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	40MHz
Connectivity	I²C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	34
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LQFP
Supplier Device Package	44-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s08aw32vfge

2.2 Device Pin Assignment

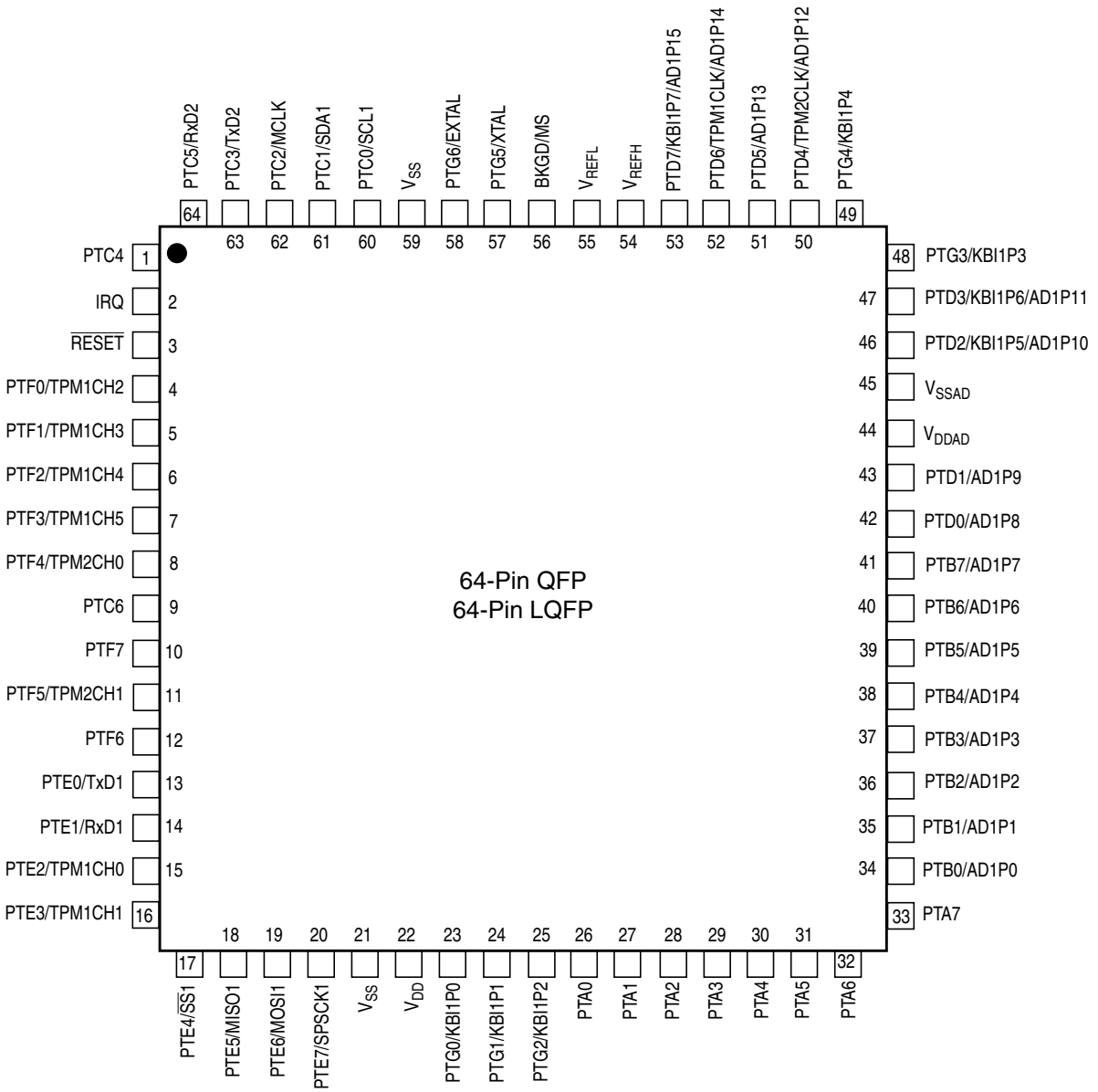


Figure 2-1. MC9S08AW60 Series in 64-Pin QFP/LQFP Package

Table 4-12. FSTAT Register Field Descriptions

Field	Description
7 FCBEF	**FLASH Command Buffer Empty Flag** — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so that a new command sequence can be executed when performing burst programming. The FCBEF bit is cleared by writing a one to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered. 0 Command buffer is full (not ready for additional commands). 1 A new burst program command may be written to the command buffer.
6 FCCF	**FLASH Command Complete Flag** — FCCF is set automatically when the command buffer is empty and no command is being processed. FCCF is cleared automatically when a new command is started (by writing 1 to FCBEF to register a command). Writing to FCCF has no meaning or effect. 0 Command in progress 1 All commands complete
5 FPVIOL	**Protection Violation Flag** — FPVIOL is set automatically when FCBEF is cleared to register a command that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL. 0 No protection violation. 1 An attempt was made to erase or program a protected location.
4 FACCERR	**Access Error Flag** — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see Section 4.4.5, “Access Errors.” FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect. 0 No access error. 1 An access error has occurred.
2 FBLANK	**FLASH Verified as All Blank (erased) Flag** — FBLANK is set automatically at the conclusion of a blank check command if the entire FLASH array was verified to be erased. FBLANK is cleared by clearing FCBEF to write a new valid command. Writing to FBLANK has no meaning or effect. 0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the FLASH array is not completely erased. 1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the FLASH array is completely erased (all $FF).

5.9.7 System Real-Time Interrupt Status and Control Register (SRTISC)

This register contains one read-only status flag, one write-only acknowledge bit, three read/write delay selects, and three unimplemented bits, which always read 0.

	7	6	5	4	3	2	1	0
R	RTIF	0	RTICLKS	RTIE	0	RTIS2	RTIS1	RTIS0
W		RTIACK						
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-9. System RTI Status and Control Register (SRTISC)

Table 5-9. SRTISC Register Field Descriptions

Field	Description
7 RTIF	**Real-Time Interrupt Flag** — This read-only status bit indicates the periodic wakeup timer has timed out. 0 Periodic wakeup timer not timed out. 1 Periodic wakeup timer timed out.
6 RTIACK	**Real-Time Interrupt Acknowledge** — This write-only bit is used to acknowledge real-time interrupt request (write 1 to clear RTIF). Writing 0 has no meaning or effect. Reads always return logic 0.
