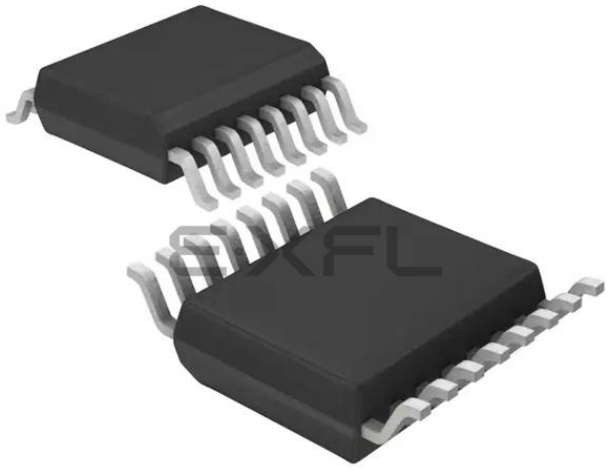


Welcome to E-XFL.COM

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	Obsolete
Core Processor	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	-
Peripherals	LVD, POR, PWM
Number of I/O	13
Program Memory Size	1.5KB (1.5K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	16-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	16-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908qy1acdte

MC68HC908QY4A
MC68HC908QT4A
MC68HC908QY2A
MC68HC908QT2A
MC68HC908QY1A
MC68HC908QT1A

Data Sheet

M68HC08
Microcontrollers

MC68HC908QY4A
Rev. 3
03/2010

freescale.com

freescale™
semiconductor

2.6.6 FLASH Block Protect Register

The FLASH block protect register is implemented as a byte within the FLASH memory, and therefore can only be written during a programming sequence of the FLASH memory. The value in this register determines the starting address of the protected range within the FLASH memory.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
Reset:	Unaffected by reset. Initial value from factory is 1.							

Write to this register is by a programming sequence to the FLASH memory.

Figure 2-5. FLASH Block Protect Register (FLBPR)

BPR[7:0] — FLASH Protection Register Bits [7:0]

These eight bits in FLBPR represent bits [13:6] of a 16-bit memory address. Bits [15:14] are 1s and bits [5:0] are 0s.

The resultant 16-bit address is used for specifying the start address of the FLASH memory for block protection. The FLASH is protected from this start address to the end of FLASH memory, at $FFFF. With this mechanism, the protect start address can be XX00, XX40, XX80, or XXC0 within the FLASH memory. See Figure 2-6 and Table 2-2.

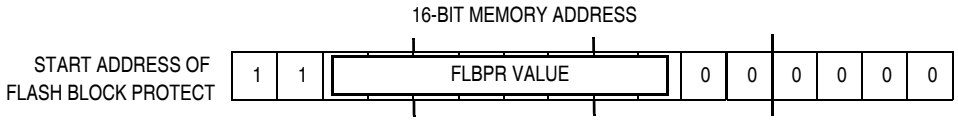


Figure 2-6. FLASH Block Protect Start Address

Table 2-2. Examples of Protect Start Address

BPR[7:0]	Start of Address of Protect Range
$00–$B8	The entire FLASH memory is protected.
$B8 (**1011 1000**)	$EE00 (11**10 1110 00**00 0000)
$B9 (**1011 1001**)	$EE40 (11**10 1110 01**00 0000)
$BA (**1011 1010**)	$EE80 (11**10 1110 10**00 0000)
$BB (**1011 1011**)	$EFC0 (11**10 1110 11**00 0000)
and so on...	
$DE (**1101 1110**)	$F780 (11**11 0111 10**00 0000)
$DF (**1101 1111**)	$F7C0 (11**11 0111 11**00 0000)
$FE (**1111 1110**)	$FF80 (11**11 1111 10**00 0000) FLBPR, internal oscillator trim values, and vectors are protected
$FF	The entire FLASH memory is not protected.

2.6.7 EEPROM Memory Emulation Using FLASH Memory

In some applications, the user may want to repeatedly store and read a set of data from an area of nonvolatile memory. This is easily implemented in EEPROM memory because single byte erase is allowed in EEPROM.

When using FLASH memory, the minimum erase size is a page. However, the FLASH can be used as EEPROM memory. This technique is called “EEPROM emulation”.

The basic concept of EEPROM emulation using FLASH is that a page is continuously programmed with a new data set without erasing the previously programmed locations. Once the whole page is completely programmed or the page does not have enough bytes to program a new data set, the user software automatically erases the page and then programs a new data set in the erased page.

In EEPROM emulation when data is read from the page, the user software must find the latest data set in the page since the previous data still remains in the same page. There are many ways to monitor the page erase timing and the latest data set. One example is unprogrammed FLASH bytes are detected by checking programmed bytes (non-$FF value) in a page. In this way, the end of the data set will contain unprogrammed data ($FF value).

A couple of application notes, describing how to emulate EEPROM using FLASH, are available on our web site. Titles and order numbers for these application notes are given at the end of this subsection.

For EEPROM emulation software to work successfully, the following items must be taken care of in the user software:

1. Each FLASH byte in a page must be programmed only one time until the page is erased.
2. A page must be erased before the FLASH cumulative program HV period (t_{HV}) is beyond the maximum t_{HV}. t_{HV} is defined as the cumulative high-voltage programming time to the same row before the next erase. For more detailed information, refer to 16.15 Memory Characteristics.
3. FLASH row erase and program cycles should not exceed 10,000 cycles, respectively.

The above EEPROM emulation software can be easily developed by using the on-chip FLASH routines implemented in the MCU. These routines are located in the ROM memory and support FLASH program and erase operations. Proper utilization of the on-chip FLASH routines guarantee conformance to the FLASH specifications.

In the on-chip FLASH programming routine called PRGRNGE, the high-voltage programming time is enabled for less than 125 μs when programming a single byte at any operating bus frequency between 1.0 MHz and 8.4 MHz. Therefore, even when a row is programmed by 32 separate single-byte programming operations, t_{HV} is less than the maximum t_{HV}. Hence, item 2 listed above is already taken care of by using this routine.

A page erased operation is provided in the FLASH erase routine called ERARNGE.

Application note AN2635 (*On-Chip FLASH Programming Routines*) describes how to use these routines.

The following application notes, available at www.freescale.com, describe how EERPOM emulation is implemented using FLASH:

AN2183 — *Using FLASH as EEPROM on the MC68HC908GP32*

AN2346 — *EEPROM Emulation Using FLASH in MC68HC908QY/QT MCUs*

AN2690 — *Low Frequency EEPROM Emulation on the MC68HC908QY4*

An EEPROM emulation driver, available at www.freescale.com, has been developed and qualified:

AN3040 — *M68HC08 EEPROM Emulation Driver*

Chapter 7
Central Processor Unit (CPU)

7.1 Introduction

The M68HC08 CPU (central processor unit) is an enhanced and fully object-code-compatible version of the M68HC05 CPU. The *CPU08 Reference Manual* (document order number CPU08RM/AD) contains a description of the CPU instruction set, addressing modes, and architecture.

