

Welcome to [E-XFL.COM](https://www.e-xfl.com)

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

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

Applications of "[Embedded - Microcontrollers](#)"

Details

Product Status	Active
Core Processor	RS08
Core Size	8-Bit
Speed	10MHz
Connectivity	-
Peripherals	LVD, POR, WDT
Number of I/O	4
Program Memory Size	1KB (1K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	63 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	8-SOIC (0.154", 3.90mm Width)
Supplier Device Package	8-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9rs08ka1cscr

Section Number	Title	Page
----------------	-------	------

Chapter 11

Modulo Timer (RS08MTIMV1)

11.1	Introduction	89
11.1.1	Features	90
11.1.2	Modes of Operation	90
11.1.2.1	Operation in Wait Mode	90
11.1.2.2	Operation in Stop Modes	90
11.1.2.3	Operation in Active Background Mode	90
11.1.3	Block Diagram	91
11.2	External Signal Description	91
11.3	Register Definition	91
11.3.1	MTIM Status and Control Register (MTIMSC)	92
11.3.2	MTIM Clock Configuration Register (MTIMCLK)	93
11.3.3	MTIM Counter Register (MTIMCNT)	93
11.3.4	MTIM Modulo Register (MTIMMOD)	94
11.4	Functional Description	95
11.4.1	MTIM Operation Example	96

Chapter 12

Development Support

12.1	Introduction	97
12.2	Features	97
12.3	RS08 Background Debug Controller (BDC)	98
12.3.1	BKGD Pin Description	99
12.3.2	Communication Details	99
12.3.3	SYNC and Serial Communication Timeout	102
12.4	BDC Registers and Control Bits	103
12.4.1	BDC Status and Control Register (BDCSCR)	103
12.4.2	BDC Breakpoint Match Register	104
12.5	RS08 BDC Commands	105

the IREFSTEN bit. For the ICS to run in stop, the LVDE and LVDSE bits in the SPMSC1 must both be set before entering stop.

3.6.1 Active BDM Enabled in Stop Mode

Entry into active background mode from run mode is enabled if the ENBDM bit in BDCSCR is set. This register is described in the [Development Support](#) chapter of this data sheet. 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 so background debug communication is still possible. In addition, the voltage regulator does not enter its low-power standby state; it 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 either 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 active background mode is entered, all background commands are available.

[Table 3-3](#) summarizes the behavior of the MCU in stop when entry into the active background mode is enabled.

Table 3-3. BDM Enabled Stop Mode Behavior

Mode	CPU	Digital Peripherals	ICS	ACMP	Regulator	I/O Pins	RTI
Stop	Standby	Standby	On	Optionally on	On	States held	Optionally on

3.6.2 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop (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 when LVD reset is enabled.

Table 3-4. LVD Enabled Stop Mode Behavior

Mode	CPU	Digital Peripherals	ICS	ACMP	Regulator	I/O Pins	RTI
Stop	Standby	Standby	Optionally on	Optionally on	On	States held	Optionally on

