

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, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	60KB (60K 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/pro/item?MUrl=&PartUrl=mc9s08rg60cfj

Section Number	Title	Page
5.8	Reset, Interrupt, and System Control Registers and Control Bits	64
5.8.1	Interrupt Pin Request Status and Control Register (IRQSC)	64
5.8.2	System Reset Status Register (SRS)	65
5.8.3	System Background Debug Force Reset Register (SBDFR)	67
5.8.4	System Options Register (SOPT)	68
5.8.5	System Device Identification Register (SDIDH, SDIDL)	69
5.8.6	System Real-Time Interrupt Status and Control Register (SRTISC)	70
5.8.7	System Power Management Status and Control 1 Register (SPMSC1)	71
5.8.8	System Power Management Status and Control 2 Register (SPMSC2)	72

Chapter 6

Parallel Input/Output

6.1	Introduction	73
6.2	Features	73
6.3	Pin Descriptions	74
6.3.1	Port A	74
6.3.2	Port B	74
6.3.3	Port C	75
6.3.4	Port D	75
6.3.5	Port E	76
6.4	Parallel I/O Controls	76
6.4.1	Data Direction Control	76
6.4.2	Internal Pullup Control	77
6.5	Stop Modes	77
6.6	Parallel I/O Registers and Control Bits	77
6.6.1	Port A Registers (PTAD, PTAPE, and PTADD)	78
6.6.2	Port B Registers (PTBD, PTBPE, and PTBDD)	79
6.6.3	Port C Registers (PTCD, PTCPE, and PTCDD)	81
6.6.4	Port D Registers (PTDD, PTDPE, and PTDDD)	82
6.6.5	Port E Registers (PTED, PTEPE, and PTEDD)	84

Chapter 7

Central Processor Unit (S08CPUV2)

7.1	Introduction	87
7.1.1	Features	87
7.2	Programmer's Model and CPU Registers	88
7.2.1	Accumulator (A)	88
7.2.2	Index Register (H:X)	88
7.2.3	Stack Pointer (SP)	89
7.2.4	Program Counter (PC)	89
7.2.5	Condition Code Register (CCR)	89
7.3	Addressing Modes	90
7.3.1	Inherent Addressing Mode (INH)	91
7.3.2	Relative Addressing Mode (REL)	91
7.3.3	Immediate Addressing Mode (IMM)	91

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

Background commands are of two types:

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

The active background mode is used to program a bootloader or user application program into the FLASH program memory before the MCU is operated in run mode for the first time. When the MC9S08RC/RD/RE/RG is shipped from the Freescale Semiconductor factory, the FLASH program memory is usually erased so there is no program that could be executed in run mode until the FLASH memory is initially programmed. The active background mode can also be used to erase and reprogram the FLASH memory after it has been previously programmed.

For additional information about the active background mode, refer to the Development Support chapter.

3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The I bit in CCR is cleared when the CPU enters the wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits the wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

Only the BACKGROUND command and memory-access-with-status commands are available when the MCU is in wait 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 wait mode and enter active background mode.

2. Writing the user-entered key values to the NVBACKKEY through NVBACKKEY+7 locations. These writes must be done in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX must not be used for these writes because these writes cannot be done on adjacent bus cycles. User software normally would get the key codes from outside the MCU system through a communication interface such as a serial I/O.
3. Writing 0 to KEYACC in the FCNFG register. If the 8-byte key that was just written matches the key stored in the FLASH locations, SEC01:SEC00 are automatically changed to 1:0 and security will be disengaged until the next reset.

The security key can be written only from RAM, so it cannot be entered through background commands without the cooperation of a secure user program. The FLASH memory cannot be accessed by read operations while KEYACC is set.

The backdoor comparison key (NVBACKKEY through NVBACKKEY+7) is located in FLASH memory locations in the nonvolatile register space so users can program these locations exactly as they would program any other FLASH memory location. The nonvolatile registers are in the same 512-byte block of FLASH as the reset and interrupt vectors, so block protecting that space also block protects the backdoor comparison key. Block protects cannot be changed from user application programs, so if the vector space is block protected, the backdoor security key mechanism cannot permanently change the block protect, security settings, or the backdoor key.

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

1. Disable any block protections by writing FPROT. FPROT can be written only with background debug commands, not from application software.
2. Mass erase FLASH if necessary.
3. Blank check FLASH. Provided FLASH is completely erased, security is disengaged until the next reset.

To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

4.6 FLASH Registers and Control Bits

The FLASH module has nine 8-bit registers in the high-page register space, three locations in the nonvolatile register space in FLASH memory that are copied into three corresponding high-page control registers at reset. There is also an 8-byte comparison key in FLASH memory. Refer to Table 4-2 and Table 4-3 for the absolute address assignments for all FLASH registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

4.6.1 FLASH Clock Divider Register (FCDIV)

Bit 7 of this register is a read-only status flag. Bits 6 through 0 may be read at any time but can be written only one time. Before any erase or programming operations are possible, write to this register to set the frequency of the clock for the nonvolatile memory system within acceptable limits.

