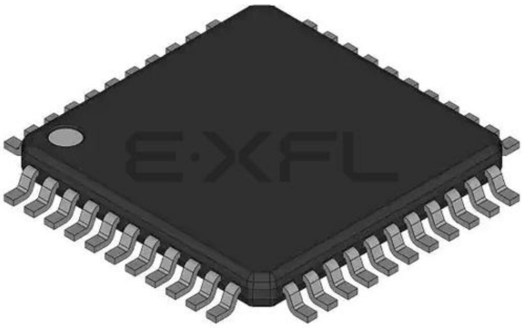
Motorola - MC68HC908BD48IFB Datasheet



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	HC08
Core Size	8-Bit
Speed	6MHz
Connectivity	I²C, USB
Peripherals	POR, PWM
Number of I/O	32
Program Memory Size	48KB (48K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 6x8b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	44-QFP
Supplier Device Package	44-QFP (10x10)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68hc908bd48ifb

Section 21. Electrical Specifications

Section 1. General Description

1.1 Contents

1.2 Introduction

The MC68HC908BD48 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 Family is based on the customer-specified integrated circuit (CSIC) design strategy. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

With special modules such as the sync processor, analog-to-digital converter, pulse modulator module, DDC12AB interface, multi-master IIC interface, and universal serial bus interface, the MC68HC908BD48 is designed specifically for use in digital monitor systems.

1.3 Features

Features of the MC68HC908BD48 MCU include the following:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 families

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0032	USB Endpoint 0 Data Register 2 (UD0R2)	Read:	UE0RD27	UE0RD26	UE0RD25	UE0RD24	UE0RD23	UE0RD22	UE0RD21	UE0RD20
		Write:	UE0TD27	UE0TD26	UE0TD25	UE0TD24	UE0TD23	UE0TD22	UE0TD21	UE0TD20
		Reset:	Indeterminate after reset							
$0033	USB Endpoint 0 Data Register 3 (UD0R3)	Read:	UE0RD37	UE0RD36	UE0RD35	UE0RD34	UE0RD33	UE0RD32	UE0RD31	UE0RD30
		Write:	UE0TD37	UE0TD36	UE0TD35	UE0TD34	UE0TD33	UE0TD32	UE0TD31	UE0TD30
		Reset:	Indeterminate after reset							
$0034	USB Endpoint 0 Data Register 4 (UD0R4)	Read:	UE0RD47	UE0RD46	UE0RD45	UE0RD44	UE0RD43	UE0RD42	UE0RD41	UE0RD40
		Write:	UE0TD47	UE0TD46	UE0TD45	UE0TD44	UE0TD43	UE0TD42	UE0TD41	UE0TD40
		Reset:	Indeterminate after reset							
$0035	USB Endpoint 0 Data Register 5 (UD0R5)	Read:	UE0RD57	UE0RD56	UE0RD55	UE0RD54	UE0RD53	UE0RD52	UE0RD51	UE0RD50
		Write:	UE0TD57	UE0TD56	UE0TD55	UE0TD54	UE0TD53	UE0TD52	UE0TD51	UE0TD50
		Reset:	Indeterminate after reset							
$0036	USB Endpoint 0 Data Register 6 (UD0R6)	Read:	UE0RD67	UE0RD66	UE0RD65	UE0RD64	UE0RD63	UE0RD62	UE0RD61	UE0RD60
		Write:	UE0TD67	UE0TD66	UE0TD65	UE0TD64	UE0TD63	UE0TD62	UE0TD61	UE0TD60
		Reset:	Indeterminate after reset							
$0037	USB Endpoint 0 Data Register 7 (UD0R7)	Read:	UE0RD77	UE0RD76	UE0RD75	UE0RD74	UE0RD73	UE0RD72	UE0RD71	UE0RD70
		Write:	UE0TD77	UE0TD76	UE0TD75	UE0TD74	UE0TD73	UE0TD72	UE0TD71	UE0TD70
		Reset:	Indeterminate after reset							
$0038	USB Endpoint 1/2 Data Register 0 (UD1R0)	Read:	▨	▨	▨	▨	▨	▨	▨	▨
		Write:	UE1TD07	UE1TD06	UE1TD05	UE1TD04	UE1TD03	UE1TD02	UE1TD01	UE1TD00
		Reset:	Indeterminate after reset							
$0039	USB Endpoint 1/2 Data Register 1 (UD1R1)	Read:	▨	▨	▨	▨	▨	▨	▨	▨
		Write:	UE1TD17	UE1TD16	UE1TD15	UE1TD14	UE1TD13	UE1TD12	UE1TD11	UE1TD10
		Reset:	Indeterminate after reset							
$003A	USB Endpoint 1/2 Data Register 2 (UD1R2)	Read:	▨	▨	▨	▨	▨	▨	▨	▨
		Write:	UE1TD27	UE1TD26	UE1TD25	UE1TD24	UE1TD23	UE1TD22	UE1TD21	UE1TD20
		Reset:	Indeterminate after reset							
$003B	USB Endpoint 1/2 Data Register 3 (UD1R3)	Read:	▨	▨	▨	▨	▨	▨	▨	▨
		Write:	UE1TD37	UE1TD36	UE1TD35	UE1TD34	UE1TD33	UE1TD32	UE1TD31	UE1TD30
		Reset:	Indeterminate after reset							

