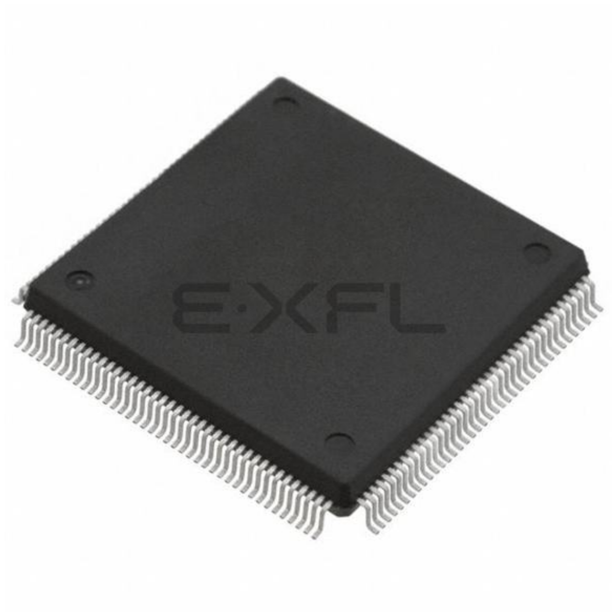
NXP USA Inc. - MC68336ACAB20 Datasheet



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	Obsolete
Core Processor	CPU32
Core Size	32-Bit Single-Core
Speed	20MHz
Connectivity	CANbus, EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	18
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	7.5K x 8
Voltage - Supply (Vcc/Vdd)	4.75V ~ 5.25V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	160-BQFP
Supplier Device Package	160-QFP (28x28)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68336acab20

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

LIST OF ILLUSTRATIONS
(Continued)

Figure **Title** **Page**

$\overline{\text{DSACK[1:0]}}$	—	Data and Size Acknowledge
DSCLK	—	Development Serial Clock
DSI	—	Development Serial Input
DSO	—	Development Serial Output
ECLK	—	MC6800 Devices and Peripherals Bus Clock
ETRIG[2:1]	—	QADC External Trigger
EXTAL	—	Crystal Oscillator Input
FC[2:0]	—	Function Codes
FREEZE	—	Freeze
$\overline{\text{HALT}}$	—	Halt
$\overline{\text{IFETCH}}$	—	Instruction Fetch
$\overline{\text{IPIPE}}$	—	Instruction Pipeline
$\overline{\text{IRQ[7:1]}}$	—	Interrupt Request
MA[2:0]	—	QADC Multiplexed Address
MISO	—	QSM Master In Slave Out
MODCLK	—	Clock Mode Select
MOSI	—	QSM Master Out Slave In
PCS[3:0]	—	QSM Peripheral Chip-Selects
PQA[7:0]	—	QADC Port A
PQB[7:0]	—	QADC Port B
PC[6:0]	—	SIM Port C
PE[7:0]	—	SIM Port E
PF[7:0]	—	SIM Port F
QUOT	—	Quotient Out
R/$\overline{\text{W}}$	—	Read/Write
$\overline{\text{RESET}}$	—	Reset
$\overline{\text{RMC}}$	—	Read-Modify-Write Cycle
RXD	—	SCI Receive Data
SCK	—	QSPI Serial Clock
SIZ[1:0]	—	Size
$\overline{\text{SS}}$	—	Slave Select
T2CLK	—	TPU Clock In
TPUCH[15:0]	—	TPU Channel Signals
TSC	—	Three-State Control
$\overline{\text{TSTME}}$	—	Test Mode Enable
V_{RH}	—	QADC High Reference Voltage
V_{RL}	—	QADC Low Reference Voltage
XFC	—	External Filter Capacitor
XTAL	—	Crystal Oscillator Output

4.2.4.2 Alternate Function Code Registers

Alternate function code registers (SFC and DFC) contain 3-bit function codes. Function codes can be considered extensions of the 24-bit linear address that optionally provide as many as eight 16-Mbyte address spaces. The processor automatically generates function codes to select address spaces for data and programs at the user and supervisor privilege levels and to select a CPU address space used for processor functions (such as breakpoint and interrupt acknowledge cycles).

Registers SFC and DFC are used by the MOVES instruction to specify explicitly the function codes of the memory address. The MOVEC instruction is used to transfer values to and from the alternate function code registers. This is a long-word transfer; the upper 29 bits are read as zeros and are ignored when written.