5.8.3 System Background Debug Force Reset Register (SBDFR)

This register contains a single write-only control bit. A serial background command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return \$00.

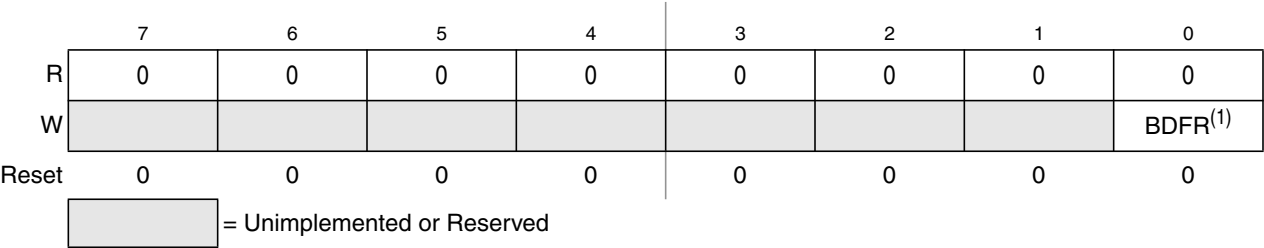


Figure 5-4. System Background Debug Force Reset Register (SBDFR)

1. BDFR is writable only through serial background debug commands, not from user programs.

Table 5-4. SBDFR Field Descriptions

Field	Description
0 BDFR	Background Debug Force Reset — A serial background command such as WRITE_BYTE may be used to allow an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program.

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.

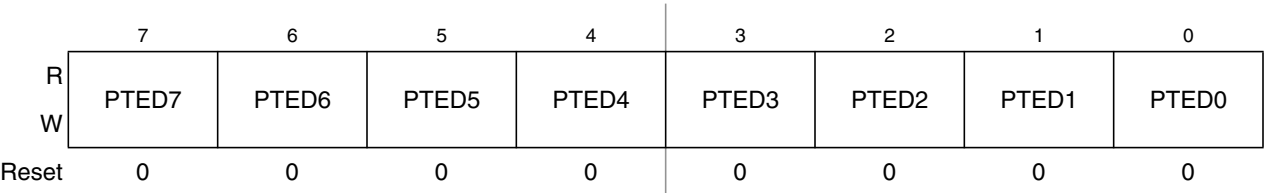


Figure 6-18. Port E Data Register (PTED)

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.

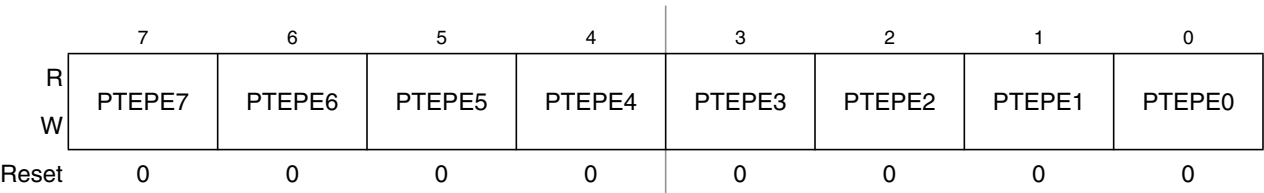


Figure 6-19. Pullup Enable for Port E (PTEPE)

Table 6-14. PTEPE Field Descriptions

Field	Description
7:0 PTEPE[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.



7.2 Programmer's Model and CPU Registers

Figure 7-1 shows the five CPU registers. CPU registers are not part of the memory map.

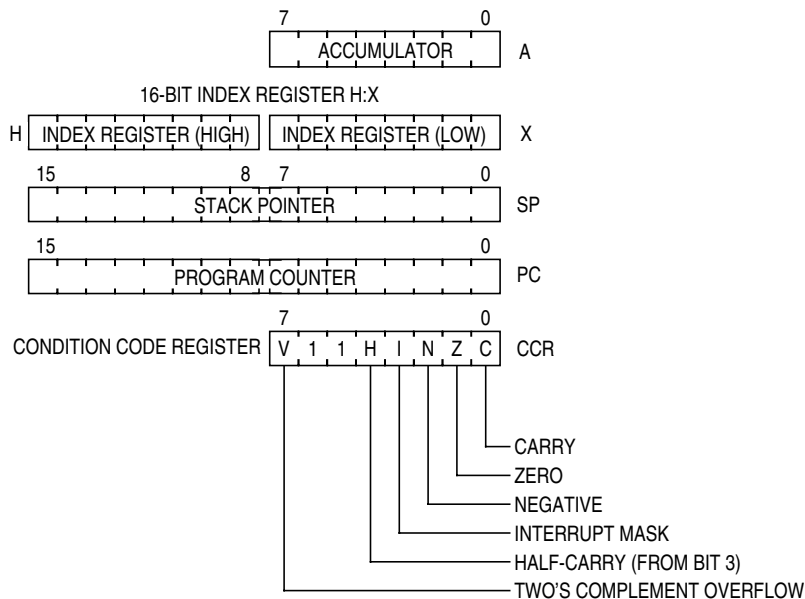


Figure 7-1. CPU Registers

7.2.1 Accumulator (A)

The A accumulator is a general-purpose 8-bit register. One operand input to the arithmetic logic unit (ALU) is connected to the accumulator and the ALU results are often stored into the A accumulator after arithmetic and logical operations. The accumulator can be loaded from memory using various addressing modes to specify the address where the loaded data comes from, or the contents of A can be stored to memory using various addressing modes to specify the address where data from A will be stored.

