

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

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

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

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

Details

Product Status	Active
Core Processor	S08
Core Size	8-Bit
Speed	8MHz
Connectivity	SCI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	23
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s08rd8dwe

List of Chapters

Chapter 1	Introduction.....	15
Chapter 2	Pins and Connections	19
Chapter 3	Modes of Operation	29
Chapter 4	Memory	35
Chapter 5	Resets, Interrupts, and System Configuration	57
Chapter 6	Parallel Input/Output	73
Chapter 7	Central Processor Unit (S08CPUV2).....	87
Chapter 8	Carrier Modulator Timer (S08CMTV1).....	107
Chapter 9	Keyboard Interrupt (S08KBIV1).....	123
Chapter 10	Timer/PWM Module (S08TPMV1).....	129
Chapter 11	Serial Communications Interface (S08SCIV1).....	145
Chapter 12	Serial Communications Interface (S08SCIV1).....	147
Chapter 13	Serial Peripheral Interface (S08SPIV3)	163
Chapter 14	Analog Comparator (S08ACMPV1)	179
Chapter 15	Development Support	183
Appendix A	Electrical Characteristics.....	205
Appendix B	Ordering Information and Mechanical Drawings.....	219

Table 2-2. Signal Properties

Pin Name	Dir ⁽¹⁾	High Current Pin	Pullup ⁽²⁾	Comments ⁽³⁾
V _{DD}		—	—	
V _{SS}		—	—	
XTAL	O	—	—	Crystal oscillator output
EXTAL	I	—	—	Crystal oscillator input
IRO	O	Y	—	Infrared output
PTA0/KBI1P0	I	N	SWC	PTA0 does not have a clamp diode to V _{DD} . PTA0 should not be driven above V _{DD} .
PTA1/KBI1P1	I/O	N	SWC	
PTA2/KBI1P2	I/O	N	SWC	
PTA3/KBI1P3	I/O	N	SWC	
PTA4/KBI1P4	I/O	N	SWC	
PTA5/KBI1P5	I/O	N	SWC	
PTA6/KBI1P6	I/O	N	SWC	
PTA7/KBI1P7	I/O	N	SWC	
PTB0/TxD1	I/O	Y	SWC	
PTB1/RxD1	I/O	Y	SWC	
PTB2	I/O	Y	SWC	
PTB3	I/O	Y	SWC	Available only in 44- and 48-pin packages
PTB4	I/O	Y	SWC	Available only in 44- and 48-pin packages
PTB5	I/O	Y	SWC	Available only in 44- and 48-pin packages
PTB6	I/O	Y	SWC	Available only in 32-, 44-, and 48-pin packages
PTB7/TPM1CH1	I/O	Y	SWC	
PTC0/KBI2P0	I/O	N	SWC	
PTC1/KBI2P1	I/O	N	SWC	
PTC2/KBI2P2	I/O	N	SWC	
PTC3/KBI2P3	I/O	N	SWC	
PTC4/MOSI1	I/O	N	SWC	
PTC5/MISO1	I/O	N	SWC	
PTC6/SPSCK1	I/O	N	SWC	
PTC7/ $\overline{SS1}$	I/O	N	SWC	
PTD0/BKGD/MS	I/O	N	SWC ⁽⁴⁾	Output-only when configured as PTD0 pin. Pullup enabled.
PTD1/ \overline{RESET}	I/O	N	SWC ⁽³⁾	Output-only when configured as PTD1 pin.