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$0000– \$000D		Fast Access RAM							
\$000E	D[X] ¹	Bit 7	6	5	4	3	2	1	Bit 0
\$000F	X	Bit 7	6	5	4	3	2	1	Bit 0
\$0010	PTAD	0	0	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
\$0011	PTADD	0	0	PTADD5	PTADD4	0	0	PTADD1	PTADD0
\$0012	Unimplemented	—	—	—	—	—	—	—	—
\$0013	ACMPSC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD	
\$0014	ICSC1	0	CLKS	0	0	0	0	0	IREFSTEN
\$0015	ICSC2	BDIV		0	0	LP	0	0	0
\$0016	ICSTRM	TRIM							
\$0017	ICSSC	0	0	0	0	0	CLKST	0	FTRIM
\$0018	MTIMSC	TOF	TOIE	TRST	TSTP	0	0	0	0
\$0019	MTIMCLK	0	0	CLKS			PS		
\$001A	MTIMCNT	COUNT							
\$001B	MTIMMOD	MOD							
\$001C	KBISC	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
\$001D	KBIPE	—	—	KBIPE5	KBIPE4	—	KBIPE2	KBIPE1	KBIPE0
\$001E	KBIES	—	—	KBEDG5	KBEDG4	—	KBEDG2	KBEDG1	KBEDG0
\$001F	PAGESEL	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6
\$0020– \$004F		RAM							
\$0050– \$00BF	Unimplemented	—	—	—	—	—	—	—	—
\$00C0– \$00FF		Paging Window							
\$0100– \$01FF	Unimplemented	—	—	—	—	—	—	—	—
\$0200	SRS	POR	PIN	COP	ILOP	ILAD	0	LVD	0
\$0201	SOPT	COPE	COPT	STOPE	0	0	0	BKGDPE	RSTPE
\$0202	SIP1	—	—	—	KBI	ACMP	MTIM	RTI	LVD
\$0203	Unimplemented	—	—	—	—	—	—	—	—
\$0204	Reserved	—	—	—	—	—	—	—	—
\$0205	Unimplemented	—	—	—	—	—	—	—	—
\$0206	SDIDH	REV3	REV2	REV1	REV0	ID			
\$0207	SDIDL	ID							
\$0208	SRTISC	RTIF	RTIACK	RTICLKS	RTIE	0	RTIS		
\$0209	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
\$020A	Reserved	—	—	—	—	—	—	—	—
\$020B	Reserved	—	—	—	—	—	—	—	—

— = Unimplemented or Reserved

Chapter 5

Resets, Interrupts, and General System Control

5.1 Introduction

This chapter discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt in the MC9RS08KA2 Series. Some interrupt sources from peripheral modules are discussed in greater detail within other chapters of this data sheet. This chapter gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and wakeup sources, including the computer operating properly (COP) watchdog and real-time interrupt (RTI), are not part of on-chip peripheral systems with their own chapters but are part of the system control logic.

5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- System reset status register (SRS) to indicate the source of the most recent reset
- System interrupt pending register (SIP1) to indicate the status of pending system interrupts
 - Analog comparator interrupt with enable
 - Keyboard interrupt with enable
 - Low-voltage detect interrupt with enable
 - Modulo timer interrupt with enable
 - Real-time interrupt with enable; available in stop with multiple rates based on a separate 1-kHz self-clocked source

5.3 MCU Reset

Resetting the MCU provides a way to start processing from a 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 MC9RS08KA2 Series has seven sources for reset:

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

Chapter 6

Parallel Input/Output Control

This section explains software controls related to parallel input/output (I/O) and pin control. The MC9RS08KA2 Series has one parallel I/O port, which includes two I/O pins in the 6-pin package or four I/O pins in the 8-pin packages, one output-only pin, and one input-only pin. See [Chapter 2, “Pins and Connections,”](#) for more information about pin assignments and external hardware considerations for 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 ($PTADDn = 0$) with pullup/pulldown devices disabled ($PTAPEn = 0$), except for output-only pin PTA3, 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 the block diagram shown in [Figure 6-1](#).