7.2 Features

Features of the CPU include:

- Object code fully upward-compatible with M68HC05 Family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with x-register manipulation instructions
- 8-MHz CPU internal bus frequency
- 64-Kbyte program/data memory space
- 16 addressing modes
- Memory-to-memory data moves without using accumulator
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- Enhanced binary-coded decimal (BCD) data handling
- Modular architecture with expandable internal bus definition for extension of addressing range beyond 64 Kbytes
- Low-power stop and wait modes

7.3 CPU Registers

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

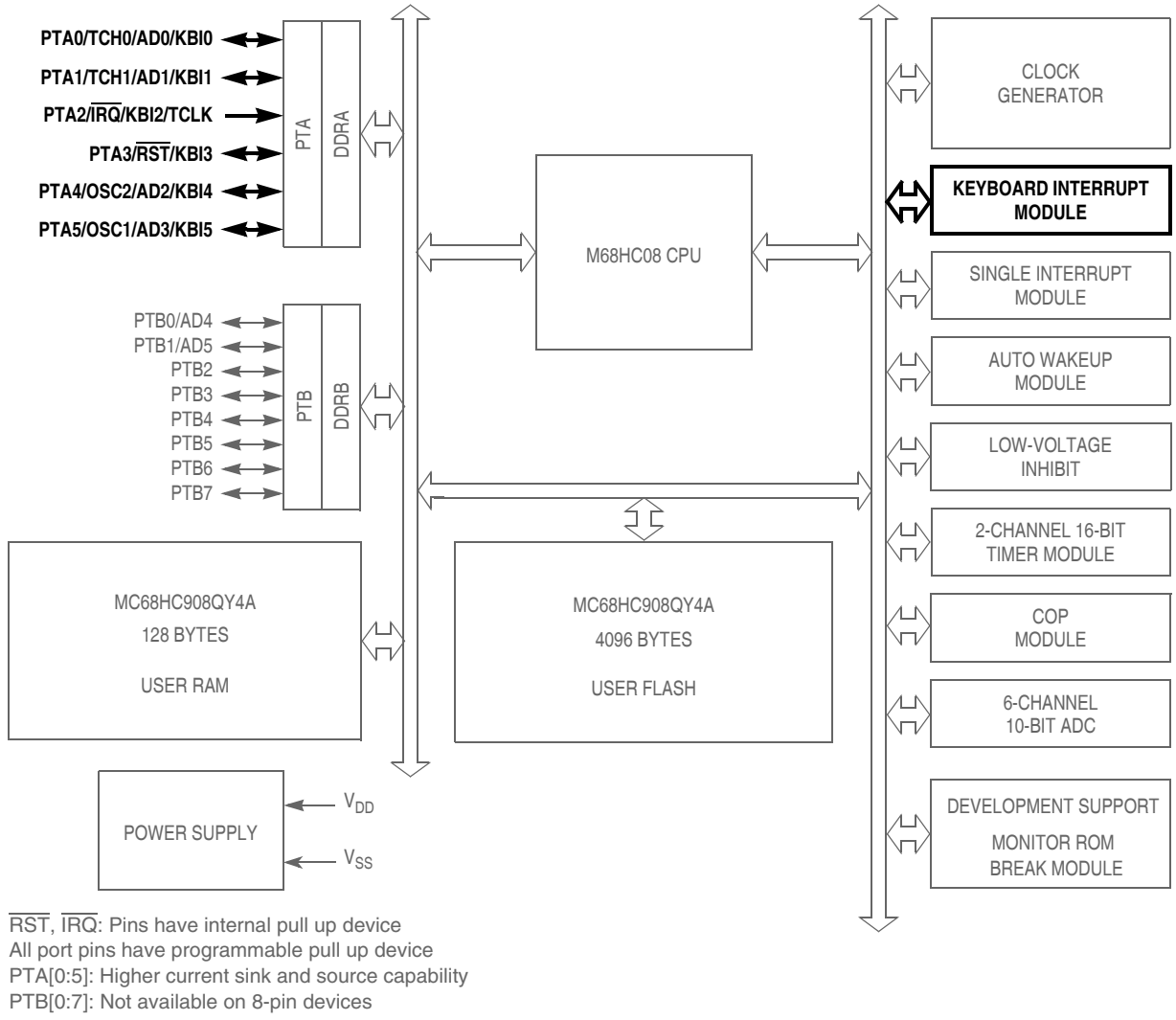
Table 7-2. Opcode Map

	Bit Manipulation		Branch	Read-Modify-Write						Control		Register/Memory							
	DIR	DIR	REL	DIR	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX
MSB / LSB	0	1	2	3	4	5	6	9E6	7	8	9	A	B	C	D	9ED	E	9EE	F
0	5 BRSET0 3 DIR	4 BSET0 2 DIR	3 BRA 2 REL	4 NEG 2 DIR	1 NEGA 1 INH	1 NEGX 1 INH	4 NEG 2 IX1	5 NEG 3 SP1	3 NEG 1 IX	7 RTI 1 INH	3 BGE 2 REL	2 SUB 2 IMM	3 SUB 2 DIR	4 SUB 3 EXT	4 SUB 3 IX2	5 SUB 4 SP2	3 SUB 2 IX1	4 SUB 3 SP1	2 SUB 1 IX
1	5 BRCLR0 3 DIR	4 BCLR0 2 DIR	3 BRN 2 REL	5 CBEQ 3 DIR	4 CBEQA 3 IMM	4 CBEQX 3 IMM	5 CBEQ 3 IX1+	6 CBEQ 4 SP1	4 CBEQ 2 IX+	4 RTS 1 INH	3 BLT 2 REL	2 CMP 2 IMM	3 CMP 2 DIR	4 CMP 3 EXT	4 CMP 3 IX2	5 CMP 4 SP2	3 CMP 2 IX1	4 CMP 3 SP1	2 CMP 1 IX
2	5 BRSET1 3 DIR	4 BSET1 2 DIR	3 BHI 2 REL		5 MUL 1 INH	7 DIV 1 INH	3 NSA 1 INH		2 DAA 1 INH		3 BGT 2 REL	2 SBC 2 IMM	3 SBC 2 DIR	4 SBC 3 EXT	4 SBC 3 IX2	5 SBC 4 SP2	3 SBC 2 IX1	4 SBC 3 SP1	2 SBC 1 IX
3	5 BRCLR1 3 DIR	4 BCLR1 2 DIR	3 BLS 2 REL	4 COM 2 DIR	1 COMA 1 INH	1 COMX 1 INH	4 COM 2 IX1	5 COM 3 SP1	3 COM 1 IX	9 SWI 1 INH	3 BLE 2 REL	2 CPX 2 IMM	3 CPX 2 DIR	4 CPX 3 EXT	4 CPX 3 IX2	5 CPX 4 SP2	3 CPX 2 IX1	4 CPX 3 SP1	2 CPX 1 IX
4	5 BRSET2 3 DIR	4 BSET2 2 DIR	3 BCC 2 REL	4 LSR 2 DIR	1 LSRA 1 INH	1 LSRX 1 INH	4 LSR 2 IX1	5 LSR 3 SP1	3 LSR 1 IX	2 TAP 1 INH	2 TXS 1 INH	2 AND 2 IMM	3 AND 2 DIR	4 AND 3 EXT	4 AND 3 IX2	5 AND 4 SP2	3 AND 2 IX1	4 AND 3 SP1	2 AND 1 IX
5	5 BRCLR2 3 DIR	4 BCLR2 2 DIR	3 BCS 2 REL	4 STHX 2 DIR	3 LDHX 3 IMM	4 LDHX 2 DIR	3 CPHX 3 IMM		4 CPHX 2 DIR	1 TPA 1 INH	2 TSX 1 INH	2 BIT 2 IMM	3 BIT 2 DIR	4 BIT 3 EXT	4 BIT 3 IX2	5 BIT 4 SP2	3 BIT 2 IX1	4 BIT 3 SP1	2 BIT 1 IX