▨ = Unimplemented   R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 12)

4. Set the HVEN bit.
5. Wait for a time, t_{Erase} (min. 2ms)
6. Clear the ERASE bit.
7. Wait for a time, t_{nvh} (min. 5µs)
8. Clear the HVEN bit.
9. After a time, t_{rcv} (min. 1µs), the memory can be accessed again in read mode.

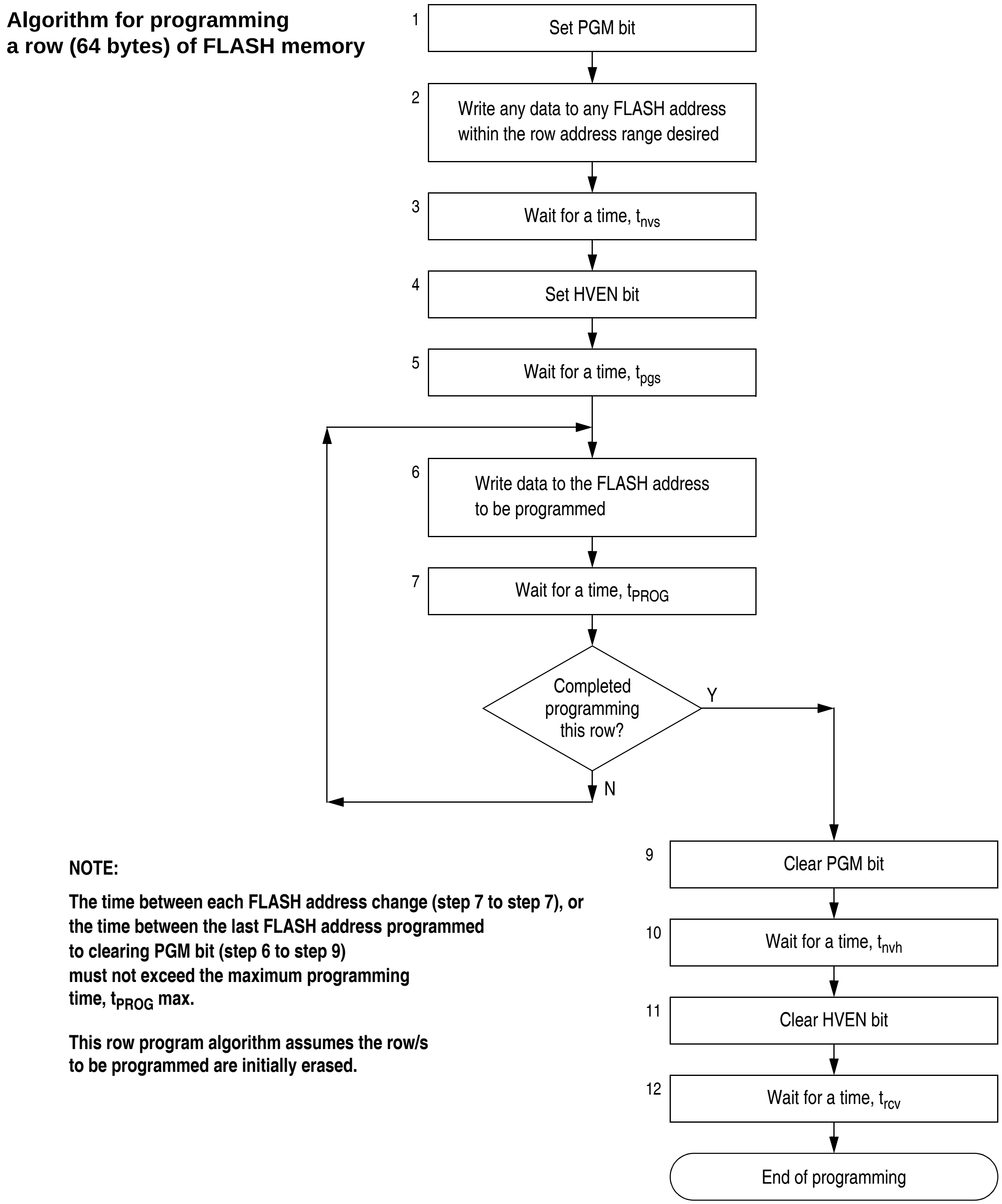
NOTE: *Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order as shown, but other unrelated operations may occur between the steps.*