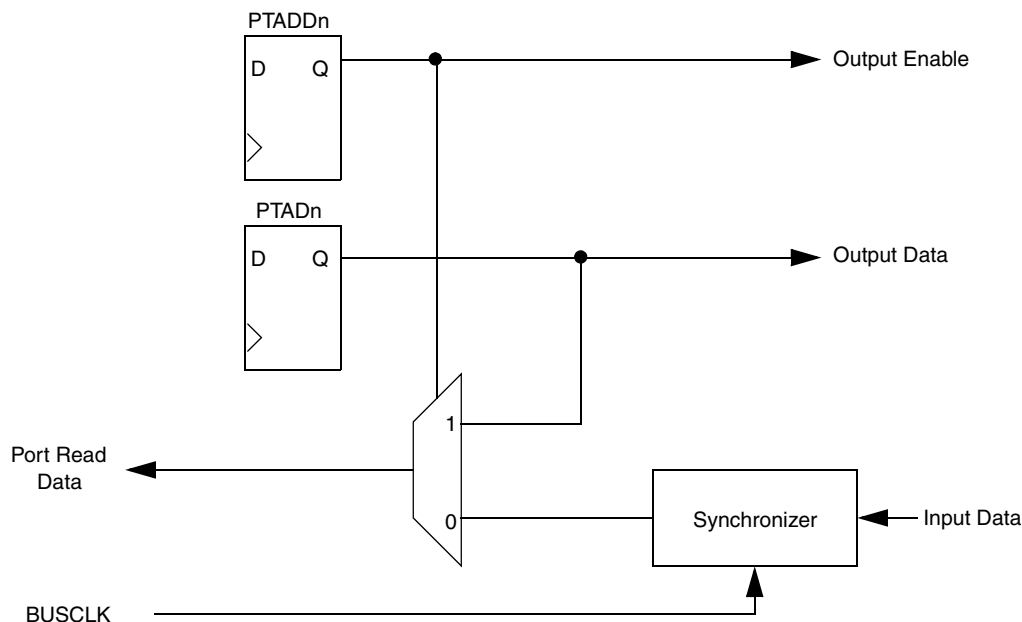


Figure 6-1. Parallel I/O Block Diagram

The data direction control bit ($PTADDn$) determines whether the output buffer for the associated pin is enabled, and also controls the source for port data register reads. The input buffer for the associated pin is always enabled unless the pin is enabled as an analog function or is an output-only pin.

Chapter 7

Keyboard Interrupt (RS08KBIV1)

