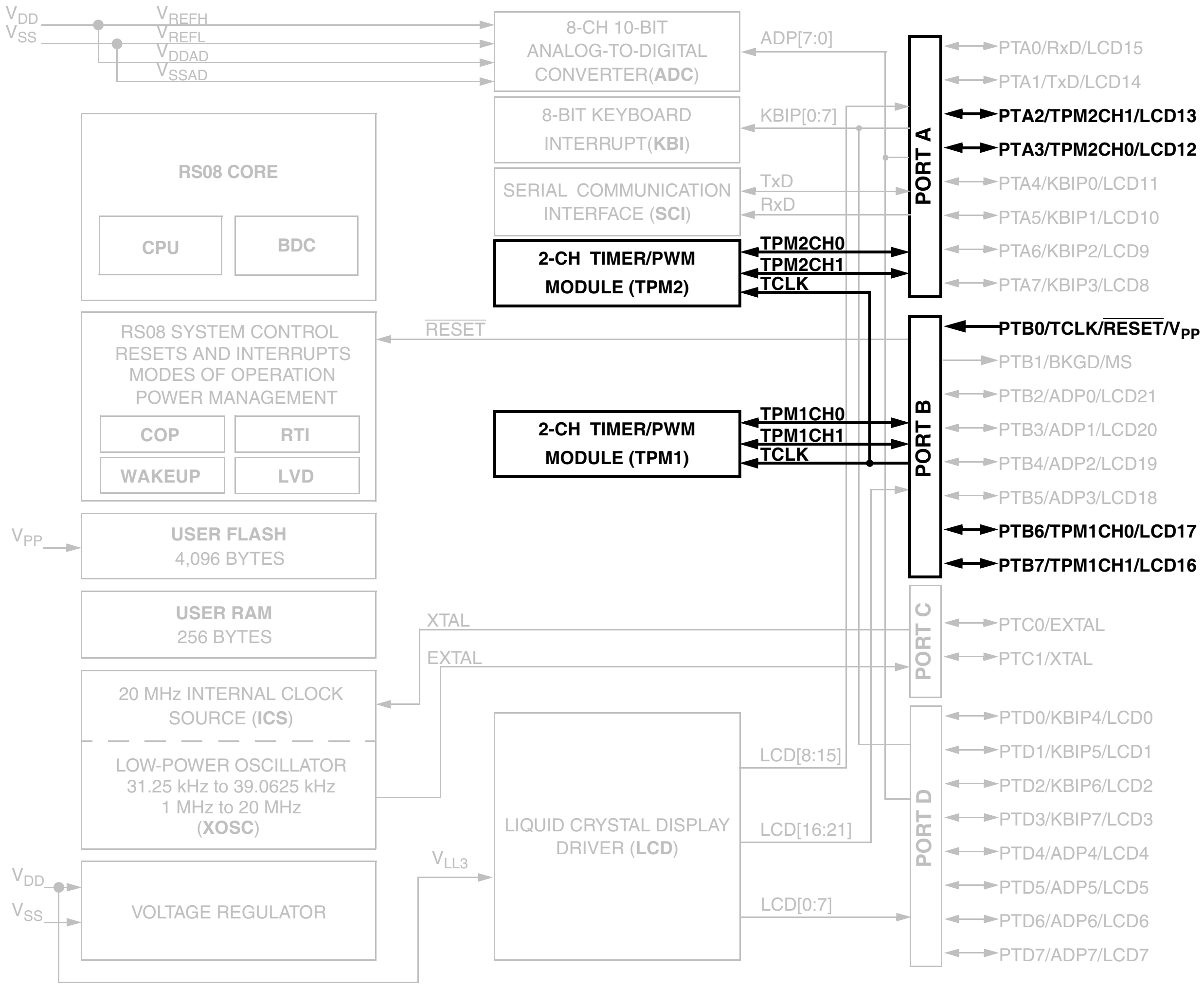
When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIC3 controls the direction of serial data on the TxD pin. When TXDIR = 0, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

Chapter 12
16-Bit Timer/PWM (S08TPMV2)

12.1 Introduction

Figure 12-1 shows the MC9RS08LE4 block diagram highlighting the TPM block and pins.



NOTES:
1. PTB0/TCLK/RESET/V_{PP} is an input-only pin when used as port pin
2. PTB1/BKGD/MS is an output-only pin

Figure 12-1. MC9RS08LE4 Series Block Diagram Highlighting TPM Block and Pins

12.1.1 Features

The TPM includes these distinctive features:

- One to eight channels:
  - Each channel may be input capture, output compare, or edge-aligned PWM
  - Rising-Edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
  - Selectable polarity on PWM outputs
- Module may be configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as prescaled bus clock, fixed system clock, or an external clock pin
  - Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128
  - Fixed system clock source are synchronized to the bus clock by an on-chip synchronization circuit
  - External clock pin may be shared with any timer channel pin or a separated input pin
- 16-bit free-running or modulo up/down count operation
- Timer system enable
- One interrupt per channel plus terminal count interrupt

12.1.2 Modes of Operation

In general, TPM channels may be independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the microcontroller is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all system clocks, including the main oscillator, are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. Provided the TPM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, the user can save power by disabling TPM functions before entering wait mode.

- Input capture mode

  When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge which triggers the input capture.
- Output compare mode

  When the value in the timer counter register matches the channel value register, an interrupt flag bit is set, and a selected output action is forced on the associated MCU pin. The output compare action may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).

- Edge-aligned PWM mode

  The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. The user may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.
- Center-aligned PWM mode

  Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

12.1.3 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMxCHn (timer channel n) where n is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 12-2 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMxCNT counter resets the counter, regardless of the data value written.

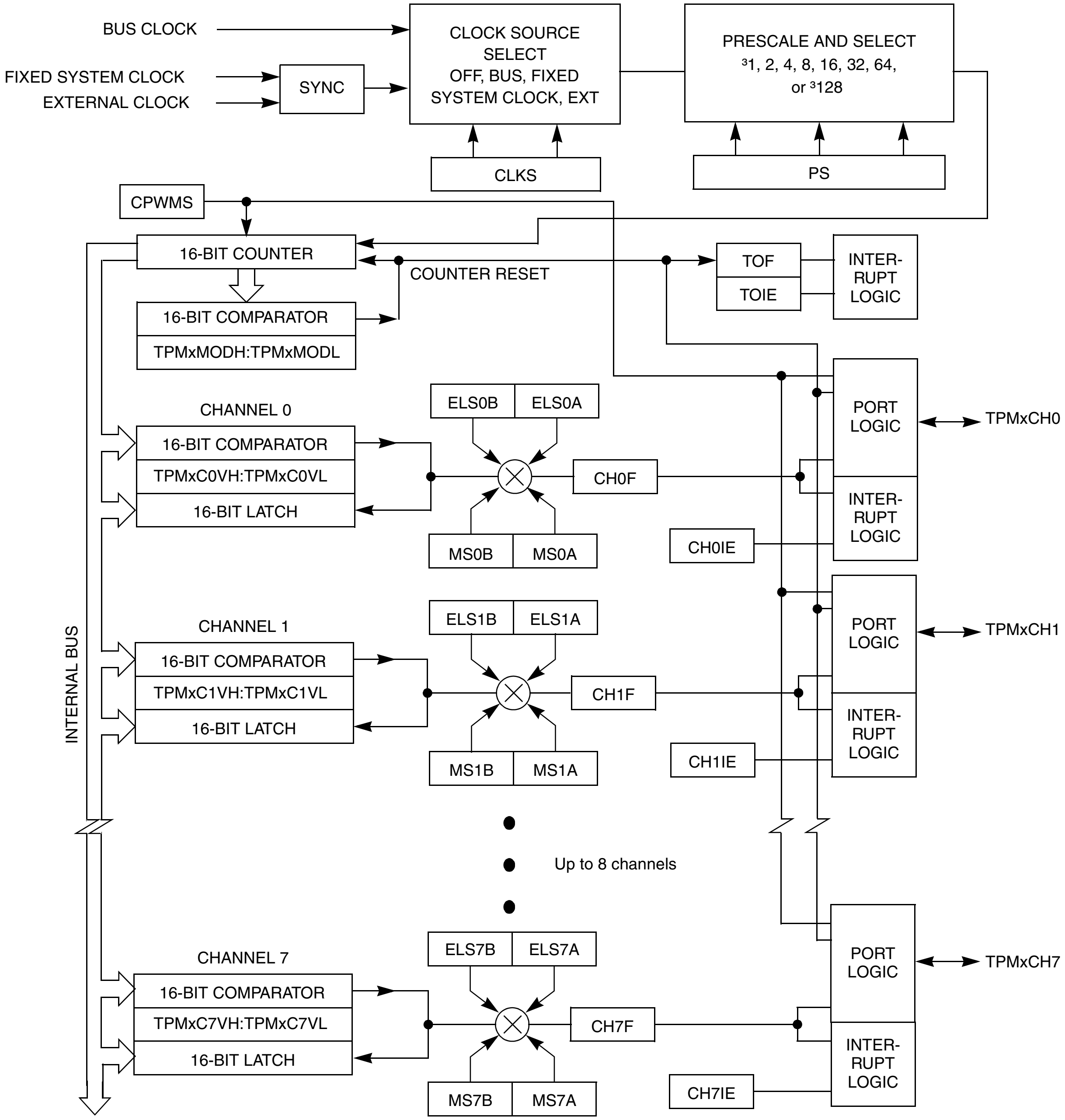


Figure 12-2. TPM Block Diagram

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

12.2 Signal Description

Table 12-1 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

Table 12-1. Signal Properties

Name	Function
EXTCLK[1]	External clock source which may be selected to drive the TPM counter.
TPMxCHn[2]	I/O pin associated with TPM channel n

[1] When preset, this signal can share any channel pin; however depending upon full-chip implementation, this signal could be connected to a separate external pin.

[2] n=channel number (1 to 8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices which can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

12.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 12-1 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Since I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

12.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

12.2.1.2 TPMxCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKS = 00) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMxCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMxCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMxCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

When a channel is configured for edge-aligned PWM (CPWMS=0, MSnB=1 and ELSnB:ELSnA not = 0:0), the data direction is overridden, the TPMxCHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB:ELSnA=1:0, the TPMxCHn pin is forced high at the start of each new period (TPMxCNT=0x0000), and the pin is forced low when the channel value register matches the timer counter. When ELSnA=1, the TPMxCHn pin is forced low at the start of each new period (TPMxCNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

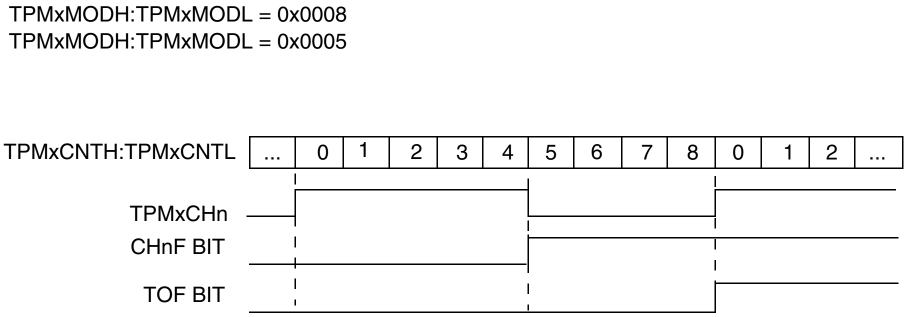


Figure 12-3. High-True Pulse of an Edge-Aligned PWM

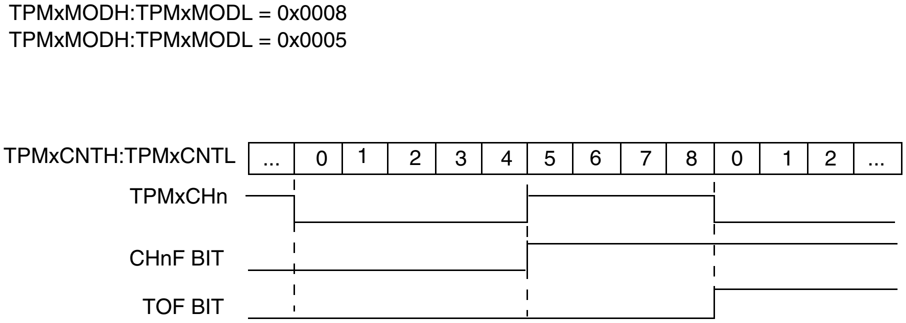


Figure 12-4. Low-True Pulse of an Edge-Aligned PWM

When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPMxCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMxCHn output. If ELSnB:ELSnA=1:0, the corresponding TPMxCHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPMxCHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPMxCHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPMxCHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

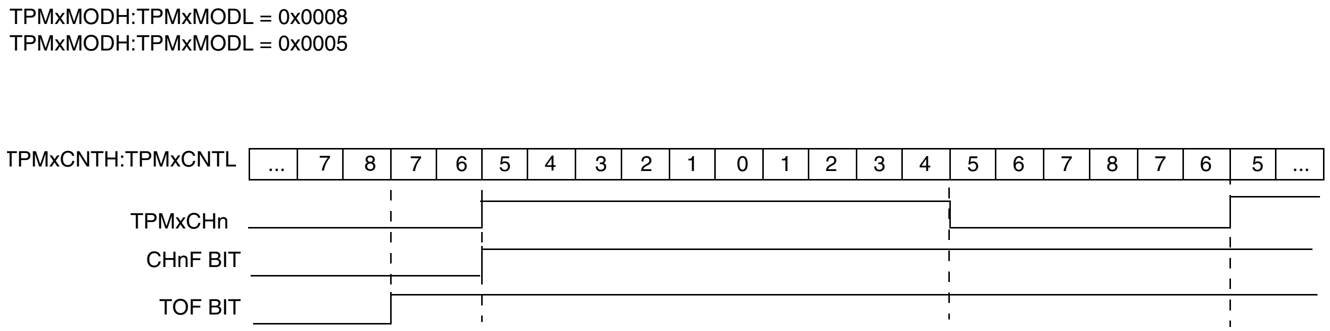


Figure 12-5. High-True Pulse of a Center-Aligned PWM

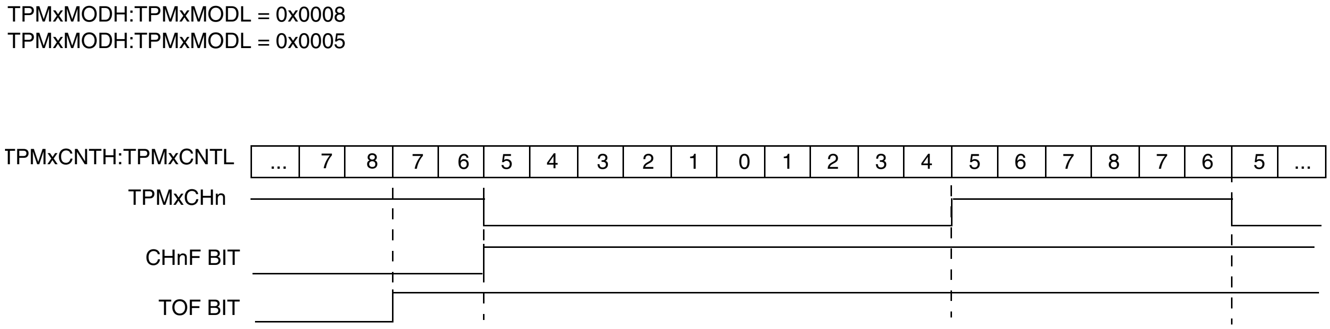


Figure 12-6. Low-True Pulse of a Center-Aligned PWM

12.3 Register Definition

This section consists of register descriptions in address order. A typical MCU system may contain multiple TPMs, and each TPM may have one to eight channels, so register names include placeholder characters to identify which TPM and which channel is being referenced. For example, TPMxCnSC refers to timer (TPM) x, channel n. TPM1C2SC would be the status and control register for channel 2 of timer 1.

