

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.

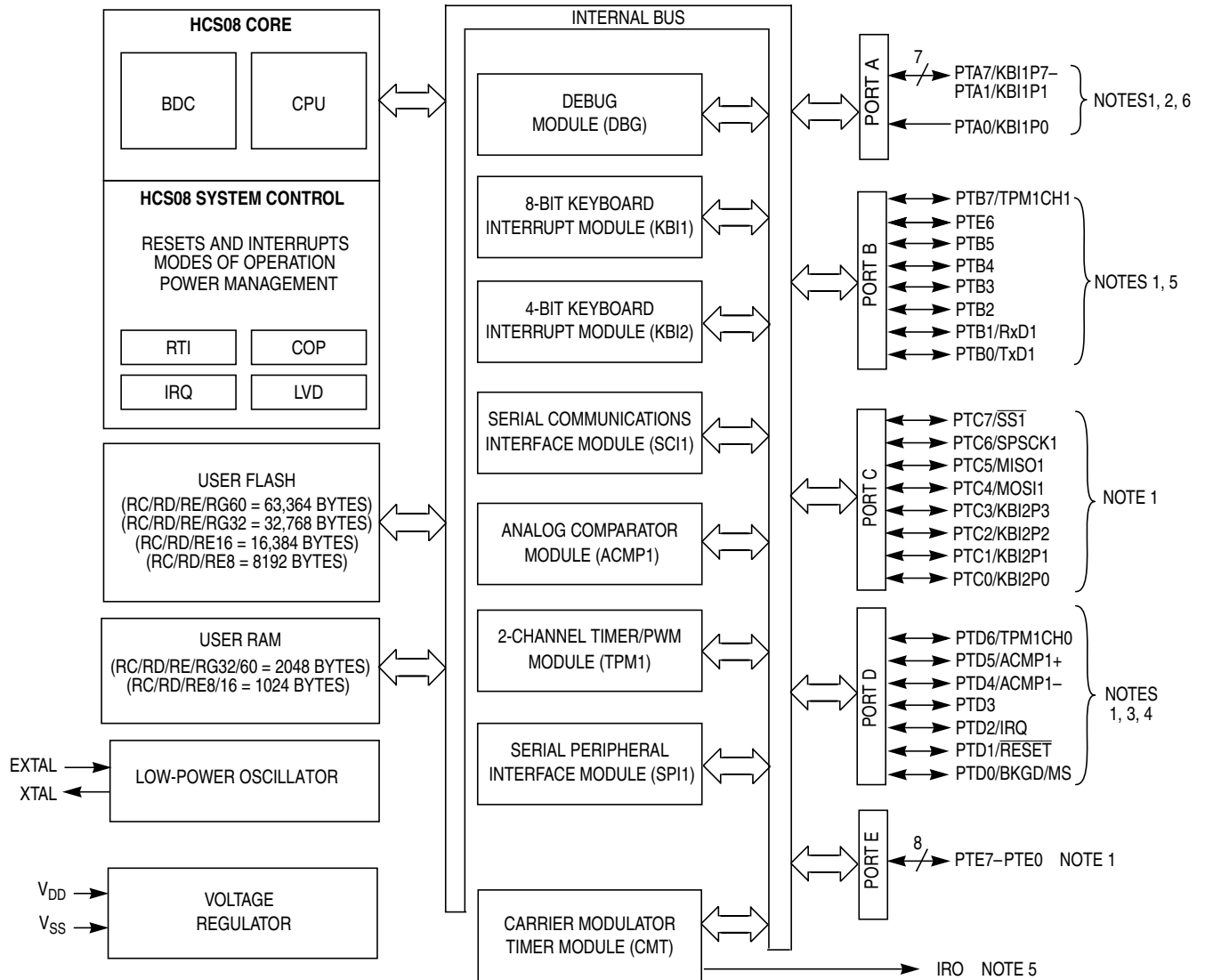
Applications of "[Embedded - Microcontrollers](#)"

Details

Product Status	Obsolete
Core Processor	S08
Core Size	8-Bit
Speed	8MHz
Connectivity	SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	32-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08rg32cfjer

1.3 MCU Block Diagram

This block diagram shows the structure of the MC9S08RC/RD/RE/RG MCUs



NOTES:

1. Port pins are software configurable with pullup device if input port
2. PTA0 does not have a clamp diode to V_{DD}. PTA0 should not be driven above V_{DD}. Also, PTA0 does not pullup to V_{DD} when internal pullup is enabled.
3. IRQ pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1)
4. The RESET pin contains integrated pullup device enabled if reset enabled (RSTPE = 1)
5. High current drive
6. Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

Figure 1-1. MC9S08RC/RD/RE/RG Block Diagram

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

Provided the key enable (KEYEN) bit is 1, the 8-byte comparison key can be used to temporarily disengage memory security. This key mechanism can be accessed only through user code running in secure memory. (A security key cannot be entered directly through background debug commands.) This security key can be disabled completely by programming the KEYEN bit to 0. If the security key is disabled, the only way to disengage security is by mass erasing the FLASH if needed (normally through the background debug interface) and verifying that FLASH is blank. To avoid returning to secure mode after the next reset, program the security bits (SEC01:SEC00) to the unsecured state (1:0).

4.3 RAM

The MC9S08RC/RD/RE/RG includes static RAM. The locations in RAM below \$0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit-manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power wait, stop2, or stop3 mode. At power-on or after wakeup from stop1, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention.

For compatibility with older M68HC05 MCUs, the HCS08 resets the stack pointer to \$00FF. In the MC9S08RC/RD/RE/RG, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale-provided equate file).

```
LDHX    #RamLast+1    ;point one past RAM
TXS                    ;SP<-(H:X-1)
```

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or through code executing from non-secure memory. See Section 4.5, “Security,” for a detailed description of the security feature.

4.4 FLASH

The FLASH memory is intended 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 no special voltages are needed for FLASH erase and programming operations, in-application programming is also possible through other software-controlled communication paths. For a more detailed discussion of in-circuit and in-application programming, refer to the *HCS08 Family Reference Manual, Volume I*, Freescale Semiconductor document order number HCS08RMv1/D.

4.4.1 Features

Features of the FLASH memory include:

- FLASH Size
 - MC9S08RC/RD/RE/RG60 — 63374 bytes (124 pages of 512 bytes each)
 - MC9S08RC/RD/RE/RG32 — 32768 bytes (64 pages of 512 bytes each)
 - MC9S08RC/RD/RE16 — 16384 bytes (32 pages of 512 bytes each)
 - MC9S08RC/RD/RE8 — 8192 bytes (16 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for FLASH and RAM
- Auto power-down for low-frequency read accesses

4.4.2 Program and Erase Times

Before any program or erase command can be accepted, the FLASH clock divider register (FCDIV) must be written to set the internal clock for the FLASH module to a frequency (f_{FCLK}) between 150 kHz and 200 kHz (see Section 4.6.1, “FLASH Clock Divider Register (FCDIV)”). This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ($1/f_{FCLK}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses are used by the command processor to complete a program or erase command.