7.1 Introduction

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

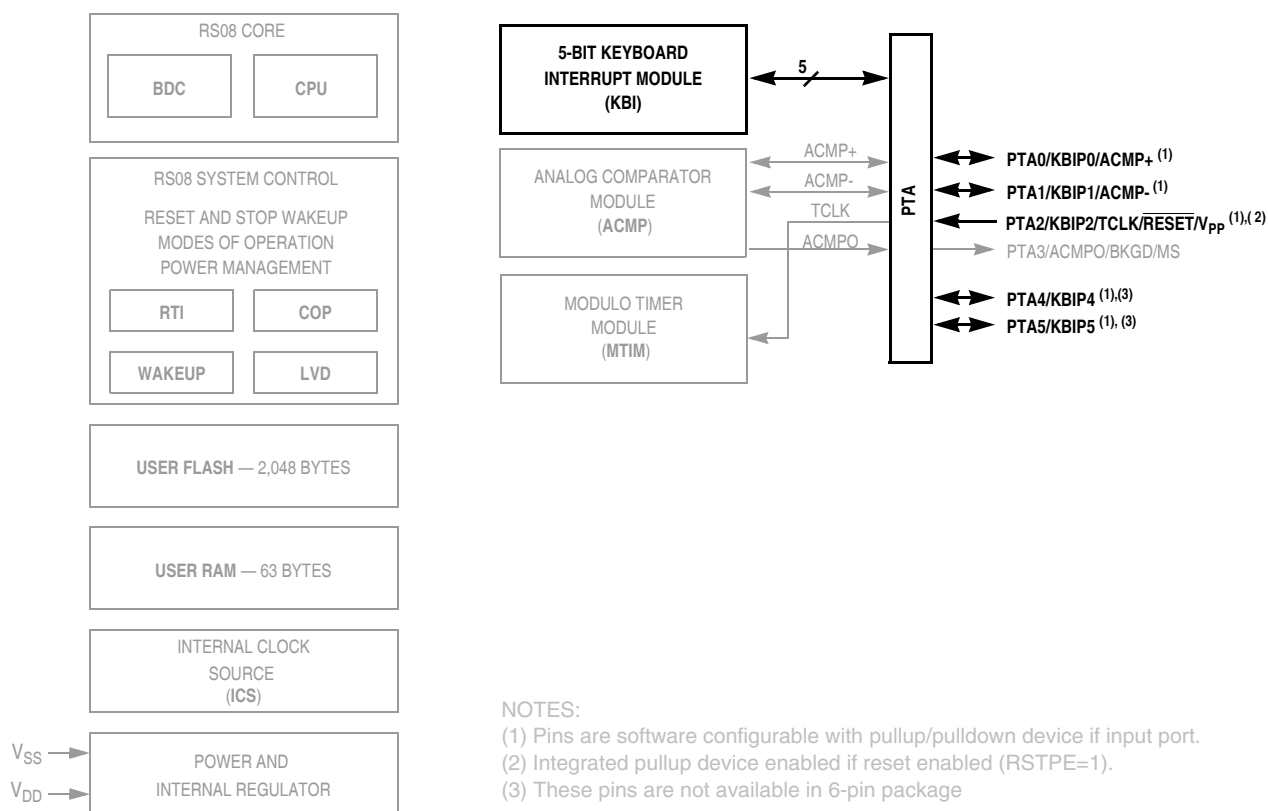


Figure 7-1. MC9RS08KA2 Series Block Diagram with KBI Block and Pins Highlighted

7.1.1 Features

The KBI features include:

- Each keyboard interrupt pin has an individual pin enable bit
- 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

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.

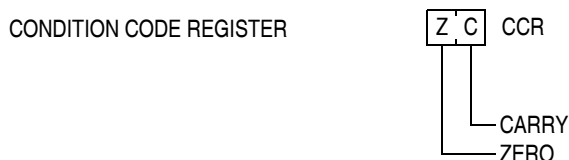


Figure 8-3. Condition Code Register (CCR)

