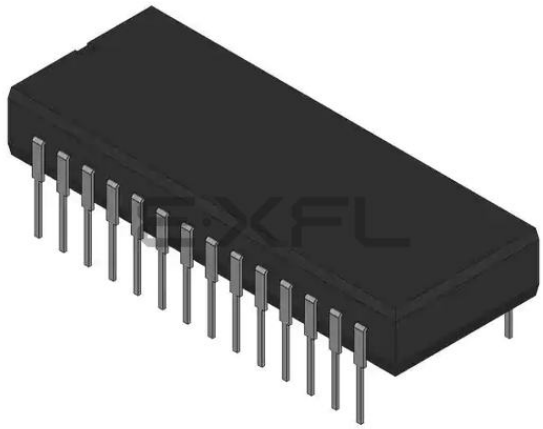


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	S08
Core Size	8-Bit
Speed	8MHz
Connectivity	SCI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	23
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	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.600", 15.24mm)
Supplier Device Package	28-PDIP
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s08rd32pe

Part Number	Package Description	Original (gold wire) package document number	Current (copper wire) package document number
MC68HC908JW32	48 QFN	98ARH99048A	98ASA00466D
MC9S08AC16			
MC9S908AC60			
MC9S08AC128			
MC9S08AW60			
MC9S08GB60A			
MC9S08GT16A			
MC9S08JM16			
MC9S08JM60			
MC9S08LL16			
MC9S08QE128			
MC9S08QE32			
MC9S08RG60			
MCF51CN128			
MC9RS08LA8	48 QFN	98ARL10606D	98ASA00466D
MC9S08GT16A	32 QFN	98ARH99035A	98ASA00473D
MC9S908QE32	32 QFN	98ARE10566D	98ASA00473D
MC9S908QE8	32 QFN	98ASA00071D	98ASA00736D
MC9S08JS16	24 QFN	98ARL10608D	98ASA00734D
MC9S08QB8			
MC9S08QG8	24 QFN	98ARL10605D	98ASA00474D
MC9S08SH8	24 QFN	98ARE10714D	98ASA00474D
MC9RS08KB12	24 QFN	98ASA00087D	98ASA00602D
MC9S08QG8	16 QFN	98ARE10614D	98ASA00671D
MC9RS08KB12	8 DFN	98ARL10557D	98ASA00672D
MC9S08QG8			
MC9RS08KA2	6 DFN	98ARL10602D	98ASA00735D

MC9S08RC8/16/32/60
MC9S08RD8/16/32/60
MC9S08RE8/16/32/60
MC9S08RG32/60

Data Sheet

HCS08
Microcontrollers

MC9S08RG60/D
Rev. 1.11
06/2005

freescale.com

freescale™
semiconductor

Chapter 9
Keyboard Interrupt (S08KBIV1)

Chapter 10
Timer/PWM Module (S08TPMV1)

Chapter 4
Memory

4.1 MC9S08RC/RD/RE/RG Memory Map

As shown in Figure 4-1, on-chip memory in the MC9S08RC/RD/RE/RG series of MCUs consists of RAM, FLASH program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers ($0000 through $0045 for 32K and 60K parts, and $0000 through $003F for 16K and 8K parts)
- High-page registers ($1800 through $182B)
- Nonvolatile registers ($FFB0 through $FFBF)