12.3.1 TPM Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits used to configure the interrupt enable, TPM configuration, clock source, and prescale factor. These controls relate to all channels within this timer module.

	7	6	5	4	3	2	1	0
R	TOF	TOIE	CPWMS	CLKS		PS		
W	0							
Reset	0	0	0	0	0	0	0	0

Figure 12-7. TPM Status and Control Register (TPMxSC)

Table 12-2. TPMxSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled
5 CPWMS	Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.

Table 12-2. TPMxSC Field Descriptions (continued)

Field	Description
4–3 CLKS	Clock source selects. As shown in Table 12-3, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based system clock. When there is no PLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock.
2–0 PS	Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in Table 12-4. This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits.

Table 12-3. TPM-Clock-Source Selection

CLKS	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

Table 12-4. Prescale Factor Selection

PS	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

12.3.2 TPM-Counter Registers (TPMxCNTH:TPMxCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMxSC).

Reset clears the TPM counter registers. Writing any value to TPMxCNTH or TPMxCNTL also clears the TPM counter (TPMxCNTH:TPMxCNTL) and resets the coherency mechanism, regardless of the data involved in the write.

	7	6	5	4	3	2	1	0
R	Bit 15	14	13	12	11	10	9	Bit 8
W	Any write to TPMxCNTH clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 12-8. TPM Counter Register High (TPMxCNTH)

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	Any write to TPMxCNTL clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 12-9. TPM Counter Register Low (TPMxCNTL)

When BDM is active, the timer counter is frozen (this is the value that will be read by user); the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution.

12.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock, and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000 which results in a free running timer counter (modulo disabled).

Writing to either byte (TPMxMODH or TPMxMODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), then the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), then the registers are updated after both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMxSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 12-10. TPM Counter Modulo Register High (TPMxMODH)

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

Figure 12-11. TPM Counter Modulo Register Low (TPMxMODL)

Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

12.3.4 TPM Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W	0						▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 12-12. TPM Channel n Status and Control Register (TPMxCnSC)

Table 12-5. TPMxCnSC Field Descriptions

Field	Description
7 CHnF	Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF will not be set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while CHnF is set and then writing a logic 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF. Reset clears the CHnF bit. Writing a logic 1 to CHnF has no effect. 0 No input capture or output compare event occurred on channel n 1 Input capture or output compare event on channel n
6 CHnIE	Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears CHnIE. 0 Channel n interrupt requests disabled (use for software polling) 1 Channel n interrupt requests enabled
5 MSnB	Mode select B for TPM channel n. When CPWMS=0, MSnB=1 configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 12-6.
4 MSnA	Mode select A for TPM channel n. When CPWMS=0 and MSnB=0, MSnA configures TPM channel n for input-capture mode or output compare mode. Refer to Table 12-6 for a summary of channel mode and setup controls. **Note:** If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 ELSnB ELSnA	Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 12-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output. Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general purpose I/O pin not related to any timer functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin.

Table 12-6. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for TPM - revert to general purpose I/O or other peripheral control	

Table 12-6. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	01	Output compare	Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

12.3.5 TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel registers are cleared by reset.

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 12-13. TPM Channel Value Register High (TPMxCnVH)

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

Figure 12-14. TPM Channel Value Register Low (TPMxCnVL)

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written (whether BDM mode is active or not). Any write to the channel registers will be ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMxCnVH and TPMxCnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKS bits and the selected mode, so:

- If (CLKS[1:0] = 00), then the registers are updated when the second byte is written.
- If (CLKS[1:0] not = 00 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKS[1:0] not = 00 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPMxCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order which is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM & output compare operation once normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

12.4 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

12.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

12.4.1.1 Counter Clock Source

The 2-bit field, CLKS, in the timer status and control register (TPMxSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See Table 12-3. After any MCU reset, CLKS=00 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKS field) does not affect the values in the counter or other timer registers.

Table 12-7. TPM Clock Source Selection

CLKS	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disabled)
01	Bus rate clock
10	Fixed system clock
11	External source

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock that is used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL or the PLL is not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL is present and engaged, a synchronizer is required between the crystal divided-by two clock source and the timer counter so counter transitions will be properly aligned to bus-clock transitions. A synchronizer will be used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin should not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. (It is the user's responsibility to avoid such settings.) The TPM channel could still be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

12.4.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE=0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE=1) where a static hardware interrupt is generated whenever the TOF flag is equal to one.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

12.4.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS=1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

12.4.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either half of TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

12.4.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

12.4.2.1 Input Capture Mode

With the input-capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input-capture channel, the TPM latches the contents of the TPM counter into the channel-value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An input capture event sets a flag bit (CHnF) which may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

12.4.2.2 Output Compare Mode

With the output-compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel-value registers of an output-compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

12.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMxMODH:TPMxMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 12-15). The time between the modulus overflow and the output compare is the pulse width. If ELSnA=0, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA=1, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.

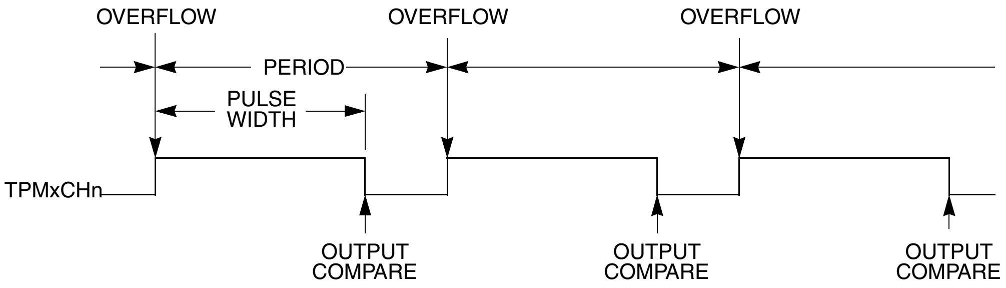


Figure 12-15. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxCnVH and TPMxCnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

12.4.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

pulse width = 2 x (TPMxCnVH:TPMxCnVL)

period = 2 x (TPMxMODH:TPMxMODL); TPMxMODH:TPMxMODL=0x0001-0x7FFF

If the channel-value register TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the (non-zero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

TPMxMODH:TPMxMODL=0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS=0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS=1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 12-16). If ELSnA=0, a compare occurred while counting up forces the CPWM output signal low and a compare occurred while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.

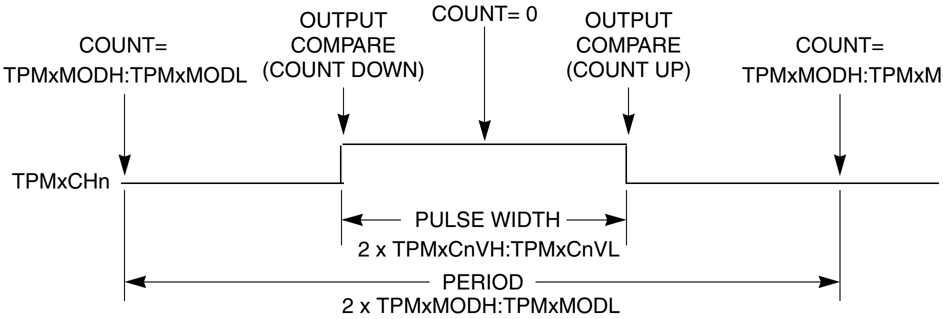


Figure 12-16. CPWM Period and Pulse Width (ELSnA=0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS=1.

The TPM may be used in an 8-bit MCU. The settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxMODH, TPMxMODL, TPMxCnVH, and TPMxCnVL, actually write to buffer registers.

In center-aligned PWM mode, the TPMxCnVH:L registers are updated with the value of their write buffer according to the value of CLKS bits, so:

- If (CLKS[1:0] = 00), the registers are updated when the second byte is written
- If (CLKS[1:0] not = 00), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

When TPMxCNTH:TPMxCNTL=TPMxMODH:TPMxMODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

Writing to TPMxSC cancels any values written to TPMxMODH and/or TPMxMODL and resets the coherency mechanism for the modulo registers. Writing to TPMxCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMxCnVH:TPMxCnVL.

12.5 Reset Overview

12.5.1 General

The TPM is reset whenever any MCU reset occurs.

12.5.2 Description of Reset Operation

Reset clears the TPMxSC register which disables clocks to the TPM and disables timer overflow interrupts (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared which configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (so all MCU pins related to the TPM revert to general purpose I/O pins).

12.6 Interrupts

12.6.1 General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel's mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in Table 12-8 which shows the interrupt name, the name of any local enable that can block the interrupt request from leaving the TPM and getting recognized by the separate interrupt processing logic.

Table 12-8. Interrupt Summary

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the timer counter reaches its terminal count (at transition to next count value which is usually 0x0000)
CHnF	CHnIE	Channel event	An input capture or output compare event took place on channel n

The TPM module will provide a high-true interrupt signal. Vectors and priorities are determined at chip integration time in the interrupt module so refer to the user’s guide for the interrupt module or to the chip’s complete documentation for details.

12.6.2 Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel-input capture, or output-compare events. This flag may be read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will generate whenever the associated interrupt flag equals one. The user’s software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt-service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set (1) followed by a write of zero (0) to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

12.6.2.1 Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

12.6.2.1.1 Normal Case

Normally TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS=0), TOF gets set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

12.6.2.1.2 Center-Aligned PWM Case

When CPWMS=1, TOF gets set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case the TOF corresponds to the end of a PWM period.

12.6.2.2 Channel Event Interrupt Description

The meaning of channel interrupts depends on the channel's current mode (input-capture, output-compare, edge-aligned PWM, or center-aligned PWM).

12.6.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select no edge (off), rising edges, falling edges or any edge as the edge which triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in Section 12.6.2, "Description of Interrupt Operation."

12.6.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described Section 12.6.2, "Description of Interrupt Operation."

12.6.2.2.3 PWM End-of-Duty-Cycle Events

For channels configured for PWM operation there are two possibilities. When the channel is configured for edge-aligned PWM, the channel flag gets set when the timer counter matches the channel value register which marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period which are the times when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described Section 12.6.2, "Description of Interrupt Operation."

Chapter 13
Liquid Crystal Display Module (S08LCDV1)

13.1 Introduction

In MC9RS08LE4 series, the Liquid Crystal Display module (LCD) controls the 22 LCD pins to generate the waveforms necessary to drive a liquid crystal display. These LCD pins can be configured for 8 × 14 or 4 × 18 based on software configuration.

13.1.1 LCD Clock Sources

The LCD module on MC9RS08LE4 can be clocked from ICSFFCLK or the external clock (ICSERCLK).

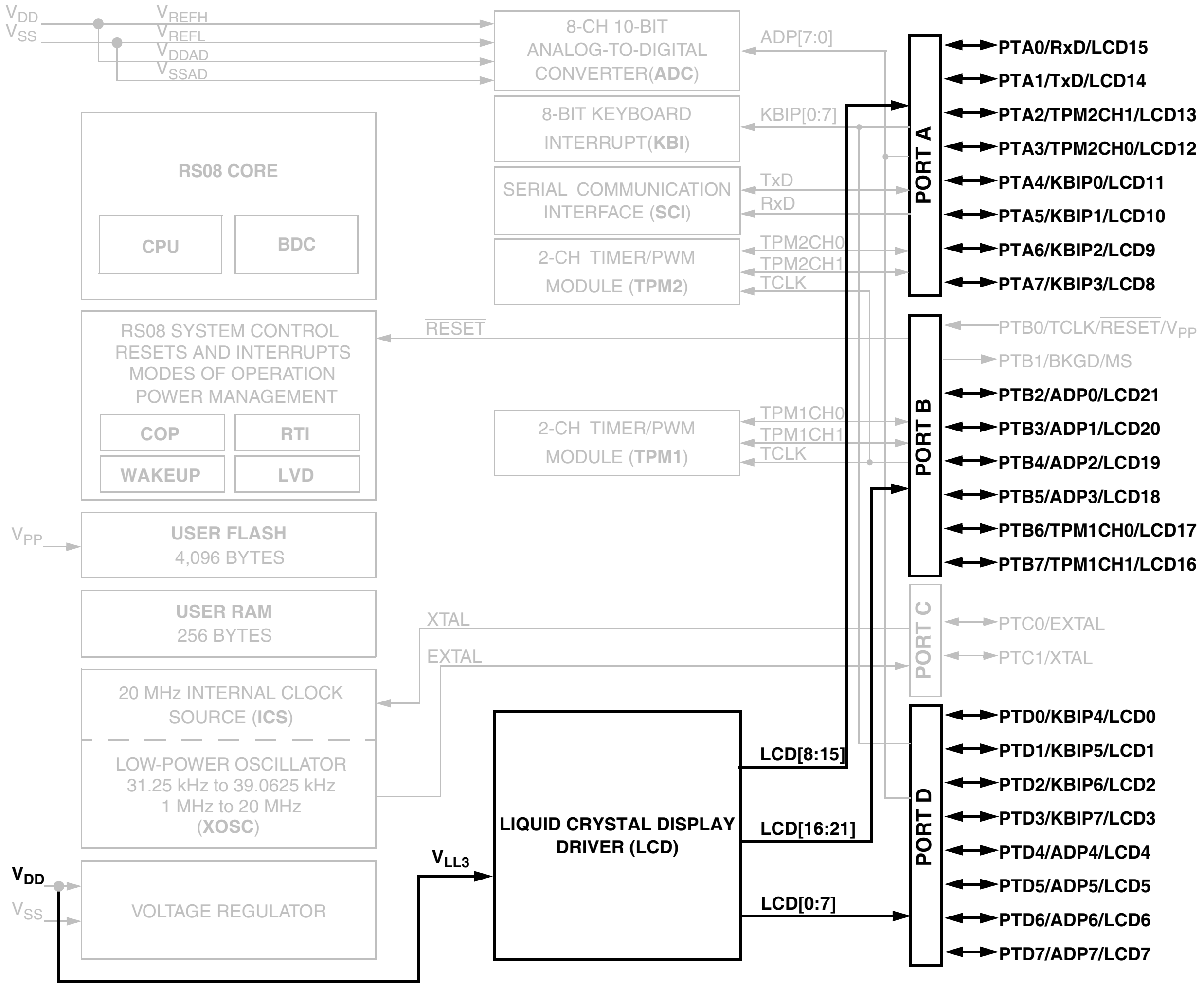
13.1.2 LCD Modes of Operation

The LCD module can be configured to operate in stop modes by clearing the LCDSTP bit. In stop mode, only ICSERCLK is available for LCD module.