4.2.5 Vector Base Register (VBR)

The VBR contains the base address of the 1024-byte exception vector table, consisting of 256 exception vectors. Exception vectors contain the memory addresses of routines that begin execution at the completion of exception processing. More information on the VBR and exception processing can be found in **4.9 Exception Processing**.

4.3 Memory Organization

Memory is organized on a byte-addressable basis in which lower addresses correspond to higher order bytes. For example, the address N of a long-word data item corresponds to the address of the most significant byte of the highest order word. The address of the most significant byte of the low-order word is N + 2, and the address of the least significant byte of the long word is N + 3. The CPU32 requires long-word and word data and all instructions to be aligned on word boundaries. Refer to **Figure 4-6**. If this does not happen, an exception will occur when the CPU32 accesses the misaligned instruction or data. Data misalignment is not supported.

5.4 System Protection

The system protection block preserves reset status, monitors internal activity, and provides periodic interrupt generation. **Figure 5-6** is a block diagram of the submodule.

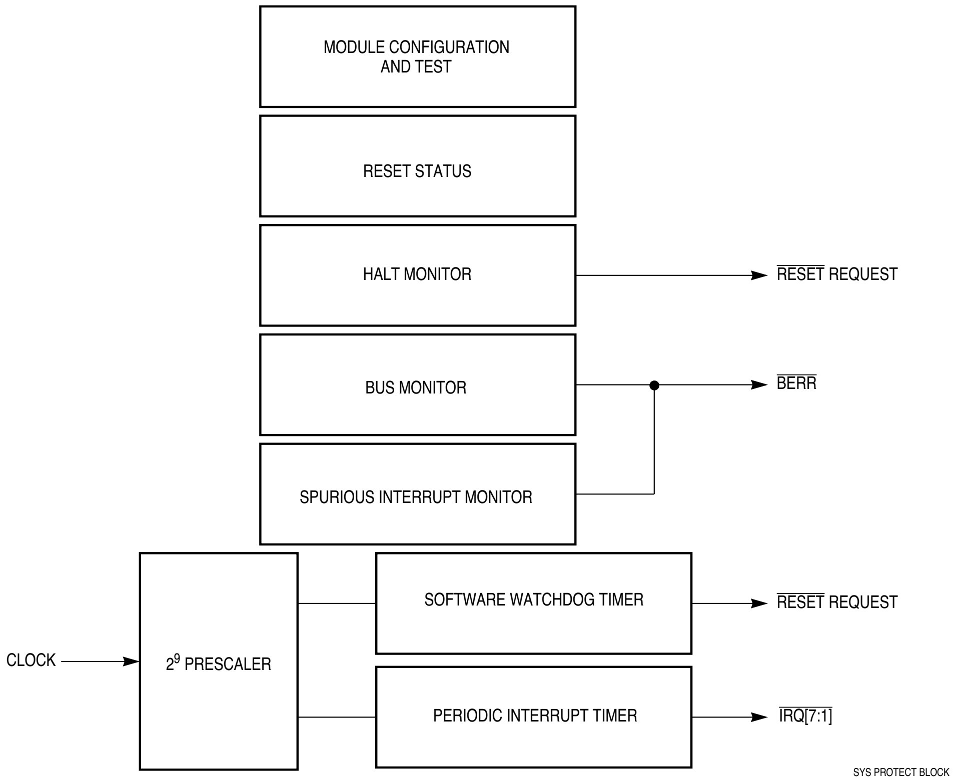


Figure 5-6 System Protection Block

5.4.1 Reset Status

The reset status register (RSR) latches internal MCU status during reset. Refer to **5.7.10 Reset Status Register** for more information.

5.4.2 Bus Monitor

The internal bus monitor checks data and size acknowledge ($\overline{\text{DSACK}}$) or autovector ($\overline{\text{AVEC}}$) signal response times during normal bus cycles. The monitor asserts the internal bus error ($\overline{\text{BERR}}$) signal when the response time is excessively long.

$\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ response times are measured in clock cycles. Maximum allowable response time can be selected by setting the bus monitor timing (BMT[1:0]) field in the system protection control register (SYPCR). **Table 5-4** shows the periods allowed.