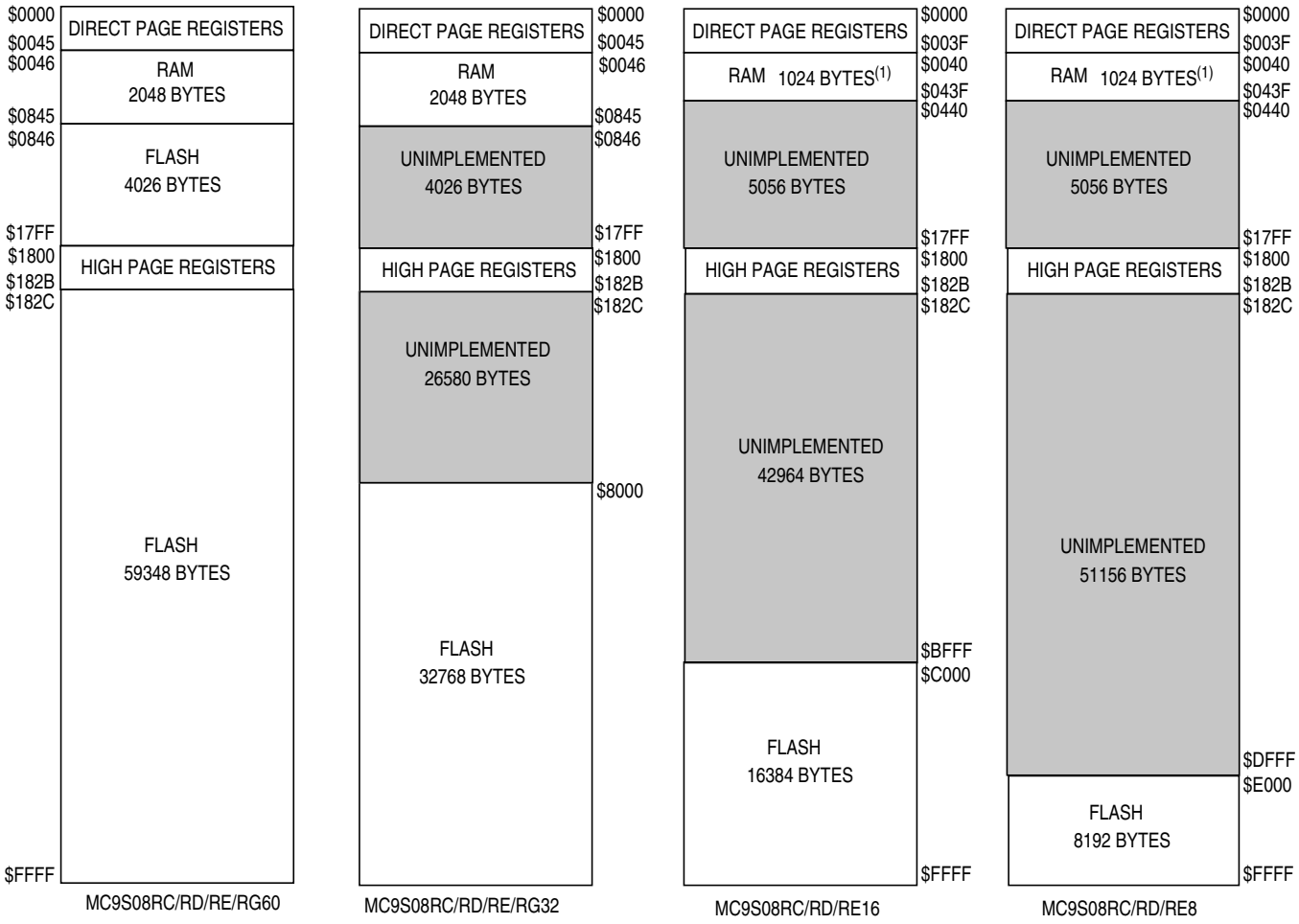


Figure 4-1. MC9S08RC/RD/RE/RG Memory Map

4.1.1 Reset and Interrupt Vector Assignments

Figure 4-2 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale-provided equate file for the MC9S08RC/RD/RE/RG. For more details about resets, interrupts, interrupt priority, and local interrupt mask controls, refer to the Chapter 5, “Resets, Interrupts, and System Configuration."