6	5 BRSET3 3 DIR	4 BSET3 2 DIR	3 BNE 2 REL	4 ROR 2 DIR	1 RORA 1 INH	1 RORX 1 INH	4 ROR 2 IX1	5 ROR 3 SP1	3 ROR 1 IX	2 PULA 1 INH		2 LDA 2 IMM	3 LDA 2 DIR	4 LDA 3 EXT	4 LDA 3 IX2	5 LDA 4 SP2	3 LDA 2 IX1	4 LDA 3 SP1	2 LDA 1 IX
7	5 BRCLR3 3 DIR	4 BCLR3 2 DIR	3 BEQ 2 REL	4 ASR 2 DIR	1 ASRA 1 INH	1 ASRX 1 INH	4 ASR 2 IX1	5 ASR 3 SP1	3 ASR 1 IX	2 PSHA 1 INH	1 TAX 1 INH	2 AIS 2 IMM	3 STA 2 DIR	4 STA 3 EXT	4 STA 3 IX2	5 STA 4 SP2	3 STA 2 IX1	4 STA 3 SP1	2 STA 1 IX
8	5 BRSET4 3 DIR	4 BSET4 2 DIR	3 BHCC 2 REL	4 LSL 2 DIR	1 LSLA 1 INH	1 LSLX 1 INH	4 LSL 2 IX1	5 LSL 3 SP1	3 LSL 1 IX	2 PULX 1 INH	1 CLC 1 INH	2 EOR 2 IMM	3 EOR 2 DIR	4 EOR 3 EXT	4 EOR 3 IX2	5 EOR 4 SP2	3 EOR 2 IX1	4 EOR 3 SP1	2 EOR 1 IX
9	5 BRCLR4 3 DIR	4 BCLR4 2 DIR	3 BHCS 2 REL	4 ROL 2 DIR	1 ROLA 1 INH	1 ROLX 1 INH	4 ROL 2 IX1	5 ROL 3 SP1	3 ROL 1 IX	2 PSHX 1 INH	1 SEC 1 INH	2 ADC 2 IMM	3 ADC 2 DIR	4 ADC 3 EXT	4 ADC 3 IX2	5 ADC 4 SP2	3 ADC 2 IX1	4 ADC 3 SP1	2 ADC 1 IX
A	5 BRSET5 3 DIR	4 BSET5 2 DIR	3 BPL 2 REL	4 DEC 2 DIR	1 DECA 1 INH	1 DECX 1 INH	4 DEC 2 IX1	5 DEC 3 SP1	3 DEC 1 IX	2 PULH 1 INH	2 CLI 1 INH	2 ORA 2 IMM	3 ORA 2 DIR	4 ORA 3 EXT	4 ORA 3 IX2	5 ORA 4 SP2	3 ORA 2 IX1	4 ORA 3 SP1	2 ORA 1 IX
B	5 BRCLR5 3 DIR	4 BCLR5 2 DIR	3 BMI 2 REL	5 DBNZ 3 DIR	3 DBNZA 2 INH	3 DBNZX 2 INH	5 DBNZ 3 IX1	6 DBNZ 4 SP1	4 DBNZ 2 IX	2 PSHH 1 INH	2 SEI 1 INH	2 ADD 2 IMM	3 ADD 2 DIR	4 ADD 3 EXT	4 ADD 3 IX2	5 ADD 4 SP2	3 ADD 2 IX1	4 ADD 3 SP1	2 ADD 1 IX
C	5 BRSET6 3 DIR	4 BSET6 2 DIR	3 BMC 2 REL	4 INC 2 DIR	1 INCA 1 INH	1 INCX 1 INH	4 INC 2 IX1	5 INC 3 SP1	3 INC 1 IX	1 CLRH 1 INH	1 RSP 1 INH		2 JMP 2 DIR	3 JMP 3 EXT	4 JMP 3 IX2		3 JMP 2 IX1		2 JMP 1 IX
D	5 BRCLR6 3 DIR	4 BCLR6 2 DIR	3 BMS 2 REL	3 TST 2 DIR	1 TSTA 1 INH	1 TSTX 1 INH	3 TST 2 IX1	4 TST 3 SP1	2 TST 1 IX		1 NOP 1 INH	4 BSR 2 REL	4 JSR 2 DIR	5 JSR 3 EXT	6 JSR 3 IX2		5 JSR 2 IX1		4 JSR 1 IX
E	5 BRSET7 3 DIR	4 BSET7 2 DIR	3 BIL 2 REL		5 MOV 3 DD	4 MOV 2 DIX+	4 MOV 3 IMD		4 MOV 2 IX+D	1 STOP 1 INH	*	2 LDX 2 IMM	3 LDX 2 DIR	4 LDX 3 EXT	4 LDX 3 IX2	5 LDX 4 SP2	3 LDX 2 IX1	4 LDX 3 SP1	2 LDX 1 IX
F	5 BRCLR7 3 DIR	4 BCLR7 2 DIR	3 BIH 2 REL	3 CLR 2 DIR	1 CLRA 1 INH	1 CLRX 1 INH	3 CLR 2 IX1	4 CLR 3 SP1	2 CLR 1 IX	1 WAIT 1 INH	1 TXA 1 INH	2 AIX 2 IMM	3 STX 2 DIR	4 STX 3 EXT	4 STX 3 IX2	5 STX 4 SP2	3 STX 2 IX1	4 STX 3 SP1	2 STX 1 IX