Table 5-4 Bus Monitor Period

BMT[1:0]	Bus Monitor Time-Out Period
00	64 system clocks
01	32 system clocks
10	16 system clocks
11	8 system clocks

The monitor does not check $\overline{\text{DSACK}}$ response on the external bus unless the CPU32 initiates a bus cycle. The BME bit in SYPCR enables the internal bus monitor for internal to external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented and the internal-to-external bus monitor option must be disabled.

When monitoring transfers to an 8-bit port, the bus monitor does not reset until both byte accesses of a word transfer are completed. Monitor time-out period must be at least twice the number of clocks that a single byte access requires.

5.4.3 Halt Monitor

The halt monitor responds to an assertion of the $\overline{\text{HALT}}$ signal on the internal bus, caused by a double bus fault. A flag in the reset status register (RSR) indicates that the last reset was caused by the halt monitor. Halt monitor reset can be inhibited by the halt monitor (HME) enable bit in SYPCR. Refer to **5.6.5.2 Double Bus Faults** for more information.

5.4.4 Spurious Interrupt Monitor

During interrupt exception processing, the CPU32 normally acknowledges an interrupt request, recognizes the highest priority source, and then either acquires a vector or responds to a request for autovectoring. The spurious interrupt monitor asserts the internal bus error signal ($\overline{\text{BERR}}$) if no interrupt arbitration occurs during interrupt exception processing. The assertion of $\overline{\text{BERR}}$ causes the CPU32 to load the spurious interrupt exception vector into the program counter. The spurious interrupt monitor cannot be disabled. Refer to **5.8 Interrupts** for more information. For detailed information about interrupt exception processing, refer to **4.9 Exception Processing**.

5.4.5 Software Watchdog

The software watchdog is controlled by the software watchdog enable (SWE) bit in SYPCR. When enabled, the watchdog requires that a service sequence be written to the software service register (SWSR) on a periodic basis. If servicing does not take place, the watchdog times out and asserts the $\overline{\text{RESET}}$ signal.

Each time the service sequence is written, the software watchdog timer restarts. The sequence to restart consists of the following steps:

1. Write $55 to SWSR.
2. Write $AA to SWSR.

5.4.8 Low-Power STOP Mode Operation

When the CPU32 executes the LPSTOP instruction, the current interrupt priority mask is stored in the clock control logic, internal clocks are disabled according to the state of the STSIM bit in the SYNCR, and the MCU enters low-power stop mode. The bus monitor, halt monitor, and spurious interrupt monitor are all inactive during low-power stop mode.

During low-power stop mode, the clock input to the software watchdog timer is disabled and the timer stops. The software watchdog begins to run again on the first rising clock edge after low-power stop mode ends. The watchdog is not reset by low-power stop mode. A service sequence must be performed to reset the timer.

The periodic interrupt timer does not respond to the LPSTOP instruction, but continues to run during LPSTOP. To stop the periodic interrupt timer, PITR must be loaded with a zero value before the LPSTOP instruction is executed. A PIT interrupt, or an external interrupt request, can bring the MCU out of low-power stop mode if it has a higher priority than the interrupt mask value stored in the clock control logic when low-power stop mode is initiated. LPSTOP can be terminated by a reset.

5.5 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. **Figure 5-8** shows a basic system with external memory and peripherals.

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For any bus access, it is assumed that the port is 16 bits wide when the bus cycle begins.

Operand bytes are designated as shown in **Figure 5-9**. OP[0:3] represent the order of access. For instance, OP0 is the most significant byte of a long-word operand, and is accessed first, while OP3, the least significant byte, is accessed last. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