Reset has no effect on the contents of the A accumulator.

7.2.2 Index Register (H:X)

This 16-bit register is actually two separate 8-bit registers (H and X), which often work together as a 16-bit address pointer where H holds the upper byte of an address and X holds the lower byte of the address. All indexed addressing mode instructions use the full 16-bit value in H:X as an index reference pointer; however, for compatibility with the earlier M68HC05 Family, some instructions operate only on the low-order 8-bit half (X).

Many instructions treat X as a second general-purpose 8-bit register that can be used to hold 8-bit data values. X can be cleared, incremented, decremented, complemented, negated, shifted, or rotated. Transfer instructions allow data to be transferred from A or transferred to A where arithmetic and logical operations can then be performed.

For compatibility with the earlier M68HC05 Family, H is forced to 0x00 during reset. Reset has no effect on the contents of X.

7.3.6.1 Indexed, No Offset (IX)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction.

7.3.6.2 Indexed, No Offset with Post Increment (IX+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction. The index register pair is then incremented ($H:X = H:X + 0x0001$) after the operand has been fetched. This addressing mode is only used for MOV and CBEQ instructions.

7.3.6.3 Indexed, 8-Bit Offset (IX1)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.3.6.4 Indexed, 8-Bit Offset with Post Increment (IX1+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction. The index register pair is then incremented ($H:X = H:X + 0x0001$) after the operand has been fetched. This addressing mode is used only for the CBEQ instruction.

7.3.6.5 Indexed, 16-Bit Offset (IX2)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.3.6.6 SP-Relative, 8-Bit Offset (SP1)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.3.6.7 SP-Relative, 16-Bit Offset (SP2)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.4 Special Operations

The CPU performs a few special operations that are similar to instructions but do not have opcodes like other CPU instructions. In addition, a few instructions such as STOP and WAIT directly affect other MCU circuitry. This section provides additional information about these operations.

- 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.
- opr8* — Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing
- opr16* — 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

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 <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> 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 <i>rel</i>	Branch Never	Uses 3 Bus Cycles	–	–	–	–	–	–	REL	21	rr	3
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> 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 <i>n,opr8a</i>	Set Bit <i>n</i> 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 <i>rel</i>	Branch to Subroutine	PC ← (PC) + 0x0002 push (PCL); SP ← (SP) – 0x0001 push (PCH); SP ← (SP) – 0x0001 PC ← (PC) + <i>rel</i>	–	–	–	–	–	–	REL	AD	rr	5
CBEQ <i>opr8a,rel</i> CBEQA # <i>opr8i,rel</i> CBEQX # <i>opr8i,rel</i> CBEQ <i>opr8,X+,rel</i> CBEQ <i>,X+,rel</i> CBEQ <i>opr8,SP,rel</i>	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 <i>opr8a</i> CLRA CLR X CLR H CLR <i>opr8,X</i> CLR <i>,X</i> CLR <i>opr8,SP</i>	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 ff	5 1 1 1 5 4 6
CMP # <i>opr8i</i> CMP <i>opr8a</i> CMP <i>opr16a</i> CMP <i>opr16,X</i> CMP <i>opr8,X</i> CMP <i>,X</i> CMP <i>opr16,SP</i> CMP <i>opr8,SP</i>	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 ff ee ff ff	2 3 4 4 3 3 5 4
COM <i>opr8a</i> COMA COM X COM <i>opr8,X</i> COM <i>,X</i> COM <i>opr8,SP</i>	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 ff	5 1 1 5 4 6
CPHX <i>opr16a</i> CPHX # <i>opr16i</i> CPHX <i>opr8a</i> CPHX <i>opr8,SP</i>	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