4.6 FLASH Mass Erase Operation

Use the following procedure to erase entire FLASH memory:

1. Set both the ERASE bit, and the MASS bit in the FLASH control register.
2. Write any data to any FLASH address within the FLASH memory address range.
3. Wait for a time, t_{nvs} (5µs).
4. Set the HVEN bit.
5. Wait for a time, t_{MErase} (4ms).
6. Clear the ERASE bit.
7. Wait for a time, t_{nvhl} (100µs).
8. Clear the HVEN bit.
9. After time, t_{rcv} (1µs)$_{,}$ the memory can be accessed again in read mode.

NOTE: *Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order as shown, but other unrelated operations may occur between the steps.*


Algorithm for programming
a row (64 bytes) of FLASH memory
1
Set PGM bit
2
Write any data to any FLASH address within the row address range desired
3
Wait for a time, t_{nvs}
4
Set HVEN bit
5
Wait for a time, t_{pgs}
6
Write data to the FLASH address to be programmed
7
Wait for a time, t_{PROG}
Completed programming this row?
Y
N
9
Clear PGM bit
10
Wait for a time, t_{nvh}
11
Clear HVEN bit
12
Wait for a time, t_{rcv}
End of programming

NOTE:

The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH address programmed to clearing PGM bit (step 6 to step 9) must not exceed the maximum programming time, t_{PROG} max.

This row program algorithm assumes the row/s to be programmed are initially erased.

Figure 4-2. FLASH Programming Flowchart

Section 6. Central Processor Unit (CPU)