13.1.3 LCD Power Supply Configurations

In MC9RS08LE4 series, the LCD power supply is connected to V_{DD} internally.

Figure 13-1 shows the MC9RS08LE4 block diagram, highlighting the LCD block and pins.



NOTES:

1. PTB0/TCLK/RESET/V_{PP} is an input-only pin when used as port pin
2. PTB1/BKGD/MS is an output-only pin

Figure 13-1. MC9RS08LE4 Series Block Diagram Highlighting LCD Block and Pins

13.1.4 Features

The LCD module driver features include:

- LCD waveforms functional in wait, and stop low-power modes
- 22 LCD (LCD[21:0]) pins with selectable frontplane/backplane configuration
  - — Generate up to 21 frontplane signals
  - — Generate up to 8 backplanes signals
- Programmable LCD frame frequency
- Programmable blink modes and frequency
  - — All segments blank during blink period
  - — Alternate display for each LCD segment in x4 or less mode
  - — Blink operation in low-power modes
- LCD power supply is configured as follows:
  - — Internal LCD power using V_{DD} (2.7to 5.5 V)
- On-chip generation of bias voltages by resistor network.
- Waveform storage registers LCDWF
- Backplane reassignment to assist in vertical scrolling on dot-matrix displays
- Software configurable LCD frame frequency interrupt
- Internal ADC channels are connected to V_{LL1} and V_{LL2} to monitor their magnitudes. This feature allows software to adjust the contrast.

13.1.5 Modes of Operation

The LCD module supports the following operation modes:

Mode	Operation
Stop	Depending on the state of the LCDSTP bit, the LCD module can operate an LCD panel in stop mode. If LCDSTP = 1, LCD module clock generation is turned off and the LCD module enters a power conservation state and is disabled. If LCDSTP = 0, the LCD module can operate an LCD panel in stop, and the LCD module continues displaying the current LCD panel contents based on the LCD operation prior to the stop event. If the LCD is enabled in stop, the selected LCD clock source, ICSERCLK or the Alternate Clock, must be enabled to operate in stop. In stop mode the LCD frame interrupt can cause the MCU to exit stop.
Wait	Depending on the configuration, the LCD module can operate an LCD panel in wait mode. If LCDWAI = 1, the LCD module clock generation is turned off and the LCD module enters a power-conservation state and is disabled. If LCDWAI = 0, the LCD module can operate an LCD panel in wait, and the LCD module continues displaying the current LCD panel contents base on the LCDWF registers. In wait mode, the LCD frame interrupt can cause the MCU to exit wait.

Stop provides the lowest power consumption state where the LCD module is functional. To operate the LCD in stop mode, use an external reference clock.

13.1.6 Block Diagram

Figure 13-2 is a block diagram of the LCD module.

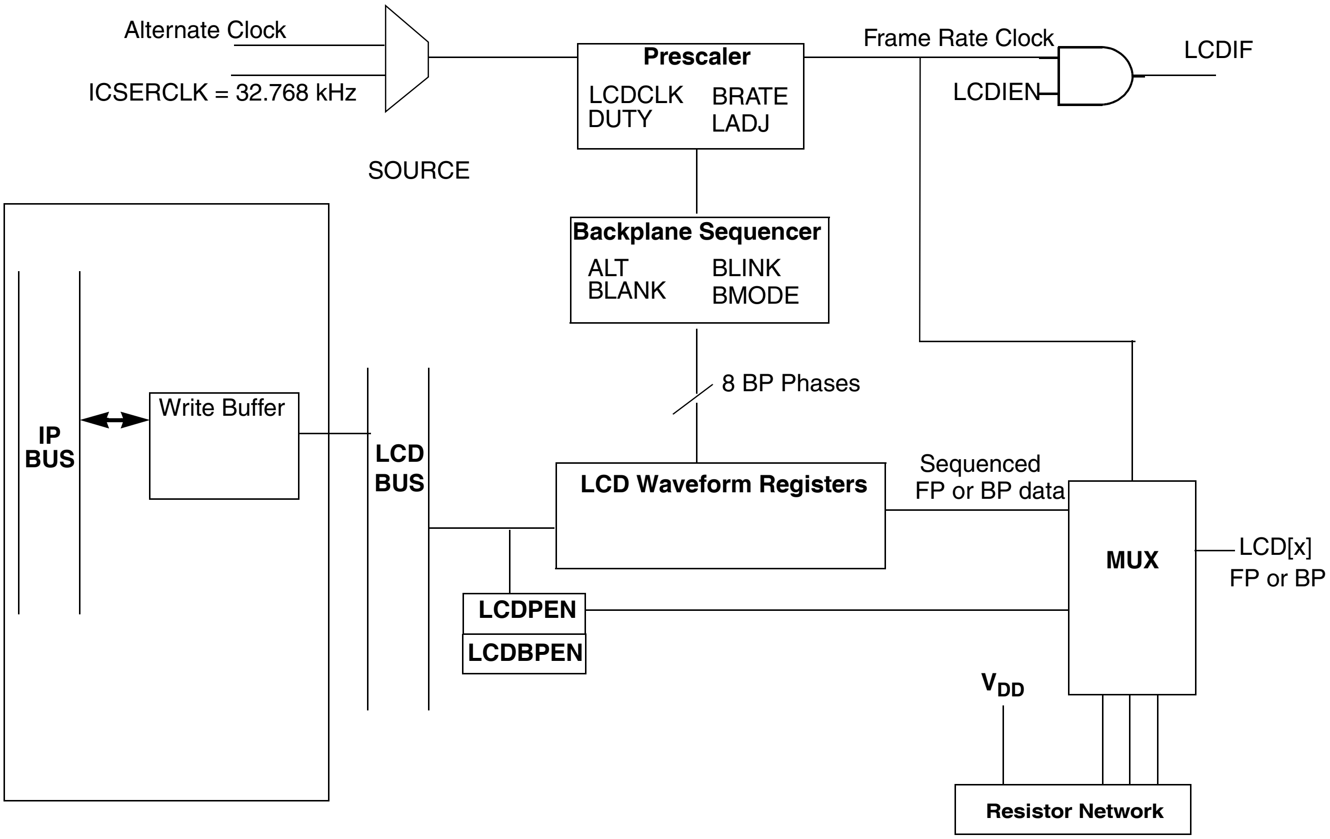


Figure 13-2. LCD Driver Block Diagram

13.2 External Signal Description

The LCD module has several external pins dedicated to LCD frontplane/backplane signaling. The LCD module can be configured to support eight backplane signals. The table below itemizes all the LCD external pins. See the Pins and Connections chapter for device-specific pin configurations.

Table 13-2. Signal Properties

Name	Port	Function	Reset State
22 LCD frontplane/backplane	LCD[21:0]	Switchable frontplane/backplane driver that connects directly to the display LCD[21:0] can operate as GPIO pins	High impedance

13.2.1 LCD[21:0]

When LCD functionality is enabled by the PEN[21:0] bits in the LCDPEN registers, the corresponding LCD[21:0] pin will generate a frontplane or backplane waveform depending on the configuration of the backplane-enable bit field (BPEN[21:0]).

13.3 Memory Map and Register Definition

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

13.3.1 LCD Control Register 0 (LCDC0)

	7	6	5	4	3	2	1	0
R W	LCDEN	SOURCE	LCLK2	LCLK1	LCLK0	DUTY2	DUTY1	DUTY0
Reset	0	0	0	0	0	0	1	1

▨ = Unimplemented or Reserved

Figure 13-3. LCD Control Register 0 (LCDC0)

Read: anytime

Write: LCDEN anytime. Do not change SOURCE LCLCK OR DUTY while LCDEN = 1.

Table 13-3. LCDC0 Field Descriptions

Field	Description
7 LCDEN	**LCD Driver Enable** — LCDEN starts LCD-module-waveform generator. 0 All frontplane and backplane pins are disabled. The LCD module system is also disabled, and all LCD waveform generation clocks are stopped. V_{LL3} is connected to V_{DD} internally 1 LCD module driver system is enabled and frontplane and backplane waveforms are generated. All LCD pins enabled using the LCD pin enable register (LCDPEN[x]) will output an LCD module driver waveform.The backplane pins will output an LCD module driver backplane waveform based on the settings of DUTY[2:0]. Chargepump or resistor bias is enabled.
6 SOURCE	**LCD Clock Source Select** — The LCD module has two possible clock sources. This bit is used to select which clock source is the basis for LCDCLK. 0 Selects the (external clock reference) as the LCD clock source. 1 Selects the alternate clock as the LCD clock source.

Table 13-3. LCDC0 Field Descriptions (continued)

Field	Description
5:3 LCLK[2:0]	**LCD Clock Prescaler** — Used as a clock divider to generate the LCD module frame frequency as shown in Equation 13-1. LCD-module-duty-cycle configuration is used to determine the LCD module frame frequency. LCD module frame frequency calculations are provided in Table 13-12. ***Eqn. 13-1*** $\text{LCD Module Frame Frequency} = \dfrac{\text{LCDCLK}}{((\text{DUTY}+1) \times 8 \times (4 + \text{LCLK[2:0]}) \times Y)}$ where $30 <$ LCDCLK < 39.063 kHz where Y = 2,2,3,3,4,5,8,16 chosen by module duty cycle configuration
2:0 DUTY[2:0]	**LCD Duty Select** — DUTY[2:0] bits select the duty cycle of the LCD module driver. 000 Use 1 BP (1/1 duty cycle). 001 Use 2 BP (1/2 duty cycle). 010 Use 3 BP (1/3 duty cycle). 011 Use 4 BP (1/4 duty cycle). (Default) 100 Use 5 BP (1/5 duty cycle). 101 Use 6 BP (1/6 duty cycle). 110 Use 7 BP (1/7 duty cycle). 111 Use 8 BP (1/8 duty cycle).

13.3.2 LCD Control Register 1 (LCDC1)

	7	6	5	4	3	2	1	0
R	LCDIEN	0	0	0	0	FCDEN	LCDWAI	LCDSTP
W		▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 13-4. LCD Control Register 1 (LCDC1)

Read: anytime

Write: anytime

Table 13-4. LCDC1 Field Descriptions

Field	Description
7 LCDIEN	**LCD Module Frame Frequency Interrupt Enable** — Enables an LCD interrupt event that coincides with the LCD module frame frequency. 0 No interrupt request is generated by this event. 1 The start of the LCD module frame causes an LCD module frame frequency interrupt request.
2 FCDEN	**Full Complementary Drive Enable** — This bit allows GPIO that are shared with LCD pins to operate as full complementary if the other conditions necessary have been met. The other conditions are: VSUPPLY = 11. 0 GPIO shared with LCD operate as open drain outputs, input levels and internal pullup resistors are referenced to V_{DD}. 1 If VSUPPLY = 11, GPIO shared with LCD operate as full complementary outputs. Input levels and internal pullup resistors are referenced to V_{LL3}.

Table 13-4. LCDC1 Field Descriptions (continued)

Field	Description
1 LCDWAI	**LCD Module Driver and Charge Pump Stop While in Wait Mode** 0 Allows the LCD driver and charge pump to continue running during wait mode. 1 Disables the LCD driver and charge pump when MCU goes into wait mode.
0 LCDSTP	**LCD Module Driver and Charge Pump Stop While in Stop Mode** 0 Allows LCD module driver and charge pump to continue running during stop. 1 Disables LCD module driver and charge pump when MCU goes into stop.

13.3.3 LCD Voltage Supply Register (LCDSUPPLY)

	7	6	5	4	3	2	1	0
R	0	0	LADJ1	LADJ0	0	0	VSUPPLY1	VSUPPLY0
W	▨	▨			▨	▨		
Reset	0	0	1	1	0	1	0	1

▨ Unimplemented or Reserved

Read: anytime

Write: anytime.

For proper operation, do not modify VSUPPLY[1:0] while the LCDEN bit is asserted. VSUPPLY[1:0] has to be configured to 01 for LCD display since LCD power supply is connected to V_{DD} internally.