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

Bit-Manipulation			Branch		Read-Modify-Write								Control				Register/Memory														
00	5	10	5	20	3	30	5	40	1	50	5	60	5	70	4	80	9	90	3	A0	2	B0	3	C0	4	D0	4	E0	3	F0	3
BRSET0	DIR	BSET0	DIR	BRA	REL	NEG	DIR	NEGA	INH	NEGX	INH	NEG	IX1	NEG	IX	RTI	INH	BGE	REL	SUB	IMM	SUB	DIR	SUB	EXT	SUB	IX2	SUB	IX1	SUB	IX
01	5	11	5	21	3	31	5	41	4	51	4	61	5	71	5	81	6	91	3	A1	2	B1	3	C1	4	D1	4	E1	3	F1	3
BRCLR0	DIR	BCLR0	DIR	BRN	REL	CBEQ	DIR	CBEQA	IMM	CBEQX	IMM	CBEQ	IX1+	CBEQ	IX+	RTS	INH	BLT	REL	CMP	IMM	CMP	DIR	CMP	EXT	CMP	IX2	CMP	IX1	CMP	IX
02	5	12	5	22	3	32	5	42	5	52	6	62	1	72	1	82	5+	92	3	A2	2	B2	3	C2	4	D2	4	E2	3	F2	3
BRSET1	DIR	BSET1	DIR	BHI	REL	LDHX	EXT	MUL	INH	DIV	INH	NSA	INH	DAA	INH	BGND	INH	BGT	REL	SBC	IMM	SBC	DIR	SBC	EXT	SBC	IX2	SBC	IX1	SBC	IX
03	5	13	5	23	3	33	5	43	1	53	1	63	5	73	4	83	11	93	3	A3	2	B3	3	C3	4	D3	4	E3	3	F3	3
BRCLR1	DIR	BCLR1	DIR	BLS	REL	COM	DIR	COMA	INH	COMX	INH	COM	IX1	COM	IX	SWI	INH	BLE	REL	CPX	IMM	CPX	DIR	CPX	EXT	CPX	IX2	CPX	IX1	CPX	IX
04	5	14	5	24	3	34	5	44	1	54	1	64	5	74	4	84	1	94	2	A4	2	B4	3	C4	4	D4	4	E4	3	F4	3
BRSET2	DIR	BSET2	DIR	BCC	REL	LSR	DIR	LSRA	INH	LSRX	INH	LSR	IX1	LSR	IX	TAP	INH	TXS	INH	AND	IMM	AND	DIR	AND	EXT	AND	IX2	AND	IX1	AND	IX
05	5	15	5	25	3	35	4	45	3	55	4	65	3	75	5	85	1	95	2	A5	2	B5	3	C5	4	D5	4	E5	3	F5	3
BRCLR2	DIR	BCLR2	DIR	BCS	REL	STHX	DIR	LDHX	IMM	LDHX	DIR	CPHX	IMM	CPHX	DIR	TPA	INH	TSX	INH	BIT	IMM	BIT	DIR	BIT	EXT	BIT	IX2	BIT	IX1	BIT	IX
06	5	16	5	26	3	36	5	46	1	56	1	66	5	76	4	86	3	96	5	A6	2	B6	3	C6	4	D6	4	E6	3	F6	3
BRSET3	DIR	BSET3	DIR	BNE	REL	ROR	DIR	RORA	INH	RORX	INH	ROR	IX1	ROR	IX	PULA	INH	STHX	EXT	LDA	IMM	LDA	DIR	LDA	EXT	LDA	IX2	LDA	IX1	LDA	IX
07	5	17	5	27	3	37	5	47	1	57	1	67	5	77	4	87	2	97	1	A7	2	B7	3	C7	4	D7	4	E7	3	F7	2
BRCLR3	DIR	BCLR3	DIR	BEQ	REL	ASR	DIR	ASRA	INH	ASRX	INH	ASR	IX1	ASR	IX	PSHA	INH	TAX	INH	AIS	IMM	STA	DIR	STA	EXT	STA	IX2	STA	IX1	STA	IX