5 RTICLKS	**Real-Time Interrupt Clock Select** — This read/write bit selects the clock source for the real-time interrupt. 0 Real-time interrupt request clock source is internal 1-kHz oscillator. 1 Real-time interrupt request clock source is external clock.
4 RTIE	**Real-Time Interrupt Enable** — This read-write bit enables real-time interrupts. 0 Real-time interrupts disabled. 1 Real-time interrupts enabled.
2:0 RTIS[2:0]	**Real-Time Interrupt Delay Selects** — These read/write bits select the wakeup delay for the RTI. The clock source for the real-time interrupt is a self-clocked source which oscillates at about 1 kHz, is independent of other MCU clock sources. Using external clock source the delays will be crystal frequency divided by value in RTIS2:RTIS1:RTIS0. See Table 5-10.

Table 5-10. Real-Time Interrupt Frequency

RTIS2:RTIS1:RTIS0	1-kHz Clock Source Delay[1]	Using External Clock Source Delay (Crystal Frequency)
0:0:0	Disable periodic wakeup timer	Disable periodic wakeup timer
0:0:1	8 ms	divide by 256
0:1:0	32 ms	divide by 1024
0:1:1	64 ms	divide by 2048
1:0:0	128 ms	divide by 4096
1:0:1	256 ms	divide by 8192
1:1:0	512 ms	divide by 16384
1:1:1	1.024 s	divide by 32768

[1] Normal values are shown in this column based on f_{RTI} = 1 kHz. See Appendix A, “Electrical Characteristics and Timing Specifications,” f_{RTI} for the tolerance on these values.

	7	6	5	4	3	2	1	0
R W	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
Reset	0	0	0	0	0	0	0	0

Figure 6-10. Data Direction for Port A Register (PTADD)

Table 6-3. PTADD Register Field Descriptions

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

6.7.2 Port A Pin Control Registers (PTAPE, PTASE, PTADS)

In addition to the I/O control, port A pins are controlled by the registers listed below.

	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 6-11. Internal Pullup Enable for Port A (PTAPE)

Table 6-4. PTADD Register Field Descriptions

Field	Description
[7:0] PTAPE[7:0]	**Internal Pullup Enable for Port A Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port A bit n. 1 Internal pullup device enabled for port A bit n.

	7	6	5	4	3	2	1	0
R W	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
Reset	0	0	0	0	0	0	0	0

Figure 6-12. Output Slew Rate Control Enable for Port A (PTASE)

Table 6-5. PTASE Register Field Descriptions

Field	Description
7:0 PTASE[7:0]	**Output Slew Rate Control Enable for Port A Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port A bit n. 1 Output slew rate control enabled for port A bit n.

	7	6	5	4	3	2	1	0
R W	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
Reset	0	0	0	0	0	0	0	0

Figure 6-13. Output Drive Strength Selection for Port A (PTASE)

Table 6-6. PTASE Register Field Descriptions

Field	Description
7:0 PTADS[7:0]	**Output Drive Strength Selection for Port A Bits** — Each of these control bits selects between low and high output drive for the associated PTA pin. 0 Low output drive enabled for port A bit n. 1 High output drive enabled for port A bit n.