6.1 Contents

6.2 Introduction

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

Table 6-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X CMP *opr*,SP CMP *opr*,SP	Compare A with M	(A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9EE1 9ED1	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
COM *opr* COMA COMX COM *opr*,X COM ,X COM *opr*,SP	Complement (One's Complement)	$M \leftarrow (\overline{M}) = \$FF - (M)$ $A \leftarrow (\overline{A}) = \$FF - (M)$ $X \leftarrow (\overline{X}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$	0	–	–	↕	↕	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	4 1 1 4 3 5
CPHX #*opr* CPHX *opr*	Compare H:X with M	(H:X) – (M:M + 1)	↕	–	–	↕	↕	↕	IMM DIR	65 75	ii ii+1 dd	3 4
CPX #*opr* CPX *opr* CPX *opr* CPX ,X CPX *opr*,X CPX *opr*,X CPX *opr*,SP CPX *opr*,SP	Compare X with M	(X) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A3 B3 C3 D3 E3 F3 9EE3 9ED3	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
DAA	Decimal Adjust A	$(A)_{10}$	U	–	–	↕	↕	↕	INH	72		2
DBNZ *opr*,*rel* DBNZA *rel* DBNZX *rel* DBNZ *opr*,X,*rel* DBNZ X,*rel* DBNZ *opr*,SP,*rel*	Decrement and Branch if Not Zero	A ← (A) – 1 or M ← (M) – 1 or X ← (X) – 1 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 4 + *rel* ? (result) ≠ 0	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	5 3 3 5 4 6
DEC *opr* DECA DECX DEC *opr*,X DEC ,X DEC *opr*,SP	Decrement	M ← (M) – 1 A ← (A) – 1 X ← (X) – 1 M ← (M) – 1 M ← (M) – 1 M ← (M) – 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	4 1 1 4 3 5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	–	–	–	–	↕	↕	INH	52		7

PIN — External Reset Bit

1 = Last reset caused by external reset pin ($\overline{\text{RST}}$)

0 = POR or read of SRSR

COP — Computer Operating Properly Reset Bit

1 = Last reset caused by COP counter

0 = POR or read of SRSR

ILOP — Illegal Opcode Reset Bit

1 = Last reset caused by an illegal opcode

0 = POR or read of SRSR

ILAD — Illegal Address Reset Bit (opcode fetches only)

1 = Last reset caused by an opcode fetch from an illegal address

0 = POR or read of SRSR

7.8.3 SIM Break Flag Control Register (SBFCR)

The SIM break flag control register contains a bit that enables software to clear status bits while the MCU is in a break state.

Address: $FE03

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

R = Reserved

Figure 7-20. SIM Break Flag Control Register (SBFCR)

BCFE — Break Clear Flag Enable Bit

This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.

1 = Status bits clearable during break

0 = Status bits not clearable during break

Table 11-1. PWM I/O Register Summary

$0051	PWM8 Data Register (8PWM)	Read: Write:	8PWM4	8PWM3	8PWM2	8PWM1	8PWM0	8BRM2	8BRM1	8BRM0
$0052	PWM9 Data Register (9PWM)	Read: Write:	9PWM4	9PWM3	9PWM2	9PWM1	9PWM0	9BRM2	9BRM1	9BRM0
$0053	PWM10 Data Register (10PWM)	Read: Write:	10PWM4	10PWM3	10PWM2	10PWM1	10PWM0	10BRM2	10BRM1	10BRM0
$0054	PWM11 Data Register (11PWM)	Read: Write:	11PWM4	11PWM3	11PWM2	11PWM1	11PWM0	11BRM2	11BRM1	11BRM0
$0055	PWM12 Data Register (12PWM)	Read: Write:	12PWM4	12PWM3	12PWM2	12PWM1	12PWM0	12BRM2	12BRM1	12BRM0
$0056	PWM13 Data Register (13PWM)	Read: Write:	13PWM4	13PWM3	13PWM2	13PWM1	13PWM0	13BRM2	13BRM1	13BRM0
$0057	PWM14 Data Register (14PWM)	Read: Write:	14PWM4	PWM3	14PWM2	14PWM1	14PWM0	14BRM2	14BRM1	14BRM0
$0058	PWM15 Data Register (15PWM)	Read: Write:	15PWM4	15PWM3	15PWM2	15PWM1	15PWM0	15BRM2	15BRM1	15BRM0
$0059	PWM Control Register 2 (PWMCR2)	Read: Write:	PWM15E	PWM14E	PWM13E	PWM12E	PWM11E	PWM10E	PWM9E	PWM8E
		Reset:	0	0	0	0	0	0	0	0

11.4 PWM Registers

The PWM module uses of 18 registers for data and control functions.