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:    decb         ;do this if A not higher than or same as 5
lower:

```


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 will 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 will 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.

9.1.1 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
 - DCO output is 512 times internal reference frequency
- Internal reference clock has 9 trim bits available
- Internal reference clock 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, and 8)
- FLL engaged internal mode is automatically selected out of reset

9.1.2 Modes of Operation

There are four modes of operation for the ICS: FEI, FBI, FBILP, and stop.

9.1.2.1 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.

9.1.2.2 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.

9.1.2.3 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.

9.1.2.4 Stop (STOP)

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

9.1.3 Block Diagram

Figure 9-2 shows the ICS block diagram.

9.3.3 ICS Trim Register (ICSTRM)

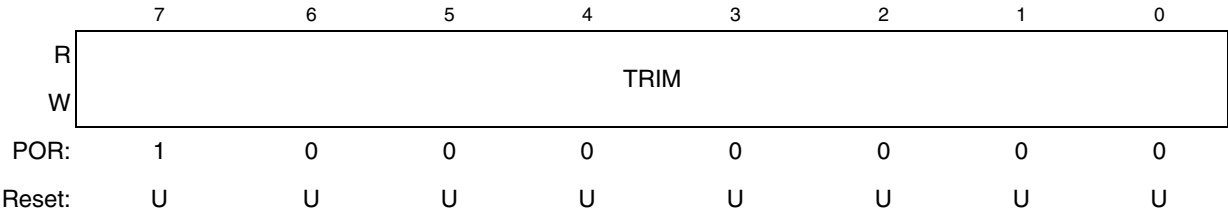


Figure 9-5. ICS Trim Register (ICSTRM)

Table 9-4. ICSTRM Field Descriptions

Field	Description
7:0 TRIM	<p>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.</p> <p>An additional fine trim bit is available in ICSSC as the FTRIM bit.</p>

9.3.4 ICS Status and Control (ICSSC)

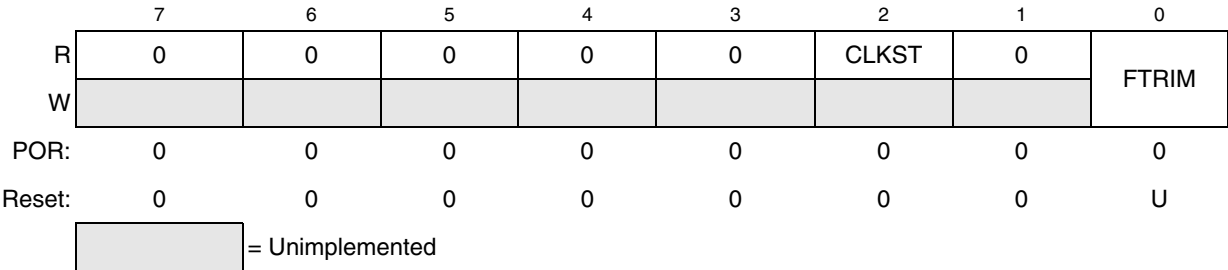


Figure 9-6. ICS Status and Control Register (ICSSC)

Table 9-5. ICSSC Field Descriptions

Field	Description
2 CLKST	<p>Clock Mode Status — The CLKST read-only bit indicate the current clock mode. The CLKST bit does not update immediately after a write to the CLKS bit due to internal synchronization between clock domains.</p> <p>0 Output of FLL is selected</p> <p>1 Internal reference clock is selected</p>
0 FTRIM	<p>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.</p>

10.1.1 Features

The ACMP has the following features:

- Full rail-to-rail supply operation
- Less than 40 mV of input offset
- Less than 15 mV of hysteresis
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output
- Option to compare to fixed internal bandgap reference voltage
- Option to allow comparator output to be visible on a pin, ACMPO
- Remains operational in stop mode

10.1.2 Modes of Operation

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

10.1.2.1 Operation in Wait Mode

The ACMP continues to operate in wait mode if enabled before executing the WAIT instruction. Therefore, the ACMP can be used to bring the MCU out of wait mode if the ACMP interrupt is enabled (ACIE = 1). For lowest possible current consumption, the ACMP should be disabled by software if not required as an interrupt source during wait mode.

10.1.2.2 Operation in Stop Mode