INH Inherent
IMM Immediate
DIR Direct
EXT Extended
DD Direct-Direct
IX+D Indexed-Direct
REL Relative
IX Indexed, No Offset
IX1 Indexed, 8-Bit Offset
IX2 Indexed, 16-Bit Offset
IMD Immediate-Direct
DIX+ Direct-Indexed
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
*Pre-byte for stack pointer indexed instructions

MSB / LSB	0
0	5 BRSET0 3 DIR

Low Byte of Opcode in Hexadecimal — High Byte of Opcode in Hexadecimal
Cycles
Opcode Mnemonic
Number of Bytes / Addressing Mode



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability
PTB[0:7]: Not available on 8-pin devices

Figure 9-1. Block Diagram Highlighting KBI Block and Pins

9.3.1.1 MODEK = 1

If the MODEK bit is set, the keyboard interrupt inputs are both edge and level sensitive. The KBIPx bit will determine whether a edge sensitive pin detects rising or falling edges and on level sensitive pins whether the pin detects low or high levels. With MODEK set, both of the following actions must occur to clear a keyboard interrupt request:

- Return of all enabled keyboard interrupt inputs to a deasserted level. As long as any enabled keyboard interrupt pin is asserted, the keyboard interrupt remains active.
- Vector fetch or software clear. A KBI vector fetch generates an interrupt acknowledge signal to clear the KBI latch. Software generates the interrupt acknowledge signal by writing a 1 to ACKK in KBSCR. The ACKK bit is useful in applications that poll the keyboard interrupt inputs and require software to clear the KBI latch. Writing to ACKK prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt inputs. An edge detect that occurs after writing to ACKK latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the KBI vector address.

13.6 Exception Control

Normal sequential program execution can be changed in three different ways:

1. Interrupts
   a. Maskable hardware CPU interrupts
   b. Non-maskable software interrupt instruction (SWI)
2. Reset
3. Break interrupts

13.6.1 Interrupts

An interrupt temporarily changes the sequence of program execution to respond to a particular event. Figure 13-7 flow charts the handling of system interrupts.

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared).

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. Figure 13-8 shows interrupt entry timing. Figure 13-9 shows interrupt recovery timing.

13.6.1.1 Hardware Interrupts

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register), and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. Figure 13-10 demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

The LDA opcode is prefetched by both the INT1 and INT2 return-from-interrupt (RTI) instructions. However, in the case of the INT1 RTI prefetch, this is a redundant operation.

NOTE

To maintain compatibility with the M6805 Family, the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.