Table 4-1. Direct-Page Register Summary

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
\$0000	PTAD	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
\$0001	PTAPE	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
\$0002	Reserved	—	—	—	—	—	—	—	—
\$0003	PTADD	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
\$0004	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
\$0005	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
\$0006	Reserved	—	—	—	—	—	—	—	—
\$0007	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
\$0008	PTCD	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
\$0009	PTCPE	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
\$000A	Reserved	—	—	—	—	—	—	—	—
\$000B	PTCDD	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
\$000C	PTDD	0	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
\$000D	PTDPE	0	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
\$000E	Reserved	—	—	—	—	—	—	—	—
\$000F	PTDDD	0	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
\$0010	PTED	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
\$0011	PTEPE	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
\$0012	Reserved	—	—	—	—	—	—	—	—
\$0013	PTEDD	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
\$0014	KBI1SC	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBF	KBACK	KBIE	KBIMOD
\$0015	KBI1PE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
\$0016	KBI2SC	0	0	0	0	KBF	KBACK	KBIE	KBIMOD
\$0017	KBI2PE	0	0	0	0	KBIPE3	KBIPE2	KBIPE1	KBIPE0
\$0018	SCI1BDH ⁽¹⁾	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
\$0019	SCI1BDL ⁽¹⁾	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
\$001A	SCI1C1 ⁽¹⁾	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
\$001B	SCI1C2 ⁽¹⁾	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
\$001C	SCI1S1 ⁽¹⁾	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
\$001D	SCI1S2 ⁽¹⁾	0	0	0	0	0	0	0	RAF
\$001E	SCI1C3 ⁽¹⁾	R8	T8	TXDIR	0	ORIE	NEIE	FEIE	PEIE
\$001F	SCI1D ⁽¹⁾	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
\$0020	CMTCGH1	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
\$0021	CMTCGL1	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0
\$0022	CMTCGH2	SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
\$0023	CMTCGL2	SL7	SL6	SL5	SL4	SL3	SL2	SL1	SL0
\$0024	CMTOC	IROL	CMTPOL	IROPEN	0	0	0	0	0
\$0025	CMTMSC	EOCF	CMTDIV1	CMTDIV0	EXSPC	BASE	FSK	EOCIE	MCGEN
\$0026	CMTCMD1	MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8
\$0027	CMTCMD2	MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0
\$0028	CMTCMD3	SB15	SB14	SB13	SB12	SB11	SB10	SB9	SB8
\$0029	CMTCMD4	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0

- 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.

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.

4.6.6 FLASH Command Register (FCMD)

Only four command codes are recognized in normal user modes as shown in Table 4-14. Refer to Section 4.4.3, “Program and Erase Command Execution,” for a detailed discussion of FLASH programming and erase operations.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
Reset	0	0	0	0	0	0	0	0

Figure 4-10. FLASH Command Register (FCMD)

Table 4-13. FCMD Field Descriptions

Field	Description
7:0 FCMD[7:0]	See Table 4-14 for FLASH commands.

Table 4-14. FLASH Commands

Command	FCMD	Equate File Label
Blank check	\$05	mBlank
Byte program	\$20	mByteProg
Byte program – burst mode	\$25	mBurstProg
Page erase (512 bytes/page)	\$40	mPageErase
Mass erase (all FLASH)	\$41	mMassErase

All other command codes are illegal and generate an access error.

It is not necessary to perform a blank check command after a mass erase operation. Only blank check is required as part of the security unlocking mechanism.

as an edge on the IRQ pin or a timer-overflow event. The debug module can also generate an SWI under certain circumstances.

If an event occurs in an enabled interrupt source, an associated read-only status flag will become set. The CPU will not respond until and unless the local interrupt enable is a logic 1 to enable the interrupt. The I bit in the CCR is logic 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset, which masks (prevents) all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

When the CPU receives a qualified interrupt request, it completes the current instruction before responding to the interrupt. The interrupt sequence uses the same cycle-by-cycle sequence as the SWI instruction and consists of:

- Saving the CPU registers on the stack
- Setting the I bit in the CCR to mask further interrupts
- Fetching the interrupt vector for the highest-priority interrupt that is currently pending
- Filling the instruction queue with the first three bytes of program information starting from the address fetched from the interrupt vector locations

While the CPU is responding to the interrupt, the I bit is automatically set to avoid the possibility of another interrupt interrupting the ISR itself (this is called nesting of interrupts). Normally, the I bit is restored to 0 when the CCR is restored from the value stacked on entry to the ISR. In rare cases, the I bit may be cleared inside an ISR (after clearing the status flag that generated the interrupt) so that other interrupts can be serviced without waiting for the first service routine to finish. This practice is not recommended for anyone other than the most experienced programmers because it can lead to subtle program errors that are difficult to debug.

The interrupt service routine ends with a return-from-interrupt (RTI) instruction that restores the CCR, A, X, and PC registers to their pre-interrupt values by reading the previously saved information off the stack.

NOTE

For compatibility with the M68HC08 Family, the H register is not automatically saved and restored. It is good programming practice to push H onto the stack at the start of the interrupt service routine (ISR) and restore it just before the RTI that is used to return from the ISR.

If two or more interrupts are pending when the I bit is cleared, the highest priority source is serviced first (see Table 5-1).