The ACMP continues to operate in stop mode if enabled and compare operation remains active. If ACOPE is enabled, comparator output operates as in the normal operating mode and comparator output is placed onto the external pin. The MCU is brought out of stop when a compare event occurs and ACIE is enabled; ACF flag sets accordingly.

If stop is exited with a reset, the ACMP will be put into its reset state.

10.1.2.3 Operation in Active Background Mode

When the MCU is in active background mode, the ACMP will continue to operate normally.

10.1.3 Block Diagram

The block diagram for the analog comparator module is shown in [Figure 10-2](#).

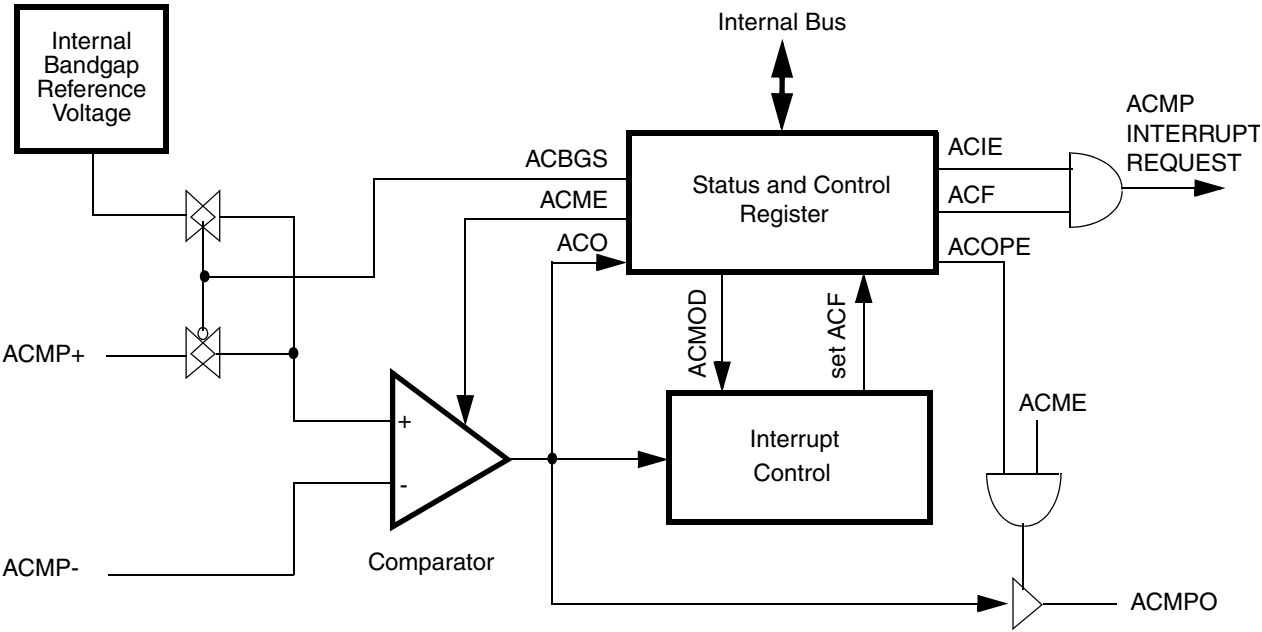


Figure 10-2. Analog Comparator (ACMP) Block Diagram

- BDC_RESET command allows host to reset MCU without using a reset pin
- One hardware address breakpoint built into BDC
- RS08 clock source runs in stop mode if BDM enabled to allow debugging when CPU is in stop mode
- COP watchdog suspended while in active background mode

12.3 RS08 Background Debug Controller (BDC)

All MCUs in the RS08 Family contain a single-wire background debug interface which supports in-circuit programming of on-chip non-volatile memory and sophisticated debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this debug system provides for minimal interference with normal application resources. It does not use any user memory or locations in the memory map. It requires use of only the output-only BKGD pin. This pin will be shared with simple user output-only functions (typically port, comparator outputs, etc.), which can be easily debugged in normal user mode.

RS08 BDM 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). The BACKGROUND command causes the target MCU to enter active background mode. Active background mode commands allow the CPU registers to be read or written and allow the user to trace one (TRACE1) 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 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 (BDC).

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 communication such as Ethernet or a universal serial bus (USB) to communicate between the host PC and the pod.

Figure 12-2 shows the standard header for connection of a RS08 BDM pod. A pod is a small interface device that connects a host computer such as a personal computer to a target RS08 system. BKGD and GND are the minimum connections required to communicate with a target MCU. The pseudo-open-drain RESET signal is included in the connector to allow a direct hardware method for the host to force or monitor (if RESET is available as output) a target system reset.

The RS08 BDM pods supply the V_{PP} voltage to the RS08 MCU when in-circuit programming is required. The V_{PP} connection from the pod is shared with RESET as shown in Figure 12-2. For V_{PP} requirements see the FLASH specifications in the electricals appendix.