Table 4-2. High-Page Register Summary (continued)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$1809	SPMSC1	LVDF	LVDACK	LVDIE	SAFE	LVDRE	—	—	—
$180A	SPMSC2	LVWF	LVWACK	0	0	PPDF	PPDACK	PDC	PPDC
$180B–$180F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
$1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
$1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
$1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
$1813	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
$1814	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8
$1815	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
$1816	DBGC	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
$1817	DBGT	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
$1818	DBGS	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
$1819–$181F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
$1820	FCDIV	DIVLD	PRDIV8	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0
$1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
$1822	Reserved	—	—	—	—	—	—	—	—
$1823	FCNFG	0	0	KEYACC	0	0	0	0	0
$1824	FPROT	FPOPEN	FPDIS	FPS2	FPS1	FPS0	0	0	0
$1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
$1826	FCMD	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
$1827–$182B	Reserved	— —	— —	— —	— —	— —	— —	— —	— —

1. The ILAD bit is only present on 16K and 8K versions of the devices.

Nonvolatile FLASH registers, shown in Table 4-3, are located in the FLASH memory. These registers include an 8-byte backdoor key that optionally can be used to gain access to secure memory resources. During reset events, the contents of NVPROT and NVOPT in the nonvolatile register area of the FLASH memory are transferred into corresponding FPROT and FOPT working registers in the high-page registers to control security and block protection options.

Table 4-3. Nonvolatile Register Summary

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$FFB0–$FFB7	NVBACKKEY	8-Byte Comparison Key							
$FFB8–$FFBC	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
$FFBD	NVPROT	FPOPEN	FPDIS	FPS2	FPS1	FPS0	0	0	0
$FFBE	Reserved	—	—	—	—	—	—	—	—
$FFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00

- Writing to a FLASH address before the internal FLASH clock frequency has been set by writing to the FCDIV register
- Writing to a FLASH address while FCBEF is not set (A new command cannot be started until the command buffer is empty.)
- Writing a second time to a FLASH address before launching the previous command (There is only one write to FLASH for every command.)
- Writing a second time to FCMD before launching the previous command (There is only one write to FCMD for every command.)
- Writing to any FLASH control register other than FCMD after writing to a FLASH address
- Writing any command code other than the five allowed codes ($05, $20, $25, $40, or $41) to FCMD
- Accessing (read or write) any FLASH control register other than the write to FSTAT (to clear FCBEF and launch the command) after writing the command to FCMD
- The MCU enters stop mode while a program or erase command is in progress (The command is aborted.)
- Writing the byte program, burst program, or page erase command code ($20, $25, or $40) with a background debug command while the MCU is secured (The background debug controller can only do blank check and mass erase commands when the MCU is secure.)
- Writing 0 to FCBEF to cancel a partial command

4.4.6 FLASH Block Protection

Block protection prevents program or erase changes for FLASH memory locations in a designated address range. Mass erase is disabled when any block of FLASH is protected. The MC9S08RC/RD/RE/RG allows a block of memory at the end of FLASH, and/or the entire FLASH memory to be block protected. A disable control bit and a 3-bit control field, for each of the blocks, allows the user to independently set the size of these blocks. A separate control bit allows block protection of the entire FLASH memory array. All seven of these control bits are located in the FPROT register (see Section 4.6.4, “FLASH Protection Register (FPROT and NVPROT)“).

At reset, the high-page register (FPROT) is loaded with the contents of the NVPROT location that is in the nonvolatile register block of the FLASH memory. The value in FPROT cannot be changed directly from application software so a runaway program cannot alter the block protection settings. If the last 512 bytes of FLASH (which includes the NVPROT register) is protected, the application program cannot alter the block protection settings (intentionally or unintentionally). The FPROT control bits can be written by background debug commands to allow a way to erase a protected FLASH memory.

One use for block protection is to block protect an area of FLASH memory for a bootloader program. This bootloader program then can be used to erase the rest of the FLASH memory and reprogram it. Because the bootloader is protected, it remains intact even if MCU power is lost during an erase and reprogram operation.