Table 4-4 shows program and erase times. The bus clock frequency and FCDIV determine the frequency of FCLK (f_{FCLK}). The time for one cycle of FCLK is $t_{FCLK} = 1/f_{FCLK}$. The times are shown as a number of cycles of FCLK and as an absolute time for the case where $t_{FCLK} = 5 \mu\text{s}$. Program and erase times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

Table 4-4. Program and Erase Times

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 μs
Byte program (burst)	4	20 μs ⁽¹⁾
Page erase	4000	20 ms
Mass erase	20,000	100 ms

1. Excluding start/end overhead

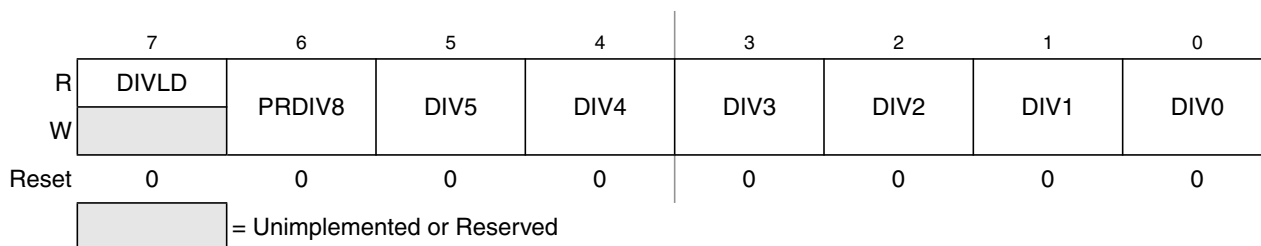


Figure 4-5. FLASH Clock Divider Register (FCDIV)

Table 4-5. FCDIV Field Descriptions

Field	Description
7 DIVLD	Divisor Loaded Status Flag — When set, this read-only status flag indicates that the FCDIV register has been written since reset. Reset clears this bit and the first write to this register causes this bit to become set regardless of the data written. 0 FCDIV has not been written since reset; erase and program operations disabled for FLASH. 1 FCDIV has been written since reset; erase and program operations enabled for FLASH.
6 PRDIV8	Prescale (Divide) FLASH Clock by 8 0 Clock input to the FLASH clock divider is the bus rate clock. 1 Clock input to the FLASH clock divider is the bus rate clock divided by 8.
5:0 DIV[5:0]	Divisor for FLASH Clock Divider — The FLASH clock divider divides the bus rate clock (or the bus rate clock divided by 8 if PRDIV8 = 1) by the value in the 6-bit DIV5:DIV0 field plus one. The resulting frequency of the internal FLASH clock must fall within the range of 200 kHz to 150 kHz for proper FLASH operations. Program/erase timing pulses are one cycle of this internal FLASH clock, which corresponds to a range of 5 μs to 6.7 μs. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. See Equation 4-1 and Equation 4-2.

$$\text{if PRDIV8} = 0 \text{ — } f_{\text{FLK}} = f_{\text{Bus}} \div ([\text{DIV5:DIV0}] + 1) \quad \text{Eqn. 4-1}$$

$$\text{if PRDIV8} = 1 \text{ — } f_{\text{FLK}} = f_{\text{Bus}} \div (8 \times ([\text{DIV5:DIV0}] + 1)) \quad \text{Eqn. 4-2}$$

Table 4-6 shows the appropriate values for PRDIV8 and DIV5:DIV0 for selected bus frequencies.

Table 4-6. FLASH Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV5:DIV0 (Decimal)	f_{FLK}	Program/Erase Timing Pulse (5 μs Min, 6.7 μs Max)
8 MHz	0	39	200 kHz	5 μs
4 MHz	0	19	200 kHz	5 μs
2 MHz	0	9	200 kHz	5 μs
1 MHz	0	4	200 kHz	5 μs
200 kHz	0	0	200 kHz	5 μs
150 kHz	0	0	150 kHz	6.7 μs

5.8.5 System Device Identification Register (SDIDH, SDIDL)

This read-only register is included so host development systems can identify the HCS08 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.

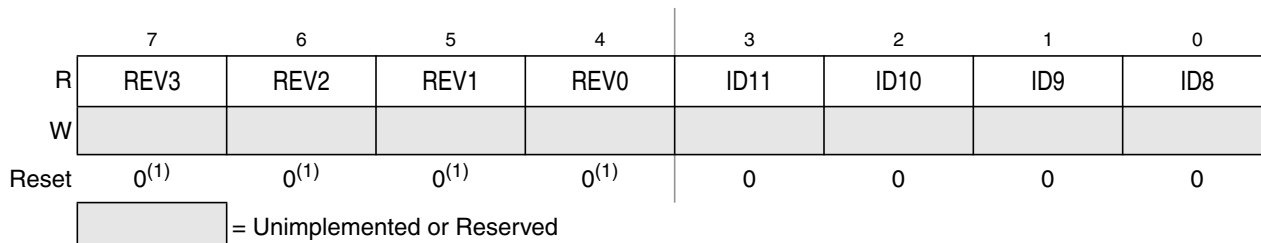


Figure 5-6. System Device Identification Register — High (SDIDH)

1. The revision number that is hard coded into these bits reflects the current silicon revision level.

Table 5-6. SDIDH Field Descriptions

Field	Description
7:4 REV[3:0]	Revision Number — The high-order 4 bits of address \$1806 are hard coded to reflect the current mask set revision number (0–F).
3:0 ID[11:8]	Part Identification Number — Each derivative in the HCS08 Family has a unique identification number. The MC9S08RC/RD/RE/RG32/60 is hard coded to the value \$004 and the MC9S08RC/RD/RE8/16 is hard coded to the value \$003.