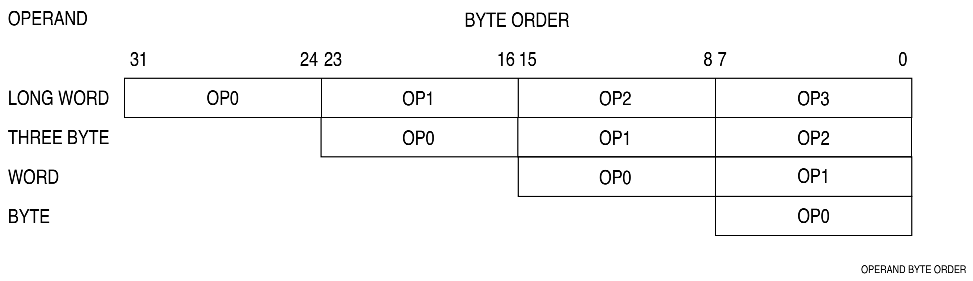


Figure 5-9 Operand Byte Order

5.5.3 Operand Alignment

The EBI data multiplexer establishes the necessary connections for different combinations of address and data sizes. The multiplexer takes the two bytes of the 16-bit bus and routes them to their required positions. Positioning of bytes is determined by the size and address outputs. SIZ1 and SIZ0 indicate the number of bytes remaining to be transferred during the current bus cycle. The number of bytes transferred is equal to or less than the size indicated by SIZ1 and SIZ0, depending on port width.

ADDR0 also affects the operation of the data multiplexer. During an operand transfer, ADDR[23:1] indicate the word base address of the portion of the operand to be accessed. ADDR0 indicates the byte offset from the base.

5.5.4 Misaligned Operands

The CPU32 uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. This is determined by the value of ADDR0. When ADDR0 = 0 (an even address), the address is on a word and byte boundary. When ADDR0 = 1 (an odd address), the address is on a byte boundary only. A byte operand is aligned at any address; a word or long-word operand is misaligned at an odd address. The CPU32 does not support misaligned transfers.

The largest amount of data that can be transferred by a single bus cycle is an aligned word. If the MCU transfers a long-word operand through a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word is transferred on a following bus cycle.

5.6.2.2 Write Cycle

During a write cycle, the MCU transfers data to an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to write two bytes at once. For a byte operation, the MCU writes one byte. The portion of the data bus upon which each byte is written depends on operand size, peripheral address, and peripheral port size.

Refer to **5.5.2 Dynamic Bus Sizing** and **5.5.4 Misaligned Operands** for more information. **Figure 5-11** is a flowchart of a write-cycle operation for a word transfer. Refer to the *SIM Reference Manual* (SIMRM/AD) for more information.