7.3 Addressing Modes

Addressing modes define the way the CPU accesses operands and data. In the HCS08, all memory, status and control registers, and input/output (I/O) ports share a single 64-Kbyte linear address space so a 16-bit binary address can uniquely identify any memory location. This arrangement means that the same instructions that access variables in RAM can also be used to access I/O and control registers or nonvolatile program space.

Some instructions use more than one addressing mode. For instance, move instructions use one addressing mode to specify the source operand and a second addressing mode to specify the destination address. Instructions such as BRCLR, BRSET, CBEQ, and DBNZ use one addressing mode to specify the location of an operand for a test and then use relative addressing mode to specify the branch destination address when the tested condition is true. For BRCLR, BRSET, CBEQ, and DBNZ, the addressing mode listed in the instruction set tables is the addressing mode needed to access the operand to be tested, and relative addressing mode is implied for the branch destination.

7.3.1 Inherent Addressing Mode (INH)

In this addressing mode, operands needed to complete the instruction (if any) are located within CPU registers so the CPU does not need to access memory to get any operands.

7.3.2 Relative Addressing Mode (REL)

Relative addressing mode is used to specify the destination location for branch instructions. A signed 8-bit offset value is located in the memory location immediately following the opcode. During execution, if the branch condition is true, the signed offset is sign-extended to a 16-bit value and is added to the current contents of the program counter, which causes program execution to continue at the branch destination address.

7.3.3 Immediate Addressing Mode (IMM)

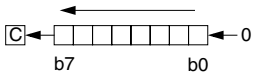
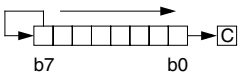
In immediate addressing mode, the operand needed to complete the instruction is included in the object code immediately following the instruction opcode in memory. In the case of a 16-bit immediate operand, the high-order byte is located in the next memory location after the opcode, and the low-order byte is located in the next memory location after that.

7.3.4 Direct Addressing Mode (DIR)

In direct addressing mode, the instruction includes the low-order eight bits of an address in the direct page (0x0000–0x00FF). During execution a 16-bit address is formed by concatenating an implied 0x00 for the high-order half of the address and the direct address from the instruction to get the 16-bit address where the desired operand is located. This is faster and more memory efficient than specifying a complete 16-bit address for the operand.

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

8.4.9 FLL Loss-of-Clock Detection

The reference clock and the DCO clock are monitored under different conditions (see Table 8-8). Provided the reference frequency is being monitored, ERCS = 1 indicates that the reference clock meets minimum frequency requirements. When the reference and/or DCO clock(s) are being monitored, if either one falls below a certain frequency, f_{LOR} and f_{LOD}, respectively, the LOCS status bit will be set to indicate the error. LOCS will remain set until it is acknowledged or until the MCU is reset. LOCS is cleared by reading ICGS1 then writing 1 to ICGIF (LOCRE = 0), or by a loss-of-clock induced reset (LOCRE = 1), or by any MCU reset.

If the ICG is in FEE, a loss of reference clock causes the ICG to enter SCM, and a loss of DCO clock causes the ICG to enter FBE mode. If the ICG is in FBE mode, a loss of reference clock will cause the ICG to enter SCM. In each case, the CLKST and CLKS bits will be automatically changed to reflect the new state.

If the ICG is in FEE mode when a loss of clock occurs and the ERCS is still set to 1, then the CLKST bits are set to 10 and the ICG reverts to FBE mode.

A loss of clock will also cause a loss of lock when in FEE or FEI modes. Because the method of clearing the LOCS and LOLS bits is the same, this would only be an issue in the unlikely case that LOLRE = 1 and LOCRE = 0. In this case, the interrupt would be overridden by the reset for the loss of lock.

Table 8-8. Clock Monitoring (When LOCD = 0)

Mode	CLKS	REFST	ERCS	External Reference Clock Monitored?	DCO Clock Monitored?
Off	0X or 11	X	Forced Low	No	No
	10	0	Forced Low	No	No
	10	1	Real-Time[1]	Yes(1)	No
SCM (CLKST = 00)	0X	X	Forced Low	No	Yes[2]
	10	0	Forced High	No	Yes(2)
	10	1	Real-Time	Yes	Yes(2)
	11	X	Real-Time	Yes	Yes(2)
FEI (CLKST = 01)	0X	X	Forced Low	No	Yes
	11	X	Real-Time	Yes	Yes
FBE (CLKST = 10)	10	0	Forced High	No	No
	10	1	Real-Time	Yes	No
FEE (CLKST = 11)	11	X	Real-Time	Yes	Yes