5.5.1 Interrupt Stack Frame

Figure 5-1 shows the contents and organization of a stack frame. Before the interrupt, the stack pointer (SP) points at the next available byte location on the stack. The current values of CPU registers are stored on the stack starting with the low-order byte of the program counter (PCL) and ending with the CCR. After stacking, the SP points at the next available location on the stack, which is the address that is one less than the address where the CCR was saved. The PC value that is stacked is the address of the instruction in the main program that would have executed next if the interrupt had not occurred.

6.3.3 Port C

Port C	Bit 7	6	5	3	3	2	1	Bit 0
MCU Pin:	PTC7/ $\overline{SS1}$	PTC6/ SPSCK1	PTC5/ MISO1	PTC4/ MOSI1	PTC3/ KBI2P3	PTC2/ KBI2P2	PTC1/ KBI2P1	PTC0/ KBI2P0

Figure 6-3. Port C Pin Names

Port C is an 8-bit general-purpose I/O port with four pins shared with the KBI2 keyboard interrupt inputs and four pins shared with the SPI.

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. Refer to Section 6.4, “Parallel I/O Controls,” for more information about general-purpose I/O control.

When the SPI module is enabled, PTC7 serves as the SPI module’s slave select pin ($\overline{SS1}$), PTC6 serves as the SPI clock pin (SPSCK1), PTC5 serves as the master-in slave-out pin (MISO1), and PTC4 serves as the master-out slave-in pin (MOSI1). Refer to the Serial Peripheral Interface (SPI) Module chapter for more information about using PTC7–PTC4 as SPI pins.

Any of the port C pins PTC3–PTC0 can be configured as a KBI2 keyboard interrupt pin. Refer to the Keyboard Interrupt (KBI) Module chapter for more information about using port C pins as keyboard interrupt pins.

6.3.4 Port D

Port D	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:		PTD6/ TPM1CH0	PTD5	PTD4	PTD3	PTD2	PTD1/ \overline{RESET}	PTD0/ BKGD/MS

Figure 6-4. Port D Pin Names

Port D is an 7-bit general-purpose I/O port with one pin shared with the BKGD/MS function, one pin shared with the \overline{RESET} function, one pin shared with the IRQ function, and one pin shared with the TPM.

Port D pins are available as general-purpose I/O pins controlled by the port D data (PTDD), data direction (PTDDD), and pullup enable (PTDPE) registers. Refer to Section 6.4, “Parallel I/O Controls,” for more information about general-purpose I/O control.

The PTD0/BKGD/MS pin is configured for the BKGD/MS function during reset and following reset. The internal pullup for this pin is enabled when the BKGD/MS function is enabled, regardless of the PTDPE0 bit. During reset, the BKGD/MS pin functions as a mode select pin. After the MCU is out of reset, the BKGD/MS pin becomes the background communications input/output pin. PTD0 can be configured to be a general-purpose output pin through software control. Refer to Chapter 3, “Modes of Operation,” Chapter 5, “Resets, Interrupts, and System Configuration,” and the Development Support chapter for more information about using this pin.

The PTD1/ \overline{RESET} pin is configured for the RESET function during reset and following reset.

7.4.3 Wait Mode Operation

The WAIT instruction enables interrupts by clearing the I bit in the CCR. It then halts the clocks to the CPU to reduce overall power consumption while the CPU is waiting for the interrupt or reset event that will wake the CPU from wait mode. When an interrupt or reset event occurs, the CPU clocks will resume and the interrupt or reset event will be processed normally.

If a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in wait mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in wait mode.

7.4.4 Stop Mode Operation

Usually, all system clocks, including the crystal oscillator (when used), are halted during stop mode to minimize power consumption. In such systems, external circuitry is needed to control the time spent in stop mode and to issue a signal to wake up the target MCU when it is time to resume processing. Unlike the earlier M68HC05 and M68HC08 MCUs, the HCS08 can be configured to keep a minimum set of clocks running in stop mode. This optionally allows an internal periodic signal to wake the target MCU from stop mode.

When a host debug system is connected to the background debug pin (BKGD) and the ENBDM control bit has been set by a serial command through the background interface (or because the MCU was reset into active background mode), the oscillator is forced to remain active when the MCU enters stop mode. In this case, if a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in stop mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in stop mode.

Recovery from stop mode depends on the particular HCS08 and whether the oscillator was stopped in stop mode. Refer to the Modes of Operation chapter for more details.