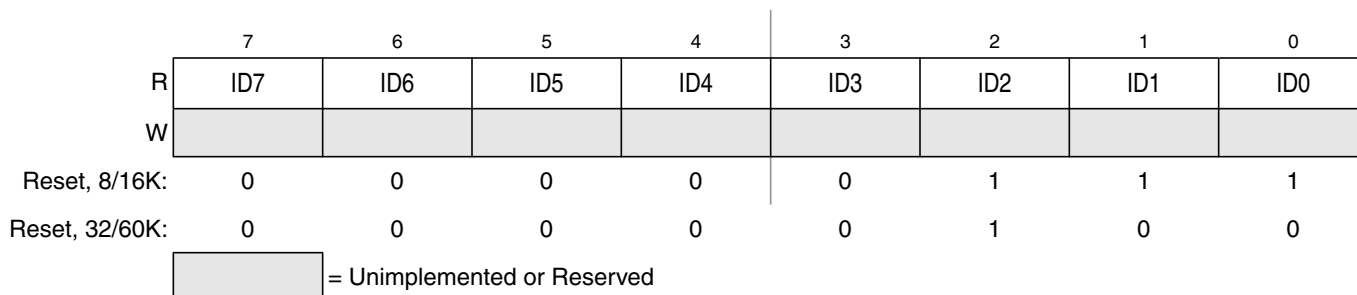


Figure 5-7. System Device Identification Register — Low (SDIDL)

Table 5-7. SDIDL Field Descriptions

Field	Description
7:0 ID[7:0]	Part Identification Number — Each derivative in the HCS08 Family has a unique identification number. The MC9S08RC/RD/RE/RG32/60 is hard coded to the value \$004 and the MC9S08RC/RD/RE8/16 is hard coded to the value \$003.

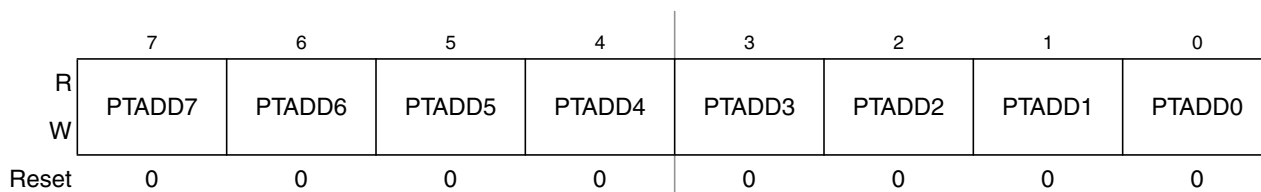


Figure 6-8. Data Direction for Port A (PTADD)

Table 6-3. PTADD 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.

6.6.2 Port B Registers (PTBD, PTBPE, and PTBDD)

Port B pins used as general-purpose I/O pins are controlled by the port B data (PTBD), data direction (PTBDD), and pullup enable (PTBPE) registers.

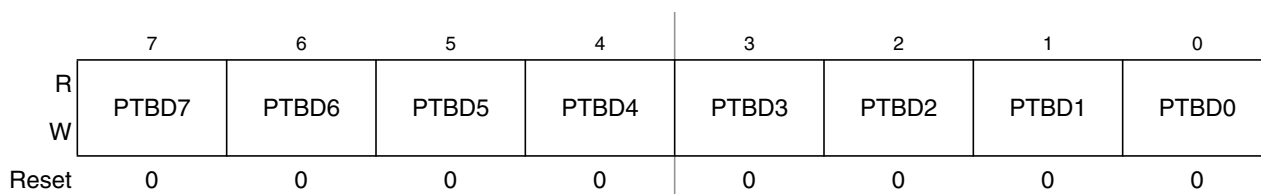


Figure 6-9. Port B Data Register (PTBD)

Table 6-4. PTBD Field Descriptions

Field	Description
7:0 PTBD[7:0]	Port B Data Register Bits — For port B pins that are 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 on the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

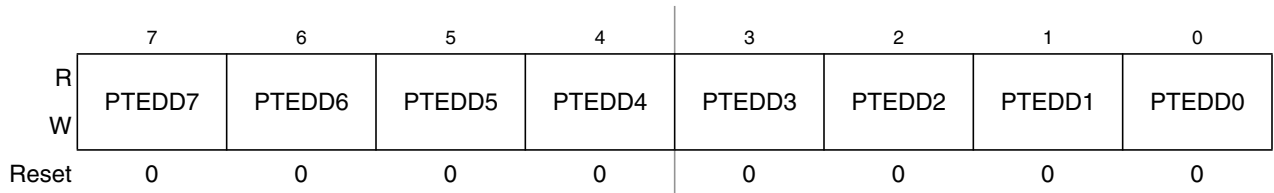


Figure 6-20. Data Direction for Port E (PTEDD)

Table 6-15. PTEDD Field Descriptions

Field	Description
7:0 PTEDD[7:0]	<p>Data Direction for Port E Bits — These read/write bits control the direction of port E pins and what is read for PTED reads.</p> <p>0 Input (output driver disabled) and reads return the pin value.</p> <p>1 Output driver enabled for port E bit n and PTED reads return the contents of PTEDn.</p>

- IX = 16-bit indexed no offset
- IX+ = 16-bit indexed no offset, post increment (CBEQ and MOV only)
- IX1 = 16-bit indexed with 8-bit offset from H:X
- IX1+ = 16-bit indexed with 8-bit offset, post increment (CBEQ only)
- IX2 = 16-bit indexed with 16-bit offset from H:X
- REL = 8-bit relative offset
- SP1 = Stack pointer with 8-bit offset
- SP2 = Stack pointer with 16-bit offset

Table 7-2. HCS08 Instruction Set Summary (Sheet 1 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
ADC #opr8i ADC opr8a ADC opr16a ADC oprx16,X ADC oprx8,X ADC ,X ADC oprx16,SP ADC oprx8,SP	Add with Carry	$A \leftarrow (A) + (M) + (C)$			-				IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 B9 C9 D9 E9 F9 9ED9 9EE9	ii dd hh ll ee ff ee ff ee ff ee ff	2 3 4 4 3 3 5 4
ADD #opr8i ADD opr8a ADD opr16a ADD oprx16,X ADD oprx8,X ADD ,X ADD oprx16,SP ADD oprx8,SP	Add without Carry	$A \leftarrow (A) + (M)$			-			IMM DIR EXT IX2 IX1 IX SP2 SP1	AB BB CB DB EB FB 9EDB 9EEB	ii dd hh ll ee ff ee ff ee ff ee ff	2 3 4 4 3 3 5 4	
AIS #opr8i	Add Immediate Value (Signed) to Stack Pointer	$SP \leftarrow (SP) + (M)$ M is sign extended to a 16-bit value	-	-	-	-	-	IMM	A7	ii	2	
AIX #opr8i	Add Immediate Value (Signed) to Index Register (H:X)	$H:X \leftarrow (H:X) + (M)$ M is sign extended to a 16-bit value	-	-	-	-	-	IMM	AF	ii	2	
AND #opr8i AND opr8a AND opr16a AND oprx16,X AND oprx8,X AND ,X AND oprx16,SP AND oprx8,SP	Logical AND	$A \leftarrow (A) \& (M)$	0	-	-			IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 B4 C4 D4 E4 F4 9ED4 9EE4	ii dd hh ll ee ff ee ff ee ff ee ff	2 3 4 4 3 3 5 4	
ASL opr8a ASLA ASLX ASL oprx8,X ASL ,X ASL oprx8,SP	Arithmetic Shift Left (Same as LSL)				-	-		DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff ff	5 1 1 5 4 6	
ASR opr8a ASRA ASRX ASR oprx8,X ASR ,X ASR oprx8,SP	Arithmetic Shift Right				-	-		DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff ff	5 1 1 5 4 6	
BCC rel	Branch if Carry Bit Clear	Branch if (C) = 0	-	-	-	-	-	REL	24	rr	3	