1 If ENABLE is high (waiting for external crystal start-up after exiting stop).

2 DCO clock will not be monitored until DCOS = 1 upon entering SCM from off or FLL bypassed external mode.

ICGTRM = $xx

Bit 7:0 TRIM — Only need to write when trimming internal oscillator; done in separate operation (see example #4)

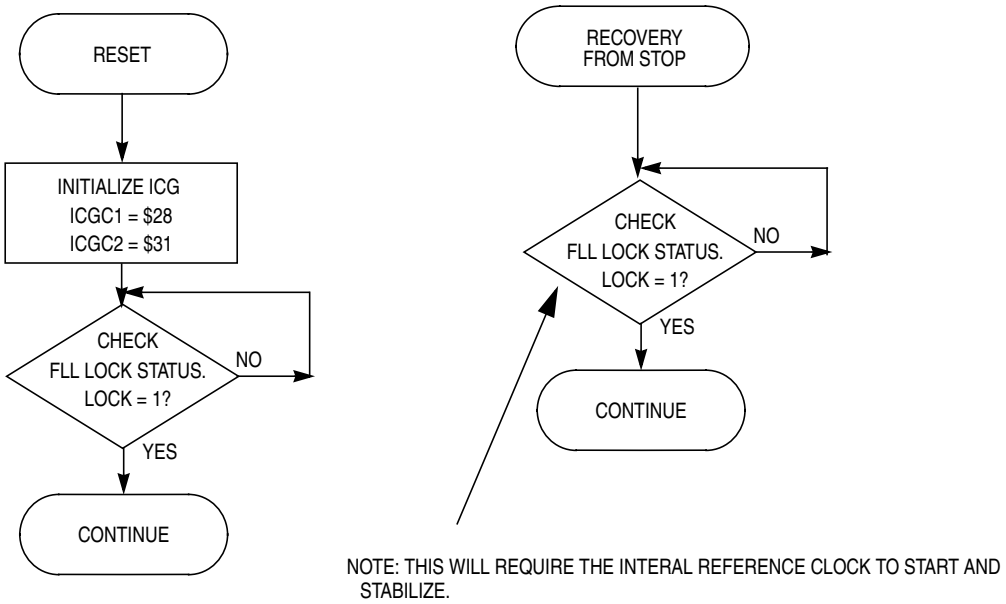


Figure 8-16. ICG Initialization and Stop Recovery for Example #3

9.3.1 KBI Block Diagram

Figure 9-2 shows the block diagram for a KBI module.

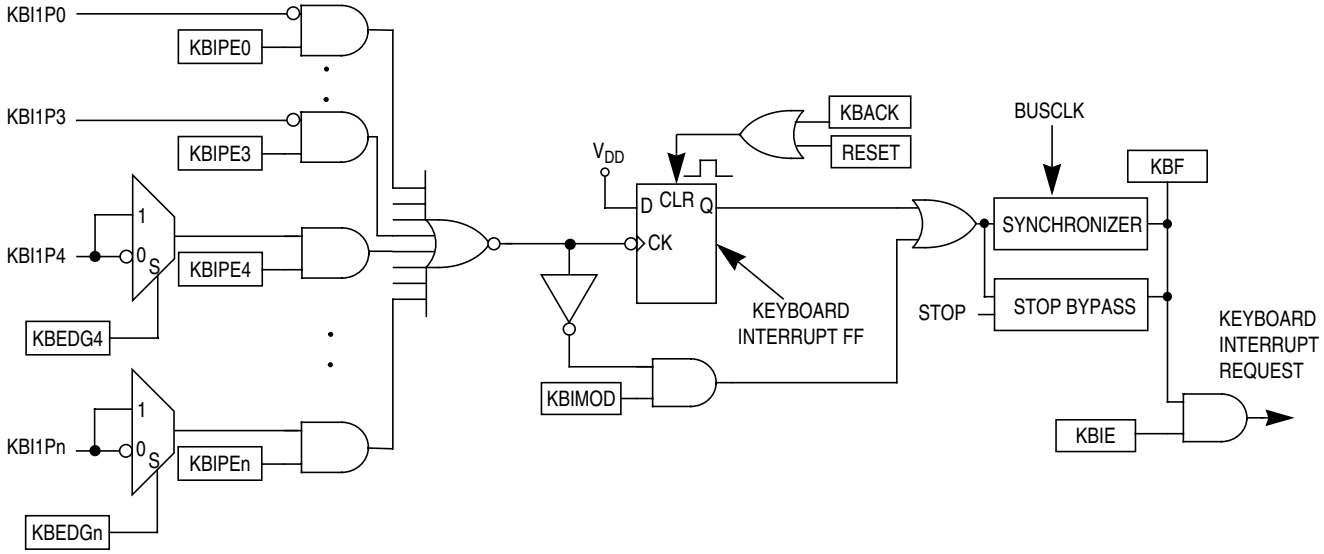


Figure 9-2. KBI Block Diagram

9.4 Register Definition

This section provides information about all registers and control bits associated with the KBI module.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