7.4.5 BGND Instruction

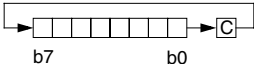
The BGND instruction is new to the HCS08 compared to the M68HC08. BGND would not be used in normal user programs because it forces the CPU to stop processing user instructions and enter the active background mode. The only way to resume execution of the user program is through reset or by a host debug system issuing a GO, TRACE1, or TAGGO serial command through the background debug interface.

Software-based breakpoints can be set by replacing an opcode at the desired breakpoint address with the BGND opcode. When the program reaches this breakpoint address, the CPU is forced to active background mode rather than continuing the user program.

Table 7-2. HCS08 Instruction Set Summary (Sheet 5 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Bus Cycles ¹
			V	H	I	N	Z	C				
LDX #opr8i LDX opr8a LDX opr16a LDX oprx16,X LDX oprx8,X LDX ,X LDX oprx16,SP LDX oprx8,SP	Load X (Index Register Low) from Memory	$X \leftarrow (M)$	0	–	–			–	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE BE CE DE EE FE 9EDE 9EEE	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 3 5 4
LSL opr8a LSLA LSLX LSL oprx8,X LSL ,X LSL oprx8,SP	Logical Shift Left (Same as ASL)			–	–				DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 5 4 6
LSR opr8a LSRA LSRX LSR oprx8,X LSR ,X LSR oprx8,SP	Logical Shift Right			–	–	0			DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	5 1 1 5 4 6
MOV opr8a,opr8a MOV opr8a,X+ MOV #opr8i,opr8a MOV ,X+,opr8a	Move	$(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ $H:X \leftarrow (H:X) + 0x0001$ in IX+/DIR and DIR/IX+ Modes	0	–	–			–	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E 5E 6E 7E	dd dd dd ii ii dd dd	5 5 4 5
MUL	Unsigned multiply	$X:A \leftarrow (X) \times (A)$	–	0	–	–	–	0	INH	42		5
NEG opr8a NEGA NEGX NEG oprx8,X NEG ,X NEG oprx8,SP	Negate (Two's Complement)	$M \leftarrow -(M) = 0x00 - (M)$ $A \leftarrow -(A) = 0x00 - (A)$ $X \leftarrow -(X) = 0x00 - (X)$ $M \leftarrow -(M) = 0x00 - (M)$ $M \leftarrow -(M) = 0x00 - (M)$ $M \leftarrow -(M) = 0x00 - (M)$		–	–				DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	5 1 1 5 4 6
NOP	No Operation	Uses 1 Bus Cycle	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap Accumulator	$A \leftarrow (A[3:0]:A[7:4])$	–	–	–	–	–	–	INH	62		1
ORA #opr8i ORA opr8a ORA opr16a ORA oprx16,X ORA oprx8,X ORA ,X ORA oprx16,SP ORA oprx8,SP	Inclusive OR Accumulator and Memory	$A \leftarrow (A) \mid (M)$	0	–	–			–	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA BA CA DA EA FA 9EDA 9EEA	ii dd hh ll ee ff ff ff	2 3 4 4 3 3 5 4
PSHA	Push Accumulator onto Stack	Push (A); $SP \leftarrow (SP) - 0x0001$	–	–	–	–	–	–	INH	87		2
PSHH	Push H (Index Register High) onto Stack	Push (H); $SP \leftarrow (SP) - 0x0001$	–	–	–	–	–	–	INH	8B		2
PSHX	Push X (Index Register Low) onto Stack	Push (X); $SP \leftarrow (SP) - 0x0001$	–	–	–	–	–	–	INH	89		2
PULA	Pull Accumulator from Stack	$SP \leftarrow (SP + 0x0001)$; Pull (A)	–	–	–	–	–	–	INH	86		3
PULH	Pull H (Index Register High) from Stack	$SP \leftarrow (SP + 0x0001)$; Pull (H)	–	–	–	–	–	–	INH	8A		3
PULX	Pull X (Index Register Low) from Stack	$SP \leftarrow (SP + 0x0001)$; Pull (X)	–	–	–	–	–	–	INH	88		3
ROL opr8a ROLA ROLX ROL oprx8,X ROL ,X ROL oprx8,SP	Rotate Left through Carry			–	–				DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	5 1 1 5 4 6

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 ee 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 ee 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 ee 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 ee 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



8.5 Functional Description

The CMT module consists of a carrier generator, a modulator, a transmitter output, and control registers. The block diagram is shown in Figure 8-2. When operating in time mode, the user independently defines the high and low times of the carrier signal to determine both period and duty cycle. The carrier generator resolution is 125 ns when operating with an 8 MHz internal bus frequency and the CMTDIV1 and CMTDIV0 bits in the CMTMSC register are both equal to 0. The carrier generator can generate signals with periods between 250 ns (4 MHz) and 127.5 μ s (7.84 kHz) in steps of 125 ns. See Table 8-1.