Table 7-2. HCS08 Instruction Set Summary (Sheet 6 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
ROR <i>opr8a</i> RORA RORX ROR <i>opr8,X</i> ROR <i>,X</i> ROR <i>opr8,SP</i>	Rotate Right through Carry			-	-				DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	5 1 1 5 4 6
RSP	Reset Stack Pointer	SP ← 0xFF (High Byte Not Affected)	-	-	-	-	-	-	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 0x0001; Pull (CCR) SP ← (SP) + 0x0001; Pull (A) SP ← (SP) + 0x0001; Pull (X) SP ← (SP) + 0x0001; Pull (PCH) SP ← (SP) + 0x0001; Pull (PCL)							INH	80		9
RTS	Return from Subroutine	SP ← SP + 0x0001; Pull (PCH) SP ← SP + 0x0001; Pull (PCL)	-	-	-	-	-	-	INH	81		6
SBC # <i>opr8i</i> SBC <i>opr8a</i> SBC <i>opr16a</i> SBC <i>opr16,X</i> SBC <i>opr8,X</i> SBC <i>,X</i> SBC <i>opr16,SP</i> SBC <i>opr8,SP</i>	Subtract with Carry	A ← (A) - (M) - (C)		-	-				IMM DIR EXT IX2 IX1 IX SP2 SP1	A2 B2 C2 D2 E2 F2 9ED2 9EE2	ii dd hh ll ee ff ff ff ff ff	2 3 4 4 3 3 5 4
SEC	Set Carry Bit	C ← 1	-	-	-	-	-	1	INH	99		1
SEI	Set Interrupt Mask Bit	I ← 1	-	-	1	-	-	-	INH	9B		1
STA <i>opr8a</i> STA <i>opr16a</i> STA <i>opr16,X</i> STA <i>opr8,X</i> STA <i>,X</i> STA <i>opr16,SP</i> STA <i>opr8,SP</i>	Store Accumulator in Memory	M ← (A)	0	-	-			-	DIR EXT IX2 IX1 IX SP2 SP1	B7 C7 D7 E7 F7 9ED7 9EE7	dd hh ll ee ff ff ff ff ff	3 4 4 3 2 5 4
STHX <i>opr8a</i> STHX <i>opr16a</i> STHX <i>opr8,SP</i>	Store H:X (Index Reg.)	(M:M + 0x0001) ← (H:X)	0	-	-			-	DIR EXT SP1	35 96 9EFF	dd hh ll ff	4 5 5
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation	I bit ← 0; Stop Processing	-	-	0	-	-	-	INH	8E		2+
STX <i>opr8a</i> STX <i>opr16a</i> STX <i>opr16,X</i> STX <i>opr8,X</i> STX <i>,X</i> STX <i>opr16,SP</i> STX <i>opr8,SP</i>	Store X (Low 8 Bits of Index Register) in Memory	M ← (X)	0	-	-			-	DIR EXT IX2 IX1 IX SP2 SP1	BF CF DF EF FF 9EDF 9EEF	dd hh ll ee ff ff ff ff ff	3 4 4 3 2 5 4
SUB # <i>opr8i</i> SUB <i>opr8a</i> SUB <i>opr16a</i> SUB <i>opr16,X</i> SUB <i>opr8,X</i> SUB <i>,X</i> SUB <i>opr16,SP</i> SUB <i>opr8,SP</i>	Subtract	A ← (A) - (M)		-	-				IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 B0 C0 D0 E0 F0 9ED0 9EE0	ii dd hh ll ee ff ff ff ff ff	2 3 4 4 3 3 5 4
SWI	Software Interrupt	PC ← (PC) + 0x0001 Push (PCL); SP ← (SP) - 0x0001 Push (PCH); SP ← (SP) - 0x0001 Push (X); SP ← (SP) - 0x0001 Push (A); SP ← (SP) - 0x0001 Push (CCR); SP ← (SP) - 0x0001 I ← 1; PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	-	-	1	-	-	-	INH	83		11

Table 7-3. Opcode Map (Sheet 1 of 2)