9.4.2 KBI Pin Enable Register (KBI1PE)

	7	6	5	4	3	2	1	0
R W	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 9-4. KBI Pin Enable Register (KBI1PE)

Table 9-3. KBI1PE Register Field Descriptions

Field	Description
7:0 KBIPE[7:0]	**Keyboard Pin Enable for KBI Port Bits** — Each of these read/write bits selects whether the associated KBI port pin is enabled as a keyboard interrupt input or functions as a general-purpose I/O pin. 0 Bit n of KBI port is a general-purpose I/O pin not associated with the KBI 1 Bit n of KBI port enabled as a keyboard interrupt input

9.5 Functional Description

9.5.1 Pin Enables

The KBIPEn control bits in the KBI1PE register allow a user to enable (KBIPEn = 1) any combination of KBI-related port pins to be connected to the KBI module. Pins corresponding to 0s in KBI1PE are general-purpose I/O pins that are not associated with the KBI module.

9.5.2 Edge and Level Sensitivity

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled keyboard inputs in a KBI module must be at the deasserted logic level.

A falling edge is detected when an enabled keyboard input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle.

A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

The KBIMOD control bit can be set to reconfigure the detection logic so that it detects edges and levels. In KBIMOD = 1 mode, the KBF status flag becomes set when an edge is detected (when one or more enabled pins change from the deasserted to the asserted level while all other enabled pins remain at their deasserted levels), but the flag is continuously set (and cannot be cleared) as long as any enabled keyboard input pin remains at the asserted level. When the MCU enters stop3 mode, the synchronous edge-detection logic is bypassed (because clocks are stopped). In stop3 mode, KBI inputs act as asynchronous level-sensitive inputs so they can wake the MCU from stop3 mode.

12.4.2 SPI Interrupts

There are three flag bits, two interrupt mask bits, and one interrupt vector associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check the flag bits to determine what event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

12.4.3 Mode Fault Detection

A mode fault occurs and the mode fault flag (MODF) becomes set when a master SPI device detects an error on the $\overline{SS}$ pin (provided the $\overline{SS}$ pin is configured as the mode fault input signal). The $\overline{SS}$ pin is configured to be the mode fault input signal when MSTR = 1, mode fault enable is set (MODFEN = 1), and slave select output enable is clear (SSOE = 0).

The mode fault detection feature can be used in a system where more than one SPI device might become a master at the same time. The error is detected when a master's $\overline{SS}$ pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

When a mode fault is detected, MODF is set and MSTR is cleared to change the SPI configuration back to slave mode. The output drivers on the SPSCK, MOSI, and MISO (if not bidirectional mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPI control register 1 (SPI1C1). User software should verify the error condition has been corrected before changing the SPI back to master mode.

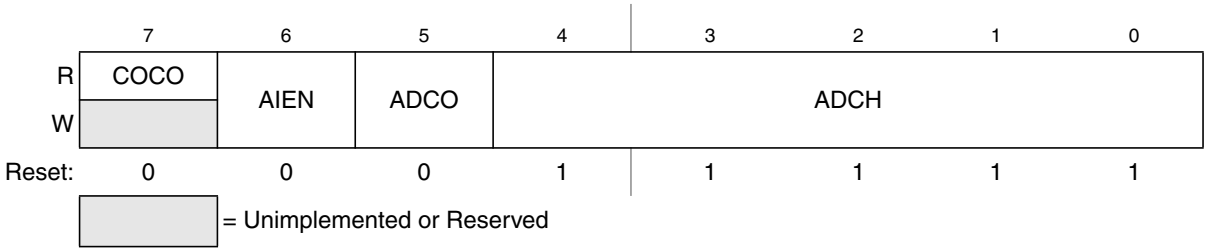


Figure 14-3. Status and Control Register (ADC1SC1)

Table 14-3. ADC1SC1 Register Field Descriptions

Field	Description
7 COCO	**Conversion Complete Flag** — The COCO flag is a read-only bit which is set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1) the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared whenever ADC1SC1 is written or whenever ADC1RL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	**Interrupt Enable** — AIEN is used to enable conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	**Continuous Conversion Enable** — ADCO is used to enable continuous conversions. 0 One conversion following a write to the ADC1SC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADC1SC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	**Input Channel Select** — The ADCH bits form a 5-bit field which is used to select one of the input channels. The input channels are detailed in Figure 14-4. The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way will prevent an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Figure 14-4. Input Channel Select

ADCH	Input Select
00000	AD0
00001	AD1
00010	AD2
00011	AD3
00100	AD4
00101	AD5
00110	AD6
00111	AD7

ADCH	Input Select
10000	AD16
10001	AD17
10010	AD18
10011	AD19
10100	AD20
10101	AD21
10110	AD22
10111	AD23

converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around 1/2LSB and will increase with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 14.7.2.3 will reduce this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values which are never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and to have no missing codes.