Chapter 5
Resets, Interrupts, and System Configuration

5.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the MC9S08RC/RD/RE/RG. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this data sheet. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog and real-time interrupt (RTI), are not part of on-chip peripheral systems having their own sections 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:
  - Power-on detection (POR)
  - Low voltage detection (LVD) with enable
  - External reset pin with enable ($\overline{\text{RESET}}$)
  - COP watchdog with enable and two timeout choices
  - Illegal opcode
  - Illegal address (on 16K and 8K devices)
  - Serial command from a background debug host
- Reset status register (SRS) to indicate source of most recent reset; flag to indicate stop2 (partial power down) mode recovery (PPDF)
- Separate interrupt vectors for each module (reduces polling overhead) (see Table 5-1)
- Safe state for protecting the MCU in low-voltage condition

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 loaded from the reset vector ($FFFE:$FFFF). On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pullup devices disabled. The I bit in the condition code register (CCR) is set to block maskable interrupts until the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to $00FF at reset.

6.6.1 Port A Registers (PTAD, PTAPE, and PTADD)

Port A pins used as general-purpose I/O pins are controlled by the port A data (PTAD), data direction (PTADD), and pullup enable (PTAPE) registers.

	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 6-6. Port A Data Register (PTAD)

Table 6-1. PTAD Field Descriptions

Field	Description
7:0 PTAD[7:0]	**Port A Data Register Bits** — For port A pins that are inputs, reads of this register return the logic level on the pin. For port A pins that are configured as outputs, reads of this register 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	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-7. Pullup Enable for Port A (PTAPE)

Table 6-2. PTAPE Field Descriptions

Field	Description
7:0 PTAPE[7:0]	**Pullup Enable for Port A Bits** — For port A pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled provided the corresponding PTADDn is a logic 0. For port A pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled. When any of bits 7 through 4 of port A are enabled as KBI inputs and are configured to detect rising edges/high levels, the pullup enable bits enable pulldown rather than pullup devices. 0 Internal pullup device disabled. 1 Internal pullup device enabled.

6.6.5 Port E Registers (PTED, PTEPE, and PTEDD)

Port E pins used as general-purpose I/O pins are controlled by the port E data (PTED), data direction (PTEDD), and pullup enable (PTEPE) registers.

	7	6	5	4	3	2	1	0
R W	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
Reset	0	0	0	0	0	0	0	0

Figure 6-18. Port E Data Register (PTED)

Table 6-13. PTED Field Descriptions

Field	Description
7:0 PTED[7:0]	**Port E Data Register Bits** — For port E pins that are inputs, reads return the logic level on the pin. For port E 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 E pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTED 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	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-19. Pullup Enable for Port E (PTED)

Table 6-14. PTED Field Descriptions

Field	Description
7:0 PTED[7:0]	**Pullup Enable for Port E Bits** — For port E pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port E pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled. 0 Internal pullup device disabled. 1 Internal pullup device enabled.

	7	6	5	4	3	2	1	0
R W	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
Reset	0	0	0	0	0	0	0	0

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

Table 6-15. PTEDD Field Descriptions

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

0	=	Bit forced to 0
1	=	Bit forced to 1
	=	Bit set or cleared according to results of operation
U	=	Undefined after the operation

Machine coding notation

dd	=	Low-order 8 bits of a direct address 0x0000–0x00FF (high byte assumed to be 0x00)
ee	=	Upper 8 bits of 16-bit offset
ff	=	Lower 8 bits of 16-bit offset or 8-bit offset
ii	=	One byte of immediate data
jj	=	High-order byte of a 16-bit immediate data value
kk	=	Low-order byte of a 16-bit immediate data value
hh	=	High-order byte of 16-bit extended address
ll	=	Low-order byte of 16-bit extended address
rr	=	Relative offset

Source form

Everything in the source forms columns, *except expressions in italic characters,* is literal information that 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
opr8i	—	Any label or expression that evaluates to an 8-bit immediate value
opr16i	—	Any label or expression that evaluates to a 16-bit immediate value
opr8a	—	Any label or expression that evaluates to an 8-bit value. The instruction treats this 8-bit value as the low order 8 bits of an address in the direct page of the 64-Kbyte address space (0x00xx).
opr16a	—	Any label or expression that evaluates to a 16-bit value. The instruction treats this value as an address in the 64-Kbyte address space.
oprx8	—	Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing
oprx16	—	Any label or expression that evaluates to a 16-bit value. Because the HCS08 has a 16-bit address bus, this can be either a signed or an unsigned value.
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)
IMM	=	8-bit or 16-bit immediate
DIR	=	8-bit direct
EXT	=	16-bit extended