- 16 PWM data registers ($0020–$0027 and $0051–$0058)
- 2 PWM control registers ($0028 and $0059)

11.4.1 PWM Data Registers 0 to 15 (0PWM–15PWM)

Address: $0020–$0027 and $0051–$0058

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	xPWM4	xPWM3	xPWM2	xPWM1	xPWM0	xBRM2	xBRM1	xBRM0
Reset:	0	0	0	0	0	0	0	0

Figure 11-1. PWM Data Registers 0 to 15 (0PWM–15PWM)

The output waveform of the 16 PWM channels are each configured by an 8-bit register, which contains a 5-bit PWM in MSB portion and a 3-bit binary rate multiplier (BRM) in LSB portion

xPWM4–xPWM0 — PWM Bits

The value programmed in the 5-bit PWM portion will determine the pulse length of the output. The clock to the 5-bit PWM portion is the system clock (CPU clock), the repetition rate of the output is hence $f_{OP} \div 32$. Examples of PWM output waveforms are shown in **Figure 11-3**.

xBRM2–xBRM0 — Binary Rate Multiplier Bits

The 3-bit BRM will generate a number of narrow pulses which are equally distributed among an 8-PWM-cycle frame. The number of pulses generated is equal to the number programmed in the 3-bit BRM portion. Examples of PWM output waveforms are shown in **Figure 11-3**.

12.3 Features

Features of the ADC module include:

- 6 Channels ADC with Multiplexed Input
- Linear Successive Approximation
- 8-Bit Resolution
- Single or Continuous Conversion
- Conversion Complete Flag or Conversion Complete Interrupt
- Selectable ADC Clock

Table 12-1. ADC Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$005D	ADC Status and Control Register (ADSCR)	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:	▨							
		Reset:	0	0	0	1	1	1	1	1
$005E	ADC Data Register (ADR)	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	Indeterminate after Reset							
$005F	ADC Input Clock Register (ADICLK)	Read:	ADIV2	ADIV1	ADIV0	0	0	0	0	0
		Write:				▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

12.4 Functional Description

Six ADC channels are available for sampling external sources at pins PTC5–PTC0. An analog multiplexer allows the single ADC converter to select one of the 6 ADC channels as ADC voltage input (ADCVIN). ADCVIN is converted by the successive approximation register-based counters. The ADC resolution is 8 bits. When the conversion is completed, ADC puts the result in the ADC data register and sets a flag or generates an interrupt. Figure 12-1 shows a block diagram of the ADC.