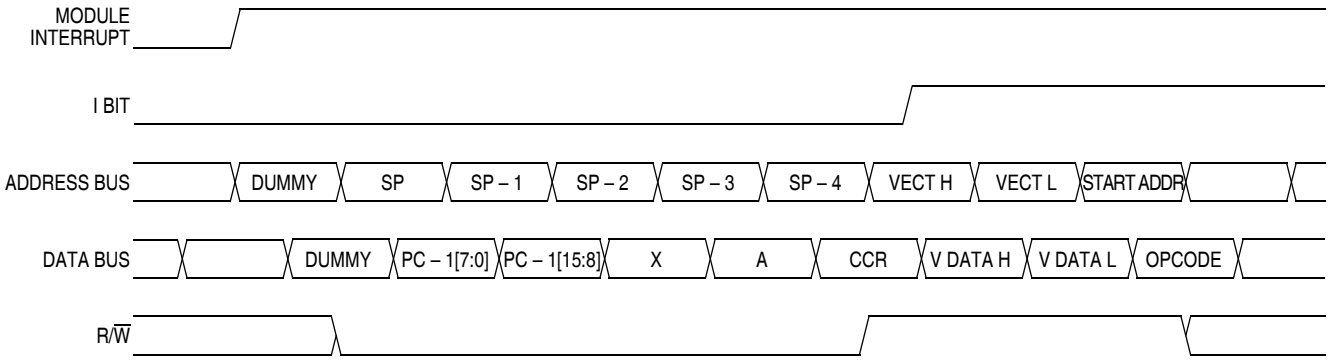


Figure 13-8. Interrupt Entry

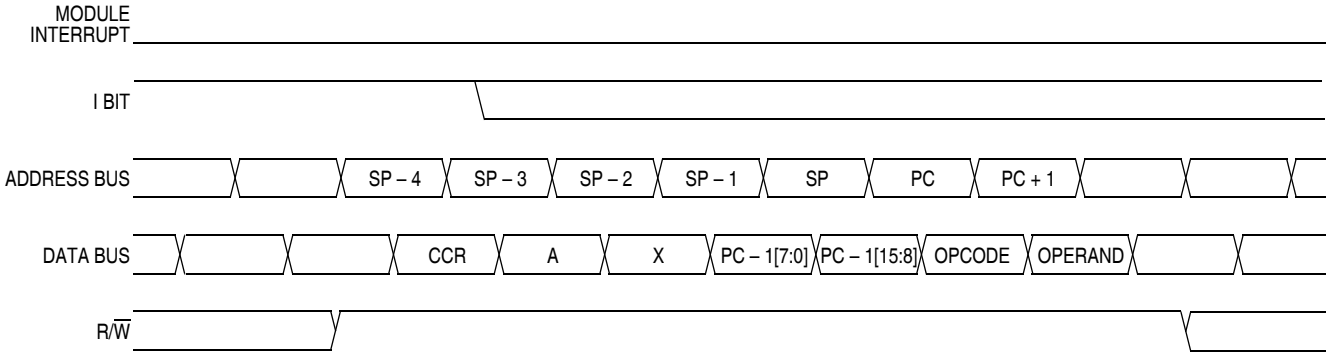


Figure 13-9. Interrupt Recovery

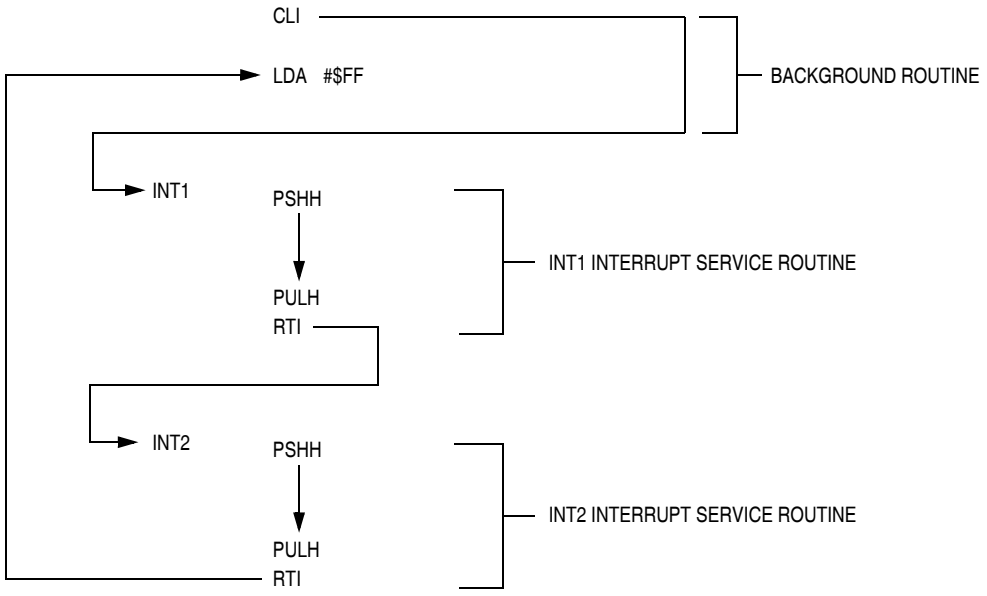


Figure 13-10. Interrupt Recognition Example

13.6.1.2 SWI Instruction

The SWI instruction is a non-maskable instruction that causes an interrupt regardless of the state of the interrupt mask (I bit) in the condition code register.

NOTE

A software interrupt pushes PC onto the stack. A software interrupt does ***not*** *push PC – 1, as a hardware interrupt does.*

13.6.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. Table 13-3 summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

Table 13-3. Interrupt Sources

Priority	Source	Flag	Mask[1]	INT Register Flag	Vector Address
Highest	Reset	—	—	—	$FFFE–$FFFF
	SWI instruction	—	—	—	$FFFC–$FFFD
	$\overline{\text{IRQ}}$ pin	IRQF	IMASK	IF1	$FFFA–$FFFB
	Timer channel 0 interrupt	CH0F	CH0IE	IF3	$FFF6–$FFF7
	Timer channel 1 interrupt	CH1F	CH1IE	IF4	$FFF4–$FFF5
	Timer overflow interrupt	TOF	TOIE	IF5	$FFF2–$FFF3
	Keyboard interrupt	KEYF	IMASKK	IF14	$FFE0–$FFE1
Lowest	ADC conversion complete interrupt	COCO	AIEN	IF15	$FFDE–$FFDF