Table 13-5. LCDSUPPLY Field Descriptions

Field	Description
5:4 LADJ[1:0]	**LCD Module Load Adjust** — The LCD load adjust bits are used to configure the LCD module to handle different LCD glass capacitance. Adjust the resistor bias network for different LCD glass capacitance 00 - Low Load (LCD glass capacitance 2000pf or lower) 01 - Low Load (LCD glass capacitance 2000pf or lower) 10 - High Load (LCD glass capacitance 8000pf or lower) 11 - High Load (LCD glass capacitance 8000pf or lower)
1:0 VSUPPLY[1:0]	**Voltage Supply Control** — Configures whether the LCD module power supply is external or internal. Avoid modifying this bit field while the LCD module is enabled (e.g., LCDEN = 1). 00 Reserved 01 Drive V_{LL3} internally from V_{DD} 10 Reserved 11 Drive V_{LL3} externally

13.3.4 LCD Blink Control Register (LCDBCTL)

	7	6	5	4	3	2	1	0
R	BLINK	ALT	BLANK	0	BMODE	BRATE2	BRATE1	BRATE0
W				▨				
Reset	0	0	0	0	0	0	0	0

▨ Unimplemented or Reserved

Figure 13-6. LCD Blink Control Register (LCDBCTL)

Read: anytime

Write: anytime

Table 13-6. LCDBCTL Field Descriptions

Field	Description
7 BLINK	**Blink Command** — Starts or stops LCD module blinking 0 Disables blinking 1 Starts blinking at blinking frequency specified by LCD blink rate calculation (see Equation 13-2)
6 ALT	**Alternate Display Mode** — For four backplanes or less the LCD backplane sequencer changes to output an alternate display. ALT bit is ignored if Duty is 5 or greater. 0 Normal Display 1 Alternate display mode
5 BLANK	**Blank Display Mode** — Asserting this bit clears all segments in the LCD display. 0 Normal or Alternate Display 1 Blank Display Mode
3 BMODE	**Blink Mode** — Selects the blink mode displayed during the blink period. See Table 13-6 for more information on how BMODE affects the LCD display. 0 Display blank during the blink period 1 Display alternate display during blink period (Ignored if duty is 5 or greater)
2:0 BRATE[2:0]	**Blink-Rate Configuration**— Selects frequency at which the LCD display blinks when the BLINK is asserted. Equation 13-2 shows how BRATE[2:0] bit field is used in the LCD blink-rate calculation. Equation 13-2 provides an expression for the LCD module blink rate *Eqn. 13-2* $$\text{LCD module blink rate} = \frac{\text{LCDCLK}}{2^{(12+\text{ BRATE[2:0]})}}$$ LCD module blink rate calculations are provided in 13.4.3.2/p.203.

13.3.5 LCD Status Register (LCDS)

	7	6	5	4	3	2	1	0
R	LCDIF	0	0	0	0	0	0	0
W		▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ Unimplemented or Reserved

Figure 13-7. LCD Status Register (LCDS)

Read: anytime

Write: anytime

Table 13-7. LCDS Field Descriptions

Field	Description
7 LCDIF	**LCD Interrupt Flag** — LCDIF indicates an interrupt condition occurred. To clear the interrupt write a 1 to LCDIF. 0 interrupt condition has not occurred. 1 interrupt condition has occurred.

13.3.6 LCD Pin Enable Registers 0–2 (LCDPEN0–LCDPEN2)

When LCDEN = 1, these bits enable the corresponding LCD pin for LCD operation.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register. Initialize these registers before enabling the LCD module.

		7	6	5	4	3	2	1	0
LCDPEN0	R W	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0
	Reset	Indeterminate after reset							
LCDPEN1	R W	PEN15	PEN14	PEN13	PEN12	PEN11	PEN10	PEN9	PEN8
	Reset	Indeterminate after reset							
LCDPEN2	R	0	0	PEN21	PEN20	PEN19	PEN18	PEN17	PEN16
	W	▨	▨						
	Reset	Indeterminate after reset							

▨ Unimplemented or Reserved

Figure 13-8. LCD Pin Enable Registers 0–2 (LCDPEN0–LCDPEN2)

Read: anytime

Write: anytime

Table 13-8. LCDPEN0–LCDPEN2 Field Descriptions

Field	Description
21:0 PEN[21:0]	**LCD Pin Enable** — The PEN[21:0] bit enables the LCD[21:0] pin for LCD operation. Each LCD[21:0] pin can be configured as a backplane or a frontplane based on the corresponding BPEN[n] bit in the Backplane Enable Register (LCDBPEN[2:0]). If LCDEN = 0, these bits have no effect on the state of the I/O pins. Set PEN[21:0] bits before LCDEN is set. 0 LCD operation disabled on LCDn. 1 LCD operation enabled on LCDn.

13.3.7 Backplane Enable Registers 0–2 (BPEN0–BPEN2)

When LCDPEN[n] = 1, these bits configure the corresponding LCD pin to operate as an LCD backplane or an LCD frontplane. Most applications set a maximum of eight of these bits. Initialize these registers before enabling the LCD module.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register.

		7	6	5	4	3	2	1	0
LCDBPEN0	R W	BPEN7	BPEN6	BPEN5	BPEN4	BPEN3	BPEN2	BPEN1	BPEN0
	Reset	Indeterminate after reset							
LCDBPEN1	R W	BPEN15	BPEN14	BPEN13	BPEN12	BPEN11	BPEN10	BPEN9	BPEN8
	Reset	Indeterminate after reset							
LCDBPEN2	R	0	0	BPEN21	BPEN20	BPEN19	BPEN18	BPEN17	BPEN16
	W	▨	▨						
	Reset	Indeterminate after reset							

▨ Unimplemented or Reserved

Figure 13-9. Backplane Enable Registers 0–2 (BPEN0–BPEN2)

Read: anytime

Write: anytime

Table 13-9. LCDBPEN0–LCDBPEN2 Field Descriptions

Field	Description
21:0 BPEN[21:0]	**Backplane Enable** — The BPEN[21:0] bit configures the LCD[21:0] pin to operate as an LCD backplane or LCD frontplane. If LCDEN = 0, these bits have no effect on the state of the I/O pins. It is recommended to set BPEN[21:0] bits before LCDEN is set. 0 Frontplane operation enabled on LCD[n]. 1 Backplane operation enabled on LCD[n].

13.3.8 LCD Waveform Registers (LCDWF[21:0])

Each frontplane segment is associated with a backplane phase (A-H). For an LCD pin configured as a frontplante the LCDWF registers control the on/off state for frontplane segments.

For an LCD pin configured as a backplane the LCDWF registers controls the phase (A-H) in which the associated backplane pin is active.

These registers should only be written with instructions that perform byte writes, using instructions that perform word writes will lead to invalid data being placed in the register.

After reset, the LCDWF contents are indeterminate as indicated by Figure 13-10.

		7	6	5	4	3	2	1	0
LCDWF0	R W	BPHLCD0	BPGLCD0	BPFLCD0	BPELCD0	BPDLCD0	BPCLCD0	BPBLCD0	BPALCD0
	Reset	Indeterminate after reset							
LCDWF1	R W	BPHLCD1	BPGLCD1	BPFLCD1	BPELCD1	BPDLCD1	BPCLCD1	BPBLCD1	BPALCD1
	Reset	Indeterminate after reset							
LCDWF2	R W	BPHLCD2	BPGLCD2	BPFLCD2	BPELCD2	BPDLCD2	BPCLCD2	BPBLCD2	BPALCD2
	Reset	Indeterminate after reset							
LCDWF3	R W	BPHLCD3	BPGLCD3	BPFLCD3	BPELCD3	BPDLCD3	BPCLCD3	BPBLCD3	BPALCD3
	Reset	Indeterminate after reset							
LCDWF4	R W	BPHLCD4	BPGLCD4	BPFLCD4	BPELCD4	BPDLCD4	BPCLCD4	BPBLCD4	BPALCD4
	Reset	Indeterminate after reset							
LCDWF5	R W	BPHLCD5	BPGLCD5	BPFLCD5	BPELCD5	BPDLCD5	BPCLCD5	BPBLCD5	BPALCD5
	Reset	Indeterminate after reset							
LCDWF6	R W	BPHLCD6	BPGLCD6	BPFLCD6	BPELCD6	BPDLCD6	BPCLCD6	BPBLCD6	BPALCD6
	Reset	Indeterminate after reset							
LCDWF7	R W	BPHLCD7	BPGLCD7	BPFLCD7	BPELCD7	BPDLCD7	BPCLCD7	BPBLCD7	BPALCD7
	Reset	Indeterminate after reset							
LCDWF8	R W	BPHLCD8	BPGLCD8	BPFLCD8	BPELCD8	BPDLCD8	BPCLCD8	BPBLCD8	BPALCD8
	Reset	Indeterminate after reset							
LCDWF9	R W	BPHLCD9	BPGLCD9	BPFLCD9	BPELCD9	BPDLCD9	BPCLCD9	BPBLCD9	BPALCD9
	Reset	Indeterminate after reset							
LCDWF10	R W	BPHLCD10	BPGLCD10	BPFLCD10	BPELCD10	BPDLCD10	BPCLCD10	BPBLCD10	BPALCD10
	Reset	Indeterminate after reset							

Figure 13-10. LCD Waveform Registers (LCDWF[21:0])

LCDWF11	R W	BPHLCD11	BPGLCD11	BPFLCD11	BPELCD11	BPDLCD11	BPCLCD11	BPBLCD11	BPALCD11
	Reset	Indeterminate after reset							
LCDWF12	R W	BPHLCD12	BPGLCD12	BPFLCD12	BPELCD12	BPDLCD12	BPCLCD12	BPBLCD12	BPALCD12
	Reset	Indeterminate after reset							
LCDWF13	R W	BPHLCD13	BPGLCD13	BPFLCD13	BPELCD13	BPDLCD13	BPCLCD13	BPBLCD13	BPALCD13
	Reset	Indeterminate after reset							
LCDWF14	R W	BPHLCD14	BPGLCD14	BPFLCD14	BPELCD14	BPDLCD14	BPCLCD14	BPBLCD14	BPALCD14
	Reset	Indeterminate after reset							
LCDWF15	R W	BPHLCD15	BPGLCD15	BPFLCD15	BPELCD15	BPDLCD15	BPCLCD15	BPBLCD15	BPALCD15
	Reset	Indeterminate after reset							
LCDWF16	R W	BPHLCD16	BPGLCD16	BPFLCD16	BPELCD16	BPDLCD16	BPCLCD16	BPBLCD16	BPALCD16
	Reset	Indeterminate after reset							
LCDWF17	R W	BPHLCD17	BPGLCD17	BPFLCD17	BPELCD17	BPDLCD17	BPCLCD17	BPBLCD17	BPALCD17
	Reset	Indeterminate after reset							
LCDWF18	R W	BPHLCD18	BPGLCD18	BPFLCD18	BPELCD18	BPDLCD18	BPCLCD18	BPBLCD18	BPALCD18
	Reset	Indeterminate after reset							
LCDWF19	R W	BPHLCD19	BPGLCD19	BPFLCD19	BPELCD19	BPDLCD19	BPCLCD19	BPBLCD19	BPALCD19
	Reset	Indeterminate after reset							
LCDWF20	R W	BPHLCD20	BPGLCD20	BPFLCD20	BPELCD20	BPDLCD20	BPCLCD20	BPBLCD20	BPALCD20
	Reset	Indeterminate after reset							
LCDWF21	R W	BPHLCD21	BPGLCD21	BPFLCD21	BPELCD21	BPDLCD21	BPCLCD21	BPBLCD21	BPALCD21
	Reset	Indeterminate after reset							

Figure 13-10. LCD Waveform Registers (LCDWF[21:0]) (continued)

Table 13-10. LCDWF Field Descriptions

Field	Description
BP[y]LCD[x]	**Segment-on-Frontplane Operation** — If the LCD[x] pin is enabled and configured to operate as a frontplane, the BP[y]LCD[x] bit in the LCDWF registers controls the on/off state for the LCD segment connected between LCD[x] and BP[y].BP[y] corresponds to an LCD[:0] pin enabled and configured to operate as a backplane that is active in phase [y]. Asserting BP[y]LCD[x] displays (turns on) the LCD segment connected between LCD[x] and BP[y]. 0 LCD segment off 1 LCD segment on **Segment-on-Backplane Operation** — If the LCD[x] pin is enabled and configured to operate as a backplane, the BP[y] LCD[x] bit in the LCDWF registers controls the phase (A-H) in which the LCD[x] pin is active.Backplane phase assignment is done using this method. 0 LCD backplane inactive for phase[y] 1 LCD backplane active for phase[y].

13.4 Functional Description

This section provides a complete functional description of the LCD block, detailing the operation of the design from the end-user perspective.

Before enabling the LCD module by asserting the LCDEN bit in the LCDC0 register, configure the LCD module based on the end application requirements. Out of reset, the LCD module is configured with default settings, but these settings are not optimal for every application. The LCD module provides several versatile configuration settings and options to support varied implementation requirements, including:

- Frame frequency
- Duty cycle (number of backplanes)
- Backplane assignment (which LCD[:0] pins operate as backplanes)
- Frame frequency interrupt enable
- Blinking frequency and options
- Power-supply configurations

The LCD module also provides an LCD pin enable control. Setting the LCD pin enable bit (PEN[x] in the LCDPEN[y] register) for a particular LCD[y] pin enables the LCD module functionality of that pin once the LCDEN bit is set. When the BPEN[x] bit in the LCDBPEN[y] is set, the associated pin operates as a backplane. The LCDWF registers can then activate (display) the corresponding LCD segments on an LCD panel.

The LCDWF registers control the on/off state for the segments controlled by the LCD pins defined as front planes and the active phase for the backplanes. Blank display modes do not use the data from the LCDWF registers. When using the LCDWF register for frontplane operation, writing a 0 turns the segment off.

For pins enabled as backplane, the phase of the backplane (A-H) is assigned by the LCDWF register for the corresponding backplane pin.

13.4.1 LCD Driver Description

The LCD module driver has 8 modes of operation:

- 1/1 duty (1 backplane) (Phase A), 1/3 bias (4 voltage levels)
- 1/2 duty (2 backplanes) (Phase A, B), 1/3 bias (4 voltage levels)
- 1/3 duty (3 backplanes) (Phase A, B, C), 1/3 bias (4 voltage levels)
- 1/4 duty (4 backplanes) (Phase A, B, C, D), 1/3 bias (4 voltage levels)
- 1/5 duty (5 backplanes) (Phase A, B, C, D, E), 1/3 bias (4 voltage levels)
- 1/6 duty (6 backplanes) (Phase A, B, C, D, E, F), 1/3 bias (4 voltage levels)
- 1/7 duty (7 backplanes) (Phase A, B, C, D, E, F, G), 1/3 bias (4 voltage levels)
- 1/8 duty (8 backplanes) (Phase A, B, C, D, E, F, G, H), 1/3 bias (4 voltage levels)

All modes are 1/3 bias. These modes of operation are described in more detail in the following sections.

13.4.1.1 LCD Duty Cycle

The denominator of the duty cycle indicates the number of LCD panel segments capable of being driven by each individual frontplane output driver. Depending on the duty cycle, the LCD waveform drive can be categorized as static or multiplexed.

In static-driving method, the LCD is driven with two square waveforms. The static-driving method is the most basic method to drive an LCD panel, but because each frontplane driver can drive only one LCD segment, static driving limits the LCD segments that can be driven with a given number of frontplane pins. In static mode, only one backplane is required.

In multiplexed mode, the LCD waveforms are multi-level and depend on the bias mode. Multiplex mode, depending on the number of backplanes, can drive multiple LCD segments with a single frontplane driver. This reduces the number of driver circuits and connections to LCD segments. For multiplex mode operation, at least two backplane drivers are needed. The LCD module is optimized for multiplex mode.