Table 8-1. Clock Divide

Bus Clock (MHz)	CMTDIV1:CMTDIV0	Carrier Generator Resolution (μ s)	Min Carrier Generator Period (μ s)	Min Modulator Period (μ s)
8	0:0	0.125	0.25	1.0
8	0:1	0.25	0.5	2.0
8	1:0	0.5	1.0	4.0
8	1:1	1.0	2.0	8.0

The possible duty cycle options will depend upon the number of counts required to complete the carrier period. For example, a 1.6 MHz signal has a period of 625 ns and will therefore require 5×125 ns counts to generate. These counts may be split between high and low times, so the duty cycles available will be 20 percent (one high, four low), 40 percent (two high, three low), 60 percent (three high, two low) and 80 percent (four high, one low).

For lower frequency signals with larger periods, higher resolution (as a percentage of the total period) duty cycles are possible.

When the BASE bit in the CMT modulator status and control register (CMTMSC) is set, the carrier output (f_{CG}) to the modulator is held high continuously to allow for the generation of baseband protocols.

A third mode allows the carrier generator to alternate between two sets of high and low times. When operating in FSK mode, the generator will toggle between the two sets when instructed by the modulator, allowing the user to dynamically switch between two carrier frequencies without CPU intervention.

The modulator provides a simple method to control protocol timing. The modulator has a minimum resolution of 1.0 μ s with an 8 MHz internal bus clock. It can count bus clocks (to provide real-time control) or it can count carrier clocks (for self-clocked protocols). See Section 8.5.2, "Modulator," for more details.

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.

A summary of the possible modes is shown in Table 8-2.

10.3 TPM Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPM1CH_n where *n* is the channel number (for example, 0–4). The TPM shares its I/O pins with general-purpose I/O port pins (refer to the Pins and Connections chapter for more information). Figure 10-2 shows the structure of a TPM. Some MCUs include more than one TPM, with various numbers of channels.