1. The I bit in the condition code register is a global mask for all interrupt sources except the SWI instruction.

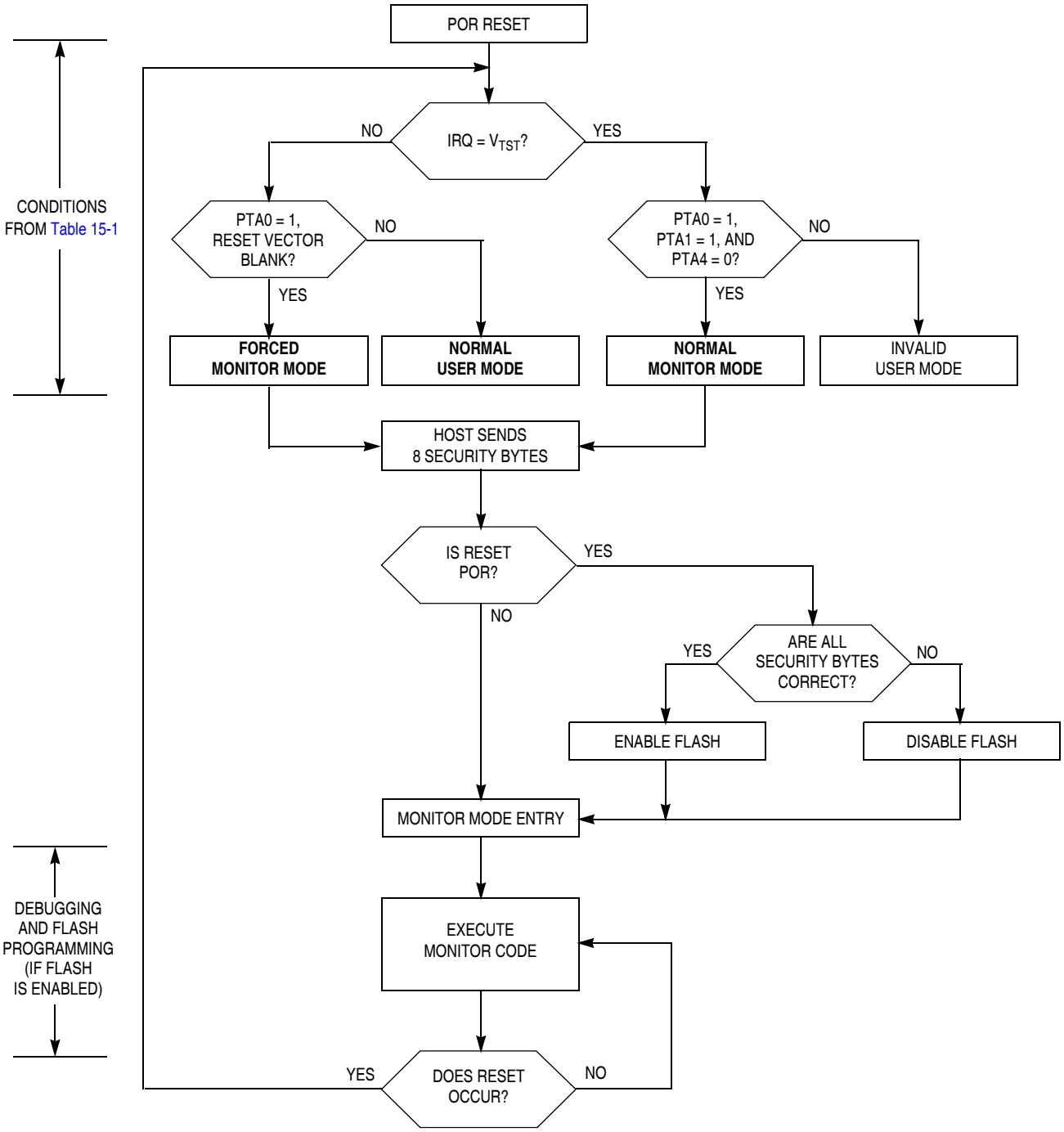


Figure 15-9. Simplified Monitor Mode Entry Flowchart

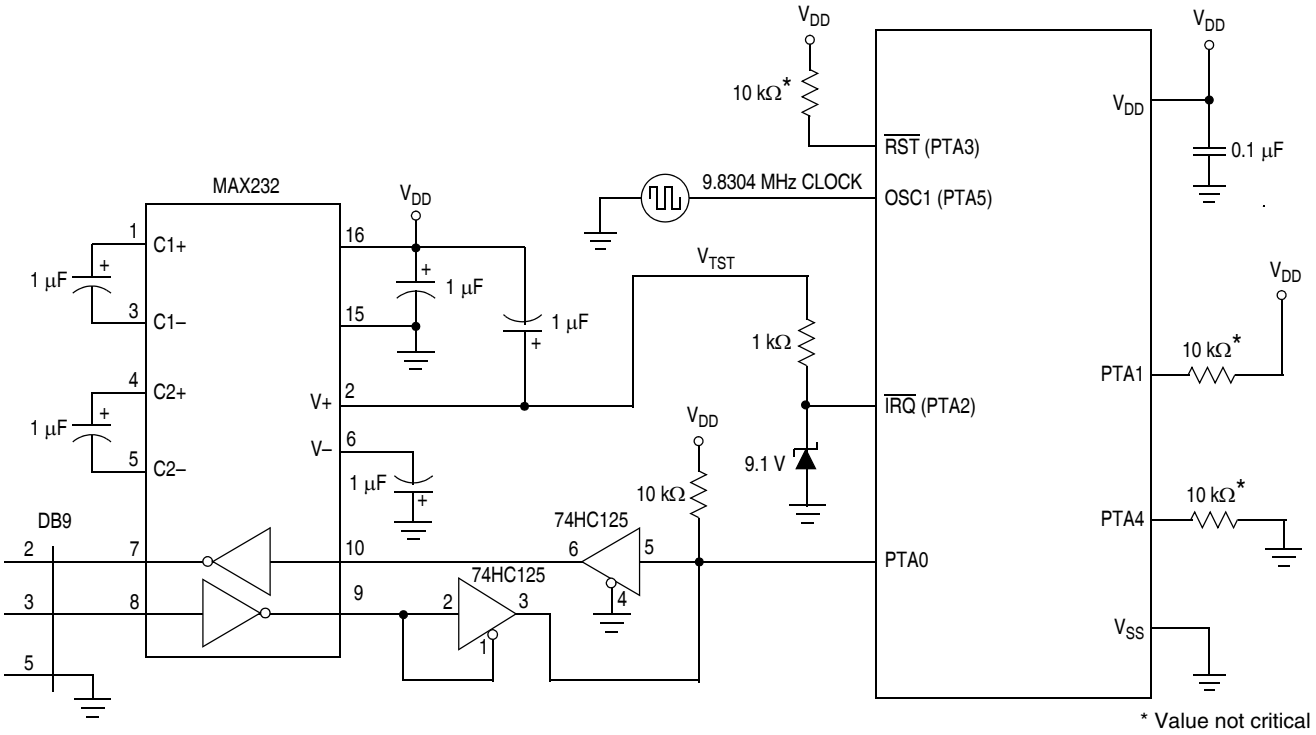


Figure 15-10. Monitor Mode Circuit (External Clock, with High Voltage)

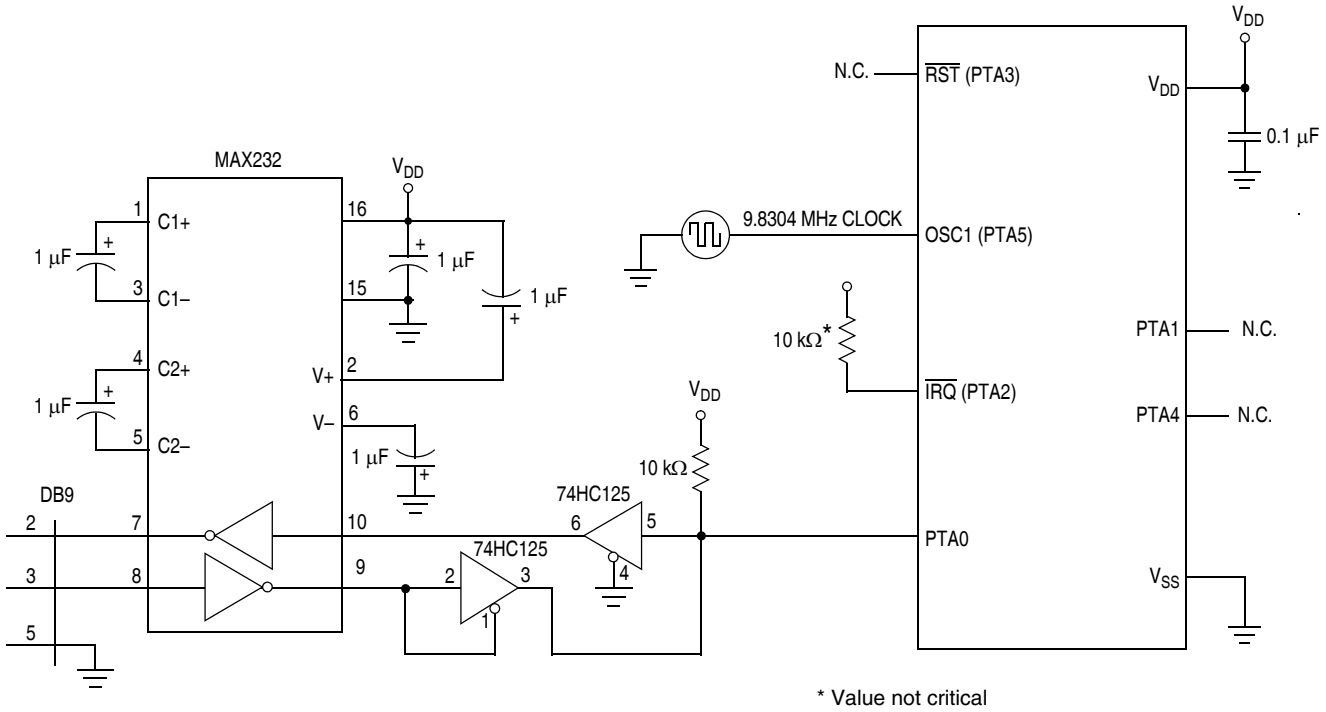


Figure 15-11. Monitor Mode Circuit (External Clock, No High Voltage)

16.3 Functional Operating Range

Characteristic	Symbol	Value	Unit	Temperature Code
Operating temperature range	T_A (T_L to T_H)	−40 to +125 −40 to +105 −40 to +85	°C	M V C
Operating voltage range	V_{DD}	2.7 to 5.5	V	—

16.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance 8-pin PDIP 8-pin SOIC 16-pin PDIP 16-pin SOIC 16-pin TSSOP	θ_{JA}	 105 142 76 90 133	°C/W
I/O pin power dissipation	$P_{I/O}$	User determined	W
Power dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273°C)$	W
Constant(2)	K	$P_D \times (T_A + 273°C) + P_D^2 \times \theta_{JA}$	W/°C
Average junction temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C
Maximum junction temperature	T_{JM}	150	°C

1. Power dissipation is a function of temperature.
2. K constant unique to the device. K can be determined for a known T_A and measured P_D. With this value of K, P_D and T_J can be determined for any value of T_A.