08	5	18	5	28	3	38	5	48	1	58	1	68	5	78	4	88	3	98	1	A8	2	B8	3	C8	4	D8	4	E8	3	F8	3
BRSET4	DIR	BSET4	DIR	BHCC	REL	LSL	DIR	LSLA	INH	LSLX	INH	LSL	IX1	LSL	IX	PULX	INH	CLC	INH	EOR	IMM	EOR	DIR	EOR	EXT	EOR	IX2	EOR	IX1	EOR	IX
09	5	19	5	29	3	39	5	49	1	59	1	69	5	79	4	89	2	99	1	A9	2	B9	3	C9	4	D9	4	E9	3	F9	3
BRCLR4	DIR	BCLR4	DIR	BHCS	REL	ROL	DIR	ROLA	INH	ROLX	INH	ROL	IX1	ROL	IX	PSHX	INH	SEC	INH	ADC	IMM	ADC	DIR	ADC	EXT	ADC	IX2	ADC	IX1	ADC	IX
0A	5	1A	5	2A	3	3A	5	4A	1	5A	1	6A	5	7A	4	8A	3	9A	1	AA	2	BA	3	CA	4	DA	4	EA	3	FA	3
BRSET5	DIR	BSET5	DIR	BPL	REL	DEC	DIR	DECA	INH	DECX	INH	DEC	IX1	DEC	IX	PULH	INH	CLI	INH	ORA	IMM	ORA	DIR	ORA	EXT	ORA	IX2	ORA	IX1	ORA	IX
0B	5	1B	5	2B	3	3B	7	4B	4	5B	4	6B	7	7B	6	8B	2	9B	1	AB	2	BB	3	CB	4	DB	4	EB	3	FB	3
BRCLR5	DIR	BCLR5	DIR	BMI	REL	DBNZ	DIR	DBNZA	INH	DBNZX	INH	DBNZ	IX1	DBNZ	IX	PSHH	INH	SEI	INH	ADD	IMM	ADD	DIR	ADD	EXT	ADD	IX2	ADD	IX1	ADD	IX
0C	5	1C	5	2C	3	3C	5	4C	1	5C	1	6C	5	7C	4	8C	1	9C	1			BC	3	CC	4	DC	4	EC	3	FC	3
BRSET6	DIR	BSET6	DIR	BMC	REL	INC	DIR	INCA	INH	INCX	INH	INC	IX1	INC	IX	CLRH	INH	RSP	INH			JMP	DIR	JMP	EXT	JMP	IX2	JMP	IX1	JMP	IX
0D	5	1D	5	2D	3	3D	4	4D	1	5D	1	6D	4	7D	3			9D	1	AD	5	BD	5	CD	6	DD	6	ED	5	FD	5
BRCLR6	DIR	BCLR6	DIR	BMS	REL	TST	DIR	TSTA	INH	TSTX	INH	TST	IX1	TST	IX			NOP	INH	BSR	REL	JSR	DIR	JSR	EXT	JSR	IX2	JSR	IX1	JSR	IX
0E	5	1E	5	2E	3	3E	6	4E	5	5E	5	6E	4	7E	5	8E	2+	9E	Page 2	AE	2	BE	3	CE	4	DE	4	EE	3	FE	3
BRSET7	DIR	BSET7	DIR	BIL	REL	CPHX	EXT	MOV	DD	MOV	DIX+	MOV	IMD	MOV	IX+D	STOP	INH			LDX	IMM	LDX	DIR	LDX	EXT	LDX	IX2	LDX	IX1	LDX	IX
0F	5	1F	5	2F	3	3F	5	4F	1	5F	1	6F	5	7F	4	8F	2+	9F	1	AF	2	BF	3	CF	4	DF	4	EF	3	FF	2
BRCLR7	DIR	BCLR7	DIR	BIH	REL	CLR	DIR	CLRA	INH	CLR	INH	CLR	IX1	CLR	IX	WAIT	INH	TXA	INH	AIX	IMM	STX	DIR	STX	EXT	STX	IX2	STX	IX1	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
 IMM 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 3
 SUB IX
 Number of Bytes 1
 HCS08 Cycles Instruction Mnemonic Addressing Mode