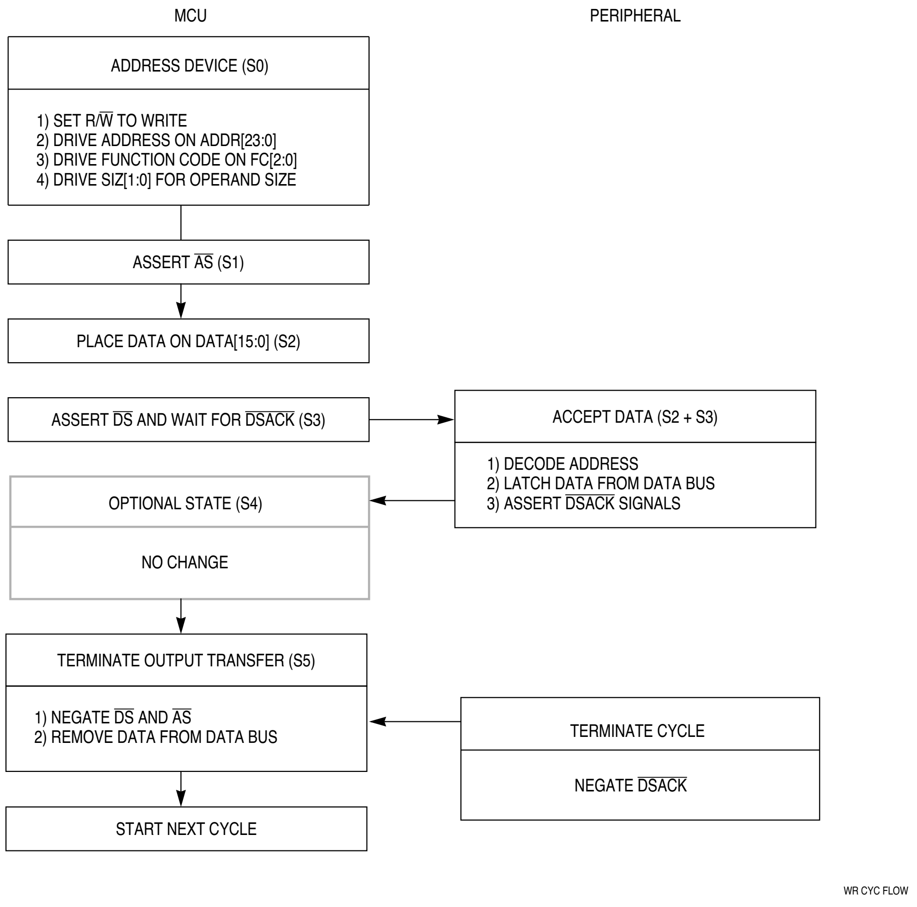

MCU
PERIPHERAL
ADDRESS DEVICE (S0)
1) SET R/$\overline{W}$ TO WRITE
2) DRIVE ADDRESS ON ADDR[23:0]
3) DRIVE FUNCTION CODE ON FC[2:0]
4) DRIVE SIZ[1:0] FOR OPERAND SIZE
ASSERT $\overline{AS}$ (S1)
PLACE DATA ON DATA[15:0] (S2)
ASSERT $\overline{DS}$ AND WAIT FOR $\overline{DSACK}$ (S3)
ACCEPT DATA (S2 + S3)
1) DECODE ADDRESS
2) LATCH DATA FROM DATA BUS
3) ASSERT $\overline{DSACK}$ SIGNALS
OPTIONAL STATE (S4)
NO CHANGE
TERMINATE OUTPUT TRANSFER (S5)
1) NEGATE $\overline{DS}$ AND $\overline{AS}$
2) REMOVE DATA FROM DATA BUS
TERMINATE CYCLE
NEGATE $\overline{DSACK}$
START NEXT CYCLE
WR CYC FLOW

Figure 5-11 Write Cycle Flowchart

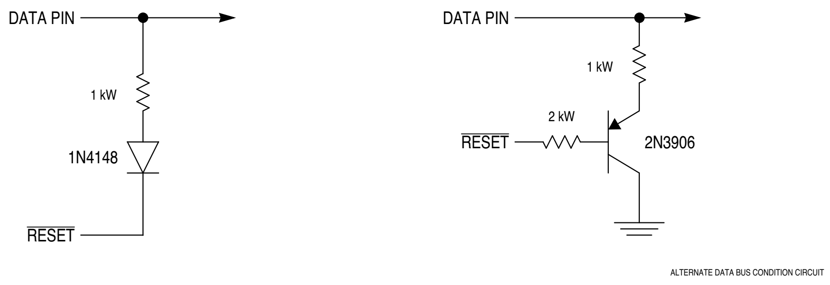


Figure 5-17 Alternate Circuit for Data Bus Mode Select Conditioning

Data bus mode select current is specified in **Table A-5**. Do not confuse pin function with pin electrical state. Refer to **5.7.5 Pin States During Reset** for more information.

Unlike other chip-select signals, the boot ROM chip-select ($\overline{\text{CSBOOT}}$) is active at the release of $\overline{\text{RESET}}$. During reset exception processing, the MCU fetches initialization vectors beginning at address $000000 in supervisor program space. An external memory device containing vectors located at these addresses can be enabled by $\overline{\text{CSBOOT}}$ after a reset.

The logic level of DATA0 during reset selects boot ROM port size for dynamic bus allocation. When DATA0 is held low, port size is eight bits; when DATA0 is held high, either by the weak internal pull-up driver or by an external pull-up, port size is 16 bits. Refer to **5.9.4 Chip-Select Reset Operation** for more information.

DATA1 and DATA2 determine the functions of $\overline{\text{CS[2:0]}}$ and $\overline{\text{CS[5:3]}}$, respectively. DATA[7:3] determine the functions of an associated chip-select and all lower-numbered chip-selects down through $\overline{\text{CS6}}$. For example, if DATA5 is pulled low during reset, $\overline{\text{CS[8:6]}}$ are assigned alternate function as ADDR[21:19], and $\overline{\text{CS[10:9]}}$ remain chip-selects. Refer to **5.9.4 Chip-Select Reset Operation** for more information.

DATA8 determines the function of the $\overline{\text{DSACK[1:0]}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, and SIZE pins. If DATA8 is held low during reset, these pins are assigned to I/O port E.

DATA9 determines the function of interrupt request pins $\overline{\text{IRQ[7:1]}}$ and the clock mode select pin (MODCLK). When DATA9 is held low during reset, these pins are assigned to I/O port F.