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)

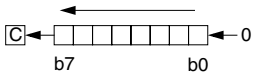
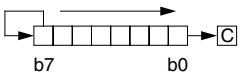
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
ADC #*opr8i* ADC *opr8a* ADC *opr16a* ADC *oprx16*,X ADC *oprx8*,X ADC ,X ADC *oprx16*,SP ADC *oprx8*,SP	Add with Carry	A ← (A) + (M) + (C)			–				IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 B9 C9 D9 E9 F9 9ED9 9EE9	ii dd hh ll ee ff 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 ← (A) + (M)			–				IMM DIR EXT IX2 IX1 IX SP2 SP1	AB BB CB DB EB FB 9EDB 9EEB	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
AIS #*opr8i*	Add Immediate Value (Signed) to Stack Pointer	SP ← (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 ← (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 ← (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 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	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	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 3 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles[1]
			V	H	I	N	Z	C				
BRCLR *n,opr8a,rel*	Branch if Bit *n* in Memory Clear	Branch if (Mn) = 0	–	–	–	–	–		DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	Uses 3 Bus Cycles	–	–	–	–	–	–	REL	21	rr	3
BRSET *n,opr8a,rel*	Branch if Bit *n* in Memory Set	Branch if (Mn) = 1	–	–	–	–	–		DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n,opr8a*	Set Bit *n* in Memory	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BSR *rel*	Branch to Subroutine	PC ← (PC) + 0x0002 push (PCL); SP ← (SP) – 0x0001 push (PCH); SP ← (SP) – 0x0001 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	5
CBEQ *oprx8,rel* CBEQA *#opr8i,rel* CBEQX *#opr8i,rel* CBEQ *oprx8*,X+,*rel* CBEQ ,X+,*rel* CBEQ *oprx8*,SP,*rel*	Compare and Branch if Equal	Branch if (A) = (M) Branch if (A) = (M) Branch if (X) = (M) Branch if (A) = (M) Branch if (A) = (M) Branch if (A) = (M)	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 5 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask Bit	I ← 0	–	–	0	–	–	–	INH	9A		1
CLR *oprx8*,X CLR ,X CLR *oprx8*,SP	Clear	M ← 0x00 A ← 0x00 X ← 0x00 H ← 0x00 M ← 0x00 M ← 0x00 M ← 0x00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	5 1 1 1 5 4 6
CMP *#opr8i* CMP *opr8a* CMP *opr16a* CMP *oprx16*,X CMP *oprx8*,X CMP ,X CMP *oprx16*,SP CMP *oprx8*,SP	Compare Accumulator with Memory	(A) – (M) (CCR Updated But Operands Not Changed)		–	–				IMM DIR EXT IX2 IX1 IX SP2 SP1	A1 B1 C1 D1 E1 F1 9ED1 9EE1	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
COM *oprx8*,X COM ,X COM *oprx8*,SP	Complement (One's Complement)	M ← (M)= 0xFF – (M) A ← (A) = 0xFF – (A) X ← (X) = 0xFF – (X) M ← (M) = 0xFF – (M) M ← (M) = 0xFF – (M) M ← (M) = 0xFF – (M)	0	–	–			1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	5 1 1 5 4 6
CPHX *opr16a* CPHX *#opr16i* CPHX *opr8a* CPHX *oprx8*,SP	Compare Index Register (H:X) with Memory	(H:X) – (M:M + 0x0001) (CCR Updated But Operands Not Changed)		–	–				EXT IMM DIR SP1	3E 65 75 9EF3	hh ll jj kk dd ff	6 3 5 6

$$t_{exspace} = t_{space} + (t_{mark} + t_{space}) \times (\text{number of modulation periods}) \quad \textit{Eqn. 8-9}$$

For an example of extended space operation, see Figure 8-7.