Bit-Manipulation		Branch		Read-Modify-Write				Control				Register/Memory																			
00 5 3	BRSET0 DIR	10 5 2	BSET0 DIR	20 3 2	BRA REL	30 5 2	NEG DIR	40 1 1	NEGA INH	50 1 1	NEGX INH	60 5 2	NEG IX1	70 4 1	NEG IX	80 9 1	RTI INH	90 3 2	BGE REL	A0 2 2	SUB IMM	B0 3 2	SUB DIR	C0 4 3	SUB EXT	D0 4 3	SUB IX2	E0 3 2	SUB IX1	F0 3 1	SUB IX
01 5 3	BRCLR0 DIR	11 5 2	BCLR0 DIR	21 3 2	BRN REL	31 5 3	CBEQ DIR	41 4 3	CBEQA IMM	51 4 3	CBEQX IMM	61 5 3	CBEQ IX1+	71 5 2	CBEQ IX+	81 6 1	RTS INH	91 3 2	BLT REL	A1 2 2	CMP IMM	B1 3 2	CMP DIR	C1 4 3	CMP EXT	D1 4 3	CMP IX2	E1 3 2	CMP IX1	F1 3 1	CMP IX
02 5 3	BRSET1 DIR	12 5 2	BSET1 DIR	22 3 2	BHI REL	32 5 3	LDHX EXT	42 5 1	MUL INH	52 6 1	DIV INH	62 1 1	NSA INH	72 4 1	DAA INH	82 5+ 1	BGND INH	92 3 2	BGT REL	A2 2 2	SBC IMM	B2 3 2	SBC DIR	C2 4 3	SBC EXT	D2 4 3	SBC IX2	E2 3 2	SBC IX1	F2 3 1	SBC IX
03 5 3	BRCLR1 DIR	13 5 2	BCLR1 DIR	23 3 2	BLS REL	33 5 3	COM DIR	43 1 1	COMA INH	53 1 1	COMX INH	63 5 2	COM IX1	73 4 1	COM IX	83 11 1	SWI INH	93 3 2	BLE REL	A3 2 2	CPX IMM	B3 3 2	CPX DIR	C3 4 3	CPX EXT	D3 4 3	CPX IX2	E3 3 2	CPX IX1	F3 3 1	CPX IX
04 5 3	BRSET2 DIR	14 5 2	BSET2 DIR	24 3 2	BCC REL	34 5 2	LSR DIR	44 1 1	LSRA INH	54 1 1	LSRX INH	64 5 2	LSR IX1	74 4 1	LSR IX	84 1 1	TAP INH	94 2 1	TXS INH	A4 2 2	AND IMM	B4 3 2	AND DIR	C4 4 3	AND EXT	D4 4 3	AND IX2	E4 3 2	AND IX1	F4 3 1	AND IX
05 5 3	BRCLR2 DIR	15 5 2	BCLR2 DIR	25 3 2	BCS REL	35 4 3	STHX DIR	45 3 3	LDHX IMM	55 4 2	LDHX DIR	65 3 3	CPHX IMM	75 5 2	CPHX DIR	85 1 1	TPA INH	95 2 1	TSX INH	A5 2 2	BIT IMM	B5 3 2	BIT DIR	C5 4 3	BIT EXT	D5 4 3	BIT IX2	E5 3 2	BIT IX1	F5 3 1	BIT IX
06 5 3	BRSET3 DIR	16 5 2	BSET3 DIR	26 3 2	BNE REL	36 5 2	ROR DIR	46 1 1	RORA INH	56 1 1	RORX INH	66 5 2	ROR IX1	76 4 1	ROR IX	86 3 1	PULA INH	96 5 3	STHX EXT	A6 2 2	LDA IMM	B6 3 2	LDA DIR	C6 4 3	LDA EXT	D6 4 3	LDA IX2	E6 3 2	LDA IX1	F6 3 1	LDA IX
07 5 3	BRCLR3 DIR	17 5 2	BCLR3 DIR	27 3 2	BEQ REL	37 5 2	ASR DIR	47 1 1	ASRA INH	57 1 1	ASRX INH	67 5 2	ASR IX1	77 4 1	ASR IX	87 2 1	PSHA INH	97 1 1	TAX INH	A7 2 2	AIS IMM	B7 3 2	STA DIR	C7 4 3	STA EXT	D7 4 3	STA IX2	E7 3 2	STA IX1	F7 3 1	STA IX
08 5 3	BRSET4 DIR	18 5 2	BSET4 DIR	28 3 2	BHCC REL	38 5 2	LSL DIR	48 1 1	LSLA INH	58 1 1	LSLX INH	68 5 2	LSL IX1	78 4 1	LSL IX	88 3 1	PULX INH	98 1 1	CLC INH	A8 2 2	EOR IMM	B8 3 2	EOR DIR	C8 4 3	EOR EXT	D8 4 3	EOR IX2	E8 3 2	EOR IX1	F8 3 1	EOR IX
09 5 3	BRCLR4 DIR	19 5 2	BCLR4 DIR	29 3 2	BHCS REL	39 5 2	ROL DIR	49 1 1	ROLA INH	59 1 1	ROLX INH	69 5 2	ROL IX1	79 4 1	ROL IX	89 2 1	PSHX INH	99 1 1	SEC INH	A9 2 2	ADC IMM	B9 3 2	ADC DIR	C9 4 3	ADC EXT	D9 4 3	ADC IX2	E9 3 2	ADC IX1	F9 3 1	ADC IX
0A 5 3	BRSET5 DIR	1A 5 2	BSET5 DIR	2A 3 2	BPL REL	3A 5 2	DEC DIR	4A 1 1	DECA INH	5A 1 1	DECX INH	6A 5 2	DEC IX1	7A 4 1	DEC IX	8A 3 1	PULH INH	9A 1 1	CLI INH	AA 2 2	ORA IMM	BA 3 2	ORA DIR	CA 4 3	ORA EXT	DA 4 3	ORA IX2	EA 3 2	ORA IX1	FA 3 1	ORA IX
0B 5 3	BRCLR5 DIR	1B 5 2	BCLR5 DIR	2B 3 2	BMI REL	3B 7 3	DBNZ DIR	4B 4 2	DBNZA INH	5B 4 2	DBNZX INH	6B 7 3	DBNZ IX1	7B 6 2	DBNZ IX	8B 2 1	PSHH INH	9B 1 1	SEI INH	AB 2 2	ADD IMM	BB 3 2	ADD DIR	CB 4 3	ADD EXT	DB 4 3	ADD IX2	EB 3 2	ADD IX1	FB 3 1	ADD IX
0C 5 3	BRSET6 DIR	1C 5 2	BSET6 DIR	2C 3 2	BMC REL	3C 5 2	INC DIR	4C 1 1	INCA INH	5C 1 1	INCX INH	6C 5 2	INC IX1	7C 4 1	INC IX	8C 1 1	CLRH INH	9C 1 1	RSP INH	AC 2 2	JMP	BC 3 2	JMP DIR	CC 4 3	JMP EXT	DC 4 3	JMP IX2	EC 3 2	JMP IX1	FC 3 1	JMP IX
0D 5 3	BRCLR6 DIR	1D 5 2	BCLR6 DIR	2D 3 2	BMS REL	3D 4 3	TST DIR	4D 1 1	TSTA INH	5D 1 1	TSTX INH	6D 4 2	TST IX1	7D 3 1	TST IX	8D 2+ 1	STOP INH	9E Page 2		AD 5 2	BSR REL	BD 5 2	JSR DIR	CD 6 3	JSR EXT	DD 6 3	JSR IX2	ED 5 2	JSR IX1	FD 5 1	JSR IX
0E 5 3	BRSET7 DIR	1E 5 2	BSET7 DIR	2E 3 2	BIL REL	3E 6 3	CPHX EXT	4E 5 3	MOV DD	5E 5 2	MOV DIX+	6E 4 3	MOV IMD	7E 5 2	MOV IX+D	8E 2+ 1	STOP INH			AE 2 2	LDX IMM	BE 3 2	LDX DIR	CE 4 3	LDX EXT	DE 4 3	LDX IX2	EE 3 2	LDX IX1	FE 3 1	LDX IX
0F 5 3	BRCLR7 DIR	1F 5 2	BCLR7 DIR	2F 3 2	BIH REL	3F 5 2	CLR DIR	4F 1 1	CLRA INH	5F 1 1	CLR INH	6F 5 2	CLR IX1	7F 4 1	CLR IX	8F 2+ 1	WAIT INH	9F 1 1	TXA INH	AF 2 2	AIX IMM	BF 3 2	STX DIR	CF 4 3	STX EXT	DF 4 3	STX IX2	EF 3 2	STX IX1	FF 3 1	STX IX