8.2 Features

The CMT consists of a carrier generator, modulator, and transmitter that drives the infrared out (IRO) pin. The features of this module include:

- Four modes of operation
 - Time with independent control of high and low times
 - Baseband
 - Frequency shift key (FSK)
 - Direct software control of IRO pin
- Extended space operation in time, baseband, and FSK modes
- Selectable input clock divide: 1, 2, 4, or 8
- Interrupt on end of cycle
 - Ability to disable IRO pin and use as timer interrupt

8.3 CMT Block Diagram

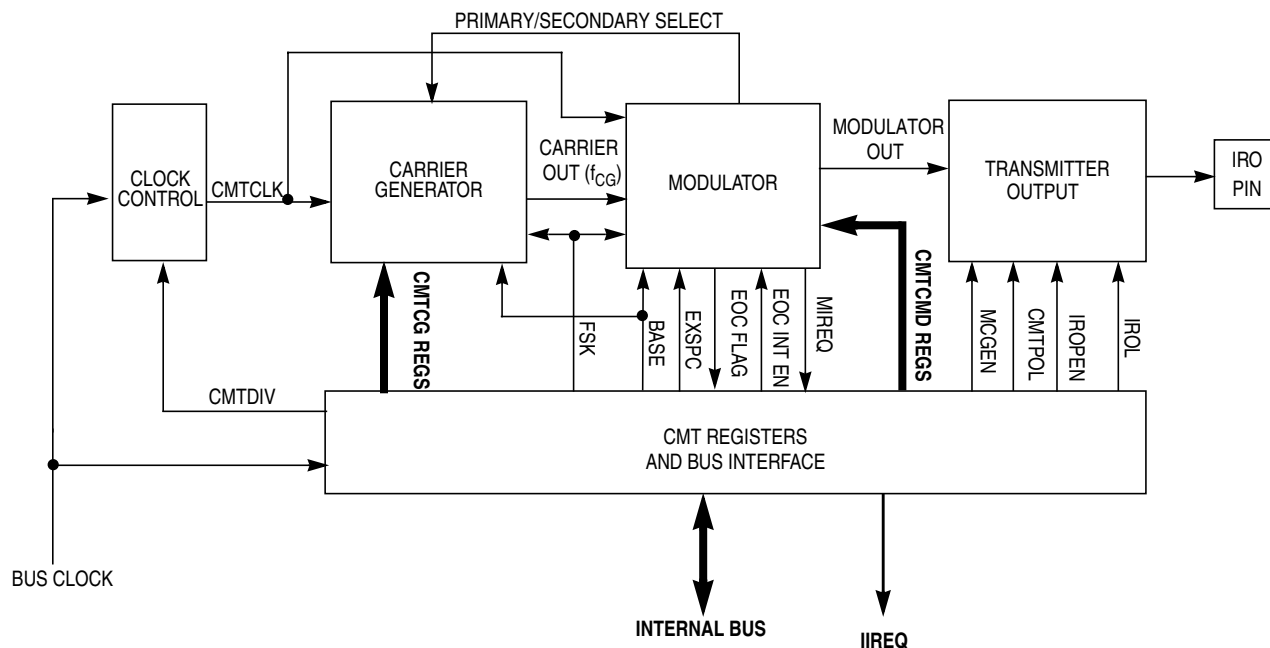


Figure 8-2. Carrier Modulator Transmitter Module Block Diagram

8.4 Pin Description

The IRO pin is the only pin associated with the CMT. The pin is driven by the transmitter output when the **MCGEN** bit in the **CMTMSC** register and the **IROPEN** bit in the **CMTOC** register are set. If the **MCGEN** bit is clear and the **IROPEN** bit is set, the pin is driven by the **IROL** bit in the **CMTOC** register. This enables user software to directly control the state of the IRO pin by writing to the **IROL** bit. If the **IROPEN** bit is clear, the pin is disabled and is not driven by the CMT module. This is so the CMT can be configured as a modulo timer for generating periodic interrupts without causing pin activity.

register (TPM1CnVH:TPM1CnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. Duty cycle cases of 0 percent and 100 percent are possible.

As Figure 10-3 shows, the output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal. The time between the modulus overflow and the output compare is the pulse width. If ELSnA = 0, the counter overflow forces the PWM signal high and the output compare forces the PWM signal low. If ELSnA = 1, the counter overflow forces the PWM signal low and the output compare forces the PWM signal high.

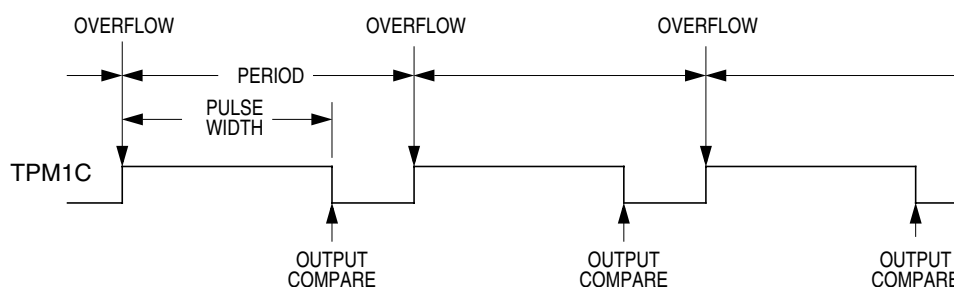


Figure 10-3. PWM Period and Pulse Width (ELSnA = 0)

When the channel value register is set to \$0000, the duty cycle is 0 percent. By setting the timer channel value register (TPM1CnVH:TPM1CnVL) to a value greater than the modulus setting, 100 percent duty cycle can be achieved. This implies that the modulus setting must be less than \$FFFF to get 100 percent duty cycle.

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to either register, TPM1CnVH or TPM1CnVL, write to buffer registers. In edge-PWM mode, values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the value in the TPM1CNTH:TPM1CNTL counter is \$0000. (The new duty cycle does not take effect until the next full period.)

10.5.3 Center-Aligned PWM Mode

This type of PWM output uses the up-/down-counting mode of the timer counter (CPWMS = 1). The output compare value in TPM1CnVH:TPM1CnVL determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in TPM1MODH:TPM1MODL.

TPM1MODH:TPM1MODL should be kept in the range of \$0001 to \$7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPM1CnVH:TPM1CnVL}) \quad \text{Eqn. 10-1}$$

$$\begin{aligned} \text{period} &= 2 \times (\text{TPM1MODH:TPM1MODL}); \\ \text{for TPM1MODH:TPM1MODL} &= \$0001-\$7FFF \end{aligned} \quad \text{Eqn. 10-2}$$

If the channel value register TPM1CnVH:TPM1CnVL is zero or negative (bit 15 set), the duty cycle will be 0 percent. If TPM1CnVH:TPM1CnVL is a positive value (bit 15 clear) and is greater than the (nonzero) modulus setting, the duty cycle will be 100 percent because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is \$0001 through \$7FFE (\$7FFF if

the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCI1D.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD1 pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD1 high, waiting for more characters to transmit.

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

12.3.2.1 Send Break and Queued Idle

The SBK control bit in SCI1C2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD1 pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD1 is an output driving a logic 1. This ensures that the TxD1 line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

12.3.3 Receiver Functional Description

In this section, the data sampling technique used to reconstruct receiver data is described in more detail; two variations of the receiver wakeup function are explained. (The receiver block diagram is shown in Figure 12-2.)