The duty cycle indicates the amount of time the LCD panel segment is energized during each LCD module frame cycle. The denominator of the duty cycle indicates the number of backplanes that are being used to drive an LCD panel.

The duty cycle is used by the backplane phase generator to set the phase outputs. The phase outputs A-H are driven according to the sequence shown below. The sequence is repeated at the LCD frame frequency. The duty cycle is configured using the DUTY[2:0] bit field in the LCDC0 register, as shown in Table 13-11.

Table 13-11. LCD Module Duty Cycle Modes

Duty	LCDC0 Register			Number of Backplanes	Phase Sequence
	DUTY2	DUTY1	DUTY0		
1/1	0	0	0	1	A
1/2	0	0	1	2	A B
1/3	0	1	0	3	A B C
1/4	0	1	1	4	A B C D

Table 13-11. LCD Module Duty Cycle Modes (continued)

Duty	LCDC0 Register			Number of Backplanes	Phase Sequence
	DUTY2	DUTY1	DUTY0		
1/5	1	0	0	5	A B C D E
1/6	1	0	1	6	A B C D E F
1/7	1	1	0	7	A B C D E F G
1/8	1	1	1	8	A B C D E F G H

13.4.1.2 LCD Bias

Because a single frontplane driver is configured to drive more and more individual LCD segments, 3 voltage levels are required to generate the appropriate waveforms to drive the segment. The LCD module is designed to operate using the 1/3 bias mode.

13.4.1.3 LCD Module Base Clock and Frame Frequency

The LCD module is optimized to operate using a 32.768-kHz clock input. Two clock sources are available to the LCD module, which are selectable by configuring the SOURCE bit in the LCDC0 register. The two clock sources include:

- External crystal — (SOURCE = 0)
- Alternate clock (SOURCE = 1)

Figure 13-11 shows the LCD clock tree. The clock tree shows the two possible clock sources and the LCD frame frequency and blink frequency clock source. The LCD blink frequency is discussed in Section 13.4.3.2, “Blink Frequency.”

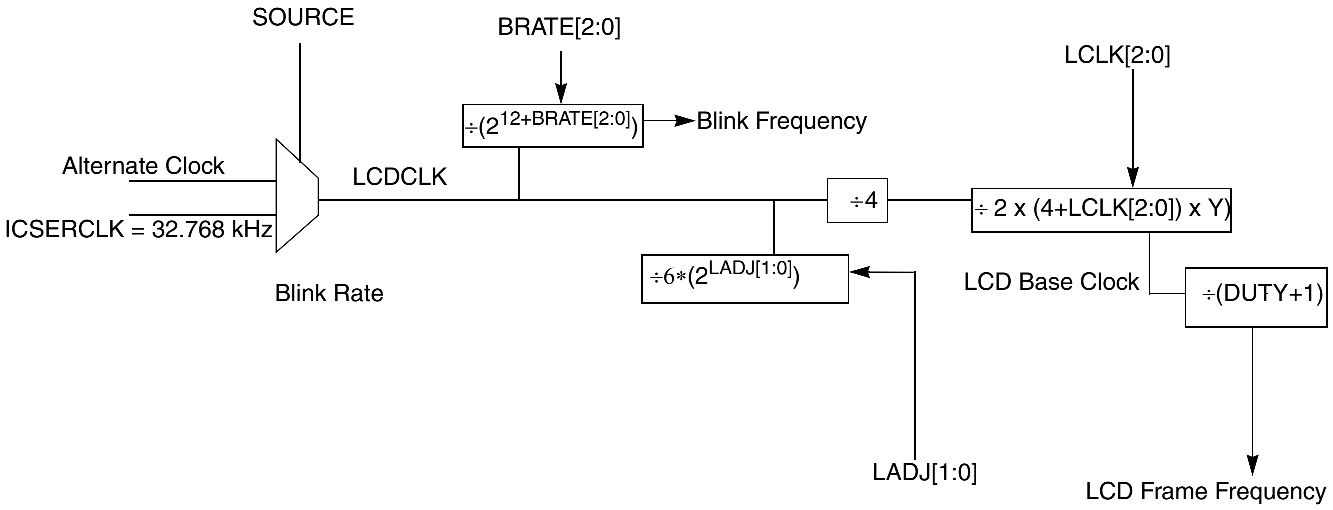


Figure 13-11. LCD Clock Tree

An external 32.768 kHz clock input is required to achieve lowest power consumption.

The value of LCDCLK is important because it is used to generate the LCD module frame frequency. Equation 13-1 provides an expression for the LCD module frame frequency calculation.

The LCD module frame frequency is a function of the LCD module duty cycle as shown in Equation 13-1. Table 13-13 and Table 13-12 show LCD module frame frequency calculations that consider several possible LCD module configurations of LCLK[2:0] and DUTY[2:0].

The LCD module frame frequency is defined as the number of times the LCD segments are energized per second. The LCD module frame frequency must be selected to prevent the LCD display from flickering (LCD module frame frequency is too low) or ghosting (LCD module frame frequency is too high). To avoid these issues, an LCD module frame frequency in the range of 28 to 58 Hz is required. LCD module frame frequencies less than 28 Hz or greater than 58 Hz are out of specification, and so are invalid. Selecting lower values for the LCD base and frame frequency results in lower current consumption for the LCD module.

The LCD module base clock frequency is the LCD module frame frequency multiplied by the number of backplane phases that are being generated. The number of backplane phases is selected using the DUTY[2:0] bits. The LCD module base clock is used by the backplane sequencer to generate the LCD waveform data for the enabled phases (A-H).

Table 13-12. LCD Module Frame Frequency Calculations[1]

Duty Cycle	1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
Y	**16**	**8**	**5**	**4**	**3**	**3**	**2**	**2**
LCLK[2:0]								
0	64	64	68.3	64	68.3	56.9	73.1	64
1	51.2	51.2	54.6	51.2	54.6	45.5	58.5	51.2
2	42.7	42.7	45.5	42.7	45.5	37.9	48.8	42.7
3	36.6	36.6	39	36.6	39	32.5	41.8	36.6
4	32	32	34.1	32	34.1	28.4	36.6	32
5	28.4	28.4	30.3	28.4	30.3	25.3	32.5	28.4
6	25.6	25.6	27.3	25.6	27.3	22.8	29.3	25.6
7	23.3	23.3	24.8	23.3	24.8	20.7	26.6	23.3

[1] LCD clock input ~ 32.768 kHz

Shaded table entries are out of specification and invalid.

Table 13-13. LCD Module Frame Frequency Calculations[1]

Duty Cycle	1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8
Y	16	8	5	4	3	3	2	2
LCLK[2:0]								
0	76.3	76.3	81.4	76.3	81.4	67.8	87.2	76.3
1	61	61	65.1	61	65.1	54.3	69.8	61
2	50.9	50.9	54.3	50.9	54.3	45.2	58.1	50.9
3	43.6	43.6	46.5	43.6	46.5	38.8	49.8	43.6
4	38.1	38.1	40.7	38.1	40.7	33.9	43.6	38.1
5	33.9	33.9	36.2	33.9	36.2	30.1	38.8	33.9
6	30.5	30.5	32.6	30.5	32.6	27.1	34.9	30.5
7	27.7	27.7	29.6	27.7	29.6	24.7	31.7	27.7

[1] LCD clock input ~ 39.063 kHz

Shaded table entries are out of specification and invalid.

13.4.1.4 LCD Waveform Examples

This section shows the timing examples of the LCD output waveforms for the several modes of operation. As shown in Table 13-14, all examples use 1/3 bias mode.

Table 13-14. Configurations for Example LCD Waveforms

	Bias Mode	DUTY[2:0]	Duty Cycle
Example 1	1/3	001	1/2
Example 2		011	1/4
Example 3		111	1/8

13.4.1.4.1 1/2 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty=1/2:DUTY[2:0] = 001

LCD pin 0 (LCD[0])and LCD pin 1, LCD[1] enabled as backplanes:

BPEN0 =1 and BPEN1 =1 in the LCDBPEN0

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

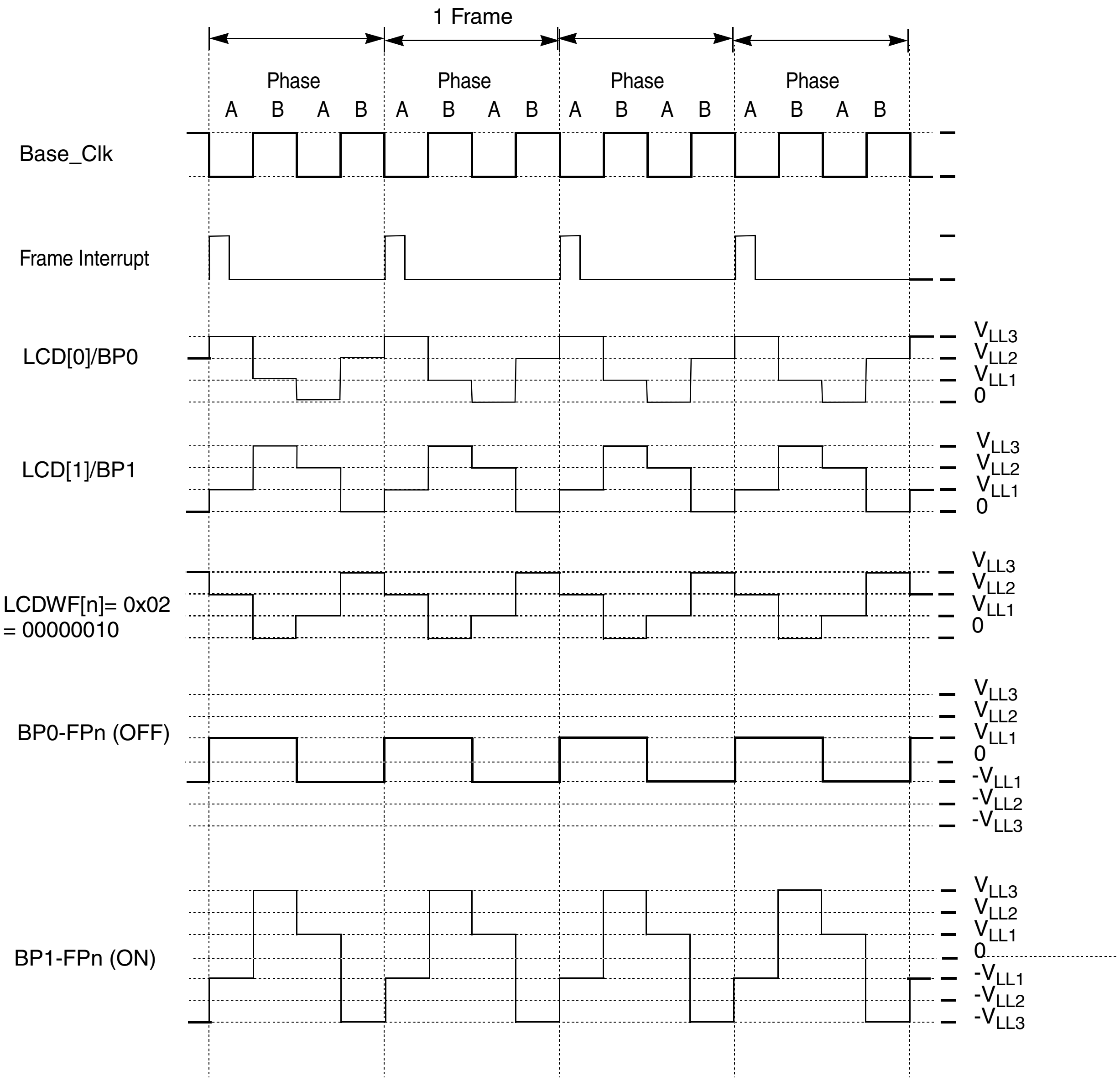


Figure 13-12. 1/2 Duty and 1/3 Bias (Low-Power Waveform)

13.4.1.4.2 1/4 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty = 1/4: DUTY[2:0] = 011

LCD pins 0 – 3 enabled as backplanes: LCDBPEN0 = 0x0F

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

LCD[2] assigned to Phase C: LCDWF2 = 0x04

LCD[3] assigned to Phase D: LCDWF3 = 0x08

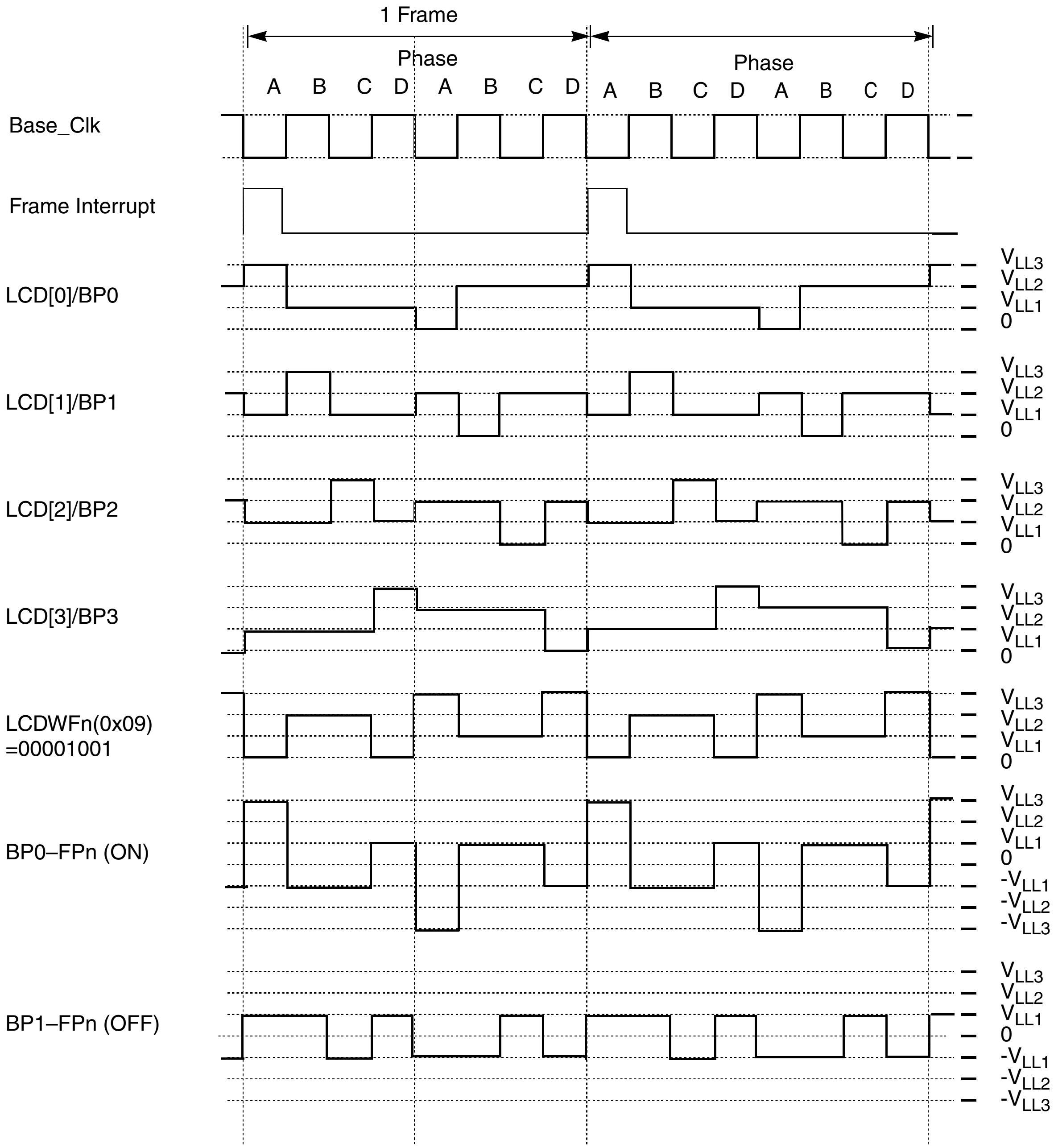


Figure 13-13. 1/4 Duty and 1/3 Bias (Low-Power Waveform)