15.1.1 Features

Features of the BDC module 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 ICE system 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:
  - Basic: A-only, A OR B
  - Sequence: A then B
  - Full: A AND B data, A AND NOT B data
  - Event (store data): Event-only B, A then event-only B
  - Range: Inside range ($A \leq address \leq B$), outside range ($address < A$ or $address > B$)

15.2 Background Debug Controller (BDC)

All MCUs in the HCS08 family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.

the host must perform ((8 – CNT) – 1) dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see Section 15.3.5, “Trigger Modes”), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When ARM = 0, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

15.3.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

15.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A.12 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 Chapter 4, "Memory."

Table A-16. FLASH Characteristics

Num	C	Characteristic	Symbol	Min	Typ[1]	Max	Unit
1	P	Supply voltage for program/erase	$V_{prog/erase}$	2.7		5.5	V
2	P	Supply voltage for read operation	V_{Read}	2.7		5.5	V
3	P	Internal FCLK frequency[2]	f_{FCLK}	150		200	kHz
4	P	Internal FCLK period (1/FCLK)	t_{Fcyc}	5		6.67	µs
5	P	Byte program time (random location)[3]	t_{prog}	9			t_{Fcyc}
6	C	Byte program time (burst mode)[3]	t_{Burst}	4			t_{Fcyc}
7	P	Page erase time[3]	t_{Page}	4000			t_{Fcyc}
8	P	Mass erase time[3]	t_{Mass}	20,000			t_{Fcyc}
9	C	Program/erase endurance[4] T_L to T_H = –40°C to + 125°C T = 25°C		10,000 —	— 100,000	— —	cycles
10	C	Data retention[5]	t_{D_ret}	15	100	—	years

1 Typical values are based on characterization data at V_{DD} = 5.0 V, 25°C unless otherwise stated.

2 The frequency of this clock is controlled by a software setting.

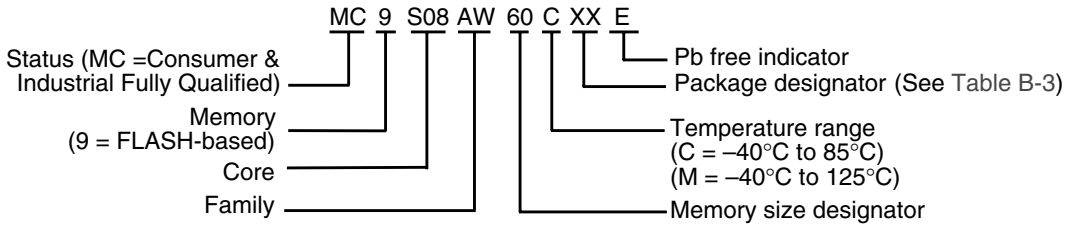
3 These values are hardware state machine controlled. User code does not need to count cycles. This information supplied for calculating approximate time to program and erase.

4 **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.*

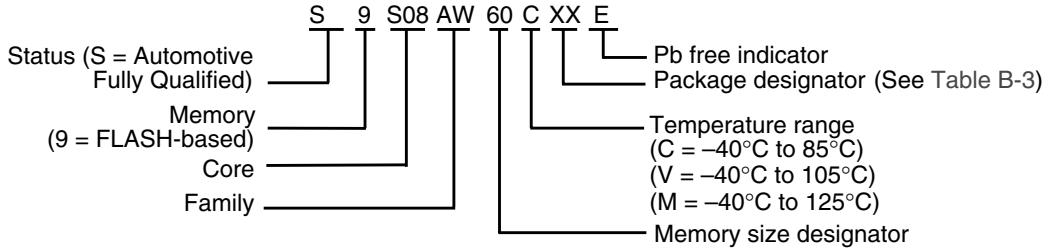
5 **Typical data retention** values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale Semiconductor defines typical data retention, please refer to Engineering Bulletin EB618/D, *Typical Data Retention for Nonvolatile Memory.*