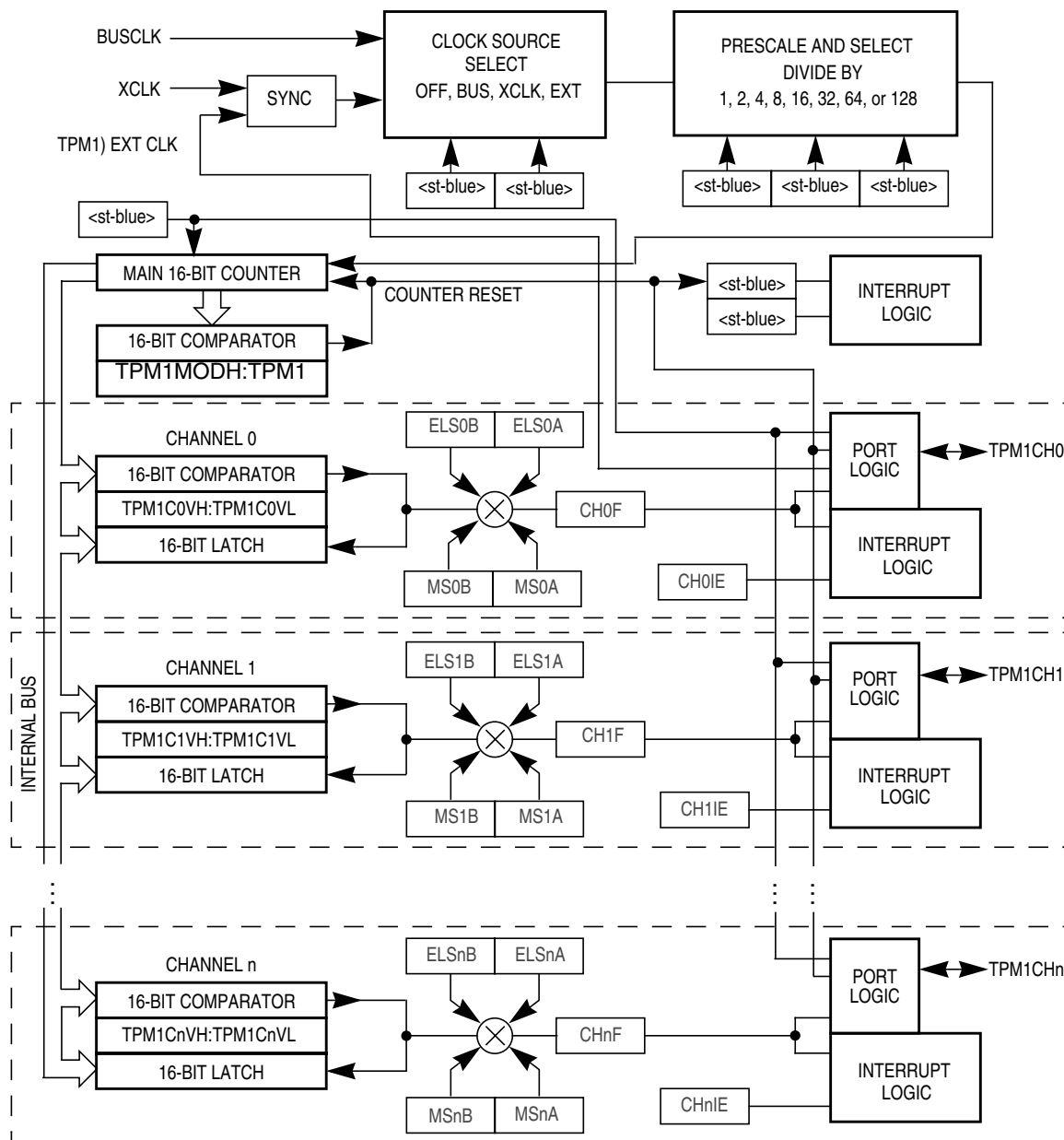


Figure 10-2. TPM Block Diagram

The central component of the TPM is the 16-bit counter that can operate as a free-running counter, a modulo counter, or an up-/down-counter when the TPM is configured for center-aligned PWM. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture,

output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPM1MODH:TPM1MODL, control the modulo value of the counter. (The values \$0000 or \$FFFF effectively make the counter free running.) Software can read the counter value at any time without affecting the counting sequence. Any write to either byte of the TPM1CNT counter resets the counter regardless of the data value written.

All TPM channels are programmable independently as input capture, output compare, or buffered edge-aligned PWM channels.

10.4 Pin Descriptions

Table 10-2 shows the MCU pins related to the TPM module. When TPM1CH0 is used as an external clock input, the associated TPM channel 0 can not use the pin. (Channel 0 can still be used in output compare mode as a software timer.) When any of the pins associated with the timer is configured as a timer input, a passive pullup can be enabled. After reset, the TPM modules are disabled and all pins default to general-purpose inputs with the passive pullups disabled.

10.4.1 External TPM Clock Sources

When control bits CLKS:CLKSA in the timer status and control register are set to 1:1, the prescaler and consequently the 16-bit counter for TPM1 are driven by an external clock source connected to the TPM1CH0 pin. A synchronizer is needed between the external clock and the rest of the TPM. This synchronizer is clocked by the bus clock so the frequency of the external source must be less than one-half the frequency of the bus rate clock. The upper frequency limit for this external clock source is specified to be one-fourth the bus frequency to conservatively accommodate duty cycle and phase-locked loop (PLL) or frequency-locked loop (FLL) frequency jitter effects.

When the TPM is using the channel 0 pin for an external clock, the corresponding ELS0B:ELS0A control bits should be set to 0:0 so channel 0 is not trying to use the same pin.

10.4.2 TPM1CHn — TPM1 Channel n I/O Pins

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the configuration of the channel. In some cases, no pin function is needed so the pin reverts to being controlled by general-purpose I/O controls. When a timer has control of a port pin, the port data and data direction registers do not affect the related pin(s). See the Pins and Connections chapter for additional information about shared pin functions.

10.5 Functional Description

All TPM functions are associated with a main 16-bit counter that allows flexible selection of the clock source and prescale divisor. A 16-bit modulo register also is associated with the main 16-bit counter in the TPM. Each TPM channel is optionally associated with an MCU pin and a maskable interrupt function.