13.4.1.4.3 1/8 Duty Multiplexed with 1/3 Bias Mode (Low-power Waveform)

Duty = 1/8:DUTY[2:0] = 111

LCD pins 0 – 7 enabled as backplanes: LCDBPEN0 = 0xFF

LCD[0] assigned to Phase A: LCDWF0 = 0x01

LCD[1] assigned to Phase B: LCDWF1 = 0x02

LCD[2] assigned to Phase C: LCDWF2 = 0x04

LCD[3] assigned to Phase D: LCDWF3 = 0x08

LCD[4] assigned to Phase E: LCDWF4 = 0x10

LCD[5] assigned to Phase F: LCDWF5 = 0x20

LCD[6] assigned to Phase G: LCDWF6 = 0x40

LCD[7] assigned to Phase H: LCDWF7 = 0x80

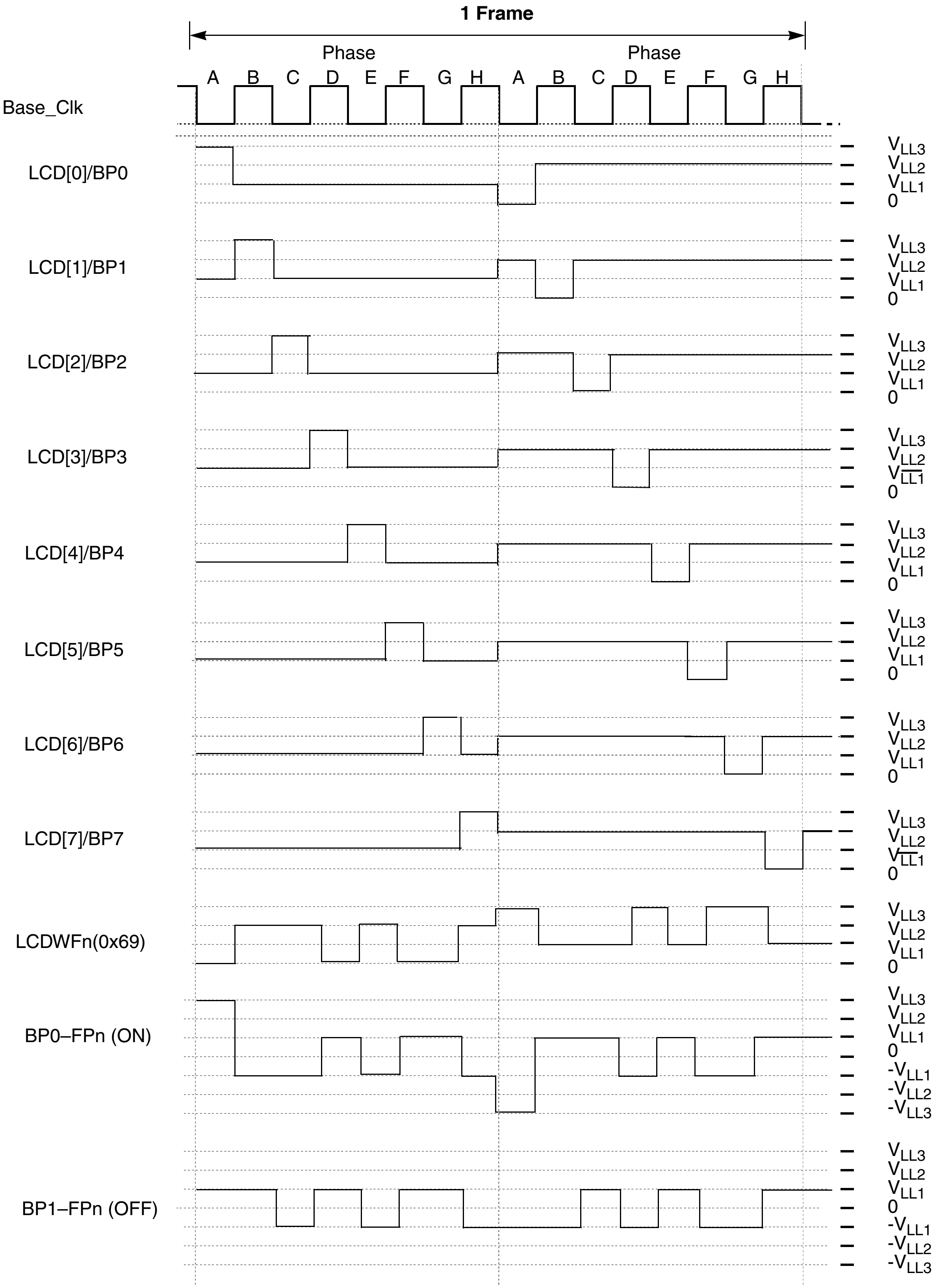


Figure 13-14. 1/8 Duty and 1/3 Bias (Low-power Waveform)

13.4.2 LCDWF Registers

For a segment on the LCD panel to be displayed, data must be written to the LCDWF registers. For LCD pins enabled as frontplanes, each bit in the LCDWF registers corresponds to a segment on an LCD panel. The different phases A-H represent the different backplanes of the LCD panel. The selected LCD duty cycle controls the number of implemented phases. Refer to Table 13-11 for normal LCD operation the phases follow the sequence shown.

For LCD pins enabled as a backplane, the LCDWF assigns the phase in which the backplane pin is active. This is how backplane assignment is done.

An example of normal operation follows: enable LCD pin 0 to operate as backplane 0. Enable the LCD pin 0 by setting PEN0 bit in the LCDPEN0 register. Configure LCD pin 0 as a backplane pin by setting the BPEN0 bit in the LCDBPEN0 register. Finally, the BPALCD0 bit in the LCDWF0 is set to associate LCD pin 0 with backplane phase A.This will configure LCD0 to operate as a backplane that is active in Phase A.

For LCD pins enabled as a frontplane, writing a 1 to a given LCDWF location results in the corresponding display segment being driven with the differential root mean square (RMS) voltage necessary to turn the segment on during the phase selected. Writing a 0 to a given location results in the corresponding display segment being driven with the differential RMS voltage necessary to turn the segment off during the phase selected.

13.4.3 LCD Display Modes

The LCD module can be configured to implement several different display modes. The bits ALT and BLANK in the LCD-blink-control register (LCDBCTL) configure the different display modes. In normal display mode (default), LCD segments are controlled by the data placed in the LCDWF registers, as described in Section 13.4.2, “LCDWF Registers.” For blank-display mode, the LCDWF data is bypassed and the frontplane and backplane pins are configured to clear all segments.

For alternate-display mode, the backplane sequence is modified for duty cycles of 1/4, 1/3, 1/2, and 1/1. For four backplanes or less, the backplane sequence is modified as shown below. The altered sequence allows two complete displays to be placed in the LDCDWF registers. The first display is placed in phases A-D and the second in phases E-H in the case of four backplanes. If the LCD duty cycle is five backplanes or greater, the ALT bit is ignored and creates a blank display. Refer to Table 13-16 for additional information.

Using the alternate display function an inverse display can be accomplished for x4 mode and less by placing inverse data in the alternate phases of the LCDWF registers.

Table 13-15. Alternate Display Backplane Sequence

Duty	Backplane Sequence	Alt. Backplane Sequence
1/1	A	E
1/2	A B	E F

Table 13-15. Alternate Display Backplane Sequence (continued)

Duty	Backplane Sequence	Alt. Backplane Sequence
1/3	A B C	E F G
1/4	A B C D	E F G H

13.4.3.1 LCD Blink Modes

The blink mode is used as a means of alternating among different LCD display modes at a defined frequency. The LCD module can be configured to implement two blink modes. The BMODE bit in the LCD-blink-control register (LCDBCTL) configures the different blink modes. Blink modes are activated by setting the BLINK bit in the LCDBCTL register. If BLINK = 0, the LCD module operates normally as described Section 13.4.3, “LCD Display Modes”. If BLINK = 1, BMODE bit configures the blinking operation. During a blink, the display data driven by the LCD module changes to the mode selected by the BMODE bit. The BMODE bit selects two different blink modes, blank and alternate modes operate in the same way, as defined in Section 13.4.3, “LCD Display Modes.” The table below shows the interaction between display modes and blink modes. If the LCD duty cycle is five backplanes or greater, BMODE = 1 is ignored and will revert to create a blank display during the blink period.

Table 13-16. Display Mode Interaction

BLANK	ALT	BMODE	LCD Duty	BLINK = 1	
				Normal Period	Blink Period
0	0	0	1-4	Normal Display	Blank Display
0	0	1	1-4	Normal Display	Alternate display
0	1	0	1-4	Alternate display	Blank Display
0	1	1	1-4	Alternate Display	Alternate display
1	X	0	1-4	Blank Display	Blank DIsplay
1	X	1	1-4	Blank Display	Alternate display
0	X	X	5-8	Normal Display	Blank Display
1	X	X	5-8	Blank Display	Blank Display

13.4.3.2 Blink Frequency

The LCD clock is the basis for the calculation of the LCD module blink frequency. The LCD module blink frequency is equal to the LCD clock (LCDCLK) divided by the factor selected by the BRATE[2:0] bits. Table 13-17 shows LCD module blink frequency calculations for all values of BRATE[2:0] at a few common LCDCLK selections.

Table 13-17. Blink Frequency Calculations
(Blink Rate = LCD Clock(Hz) ÷ Blink Divider)

BRATE[2:0]	0	1	2	3	4	5	6	7
LCD Clock	**Blink Frequency (Hz)**							
30 khz	7.32	3.66	1.831	.916	.46	.23	.11	.06
32.768 khz	8	4	2	1	.5	.25	.13	.06
39.063 khz	9.54	4.77	2.38	1.19	.6	.30	.15	.075

13.4.4 LCD Voltage Divider, and Power Supply Operation

LCD power supply is connected to V_{DD} internally. The bias voltages are generated by on-chip resistor network.

13.4.5 Resets

During a reset, the LCD module system is configured in the default mode. The default mode includes the following settings:

- LCDEN is cleared, thereby forcing all frontplane and backplane driver outputs to the high impedance state.
- 1/4 duty
- 1/3 bias
- LCLK[2:0], VSUPPLY[1:0], and BRATE[2:0] revert to their reset values

13.4.6 Interrupts

When an LCD module frame-frequency interrupt event occurs, the LCDIF bit in the LCDS register is asserted. The LCDIF bit remains asserted until software clears the LCD-module-frame-frequency interrupt. The interrupt can be cleared by software writing a 1 to the LCDIF bit.

If both the LCDIF bit in the LCDS register and the LCDIEN bit in the LCDC1 register are set, an LCD interrupt signal asserts.

13.5 Initialization Section

This section provides a recommended initialization sequence for the LCD module and also includes initialization examples for several LCD application scenarios.

13.5.1 Initialization Sequence

The below list provides a recommended initialization sequence for the LCD module.

You must write to all LCDPEN, LCDBPEN, and LCDWF registers to initialize their values after a reset.

1. LCDC0 register
   a) Configure LCD clock source (SOURCE bit).
2. LCDC1 register
   a) Configure LCD frame frequency interrupt (LCDIEN bit).
   b) Configure LCD behavior in low power mode (LCDWAI and LCDSTP bits).
3. LCDC0 register
   a) Configure LCD duty cycle (DUTY[2:0]).
   b) Select and configure LCD frame frequency (LCLK[2:0]).
4. LCDBCTL register
   a) Configure display mode (ALT and BLANK bits).
   b) Configure blink mode (BMODE).
   c) Configure blink frequency (BRATE[2:0]).
5. LCDPEN[7:0] register
   a) Enable LCD module pins (PEN[:0] bits).
6. LCDBPEN[7:0]
   a) Enable LCD pins to operate as an LCD backplane (BPEN[:0]).
7. LCDC0 register
   a) Enable LCD module (LCDEN bit).