The receiver is enabled by setting the RE bit in SCI1C2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 12.3.5.1, “8- and 9-Bit Data Modes.” For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program

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)

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

Table 15-2. BDCSCR Register Field Descriptions (continued)

Field	Description
2 WS	<p>Wait or Stop Status — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands.</p> <p>0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active)</p> <p>1 Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode</p>
1 WSF	<p>Wait or Stop Failure Status — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.)</p> <p>0 Memory access did not conflict with a wait or stop instruction</p> <p>1 Memory access command failed because the CPU entered wait or stop mode</p>
0 DVF	<p>Data Valid Failure Status — This status bit is not used in the MC9S08RC/RD/RE/RG because it does not have any slow access memory.</p> <p>0 Memory access did not conflict with a slow memory access</p> <p>1 Memory access command failed because CPU was not finished with a slow memory access</p>

15.4.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to Section 15.2.4, “BDC Hardware Breakpoint.”

15.4.2 System Background Debug Force Reset Register (SBDFR)

This register contains a single write-only control bit. A serial active background mode command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

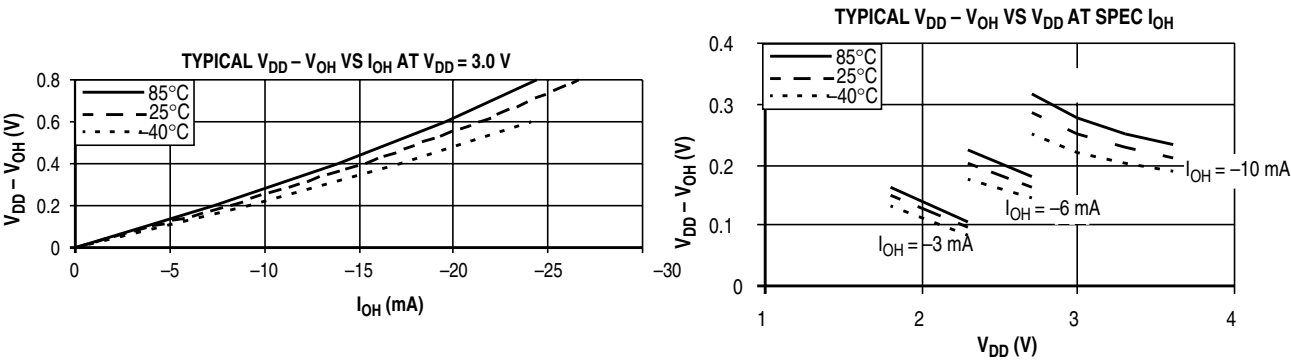


Figure A-4. Typical High-Side Driver (Source) Characteristics (Port B and IRO)

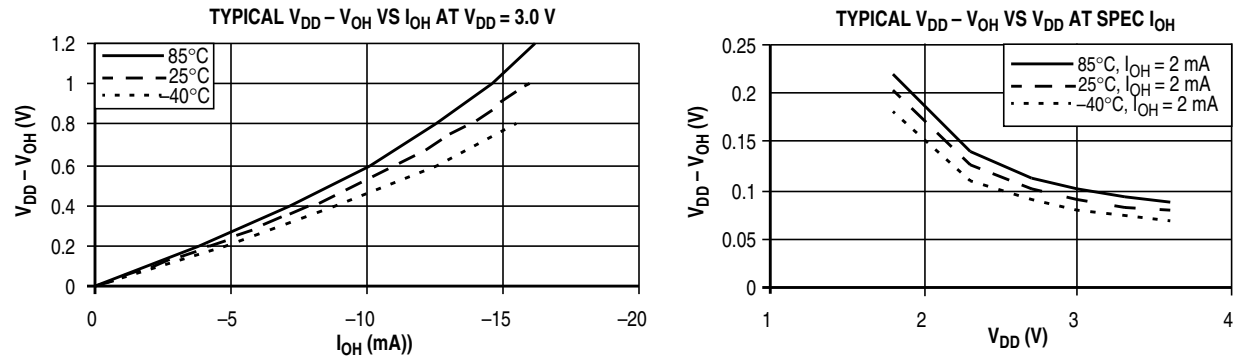


Figure A-5. Typical High-Side (Source) Characteristics (Ports A, C, D and E)