B.2 Orderable Part Numbering System

B.2.1 Consumer and Industrial Orderable Part Numbering System



B.2.2 Automotive Orderable Part Numbering System

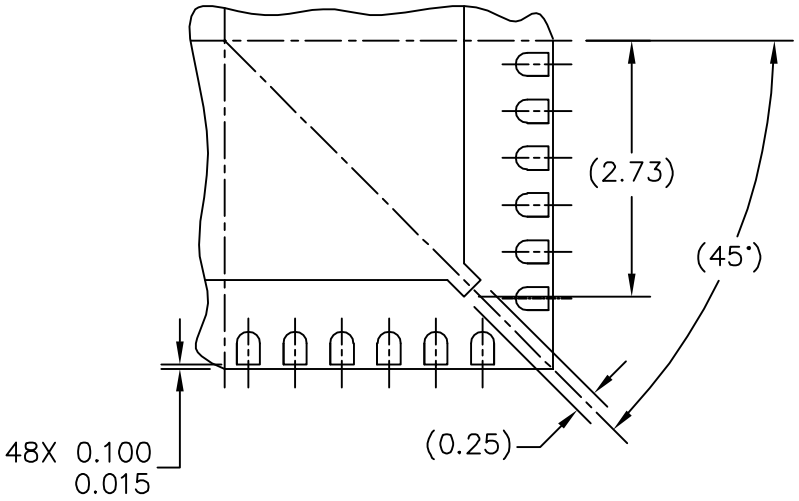


B.3 Mechanical Drawings

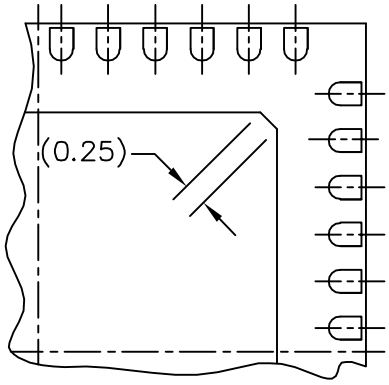
This following pages contain mechanical specifications for MC9S08AW60 Series package options. See Table B-3 for the document numbers that correspond to each package type.

Table B-3. Package Information

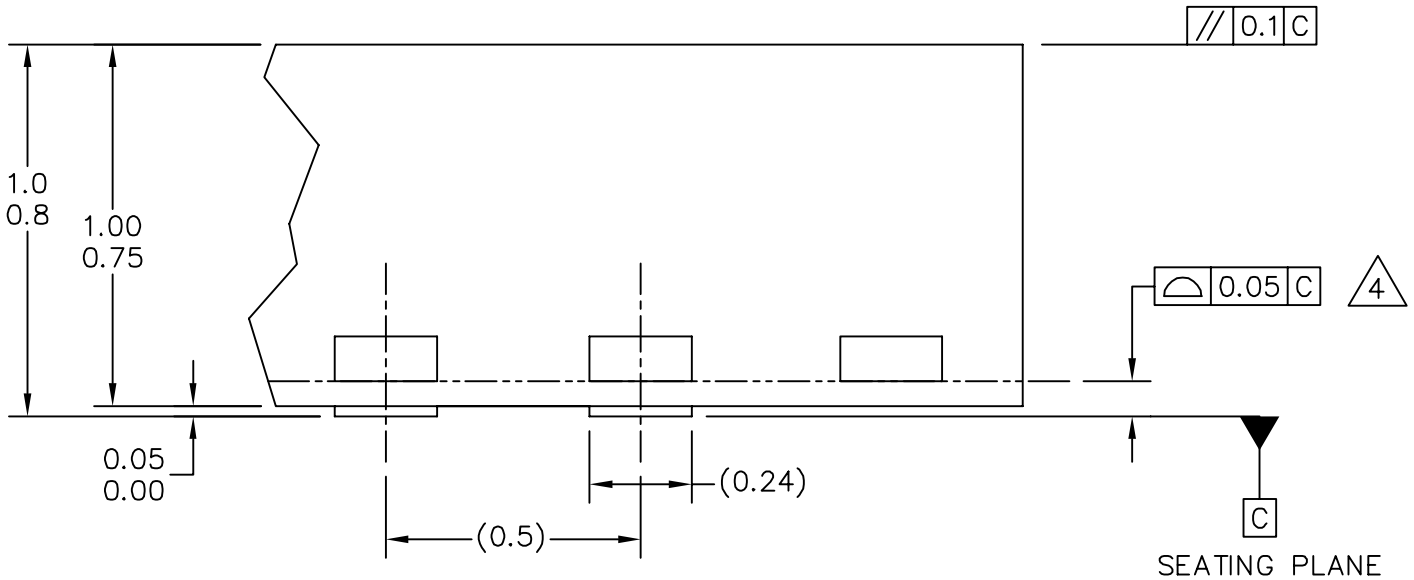
Pin Count	Type	Designator	Document No.
44	LQFP	FG	98ASS23225W
48	QFN	FD	98ARH99048A
64	LQFP	PU	98ASS23234W
64	QFP	FU	98ASB42844B