INH Inherent
 IMM Immediate
 DIR Direct
 EXT Extended
 DD DIR to DIR
 IX+D IX+ to DIR

REL Relative
 IX Indexed, No Offset
 IX1 Indexed, 8-Bit Offset
 IX2 Indexed, 16-Bit Offset
 IMD IMM to DIR
 DIX+ DIR to IX+

SP1 Stack Pointer, 8-Bit Offset
 SP2 Stack Pointer, 16-Bit Offset
 IX+ Indexed, No Offset with Post Increment
 IX1+ Indexed, 1-Byte Offset with Post Increment

Opcode in Hexadecimal F0 SUB 3
 Number of Bytes 1 IX HCS08 Cycles Instruction Mnemonic Addressing Mode

The space period provides an interpulse gap (no carrier). If CMTCMD3:CMTCMD4 = \$0000, then the modulator and carrier generator will switch between carrier frequencies without a gap or any carrier glitches (zero space).

Using timing data for carrier burst and interpulse gap length calculated by the CPU, FSK mode can automatically generate a phase-coherent, dual-frequency FSK signal with programmable burst and interburst gaps.

The mark and space time equations for FSK mode are:

$$t_{\text{mark}} = (\text{CMTCMD1:CMTCMD2} + 1) \div f_{\text{CG}} \tag{Eqn. 8-7}$$

$$t_{\text{space}} = \text{CMTCMD3:CMTCMD4} \div f_{\text{CG}} \tag{Eqn. 8-8}$$

Where f_{CG} is the frequency output from the carrier generator. The example in Figure 8-6 shows what the IRO pin output looks like in FSK mode with the following values: CMTCMD1:CMTCMD2 = \$0003, CMTCMD3:CMTCMD4 = \$0002, primary carrier high count = \$01, primary carrier low count = \$02, secondary carrier high count = \$03, and secondary carrier low count = \$01.

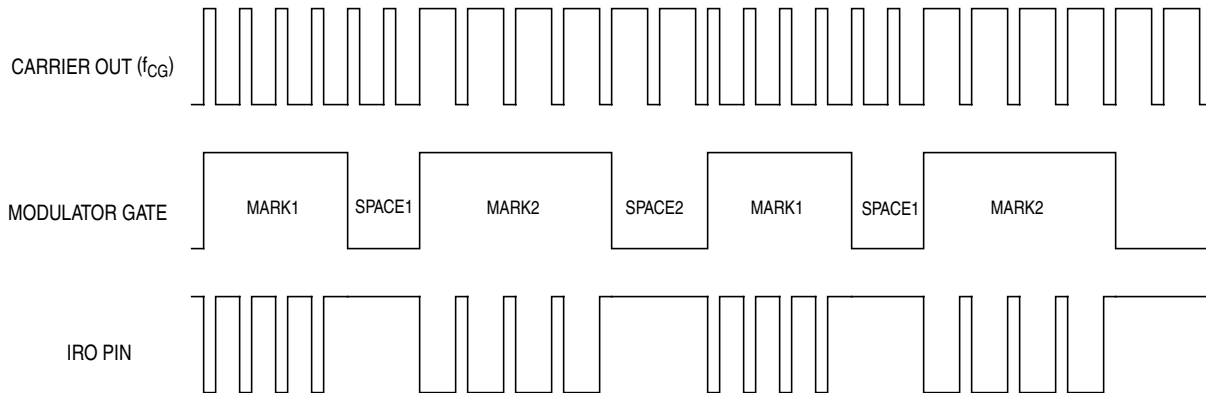


Figure 8-6. Example CMT Output in FSK Mode

8.5.3 Extended Space Operation

In time, baseband, or FSK mode, the space period can be made longer than the maximum possible value of the space period register. Setting the EXSPC bit in the CMTMSC register will force the modulator to treat the next modulation period (beginning with the next load of the counter and space period register) as a space period equal in length to the mark and space counts combined. Subsequent modulation periods will consist entirely of these extended space periods with no mark periods. Clearing EXSPC will return the modulator to standard operation at the beginning of the next modulation period.

8.5.3.1 EXSPC Operation in Time Mode

To calculate the length of an extended space in time or baseband modes, add the mark and space times and multiply by the number of modulation periods that EXSPC is set.

8.6.3 CMT Modulator Status and Control Register (CMTMSC)

The CMT modulator status and control register (CMTMSC) contains the modulator and carrier generator enable (MCGEN), end of cycle interrupt enable (EOCIE), FSK mode select (FSK), baseband enable (BASE), extended space (EXSPC), prescaler (CMTDIV1:CMTDIV0) bits, and the end of cycle (EOCF) status bit.

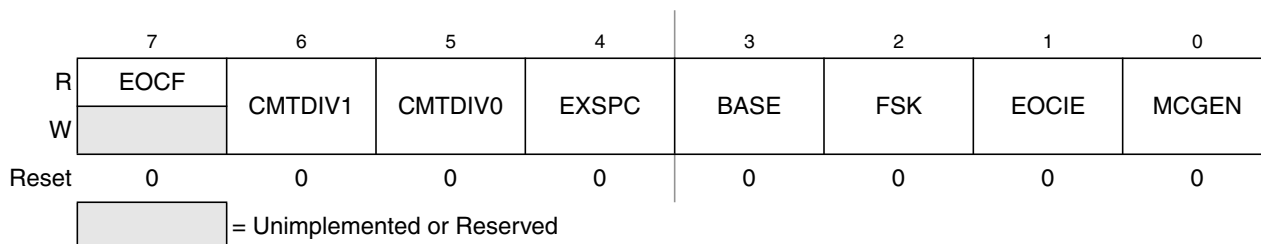


Figure 8-13. CMT Modulator Status and Control Register (CMTMSC)