NOTE

The EXSPC feature can be used to emulate a zero mark event.

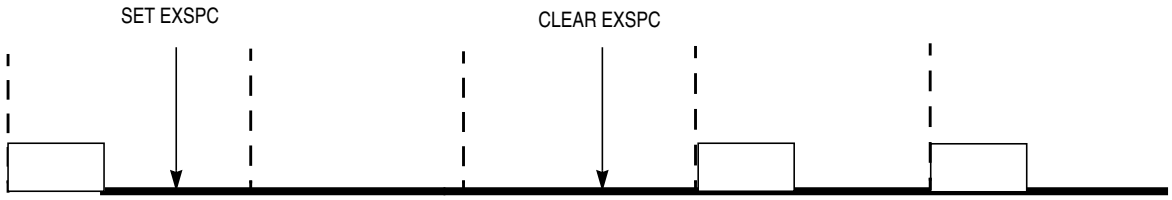


Figure 8-7. Extended Space Operation

8.5.3.2 EXSPC Operation in FSK Mode

In FSK mode, the modulator continues to count carrier out clocks, alternating between the primary and secondary registers at the end of each modulation period.

To calculate the length of an extended space in FSK mode, the user must know whether the EXSPC bit was set on a primary or secondary modulation period, as well as the total number of both primary and secondary modulation periods completed while the EXSPC bit is high. A status bit for the current modulation is not accessible to the CPU. If necessary, software should maintain tracking of the current modulation cycle (primary or secondary). The extended space period ends at the completion of the space period time of the modulation period during which the EXSPC bit is cleared.

If the EXSPC bit was set during a primary modulation cycle, use the equation:

$$t_{exspace} = (t_{space})_p + (t_{mark} + t_{space})_s + (t_{mark} + t_{space})_p + \ldots \quad \textit{Eqn. 8-10}$$

Where the subscripts p and s refer to mark and space times for the primary and secondary modulation cycles.

If the EXSPC bit was set during a secondary modulation cycle, use the equation:

$$t_{exspace} = (t_{space})_s + (t_{mark} + t_{space})_p + (t_{mark} + t_{space})_s + \ldots \quad \textit{Eqn. 8-11}$$

8.5.4 Transmitter

The transmitter output block controls the state of the infrared out pin (IRO). The modulator output is gated on to the IRO pin when the modulator/carrier generator is enabled. When the modulator/carrier generator is disabled, the IRO pin is controlled by the state of the IRO latch.

A polarity bit in the CMTOC register enables the IRO pin to be high true or low true.

Table 8-8. CMTMSC Field Descriptions (continued)

Field	Description
1 EOCIE	**End of Cycle Interrupt Enable** — A CPU interrupt will be requested when EOCF is set if EOCIE is high. 0 CPU interrupt disabled 1 CPU interrupt enabled
0 MCGEN	**Modulator and Carrier Generator Enable** — Setting MCGEN will initialize the carrier generator and modulator and enable all clocks. After it is enabled, the carrier generator and modulator will function continuously. When MCGEN is cleared, the current modulator cycle will be allowed to expire before all carrier and modulator clocks are disabled (to save power) and the modulator output is forced low. To prevent spurious operation, the user should initialize all data and control registers before enabling the system. 0 Modulator and carrier generator disabled 1 Modulator and carrier generator enabled

8.6.4 CMT Modulator Data Registers (CMTCMD1, CMTCMD2, CMTCMD3, and CMTCMD4)

The modulator data registers control the mark and space periods of the modulator for all modes. The contents of these registers are transferred to the modulator down counter and space period register upon the completion of a modulation period.