- Subsequent bits must occur within 512 BDC cycles of the last bit sent.

12.4 BDC Registers and Control Bits

The BDC contains two non-CPU accessible 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 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 status bits (BDMACT, WS, and WSF) are read-only status indicators and can never be written by the WRITE_CONTROL serial BDC command.

12.4.1 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	0	WS	WSF	0
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	0	0	0	0

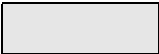
 = Unimplemented or Reserved

Figure 12-6. BDC Status and Control Register (BDCSCR)

Table 12-1. 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. If the application can go into stop mode, this bit is required to be set if debugging capabilities are required. 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.

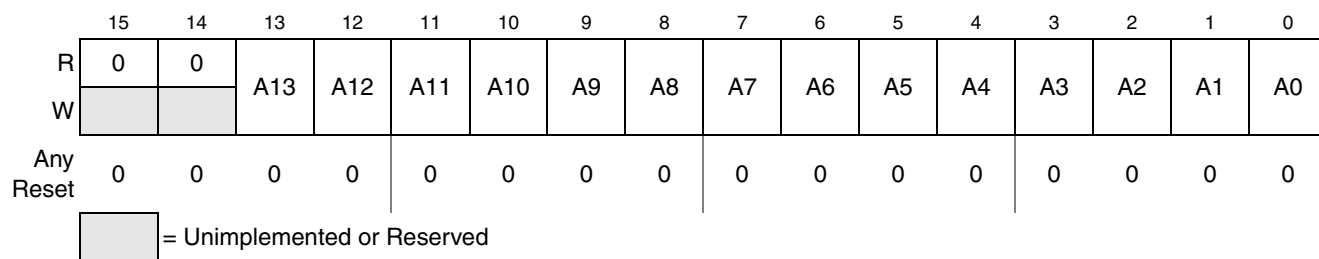


Figure 12-7. BDC Breakpoint Match Register (BDCBKPT)

12.5 RS08 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target 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 12-2 shows all RS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

Coding Structure Nomenclature

The following nomenclature is used in Table 12-2 to describe the coding structure of the BDC commands.

Commands begin with an 8-bit command code in the host-to-target direction (most significant bit first)

/ = Separates parts of the command

d = Delay 16 to 511 target BDC clock cycles

soft-reset = Delay of at least 512 BDC clock cycles from last host falling-edge

AAAA = 16-bit address in the host-to-target direction¹

RD = Eight bits of read data in the target-to-host direction

WD = Eight 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 = Eight 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)

1. The RS08 CPU uses only 14 bits of address and occupies the lower 14 bits of the 16-bit AAAA address field. The values of address bits 15 and 14 in AAAA are truncated and thus do not matter.

Table 12-2. RS08 BDC Command Summary (continued)

Command Mnemonic	Active Background Mode/ Non-Intrusive	Coding Structure	Description
WRITE_A	Active background mode	48/WD/d	Write accumulator (A)
READ_CCR_PC	Active background mode	6B/d/RD16 ⁵	Read the CCR bits z, c concatenated with the 14-bit program counter (PC) RD16=zc:PC
WRITE_CCR_PC	Active background mode	4B/WD16/d ⁶	Write the CCR bits z, c concatenated with the 14-bit program counter (PC) WD16=zc:PC
READ_SPC	Active background mode	6F/d/RD16 ⁷	Read the 14-bit shadow program counter (SPC) RD16=0:0:SPC
WRITE_SPC	Active background mode	4F/WD16/d ⁸	Write 14-bit shadow program counter (SPC) WD16 = x:x:SPC, the two most significant bits shown by “x” are ignored by target