Table 8-8. CMTMSC Field Descriptions

Field	Description
7 EOCF	<p>End of Cycle Status Flag — The EOCF bit is set when:</p> <ul style="list-style-type: none"> • The modulator is not currently active and the MCGEN bit is set to begin the initial CMT transmission. • At the end of each modulation cycle while the MCGEN bit is set. This is recognized when a match occurs between the contents of the space period register and the down-counter. At this time, the counter is initialized with the (possibly new) contents of the mark period buffer, CMTCMD1 and CMTCMD2. The space period register is loaded with the (possibly new) contents of the space period buffer, CMTCMD3 and CMTCMD4. <p>This flag is cleared by a read of the CMTMSC register followed by an access of CMTCMD2 or CMTCMD4. In the case where the MCGEN bit is cleared and then set before the end of the modulation cycle, EOCF will not be set when MCGEN is set, but will be set at the end of the current modulation cycle.</p> <p>0 No end of modulation cycle occurrence since flag last cleared 1 End of modulator cycle has occurred</p>
6:5 CMTDIV[1:0]	<p>CMT Clock Divide Prescaler — The CMT clock divide prescaler causes the CMT to be clocked at the BUS CLOCK frequency, or the BUS CLOCK frequency divided by 1, 2, 4, or 8. Because these bits are not double buffered, they should not be changed during a transmission.</p> <p>00 Bus clock ÷ 1 01 Bus clock ÷ 2 10 Bus clock ÷ 4 11 Bus clock ÷ 8</p>
4 EXSPC	<p>Extended Space Enable — The EXSPC bit enables extended space operation.</p> <p>0 Extended space disabled 1 Extended space enabled</p>
3 BASE	<p>Baseband Enable — When set, the BASE bit disables the carrier generator and forces the carrier output high for generation of baseband protocols. When BASE is clear, the carrier generator is enabled and the carrier output toggles at the frequency determined by values stored in the carrier data registers. See Section 8.5.2.2, "Baseband Mode." This bit is cleared by reset. This bit is not double buffered and should not be written to during a transmission.</p> <p>0 Baseband mode disabled 1 Baseband mode enabled</p>
2 FSK	<p>FSK Mode Select — The FSK bit enables FSK operation.</p> <p>0 CMT operates in time or baseband mode 1 CMT operates in FSK mode</p>

Chapter 9

Keyboard Interrupt (S08KBIV1)

9.1 Introduction

The MC9S08RC/RD/RE/RG has two KBI modules. One has eight keyboard interrupt inputs that share port A pins. The other KBI module has four inputs that are shared on the upper four pins of port C. See the Pins and Connections chapter for more information about the logic and hardware aspects of these pins.

Port A is an 8-bit port that is shared between the KBI1 keyboard interrupt inputs and general-purpose I/O. The eight KBI1PEn control bits in the KBI1PE register allow selection of any combination of port A pins to be assigned as KBI1 inputs. Any pins that are enabled as KBI1 inputs will be forced to act as inputs and the remaining port A pins are available as general-purpose I/O pins controlled by the port A data (PTAD), data direction (PTADD) and pullup enable (PTAPE) registers. The eight PTAPEn control bits in the PTAPE register allow the user to select whether an internal pullup device is enabled on each port A pin that is configured as a port input or a KBI1 input.

KBI1 inputs can be configured for edge-only sensitivity or edge-and-level sensitivity. Bits 3 through 0 of port A are falling-edge/low-level sensitive and bits 7 through 4 can be configured for rising-edge/high-level or for falling-edge/low-level sensitivity.

Port C is an 8-bit port with its lower four pins shared between the KBI2 keyboard interrupt inputs and general-purpose I/O. The four KBI2PEn control bits in the KBI2PE register allow selection of any combination of the lower four port C pins to be assigned as KBI2 inputs. Any pins that are enabled as KBI2 inputs will be forced to act as inputs and the remaining port C pins are available as general-purpose I/O pins controlled by the port C data (PTCD), data direction (PTCDD) and pullup enable (PTCPE) registers. The eight PTCPEn control bits in the PTCPE register allow the user to select whether an internal pullup device is enabled on each port C pin that is configured as a port input or a KBI2 input.

Any enabled keyboard interrupt can be used to wake the MCU from wait, standby (stop3), partial power-down (stop2) or power-down modes (stop1). In either stop1 or stop2 mode, an input functions as a falling edge/low-level wakeup, therefore it should be configured to use falling-edge sensing if the MCU will be used in stop1 or stop2 modes.

Either KBI1 or KBI2 can be used to wake the MCU from wait or standby (stop3). Only KBI1 can be used to wake the MCU from partial power down (stop2) or power down (stop1). When using KBI1 to wake up from stop2 or stop1, the pins must be configured to use falling-edge/low-level sensing (KBEDG = 0). The KBF bits for both KBI modules must be cleared before entering stop mode, regardless of whether the interrupt is enabled.

NOTE

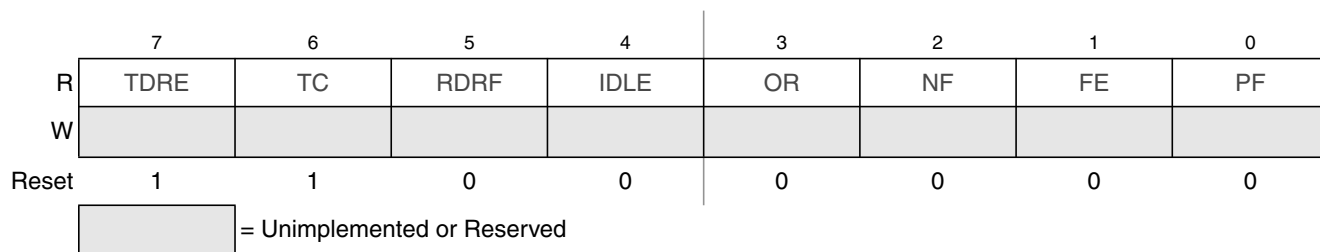
The voltage measured on the pulled up PTA0 pin will be less than V_{DD} . The internal gates connected to this pin are pulled all the way to V_{DD} . All other pins with enabled pullup resistors will have an unloaded measurement of V_{DD} .

Table 12-4. SCI1C2 Register Field Descriptions (continued)

Field	Description
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 12.3.3.2, “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 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 12.3.2.1, “Send Break and Queued Idle,” for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

12.2.4 SCI Status Register 1 (SCI1S1)

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.


Figure 12-7. SCI Status Register 1 (SCI1S1)
Table 12-5. SCI1S1 Register Field Descriptions

Field	Description
7 TDRE	Transmit Data Register Empty Flag — TDRE is set immediately after 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 SCI1S1 with TDRE = 1 and then write to the SCI data register (SCI1D). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	Transmission Complete Flag — TC is set immediately after 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 SCI1S1 with TC = 1 and then doing one of the following three things: <ul style="list-style-type: none"> • Write to the SCI data register (SCI1D) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to SBK in SCI1C2

SSOE — Slave Select Output Enable

This bit is used in combination with the mode fault enable (MODFEN) bit in SPCR2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS1}$ pin as shown in Table 13-1.