DETAIL N
PREFERRED CORNER CONFIGURATION



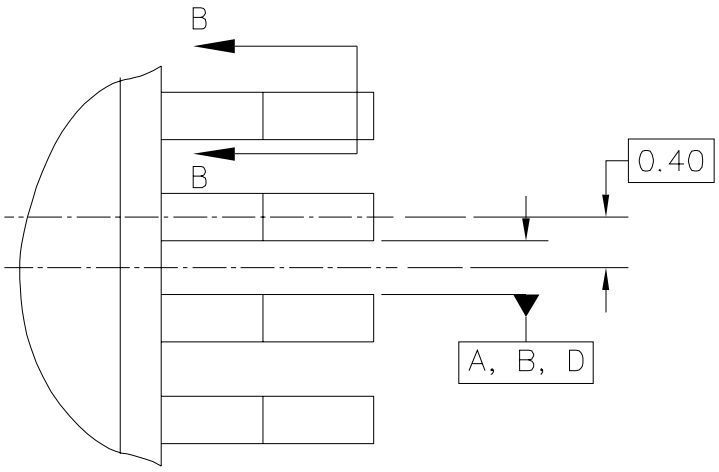
DETAIL M
PREFERED PIN 1 BACKSIDE IDENTIFIER



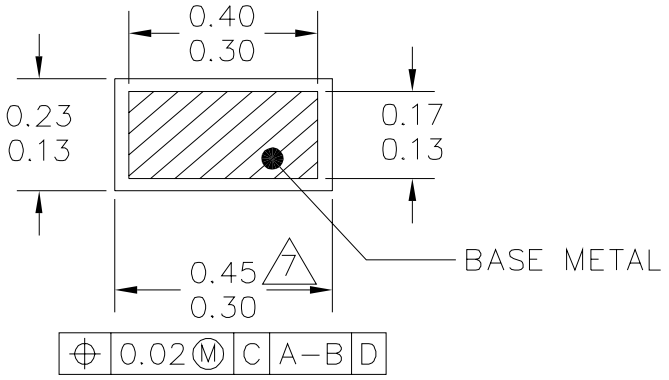
DETAIL G
VIEW ROTATED 90° CW

MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE: THERMALLY ENHANCED QUAD FLAT NON-LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)	DOCUMENT NO: 98ARH99048A	REV: F
	CASE NUMBER: 1314-05	05 DEC 2005
	STANDARD: JEDEC-MO-220 VKKD-2	

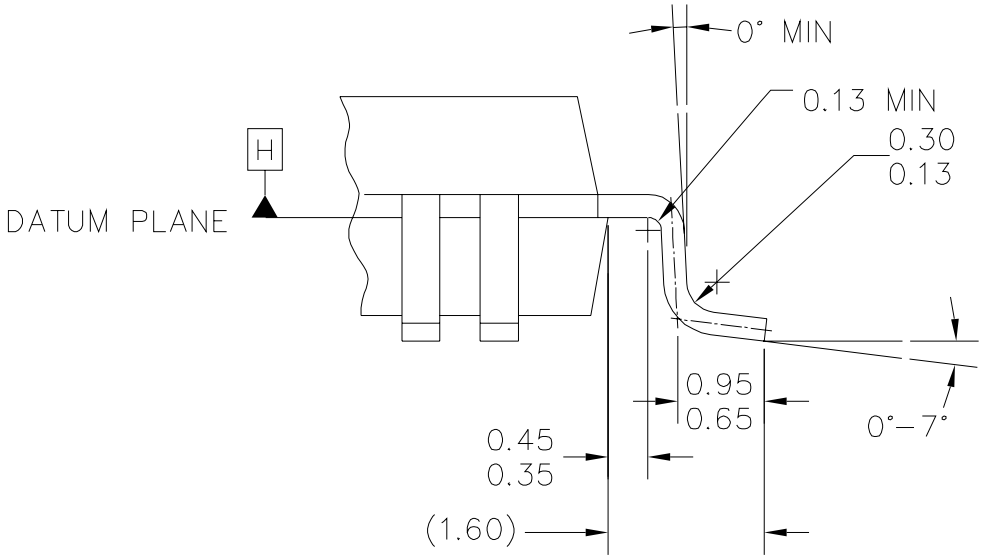


DETAIL "A"



VIEW ROTATED 90° CLOCKWISE

SECTION B–B



DETAIL "C"

MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE:	DOCUMENT NO: 98ASB42844B	REV: B
64LD QFP (14 X 14)	CASE NUMBER: 840B-01	20 MAY 2005
	STANDARD: NON-JEDEC	