A.10 FLASH Specifications

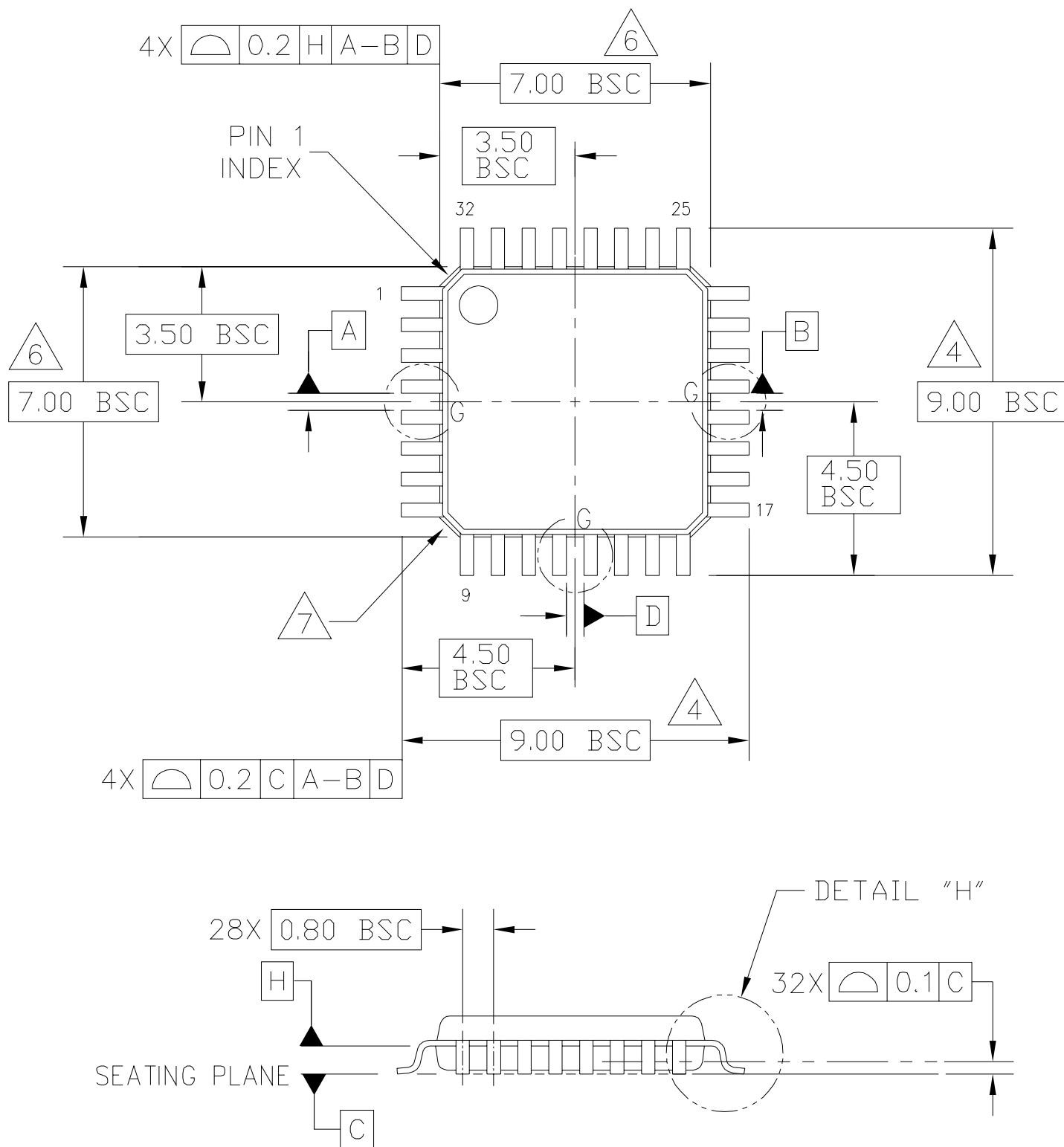
This section provides details about program/erase times and program-erase endurance for the FLASH memory.

Program and erase operations do not require any special power sources other than the normal V_{DD} supply. For more detailed information about program/erase operations, see the [Memory](#) chapter.

Table A-12. FLASH Characteristics

Characteristic	Symbol	Min	Typical	Max	Unit
Supply voltage for program/erase	$V_{\text{prog/erase}}$	2.05		3.6	V
Supply voltage for read operation $0 < f_{\text{BUS}} < 8 \text{ MHz}$	V_{Read}	1.8		3.6	V
Internal FCLK frequency ⁽¹⁾	f_{FCLK}	150		200	kHz
Internal FCLK period (1/FCLK)	t_{Fcyc}	5		6.67	μs
Byte program time (random location) ⁽²⁾	t_{prog}	9			t_{Fcyc}
Byte program time (burst mode) ⁽²⁾	t_{Burst}	4			t_{Fcyc}
Page erase time ⁽²⁾	t_{Page}	4000			t_{Fcyc}
Mass erase time ⁽²⁾	t_{Mass}	20,000			t_{Fcyc}
Program/erase endurance ⁽³⁾ T_L to $T_H = -40^\circ\text{C}$ to $+85^\circ\text{C}$ $T = 25^\circ\text{C}$		10,000	100,000	—	cycles
Data retention ⁽⁴⁾	$t_{\text{D-ret}}$	15	100	—	years

1. The frequency of this clock is controlled by a software setting.
2. These values are hardware state machine controlled. User code does not need to count cycles. This information supplied for calculating approximate time to program and erase.
3. **Typical endurance for FLASH** was evaluated for this product family on the 9S12Dx64. For additional information on how Freescale Semiconductor defines typical endurance, please refer to Engineering Bulletin EB619/D, *Typical Endurance for Nonvolatile Memory*.
4. **Typical data retention** values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale Semiconductor defines typical data retention, please refer to Engineering Bulletin EB618/D, *Typical Data Retention for Nonvolatile Memory*.



TITLE:

LOW PROFILE QUAD FLAT PACK (LQFP)
32 LEAD, 0.8 PITCH (7 X 7 X 1.4)

CASE NUMBER: 873A-04

STANDARD: JEDEC MS-026 BBA

PACKAGE CODE: 6300

SHEET: 1 OF 3