Table 13-1. $\overline{SS1}$ Pin Function

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	\overline{SS} input for mode fault	Slave select input
1	1	Automatic \overline{SS} output	Slave select input

LSBFE — LSB First (Shifter Direction)

1 = SPI serial data transfers start with least significant bit.

0 = SPI serial data transfers start with most significant bit.

13.4.2 SPI Control Register 2 (SPI1C2)

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	<st-blue>	<st-blue>	0	<st-blue>	<st-blue>
Write:				MODFEN	BIDIROE		SPISWAI	SPC0
Reset:	0	0	0	0	0	0	0	0


 = Unimplemented or Reserved

Figure 13-8. SPI Control Register 2 (SPI1C2)

MODFEN — Master Mode-Fault Function Enable

When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS1}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS1}$ pin is used (refer to Table 13-1 for more details).

1 = Mode fault function enabled, master $\overline{SS1}$ pin acts as the mode fault input or the slave select output.

0 = Mode fault function disabled, master $\overline{SS1}$ pin reverts to general-purpose I/O not controlled by SPI.

15.2.3 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 15-1 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 15-1 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 15-1. 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.

15.3 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 15.3.6, "Hardware Breakpoints."

15.3.1 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)

15.3.2 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

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 ($A \leq \text{Address} \leq B$) — 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 ($\text{Address} < A$ or $\text{Address} > B$) — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

7. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.
8. Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions. If positive injection current ($V_{in} > V_{DD}$) is greater than I_{DD} , the injection current may flow out of V_{DD} and could result in external power supply going out of regulation. Ensure external V_{DD} load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if clock rate is very low (which would reduce overall power consumption).
9. PTA0 does not have a clamp diode to V_{DD} . Do not drive PTA0 above V_{DD} .

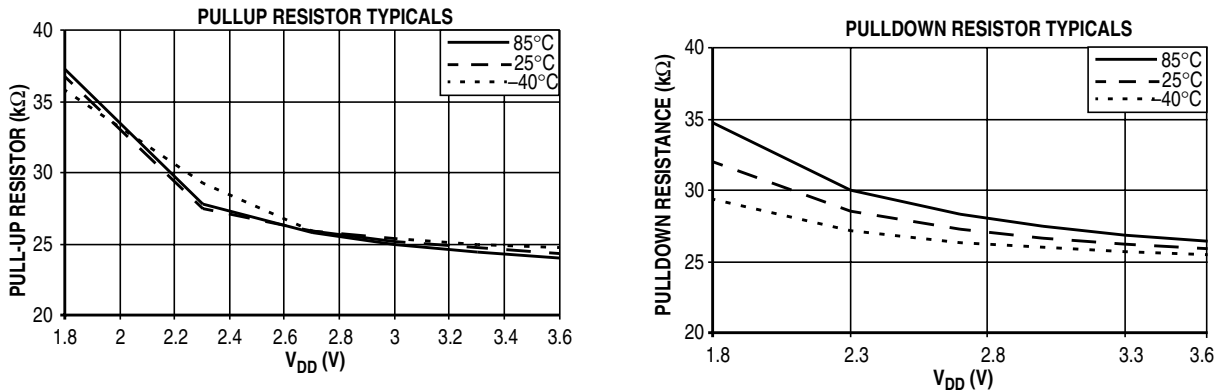
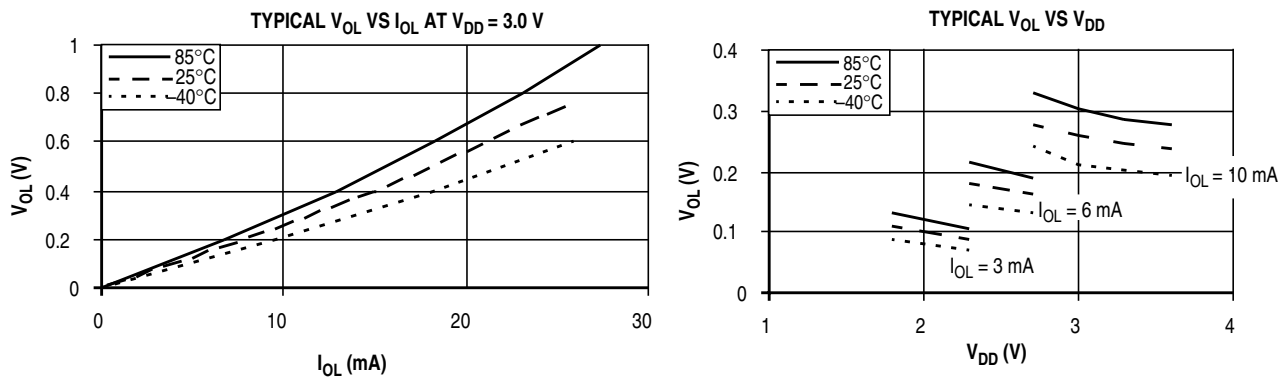

 Figure A-1. Pullup and Pulldown Typical Resistor Values ($V_{DD} = 3.0$ V)


Figure A-2. Typical Low-Side Driver (Sink) Characteristics (Port B and IRO)

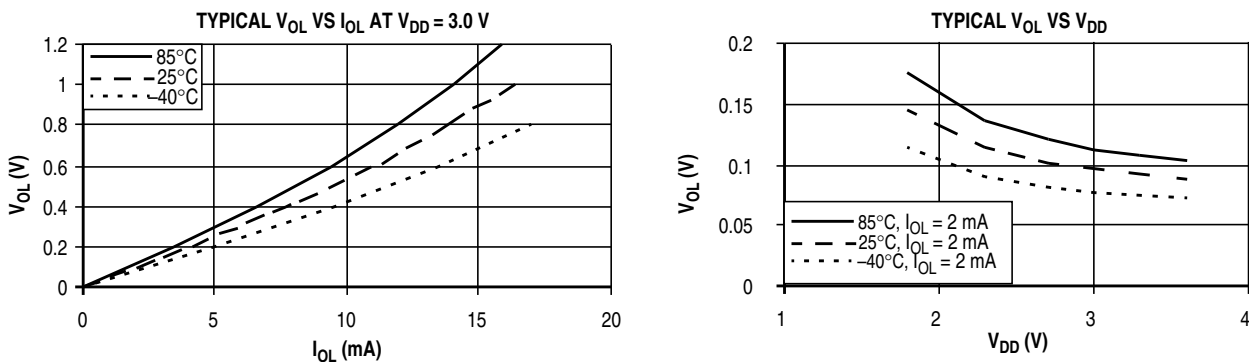


Figure A-3. Typical Low-Side Driver (Sink) Characteristics (Ports A, C, D and E)