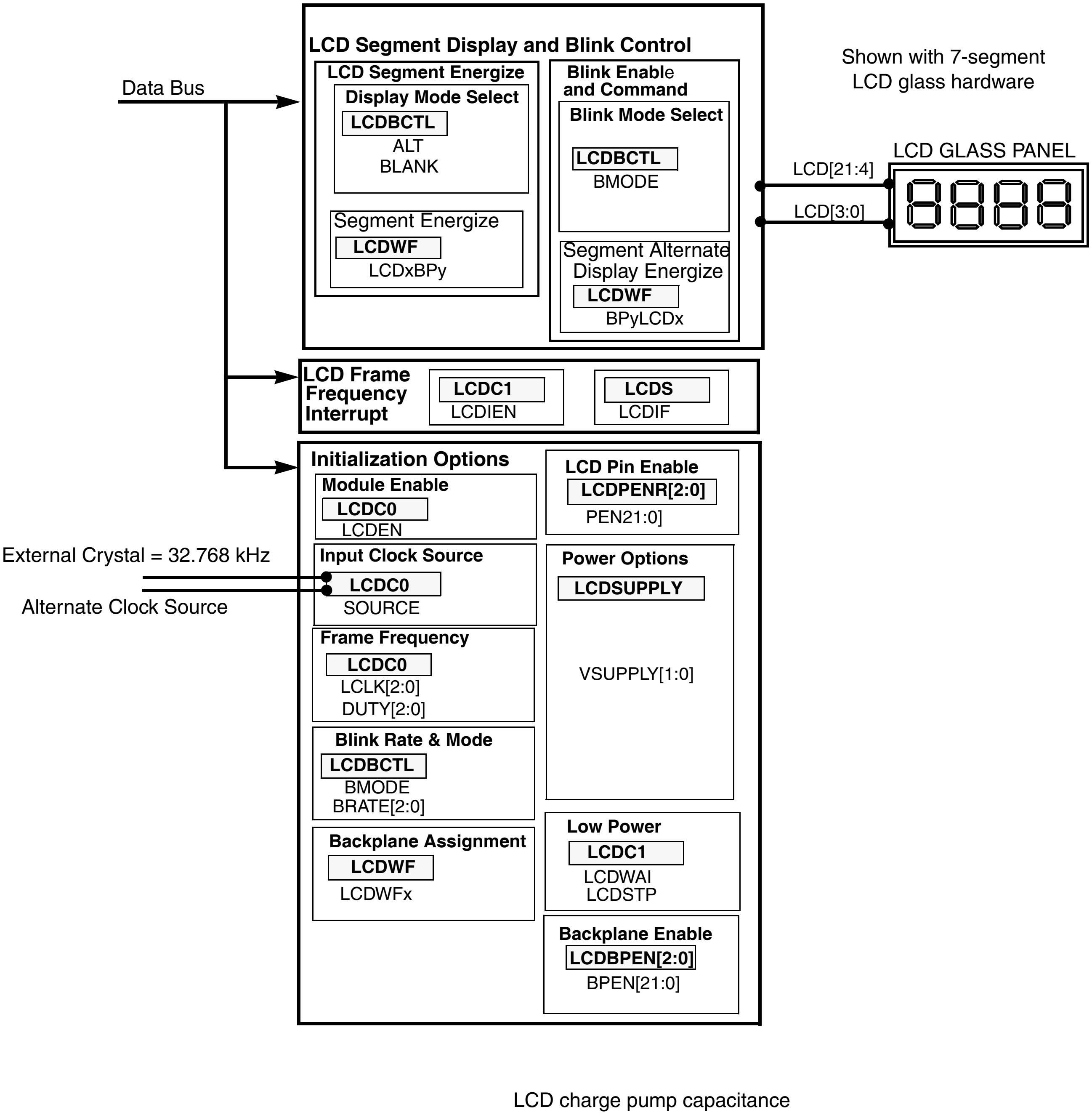


Figure 13-15. LCD Programmer's Model Diagram

Chapter 14
Development Support

14.1 Introduction

Development support systems in the RS08 family include the RS08 background debug controller (BDC).

The BDC provides a single-wire debug interface to the target MCU. This interface provides a convenient means for programming the on-chip flash memory and other nonvolatile memories. Also, the BDC is the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modification, breakpoint, and single-instruction trace commands.

In the RS08 family, address and data bus signals are not available on external pins. Debug is done through commands fed into the target MCU via the single-wire background debug interface, including resetting the device without using a reset pin.

14.1.1 Features

Features of the BDC module include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

Features of the ICE system include:

- Two trigger comparators: Two address + read/write (R/W) or one full address + data + R/W
- Flexible 8-word by 16-bit FIFO (first-in, first-out) buffer for capture information:
  - Change-of-flow addresses or
  - Event-only data
- Two types of breakpoints:
  - Tag breakpoints for instruction opcodes
  - Force breakpoints for any address access
- Nine trigger modes:
  - Basic: A-only, A OR B
  - Sequence: A then B
  - Full: A AND B data, A AND NOT B data
  - Event (store data): Event-only B, A then event-only B
  - Range: Inside range (A ≤ address ≤ B), outside range (address < A or address > B)

14.2 Background Debug Controller (BDC)

All MCUs in the HCS08 Family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.

- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes V_{DD}. An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V_{DD} can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

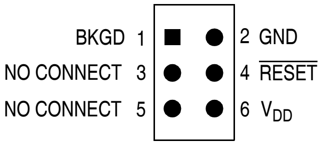


Figure 14-1. BDM Tool Connector

14.2.1 BKGD Pin Description

BKGD is the single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of active background mode commands and data. During reset, this pin is used to select between starting in active background mode or starting the user's application program. This pin is also used to request a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers. This protocol assumes the host knows the communication clock rate that is determined by the target BDC clock rate. All communication is initiated and controlled by the host that drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit first (MSB first). For a detailed description of the communications protocol, refer to Section 14.2.2, "Communication Details."

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed sync response signal from which the host can determine the correct communication speed.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to Section 14.2.2, “Communication Details,” for more detail.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a debug pod is connected to BKGD it is possible to force the MCU into active background mode after reset. The specific conditions for forcing active background depend upon the HCS08 derivative (refer to the introduction to this Development Support section). It is not necessary to reset the target MCU to communicate with it through the background debug interface.

14.2.2 Communication Details

The BDC serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress when this timeout occurs is aborted without affecting the memory or operating mode of the target MCU system.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed.

The clock switch (CLKSW) control bit in the BDC status and control register allows the user to select the BDC clock source. The BDC clock source can either be the bus or the alternate BDC clock source.

The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

Figure 14-2 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target HCS08 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.

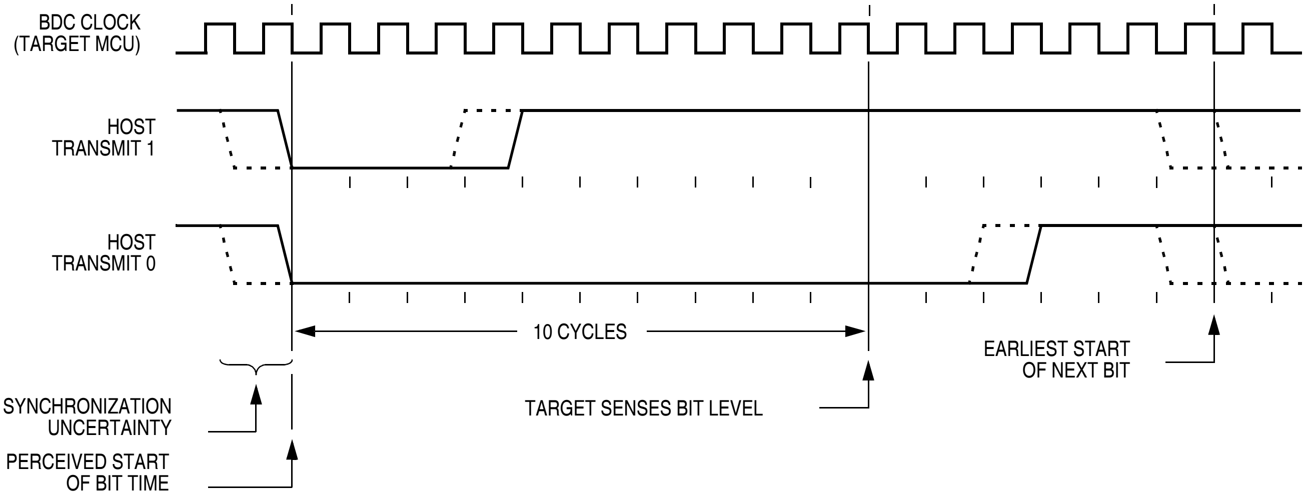


Figure 14-2. BDC Host-to-Target Serial Bit Timing

Figure 14-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

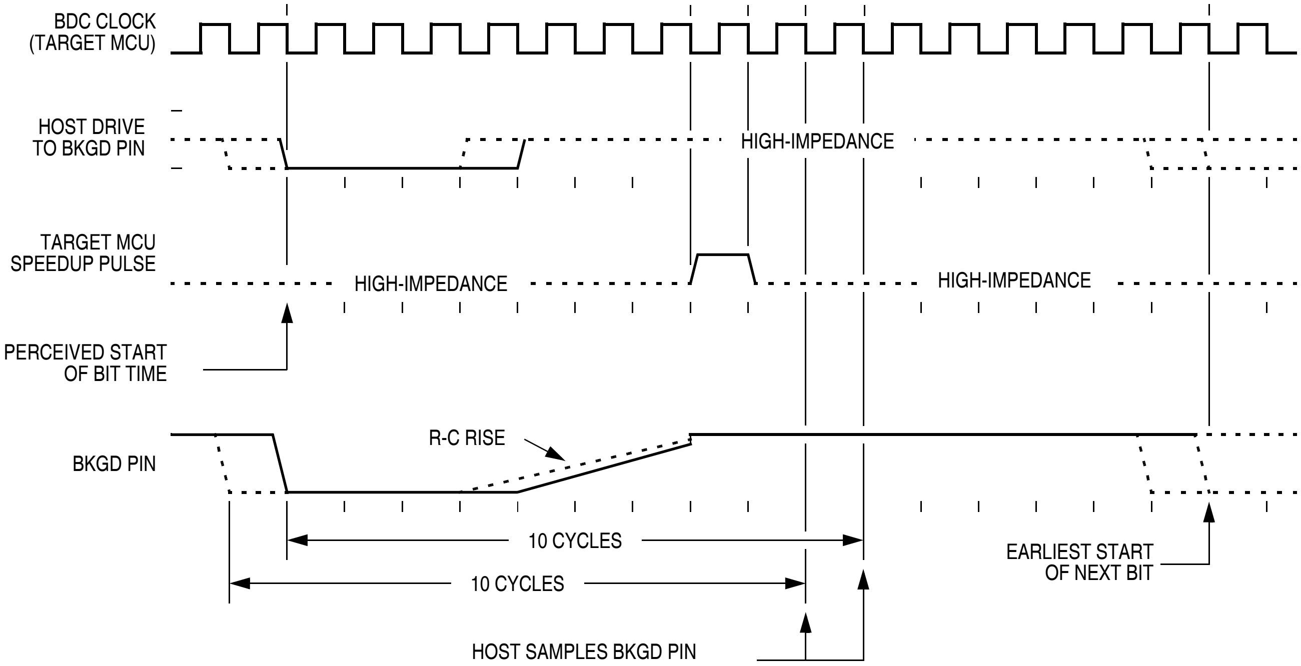


Figure 14-3. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 14-4 shows the host receiving a logic 0 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target HCS08 finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

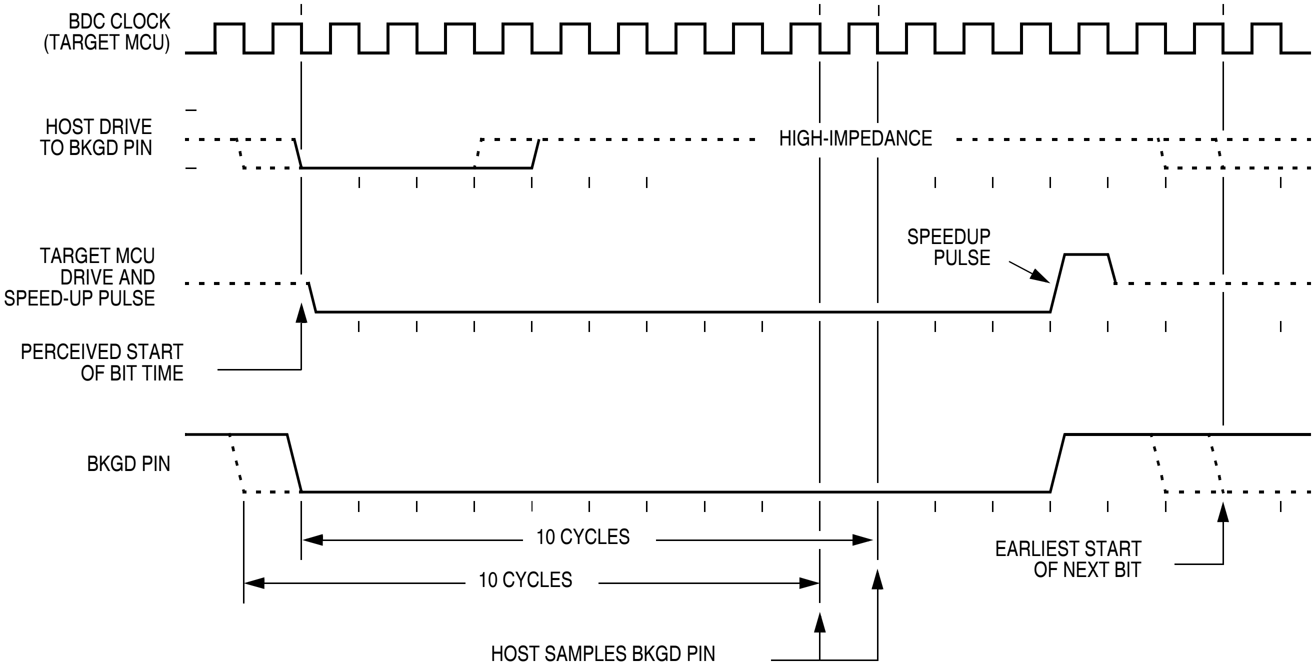


Figure 14-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

14.2.3 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target HCS08 MCU. All commands and data are sent MSB-first using a custom BDC communications protocol. Active background mode commands require that the target MCU is currently in the active background mode while non-intrusive commands may be issued at any time whether the target MCU is in active background mode or running a user application program.

Table 14-1 shows all HCS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

Coding Structure Nomenclature

This nomenclature is used in Table 14-1 to describe the coding structure of the BDC commands.

Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)

/ = separates parts of the command

d = delay 16 target BDC clock cycles

AAAA = a 16-bit address in the host-to-target direction

RD = 8 bits of read data in the target-to-host direction

WD = 8 bits of write data in the host-to-target direction

RD16 = 16 bits of read data in the target-to-host direction

WD16 = 16 bits of write data in the host-to-target direction

SS = the contents of BDCSCR in the target-to-host direction (STATUS)

CC = 8 bits of write data for BDCSCR in the host-to-target direction (CONTROL)

RBKP = 16 bits of read data in the target-to-host direction (from BDCBKPT breakpoint register)

WBKP = 16 bits of write data in the host-to-target direction (for BDCBKPT breakpoint register)

Table 14-1. BDC Command Summary

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a[1]	Request a timed reference pulse to determine target BDC communication speed
ACK_ENABLE	Non-intrusive	D5/d	Enable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D.
ACK_DISABLE	Non-intrusive	D6/d	Disable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D.
BACKGROUND	Non-intrusive	90/d	Enter active background mode if enabled (ignore if ENBDM bit equals 0)
READ_STATUS	Non-intrusive	E4/SS	Read BDC status from BDCSCR
WRITE_CONTROL	Non-intrusive	C4/CC	Write BDC controls in BDCSCR
READ_BYTE	Non-intrusive	E0/AAAA/d/RD	Read a byte from target memory
READ_BYTE_WS	Non-intrusive	E1/AAAA/d/SS/RD	Read a byte and report status
READ_LAST	Non-intrusive	E8/SS/RD	Re-read byte from address just read and report status
WRITE_BYTE	Non-intrusive	C0/AAAA/WD/d	Write a byte to target memory
WRITE_BYTE_WS	Non-intrusive	C1/AAAA/WD/d/SS	Write a byte and report status
READ_BKPT	Non-intrusive	E2/RBKP	Read BDCBKPT breakpoint register
WRITE_BKPT	Non-intrusive	C2/WBKP	Write BDCBKPT breakpoint register
GO	Active BDM	08/d	Go to execute the user application program starting at the address currently in the PC
TRACE1	Active BDM	10/d	Trace 1 user instruction at the address in the PC, then return to active background mode
TAGGO	Active BDM	18/d	Same as GO but enable external tagging (HCS08 devices have no external tagging pin)
READ_A	Active BDM	68/d/RD	Read accumulator (A)
READ_CCR	Active BDM	69/d/RD	Read condition code register (CCR)
READ_PC	Active BDM	6B/d/RD16	Read program counter (PC)
READ_HX	Active BDM	6C/d/RD16	Read H and X register pair (H:X)
READ_SP	Active BDM	6F/d/RD16	Read stack pointer (SP)
READ_NEXT	Active BDM	70/d/RD	Increment H:X by one then read memory byte located at H:X
READ_NEXT_WS	Active BDM	71/d/SS/RD	Increment H:X by one then read memory byte located at H:X. Report status and data.
WRITE_A	Active BDM	48/WD/d	Write accumulator (A)
WRITE_CCR	Active BDM	49/WD/d	Write condition code register (CCR)
WRITE_PC	Active BDM	4B/WD16/d	Write program counter (PC)
WRITE_HX	Active BDM	4C/WD16/d	Write H and X register pair (H:X)
WRITE_SP	Active BDM	4F/WD16/d	Write stack pointer (SP)
WRITE_NEXT	Active BDM	50/WD/d	Increment H:X by one, then write memory byte located at H:X
WRITE_NEXT_WS	Active BDM	51/WD/d/SS	Increment H:X by one, then write memory byte located at H:X. Also report status.

[1] The SYNC command is a special operation that does not have a command code.

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct communications speed to use for BDC communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

The target, upon detecting the SYNC request from the host (which is a much longer low time than would ever occur during normal BDC communications):

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

14.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

The on-chip debug module (DBG) includes circuitry for two additional hardware breakpoints that are more flexible than the simple breakpoint in the BDC module.

14.3 On-Chip Debug System (DBG)

Because HCS08 devices do not have external address and data buses, the most important functions of an in-circuit emulator have been built onto the chip with the MCU. The debug system consists of an 8-stage FIFO that can store address or data bus information, and a flexible trigger system to decide when to capture bus information and what information to capture. The system relies on the single-wire background debug system to access debug control registers and to read results out of the eight stage FIFO.

The debug module includes control and status registers that are accessible in the user’s memory map. These registers are located in the high register space to avoid using valuable direct page memory space.

Most of the debug module’s functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is discussed in greater detail in Section 14.3.6, “Hardware Breakpoints.”

14.3.1 Comparators A and B

Two 16-bit comparators (A and B) can optionally be qualified with the R/W signal and an opcode tracking circuit. Separate control bits allow you to ignore R/W for each comparator. The opcode tracking circuitry optionally allows you to specify that a trigger will occur only if the opcode at the specified address is actually executed as opposed to only being read from memory into the instruction queue. The comparators are also capable of magnitude comparisons to support the inside range and outside range trigger modes. Comparators are disabled temporarily during all BDC accesses.

The A comparator is always associated with the 16-bit CPU address. The B comparator compares to the CPU address or the 8-bit CPU data bus, depending on the trigger mode selected. Because the CPU data bus is separated into a read data bus and a write data bus, the RWAEN and RWA control bits have an additional purpose, in full address plus data comparisons they are used to decide which of these buses to use in the comparator B data bus comparisons. If RWAEN = 1 (enabled) and RWA = 0 (write), the CPU’s write data bus is used. Otherwise, the CPU’s read data bus is used.

The currently selected trigger mode determines what the debugger logic does when a comparator detects a qualified match condition. A match can cause:

- Generation of a breakpoint to the CPU
- Storage of data bus values into the FIFO
- Starting to store change-of-flow addresses into the FIFO (begin type trace)
- Stopping the storage of change-of-flow addresses into the FIFO (end type trace)

14.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, you would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full (CNT = 1:0:0:0), the information is shifted by one position and the host must perform ((8 – CNT) – 1) dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see Section 14.3.5, “Trigger Modes”), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When ARM = 0, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

14.3.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

14.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGT register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

14.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGT register selects one of nine trigger modes. When TRGSEL = 1 in the DBGT register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGT chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGS. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGC.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

A-Only — Trigger when the address matches the value in comparator A

A OR B — Trigger when the address matches either the value in comparator A or the value in comparator B

A Then B — Trigger when the address matches the value in comparator B but only after the address for another cycle matched the value in comparator A. There can be any number of cycles after the A match and before the B match.

A AND B Data (Full Mode) — This is called a full mode because address, data, and R/W (optionally) must match within the same bus cycle to cause a trigger event. Comparator A checks address, the low byte of comparator B checks data, and R/W is checked against RWA if RWAEN = 1. The high-order half of comparator B is not used.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

A AND NOT B Data (Full Mode) — Address must match comparator A, data must not match the low half of comparator B, and R/W must match RWA if RWAEN = 1. All three conditions must be met within the same bus cycle to cause a trigger.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

Event-Only B (Store Data) — Trigger events occur each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

A Then Event-Only B (Store Data) — After the address has matched the value in comparator A, a trigger event occurs each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

Inside Range (A ≤ Address ≤ B) — A trigger occurs when the address is greater than or equal to the value in comparator A and less than or equal to the value in comparator B at the same time.

Outside Range (Address < A or Address > B) — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

14.3.6 Hardware Breakpoints

The BRKEN control bit in the DBGC register may be set to 1 to allow any of the trigger conditions described in Section 14.3.5, “Trigger Modes,” to be used to generate a hardware breakpoint request to the CPU. TAG in DBGC controls whether the breakpoint request will be treated as a tag-type breakpoint or a force-type breakpoint. A tag breakpoint causes the current opcode to be marked as it enters the instruction queue. If a tagged opcode reaches the end of the pipe, the CPU executes a BGND instruction to go to active background mode rather than executing the tagged opcode. A force-type breakpoint causes the CPU to finish the current instruction and then go to active background mode.

If the background mode has not been enabled (ENBDM = 1) by a serial WRITE_CONTROL command through the BKGD pin, the CPU will execute an SWI instruction instead of going to active background mode.

14.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

Refer to the high-page register summary in the device overview chapter of this data sheet for the absolute address assignments for all DBG registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

14.4.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

14.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ_STATUS and WRITE_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
W		▨				▨	▨	▨
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	1	0	0	0

▨ = Unimplemented or Reserved

Figure 14-5. BDC Status and Control Register (BDCSCR)

Table 14-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	**Enable BDM (Permit Active Background Mode)** — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 0 BDM cannot be made active (non-intrusive commands still allowed) 1 BDM can be made active to allow active background mode commands
6 BDMACT	**Background Mode Active Status** — This is a read-only status bit. 0 BDM not active (user application program running) 1 BDM active and waiting for serial commands
5 BKPTEN	**BDC Breakpoint Enable** — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored. 0 BDC breakpoint disabled 1 BDC breakpoint enabled
4 FTS	**Force/Tag Select** — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction 1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode)
3 CLKSW	**Select Source for BDC Communications Clock** — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

Table 14-2. BDCSCR Register Field Descriptions (continued)

Field	Description
2 WS	**Wait or Stop Status** — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands. 0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active) 1 Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode
1 WSF	**Wait or Stop Failure Status** — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.) 0 Memory access did not conflict with a wait or stop instruction 1 Memory access command failed because the CPU entered wait or stop mode
0 DVF	**Data Valid Failure Status** — This status bit is not used in the MC9RS08LE4 Series because it does not have any slow access memory. 0 Memory access did not conflict with a slow memory access 1 Memory access command failed because CPU was not finished with a slow memory access

14.4.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to Section 14.2.4, “BDC Hardware Breakpoint.”

14.4.2 System Background Debug Force Reset Register (SBDFR)

This register contains a single write-only control bit. A serial background mode command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	▨	▨	▨	▨	▨	▨	▨	BDFR[1]
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

[1] BDFR is writable only through serial background mode debug commands, not from user programs.

Figure 14-6. System Background Debug Force Reset Register (SBDFR)

Table 14-3. SBDFR Register Field Description

Field	Description
0 BDFR	**Background Debug Force Reset** — A serial active background mode command such as WRITE_BYTE allows an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program.

14.4.3 DBG Registers and Control Bits

The debug module includes nine bytes of register space for three 16-bit registers and three 8-bit control and status registers. These registers are located in the high register space of the normal memory map so they are accessible to normal application programs. These registers are rarely if ever accessed by normal user application programs with the possible exception of a ROM patching mechanism that uses the breakpoint logic.

14.4.3.1 Debug Comparator A High Register (DBGCAH)

This register contains compare value bits for the high-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

14.4.3.2 Debug Comparator A Low Register (DBGCAL)

This register contains compare value bits for the low-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

14.4.3.3 Debug Comparator B High Register (DBGCBH)

This register contains compare value bits for the high-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

14.4.3.4 Debug Comparator B Low Register (DBGCBL)

This register contains compare value bits for the low-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

14.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

14.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It isn't necessary to read DBGFH in this case.

Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.

14.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R W	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
Reset	0	0	0	0	0	0	0	0

Figure 14-7. Debug Control Register (DBGC)

Table 14-4. DBGC Register Field Descriptions

Field	Description
7 DBGEN	**Debug Module Enable** — Used to enable the debug module. DBGEN cannot be set to 1 if the MCU is secure. 0 DBG disabled 1 DBG enabled
6 ARM	**Arm Control** — Controls whether the debugger is comparing and storing information in the FIFO. A write is used to set this bit (and ARMF) and completion of a debug run automatically clears it. Any debug run can be manually stopped by writing 0 to ARM or to DBGEN. 0 Debugger not armed 1 Debugger armed
5 TAG	**Tag/Force Select** — Controls whether break requests to the CPU will be tag or force type requests. If BRKEN = 0, this bit has no meaning or effect. 0 CPU breaks requested as force type requests 1 CPU breaks requested as tag type requests
4 BRKEN	**Break Enable** — Controls whether a trigger event will generate a break request to the CPU. Trigger events can cause information to be stored in the FIFO without generating a break request to the CPU. For an end trace, CPU break requests are issued to the CPU when the comparator(s) and R/W meet the trigger requirements. For a begin trace, CPU break requests are issued when the FIFO becomes full. TRGSEL does not affect the timing of CPU break requests. 0 CPU break requests not enabled 1 Triggers cause a break request to the CPU
3 RWA	**R/W Comparison Value for Comparator A** — When RWAEN = 1, this bit determines whether a read or a write access qualifies comparator A. When RWAEN = 0, RWA and the R/W signal do not affect comparator A. 0 Comparator A can only match on a write cycle 1 Comparator A can only match on a read cycle
2 RWAEN	**Enable R/W for Comparator A** — Controls whether the level of R/W is considered for a comparator A match. 0 R/W is not used in comparison A 1 R/W is used in comparison A
1 RWB	**R/W Comparison Value for Comparator B** — When RWBEN = 1, this bit determines whether a read or a write access qualifies comparator B. When RWBEN = 0, RWB and the R/W signal do not affect comparator B. 0 Comparator B can match only on a write cycle 1 Comparator B can match only on a read cycle
0 RWBEN	**Enable R/W for Comparator B** — Controls whether the level of R/W is considered for a comparator B match. 0 R/W is not used in comparison B 1 R/W is used in comparison B

14.4.3.8 Debug Trigger Register (DBGT)

This register can be read any time, but may be written only if ARM = 0, except bits 4 and 5 are hard-wired to 0s.

	7	6	5	4	3	2	1	0
R	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
W			▨	▨				
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 14-8. Debug Trigger Register (DBGT)

Table 14-5. DBGT Register Field Descriptions

Field	Description
7 TRGSEL	**Trigger Type** — Controls whether the match outputs from comparators A and B are qualified with the opcode tracking logic in the debug module. If TRGSEL is set, a match signal from comparator A or B must propagate through the opcode tracking logic and a trigger event is only signalled to the FIFO logic if the opcode at the match address is actually executed. 0 Trigger on access to compare address (force) 1 Trigger if opcode at compare address is executed (tag)
6 BEGIN	**Begin/End Trigger Select** — Controls whether the FIFO starts filling at a trigger or fills in a circular manner until a trigger ends the capture of information. In event-only trigger modes, this bit is ignored and all debug runs are assumed to be begin traces. 0 Data stored in FIFO until trigger (end trace) 1 Trigger initiates data storage (begin trace)
3:0 TRG[3:0]	**Select Trigger Mode** — Selects one of nine triggering modes, as described below. 0000 A-only 0001 A OR B 0010 A Then B 0011 Event-only B (store data) 0100 A then event-only B (store data) 0101 A AND B data (full mode) 0110 A AND NOT B data (full mode) 0111 Inside range: A ≤ address ≤ B 1000 Outside range: address < A or address > B 1001 – 1111 (No trigger)

14.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

	7	6	5	4	3	2	1	0
R	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
W	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 14-9. Debug Status Register (DBGS)

Table 14-6. DBGS Register Field Descriptions

Field	Description
7 AF	**Trigger Match A Flag** — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming. 0 Comparator A has not matched 1 Comparator A match
6 BF	**Trigger Match B Flag** — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming. 0 Comparator B has not matched 1 Comparator B match
5 ARMF	**Arm Flag** — While DBGEN = 1, this status bit is a read-only image of ARM in DBGC. This bit is set by writing 1 to the ARM control bit in DBGC (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBGC. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	**FIFO Valid Count** — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. Freescale reserves the right to make changes without further notice to any products herein.

Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

freescale™