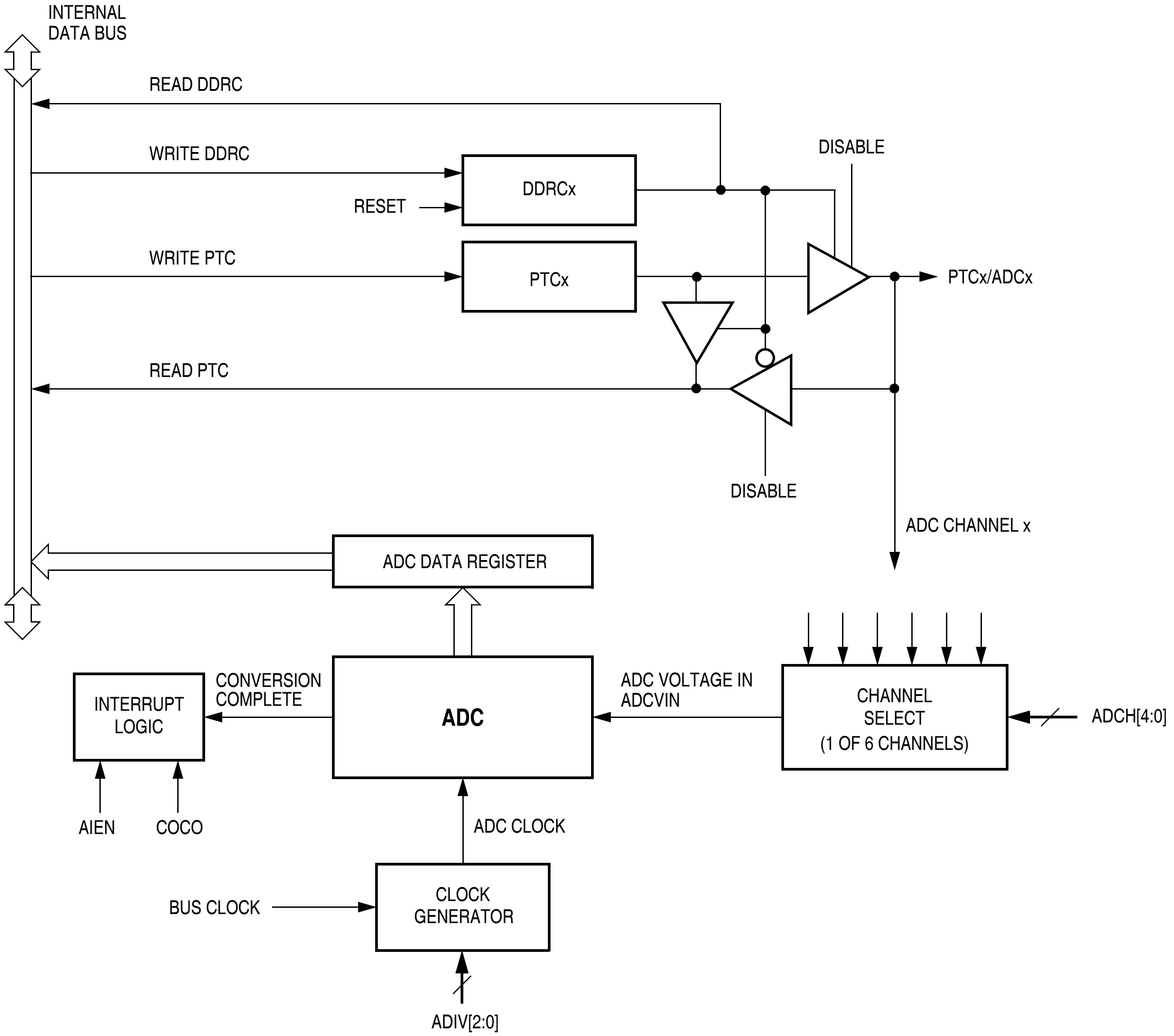


Figure 12-1. ADC Block Diagram

12.4.1 ADC Port I/O Pins

PTC5–PTC0 are general-purpose I/O pins that are shared with the ADC channels. The channel select bits (ADC status control register, $005D), define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC. The remaining ADC channels/port pins are controlled by the port I/O logic and can be used as general-purpose I/O. Writes to the port register