16.6 Typical 5-V Output Drive Characteristics

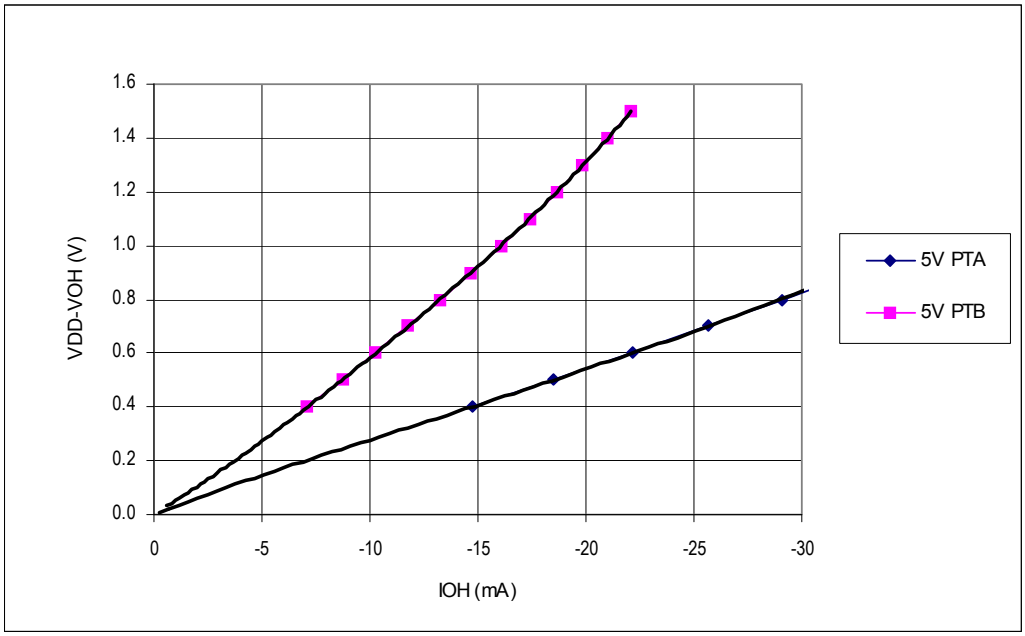


Figure 16-1. Typical 5-Volt Output High Voltage versus Output High Current (25°C)

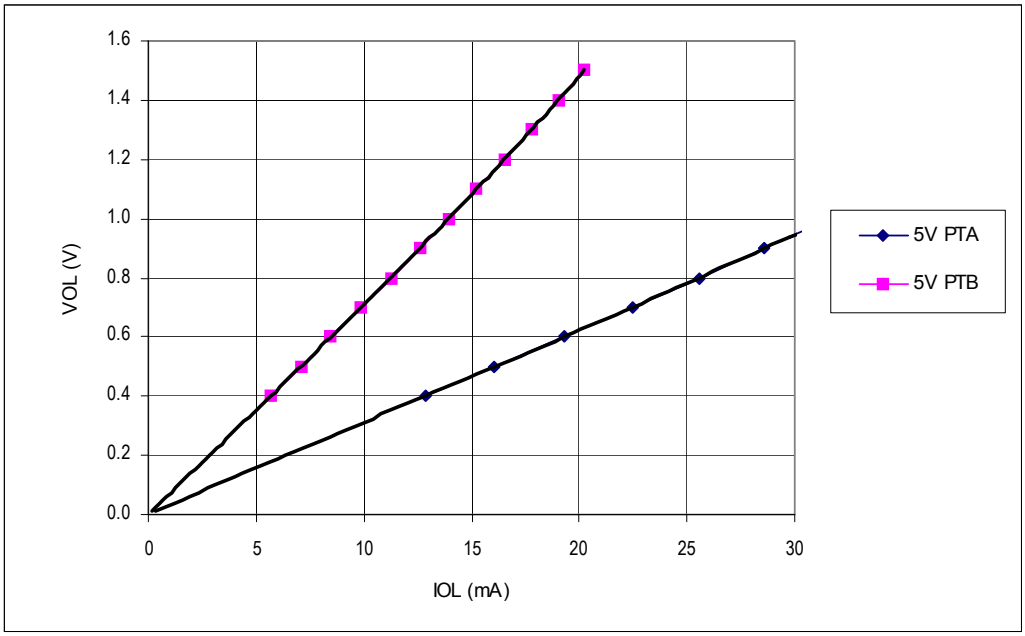


Figure 16-2. Typical 5-Volt Output Low Voltage versus Output Low Current (25°C)

16.13 ADC10 Characteristics

Characteristic	Conditions	Symbol	Min	Typ(1)	Max	Unit	Comment
Supply voltage	Absolute	V_{DD}	2.7	—	5.5	V	
Supply Current ADLPC = 1 ADLSMP = 1 ADCO = 1	$V_{DD} \leq 3.3$ V (3.0 V Typ)	I_{DD}(2)	—	55	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	75	—		
Supply current ADLPC = 1 ADLSMP = 0 ADCO = 1	$V_{DD} \leq 3.3$ V (3.0 V Typ)	I_{DD}(2)	—	120	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	175	—		
Supply current ADLPC = 0 ADLSMP = 1 ADCO = 1	$V_{DD} \leq 3.3$ V (3.0 V Typ)	I_{DD}(2)	—	140	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	180	—		
Supply current ADLPC = 0 ADLSMP = 0 ADCO = 1	$V_{DD} \leq 3.3$ V (3.0 V Typ)	I_{DD}(2)	—	340	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	440	615		
ADC internal clock	High speed (ADLPC = 0)	f_{ADCK}	0.40(3)	—	2.00	MHz	$t_{ADCK} = 1/f_{ADCK}$
	Low power (ADLPC = 1)		0.40(3)	—	1.00		
Conversion time (4) 10-bit Mode	Short sample (ADLSMP = 0)	t_{ADC}	19	19	21	t_{ADCK} cycles	
	Long sample (ADLSMP = 1)		39	39	41		
Conversion time (4) 8-bit Mode	Short sample (ADLSMP = 0)	t_{ADC}	16	16	18	t_{ADCK} cycles	
	Long sample (ADLSMP = 1)		36	36	38		
Sample time	Short sample (ADLSMP = 0)	t_{ADS}	4	4	4	t_{ADCK} cycles	
	Long sample (ADLSMP = 1)		24	24	24		
Input voltage		V_{ADIN}	V_{SS}	—	V_{DD}	V	
Input capacitance		C_{ADIN}	—	7	10	pF	Not tested
Input impedance		R_{ADIN}	—	5	15	kΩ	Not tested
Analog source impedance		R_{AS}	—	—	10	kΩ	External to MCU
Ideal resolution (1 LSB)	10-bit mode	RES	1.758	5	5.371	mV	$V_{REFH}/2^N$
	8-bit mode		7.031	20	21.48		
Total unadjusted error	10-bit mode	E_{TUE}	0	±1.5	±2.5	LSB	Includes quantization
	8-bit mode		0	±0.7	±1.0		
Differential non-linearity	10-bit mode	DNL	0	±0.5	—	LSB	
	8-bit mode		0	±0.3	—		
	Monotonicity and no-missing-codes guaranteed						