¹ The SYNC command is a special operation which does not have a command code.

² 18 was HCS08 BDC command for TAGGO.

³ Each RD requires a delay between host read data byte and next read, command ends when target detects a soft-reset.

⁴ Each WD requires a delay between host write data byte and next byte, command ends when target detects a soft-reset.

⁵ HCS08 BDC had separate READ_CCR and READ_PC commands, the RS08 BDC combined this commands.

⁶ HCS08 BDC had separate WRITE_CCR and WRITE_PC commands, the RS08 BDC combined this commands.

⁷ 6F is READ_SP (read stack pointer) for HCS08 BDC.

⁸ 4F is WRITE_SP (write stack pointer) for HCS08 BDC.



Typical IOH vs VDD-VOH at VDD=5V

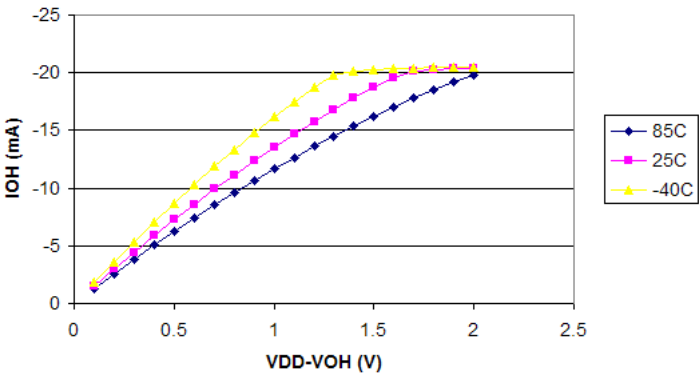


Figure 12-8. Typical IOH vs. VDD-VOH
VDD = 5 V

Typical IOH vs VDD-VOH at VDD=3V

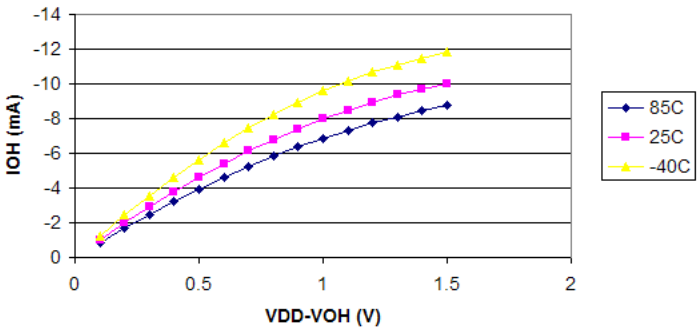


Figure 12-9. Typical IOH vs. VDD-VOH
VDD = 3 V

Typical IOH vs VDD-VOH at VDD=1.8V

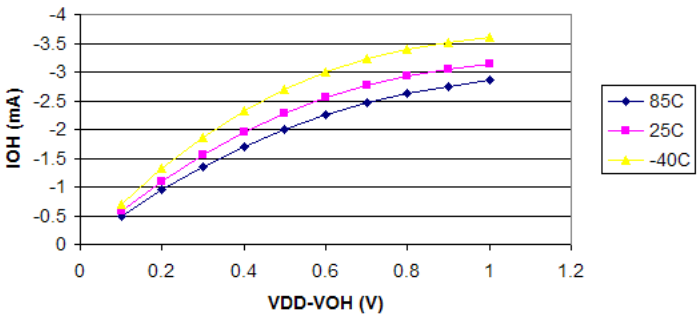


Figure 12-10. Typical IOH vs. VDD-VOH
VDD = 1.8 V

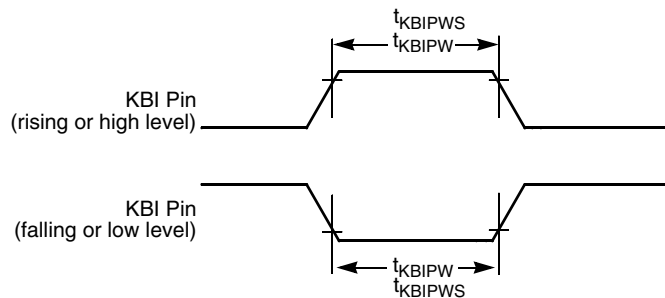


Figure A-2. KBI Pulse Width

A.10 FLASH Specifications

This section provides details about program/erase times and program-erase endurance for the FLASH memory.

For detailed information about program/erase operations, see [Chapter 4, “Memory.”](#)

Table A-9. FLASH Characteristics

Characteristic	Symbol	Min	Typical ¹	Max	Unit
Supply voltage for program/erase	V_{DD}	2.7	—	5.5	V
Program/Erase voltage	V_{PP}	11.8	12	12.2	V
V_{PP} current					
Program	I_{VPP_prog}	—	—	200	μA
Mass erase	I_{VPP_erase}	—	—	100	μA
Supply voltage for read operation $0 < f_{BUS} < 10$ MHz	V_{Read}	1.8	—	5.5	V
Byte program time	t_{prog}	20	—	40	μs
Mass erase time	t_{me}	500	—	—	ms
Cumulative program HV time ²	t_{hv}	—	—	8	ms
Total cumulative HV time (total of t_{me} & t_{hv} applied to device)	t_{hv_total}	—	—	2	hours
HVEN to program setup time	t_{pgs}	10	—	—	μs
PGM/MASS to HVEN setup time	t_{nvs}	5	—	—	μs
HVEN hold time for PGM	t_{nvh}	5	—	—	μs
HVEN hold time for MASS	t_{nvh1}	100	—	—	μs
V_{PP} to PGM/MASS setup time	t_{vps}	20	—	—	ns
HVEN to V_{PP} hold time	t_{vph}	20	—	—	ns
V_{PP} rise time ³	t_{vrs}	200	—	—	ns
Recovery time	t_{rcv}	1	—	—	μs
Program/erase endurance T_L to $T_H = -40^\circ C$ to $+85^\circ C$	—	1000	—	—	cycles
Data retention	t_{D_ret}	15	100	—	years

¹ Typicals are measured at 25°C.

² t_{hv} is the cumulative high voltage programming time to the same row before next erase. Same address can not be programmed more than twice before next erase.

³ Fast V_{PP} rise time may potentially trigger the ESD protection structure, which may result in over current flowing into the pad and cause permanent damage to the pad. External filtering for the V_{PP} power source is recommended. An example VPP filter is shown in [Figure A-3](#).