Table 8-9. Sample Register Summary

Name		7	6	5	4	3	2	1	0
CMTCMD1	R	MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8
	W								
CMTCMD2	R	MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0
	W								
CMTCMD3	R	SB15	SB14	SB13	SB12	SB11	SB10	SB9	SB8
	W								
CMTCMD4	R	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0
	W								

9.4.1 KBI x Status and Control Register (KBIxSC)

	7	6	5	4	3	2	1	0
R	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBF	0	KBIE	KBIMOD
W						KBACK		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-3. KBI x Status and Control Register (KBIxSC)

Table 9-1. KBIxSC Field Descriptions

Field	Description
7:4 KBEDG[7:4]	**Keyboard Edge Select for KBI Port Bits** — Each of these read/write bits selects the polarity of the edges and/or levels that are recognized as trigger events on the corresponding KBI port pin when it is configured as a keyboard interrupt input (KBIPEn = 1). Also see the KBIMOD control bit, which determines whether the pin is sensitive to edges-only or edges and levels. 0 Falling edges/low levels. 1 Rising edges/high levels.
3 KBF	**Keyboard Interrupt Flag** — This read-only status flag is set whenever the selected edge event has been detected on any of the enabled KBI port pins. This flag is cleared by writing a logic 1 to the KBACK control bit. The flag will remain set if KBIMOD = 1 to select edge-and-level operation and any enabled KBI port pin remains at the asserted level. 0 No KBI interrupt pending. 1 KBI interrupt pending. KBF can be used as a software pollable flag (KBIE = 0) or it can generate a hardware interrupt request to the CPU (KBIE = 1). KBF must be cleared before entering stop mode.
2 KBACK	**Keyboard Interrupt Acknowledge** — This write-only bit (reads always return 0) is used to clear the KBF status flag by writing a logic 1 to KBACK. When KBIMOD = 1 to select edge-and-level operation and any enabled KBI port pin remains at the asserted level, KBF is being continuously set so writing 1 to KBACK does not clear the KBF flag.
1 KBIE	**Keyboard Interrupt Enable** — This read/write control bit determines whether hardware interrupts are generated when the KBF status flag equals 1. When KBIE = 0, no hardware interrupts are generated, but KBF can still be used for software polling. 0 KBF does not generate hardware interrupts (use polling). 1 KBI hardware interrupt requested when KBF = 1.
0 KBIMOD	**Keyboard Detection Mode** — This read/write control bit selects either edge-only detection or edge-and-level detection. KBI port bits 3 through 0 can detect falling edges-only or falling edges and low levels. KBI port bits 7 through 4 can be configured to detect either: • Rising edges-only or rising edges and high levels (KBEDGn = 1) • Falling edges-only or falling edges and low levels (KBEDGn = 0) 0 Edge-only detection. 1 Edge-and-level detection.

12.2.5 SCI Status Register 2 (SCI1S2)

This register has one read-only status flag. Writes have no effect.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	RAF
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 12-8. SCI Status Register 2 (SCI1S2)

Table 12-6. SCI1S2 Register Field Descriptions

Field	Description
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).

12.2.6 SCI Control Register 3 (SCI1C3)

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

= Unimplemented or Reserved

Figure 12-9. SCI Control Register 3 (SCI1C3)

Table 12-7. SCI1C3 Register 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 SCI1D register. When reading 9-bit data, read R8 before reading SCI1D because reading SCI1D completes automatic flag clearing sequences which could allow R8 and SCI1D 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 SCI1D register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCI1D is written so T8 should be written (if it needs to change from its previous value) before SCI1D 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 SCI1D 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.

13.4.5 SPI Data Register (SPI1D)

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-11. SPI Data Register (SPI1D)

Reads of this register return the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. When the SPI is configured as a master, writing data to the transmit data buffer initiates an SPI transfer.

Data should not be written to the transmit data buffer unless the SPI transmit buffer empty flag (SPTEF) is set, indicating there is room in the transmit buffer to queue a new transmit byte.

Data may be read from SPI1D any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

Chapter 15
Development Support

15.1 Introduction

15.1.1 Features

Features of the background debug controller (BDC) 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 debug module (DBG) 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:
  - A-only
  - A OR B
  - A then B
  - A AND B data (full mode)
  - A AND NOT B data (full mode)
  - Event-only B (store data)
  - A then event-only B (store data)