— Continued on next page

Case 626 page 3 of 3

Case 648 page 2 of 3

- The ADC that is on the QYxA can operate while the MCU is in stop mode allowing lower power operation. This also adds a lower noise environment for precise ADC results.
- Enabling an ADC channel no longer overrides the digital I/O function of the associated pin. To prevent the digital I/O from interfering with the ADC read of the pin, the data direction bit associated with the port pin must be set as input.
- Finally, the new ADC can be configured to select two different reference clock sources:
  – The internal bus x 4
  – An internal asynchronous source

  The internal asynchronous clock source allows the ADC to be clocked for operation in stop mode.

A.2.1.1 Registers Affected

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
Write:								
Reset:	0	0	0	1	1	1	1	1

= Unimplemented

Figure A-1. ADC10 Status and Control Register (ADSCR)

The ADCHx bits can be used to select additional ADC channels or bandgap measurement.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	AD9	AD8
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure A-2. ADC10 Data Register High (ADRH), 10-Bit Mode

10-bit ADC uses the new ADRH register for the upper 2 bits.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ACLKEN
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure A-3. ADC10 Clock Register (ADCLK)

A long sample time option has been added to conserve power at the expense of longer conversion times. This option is selected using the new ADLSMP bit in the ADCLK register. (The bit location was previously reserved.)

The ADC will now run in stop mode if the ACLKEN bit is set to enable the asynchronous clock inside the ADC module. Utilizing stop mode for an ADC conversion gives the quietest operating mode to get extremely accurate ADC readings. (This bit location now used by ACLKEN was reserved — it always read as a 0 and writes to that location had no affect.)

A.2.5 Keyboard Interface Module (KBI) Functionality

The KBI module for the QYxA has the added capability of:

- Triggering a KBI interrupt on the rising or falling edge of an input while the QYx Classic has the capability of triggering on falling edges only.
  - A new register (Keyboard Interrupt Polarity Register) determines the polarity of KBI and the default state of this register configures the QYxA for triggering on falling edges to be compatible with QYx Classic.
  - The QYxA now has pull down resistors for the input pins that are configured for rising edge operation.

A.2.5.1 Registers Affected

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	KBIP5	KBIP4	KBIP3	KBIP2	KBIP1	KBIP0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure A-6. Keyboard Interrupt Polarity Register (KBIPR)

The KBIPR allows the selection of polarity, if any of these bits are set the corresponding interrupt pin will be configured for rising edge and a pulldown resistor will be added to the pin.

A.2.6 On-Chip Routine Enhancements

Enhancements have been made to the on-chip routines that are used for FLASH as EEPROM. Refer to AN2346 for information about using FLASH as EEPROM.

- A new mass erase routine requires a valid FLASH address loaded into the H:X register to perform an erase. This added step helps ensure that the erase routine is not inadvertently used to cause an unwanted erase. Also, on-chip FLASH programming routine ERARNGE variable CTRLBYT requires $00 for page erase and $40 for mass erase. The entire control byte must be set for proper operation.
- Separate routines will allow easy access to perform software SCI (Serial Communications Interface). For information on how to use on-chip FLASH programming routines refer to AN2635.
- Finally, there is improved security and robustness. The latest Monitor ROM implements updated security checks to make the program memory more secure.

A.3 Conversion Considerations

Enhancements lead to slight differences in operation from QYx Classic to the QYxA. There are a few points that should be considered in the conversion process.

- The Monitor ROM changed from 2 K to 1 K in size. This has led to the limitation that programming across page boundaries is no longer supported by the on-chip program range routine. Also, in very rare cases, ROM code improvements could cause customers to have to modify a few instructions in their application code. For example, when performing a mass erase, a valid address is required instead of an unspecified address.
- The QYxA contains new modules like the 10-bit ADC and OSC. In rare cases, new modules could cause customers to have to modify a few instructions in their application code. For example, if ADC code was written so that entire registers are configured without respect to reserve bits, then the ADC code will need to be revised to work correctly on the QYxA.
- The Reference Clock for ADC conversions has changed from the bus clock to the system clock (Bus Clock * 4). A change to the divide register may be necessary to set the reference clock to a specified value.

A.4 Code Changes Checklist

Below is a checklist that should be reviewed in the conversion process. This checklist will point out all the issues that should be addressed as your code is ported.

1. Does the original software use Auxiliary ROM routines (for example, Getbyte, Putbyte, delnus)?
   If so, the software will have to be changed to handle new Auxiliary ROM routines, addresses of these routines have changed in QYxA. Code will have to be changed to use the proper addresses.
2. Does the software use FLASH as EEPROM?
   If so, there are several possible issues for the page erase and mass erase routine. Software will have to be checked to ensure that proper procedure is used and the CTRLBYT is set with a MOV instruction not a BSET. Also, on-chip FLASH programming routines can no longer program across row boundaries
3. Does the code use the auto wake up timer and does the application depend on the typical auto wake time out?
   Since the timeout has been improved for QYxA it may be necessary to modify software to compensate for the change in timeout.
4. Bits changed in the OSCSC, CONFIG2, and ADC registers?
   Any code that writes to these registers should be reviewed to ensure that the writes are not affecting the changed bits
5. Does the code use external OSC, crystal, or RC?
   If so, since the OSCOPT bits have changed locations code will have to be updated to update these bits in their proper locations.
6. Does the code use the ADC?
   If so, because on QYxA the ADC clock is driven from 4XBUSCLK instead of BUSCLK changes to the ADC clock divider bits may be needed to maintain proper operation.