The TPM has center-aligned PWM capabilities controlled by the CPWMS control bit in TPM1SC. When CPWMS is set to 1, timer counter TPM1CNT changes to an up-/down-counter and all channels in the associated TPM act as center-aligned PWM channels. When CPWMS = 0, each channel can

13.4.5 SPI Data Register (SPI1D)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	6	5	4	3	2	1	Bit 0
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.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGTC register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

15.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGTC register selects one of nine trigger modes. When TRGSEL = 1 in the DBGTC register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGTC chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGTC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGSR. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGGEN in DBGTC.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

Appendix A

Electrical Characteristics

A.1 Introduction

This section contains electrical and timing specifications.

A.2 Absolute Maximum Ratings

Absolute maximum ratings are stress ratings only, and functional operation at the maxima is not guaranteed. Stress beyond the limits specified in Table A-1 may affect device reliability or cause permanent damage to the device. For functional operating conditions, refer to the remaining tables in this section.

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (for instance, either V_{SS} or V_{DD}) or the programmable pull-up resistor associated with the pin is enabled.

Table A-1. Absolute Maximum Ratings

Rating	Symbol	Value	Unit
Supply voltage	V_{DD}	-0.3 to +3.8	V
Maximum current into V_{DD}	I_{DD}	120	mA
Digital input voltage	V_{In}	-0.3 to $V_{DD} + 0.3$	V
Instantaneous maximum current Single pin limit (applies to all port pins) ^{(1), (2), (3)}	I_D	± 25	mA
Storage temperature range	T_{stg}	-55 to 150	°C

1. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive (V_{DD}) and negative (V_{SS}) clamp voltages, then use the larger of the two resistance values.
2. All functional non-supply pins are internally clamped to V_{SS} and V_{DD} .
3. 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 the clock rate is very low (which would reduce overall power consumption).

A.4 Electrostatic Discharge (ESD) Protection Characteristics

Although damage from static discharge is much less common on these devices than on early CMOS circuits, normal handling precautions should be used to avoid exposure to static discharge. Qualification tests are performed to ensure that these devices can withstand exposure to reasonable levels of static without suffering any permanent damage. All ESD testing is in conformity with CDF-AEC-Q00 Stress Test Qualification for Automotive Grade Integrated Circuits. (<http://www.aecouncil.com/>) A device is considered to have failed if, after exposure to ESD pulses, the device no longer meets the device specification requirements. Complete dc parametric and functional testing is performed per the applicable device data sheet at room temperature followed by hot temperature, unless specified otherwise in the device data sheet.

Table A-3. ESD Protection Characteristics

Parameter	Symbol	Value	Unit
ESD Target for Machine Model (MM) MM circuit description	V_{THMM}	200	V
ESD Target for Human Body Model (HBM) HBM circuit description	V_{THHBM}	2000	V

A.5 DC Characteristics

This section includes information about power supply requirements, I/O pin characteristics, and power supply current in various operating modes.

Table A-4. DC Characteristics (Temperature Range = 0 to 70°C Ambient)

Parameter	Symbol	Min	Typical	Max	Unit
Low-voltage detection threshold	V_{LVD}	1.82	1.875	1.90	V
Power on reset (POR) voltage	V_{POR}	0.8	0.9	1.1	V
Maximum low-voltage safe state re-arm ⁽¹⁾	V_{REARM}	1.90	2.24	2.60	V

1. If SAFE bit is set, V_{DD} must be above re-arm voltage to allow MCU to accept interrupts, refer to Section 5.6, “Low-Voltage Detect (LVD) System.”

Table A-5. DC Characteristics (Temperature Range = –40 to 85°C Ambient)

Parameter	Symbol	Min	Typical	Max	Unit
Supply voltage (run, wait and stop modes.) $0 < f_{BUS} < 8 \text{ MHz}$	V_{DD}	1.8		3.6	V
Minimum RAM retention supply voltage applied to V_{DD}	V_{RAM}	$V_{POR}^{(1), (2)}$		—	V
Low-voltage detection threshold (V_{DD} falling) (V_{DD} rising)	V_{LVD}	1.82 1.92	1.88 1.96	1.93 2.01	V
Low-voltage warning threshold (V_{DD} falling) (V_{DD} rising)	V_{LVW}	2.07 2.16	2.13 2.21	2.18 2.26	V
Power on reset (POR) voltage	V_{POR}	0.85	1.0	1.2	V