Table 13-2. USB I/O Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0029	USB Address Register (UADR)	Read:	USBEN	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002A	USB Interrupt Register (UINTR)	Read:	TBEF	RBFF	EOPIF	RSTIF	TBIE	RBIE	EOPIE	RSTIE
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	0
$002B	USB Control Register 0 (UCR0)	Read:	T0SEQ	STALL0	TX0E	RX0E	TP0SIZ3	TP0SIZ2	TP0SIZ1	TP0SIZ0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002C	USB Status Register (USR)	Read:	RSEQ	SETUP	TX1ST	0	RPSIZ3	RPSIZ2	RPSIZ1	RPSIZ0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	Indeterminate after reset							
$002D	USB Control Register 2 (UCR2)	Read:	0	0	PULLEN	SUSPND	ENABLE2	ENABLE1	STALL2	STALL1
		Write:	▨	▨						
		Reset:	0	0	0	0	0	0	0	0
$002E	USB Interrupt Register 1 (UIR1)	Read:	TXD1F	TXD1IE	RESUMF	0	0	0	0	0
		Write:	▨		▨	RESUMFR	TBEFR	RBFFR	TXD1FR	EOPFR
		Reset:	0	0	0	0	0	0	0	0
$002F	USB Control Register 1 (UCR1)	Read:	T1SEQ	ENDADD	TX1E	FRESUM	TP1SIZ3	TP1SIZ2	TP1SIZ1	TP1SIZ0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0030 to $0037	USB Endpoint 0 Data Register 0–7 (UD0R0–UD0R7)	Read:	UE0RDx7	UE0RDx6	UE0RDx5	UE0RDx4	UE0RDx3	UE0RDx2	UE0RDx1	UE0RDx0
		Write:	UE0TDx7	UE0TDx6	UE0TDx5	UE0TDx4	UE0TDx3	UE0TDx2	UE0TDx1	UE0TDx0
		Reset:	Indeterminate after reset							
$0038 to $003F	USB Endpoint 1/2 Data Register 0–7 (UD1R0–UD1R7)	Read:	▨	▨	▨	▨	▨	▨	▨	▨
		Write:	UE1TDx7	UE1TDx6	UE1TDx5	UE1TDx4	UE1TDx3	UE1TDx2	UE1TDx1	UE1TDx0
		Reset:	Indeterminate after reset							

13.5 Registers

There are seven control/status registers and 24 data buffers in the USB module. These registers are discussed in the following paragraphs.

RXBF — DDC Receive Buffer Full

This flag indicates the status of the data receive register (DDRR). When the CPU reads the data from the DDRR, the RXBF flag will be cleared. RXBF is set when DDRR is full by a transfer of data from the input circuit to the DDRR. Reset clears this bit.

1 = Data receive register full
0 = Data receive register empty

15.6.6 DDC Data Transmit Register (DDTR)

Address: $001A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DTD7	DTD6	DTD5	DTD4	DTD3	DTD2	DTD1	DTD0
Reset:	1	1	1	1	1	1	1	1

Figure 15-6. DDC Data Transmit Register (DDTR)

When the DDC module is enabled, DEN = 1, data written into this register depends on whether module is in master or slave mode.

In slave mode, the data in DDTR will be transferred to the output circuit when:

- the module detects a matched calling address (MATCH = 1), with the calling master requesting data (SRW = 1); or
- the previous data in the output circuit has be transmitted and the receiving master returns an acknowledge bit, indicated by a received acknowledge bit (RXAK = 0).

If the calling master does not return an acknowledge bit (RXAK = 1), the module will release the SDA line for master to generate a "stop" or "repeated start" condition. The data in the DDTR will not be transferred to the output circuit until the next calling from a master. The transmit buffer empty flag remains cleared (TXBE = 0).

In master mode, the data in DDTR will be transferred to the output circuit when:

When the DDRR is read by the CPU, the receive buffer full flag is cleared (RXBF = 0), and the next received data is loaded to the DDRR. Each time when new data is loaded to the DDRR, the RXIF interrupt flag is set, indicating that new data is available in DDRR.

The sequence of events for slave receive and master receive are illustrated in **Figure 15-8**.

15.7 Programming Considerations

When the DDC module detects an arbitration loss in master mode, it will release both SDA and SCL lines immediately. But if there are no further STOP conditions detected, the module will hang up. Therefore, it is recommended to have time-out software to recover from such ill condition. The software can start the time-out counter by looking at the BB (Bus Busy) flag in the DMCR and reset the counter on the completion of one byte transmission. If a time-out occur, software can clear the DEN bit (disable DDC module) to release the bus, and hence clearing the BB flag. This is the only way to clear the BB flag by software if the module hangs up due to a no STOP condition received. The DDC can resume operation again by setting the DEN bit.

Table 16-2. Sync Processor I/O Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0040	Sync Processor Control and Status Register (SPCSR)	Read:	VSIE	VEDGE	VSIF	COMP	VINVO	HINVO	VPOL	HPOL
		Write:			0				▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0041	Vertical Frequency High Register (VFHR)	Read:	VOF	0	0	VF12	VF11	VF10	VF9	VF8
		Write:	▨	CPW1	CPW0	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0042	Vertical Frequency Low Register (VFLR)	Read:	VF7	VF6	VF5	VF4	VF3	VF2	VF1	VF0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0043	Hsync Frequency High Register (HFHR)	Read:	HFH7	HFH6	HFH5	HFH4	HFH3	HFH2	HFH1	HFH0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0044	Hsync Frequency Low Register (HFLR)	Read:	HOVER	0	0	HFL4	HFL3	HFL2	HFL1	HFL0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0045	Sync Processor I/O Control Register (SPIOCR)	Read:	VSYNCS	HSYNCS	COINV	R	SOGSEL	CLAMPOE	BPOR	SOUT
		Write:	▨	▨						
		Reset:	0	0	0	0	0	0	0	0
$0046	Sync Processor Control Register 1 (SPCR1)	Read:	LVSIE	LVSIF	HPS1	HPS0	R	R	ATPOL	FSHF
		Write:		0						
		Reset:	0	0	0	0	0	0	0	0
$0047	H&V Sync Output Control Register (HVOCR)	Read:	R	0	0	0	0	HVOCR2	HVOCR1	HVOCR0
		Write:		▨	▨	▨	▨			
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

16.5.5 Low Vertical Frequency Detect

Logic monitors the value of the Vsync Frequency Register (VFR), and sets the low vertical frequency flag (LVSIF) when the value of VFR is higher than $C00 (frequency below 40.7Hz). LVSIF bit can generate an interrupt request to the CPU when the LVSIE bit is set and I-bit in the Condition Code Register is "0". The LVSIF bit can help the system to detect video off mode fast.

16.6 Registers

Eight registers are associated with the Sync Processor, they outlined in the following sections.

16.6.1 Sync Processor Control & Status Register (SPCSR)

Address: $0040

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	VSIE	VEDGE	VSIF	COMP	VINVO	HINVO	VPOL	HPOL
Write:			0				▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 16-3. Sync Processor Control & Status Register (SPCSR)

VSIE — VSync Interrupt Enable

When this bit is set, the VSIF flag is enabled to generate an interrupt request to the CPU. When VSIE is cleared, the VSIF flag is prevented from generating an interrupt request to the CPU. Reset clears this bit.

1 = VSIF bit set will generate interrupt request to CPU
0 = VSIF bit set does not generate interrupt request to CPU

19.8.1 Wait Mode

The COP remains active during wait mode. To prevent a COP reset during wait mode, periodically clear the COP counter in a CPU interrupt routine.

19.8.2 Stop Mode

Stop mode turns off the OSCXCLK input to the COP and clears the COP prescaler. Service the COP immediately before entering or after exiting stop mode to ensure a full COP timeout period after entering or exiting stop mode.

To prevent inadvertently turning off the COP with a STOP instruction, a configuration option is available that disables the STOP instruction. When the STOP bit in the configuration register has the STOP instruction is disabled, execution of a STOP instruction results in an illegal opcode reset.

19.9 COP Module During Break Mode

The COP is disabled during a break interrupt when V_{TST} is present on the $\overline{\text{RST}}$ pin.

20.4.1 Flag Protection During Break Interrupts

The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state.

20.4.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with $FFFC and $FFFD ($FEFC and $FEFD in monitor mode)

The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

20.4.3 TIM During Break Interrupts

A break interrupt stops the timer counters.

20.4.4 COP During Break Interrupts

The COP is disabled during a break interrupt when V_{TST} is present on the $\overline{\text{RST}}$ pin.

20.5 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

20.5.1 Wait Mode

If enabled, the break module is active in wait mode. In the break routine, the user can subtract one from the return address on the stack if SBSW is set (see **Section 7. System Integration Module (SIM)**). Clear the SBSW bit by writing logic 0 to it.