5.7.3.2 Clock Mode Selection

The state of the clock mode (MODCLK) pin during reset determines what clock source the MCU uses. When MODCLK is held high during reset, the clock signal is generated from a reference frequency using the clock synthesizer. When MODCLK is held low during reset, the clock synthesizer is disabled, and an external system clock signal must be applied. Refer to **5.3 System Clock** for more information.

Table 5-16 Module Pin Functions During Reset

Module	Pin Mnemonic	Function
CPU32	DSI/$\overline{\text{IFETCH}}$	DSI/$\overline{\text{IFETCH}}$
	DSO/$\overline{\text{IPIPE}}$	DSO/$\overline{\text{IPIPE}}$
	$\overline{\text{BKPT}}$/DSCLK	$\overline{\text{BKPT}}$/DSCLK
CTM4	CPWM[8:5]	Discrete output
	CTD[10:9]/[4:3]	Discrete input
	CTM4C	Discrete input
QADC	PQA[7:5]/AN[59:57]	Discrete input
	PQA[4:3]/AN[56:55]/ETRIG[2:1]	Discrete input
	PQA[2:0]/AN[54:52]/MA[2:0]	Discrete input
	PQB[7:4]/AN[51:48]	Discrete input
	PQB[3:0]/AN[z, y, x, w]/AN[3:0]	Discrete input
QSM	PQS0/MISO	Discrete input
	PQS1/MOSI	Discrete input
	PQS2/SCK	Discrete input
	PQS3/PCS0/$\overline{\text{SS}}$	Discrete input
	PQS[6:4]/PCS[3:1]	RXD
	PQS7/TXD	Discrete input
	RXD	Discrete input
TouCAN (MC68376 only)	CANRX0	TouCAN receive
	CANTX0	TouCAN transmit
TPU	TPUCH[15:0]	TPU input
	T2CLK	TCR2 clock

5.7.5 Pin States During Reset

It is important to keep the distinction between pin function and pin electrical state clear. Although control register values and mode select inputs determine pin function, a pin driver can be active, inactive or in high-impedance state while reset occurs. During power-on reset, pin state is subject to the constraints discussed in **5.7.7 Power-On Reset**.

NOTE

Pins that are not used should either be configured as outputs, or (if configured as inputs) pulled to the appropriate inactive state. This decreases additional I_{DD} caused by digital inputs floating near mid-supply level.

5.7.5.1 Reset States of SIM Pins

Generally, while $\overline{\text{RESET}}$ is asserted, SIM pins either go to an inactive high-impedance state or are driven to their inactive states. After $\overline{\text{RESET}}$ is released, mode selection occurs and reset exception processing begins. Pins configured as inputs must be driven to the desired active state. Pull-up or pull-down circuitry may be necessary. Pins configured as outputs begin to function after $\overline{\text{RESET}}$ is released. **Table 5-17** is a summary of SIM pin states during reset.

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion. Writing a value to DTL[7:0] in SPCR1 specifies a delay period. The DT bit in each command RAM byte determines whether the standard delay period (DT = 0) or the specified delay period (DT = 1) is used. The following expression is used to calculate the delay:

$$\text{Delay after Transfer} = \frac{32 \times \text{DTL[7:0]}}{\text{System Clock}}$$

where DTL equals {1, 2, 3,..., 255}.

A zero value for DTL[7:0] causes a delay-after-transfer value of 8192/System Clock.

$$\text{Standard Delay after Transfer} = \frac{17}{\text{System Clock}}$$

Adequate delay between transfers must be specified for long data streams because the QSPI requires time to load a transmit RAM entry for transfer. Receiving devices need at least the standard delay between successive transfers. If the system clock is operating at a slower rate, the delay between transfers must be increased proportionately.

Operation is initiated by setting the SPE bit in SPCR1. Shortly after SPE is set, the QSPI executes the command at the command RAM address pointed to by NEWQP. Data at the pointer address in transmit RAM is loaded into the data serializer and transmitted. Data that is simultaneously received is stored at the pointer address in receive RAM.

When the proper number of bits have been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads the next data for transfer from transmit RAM. The command pointed to by the incremented working queue pointer is executed next, unless a new value has been written to NEWQP. If a new queue pointer value is written while a transfer is in progress, that transfer is completed normally.

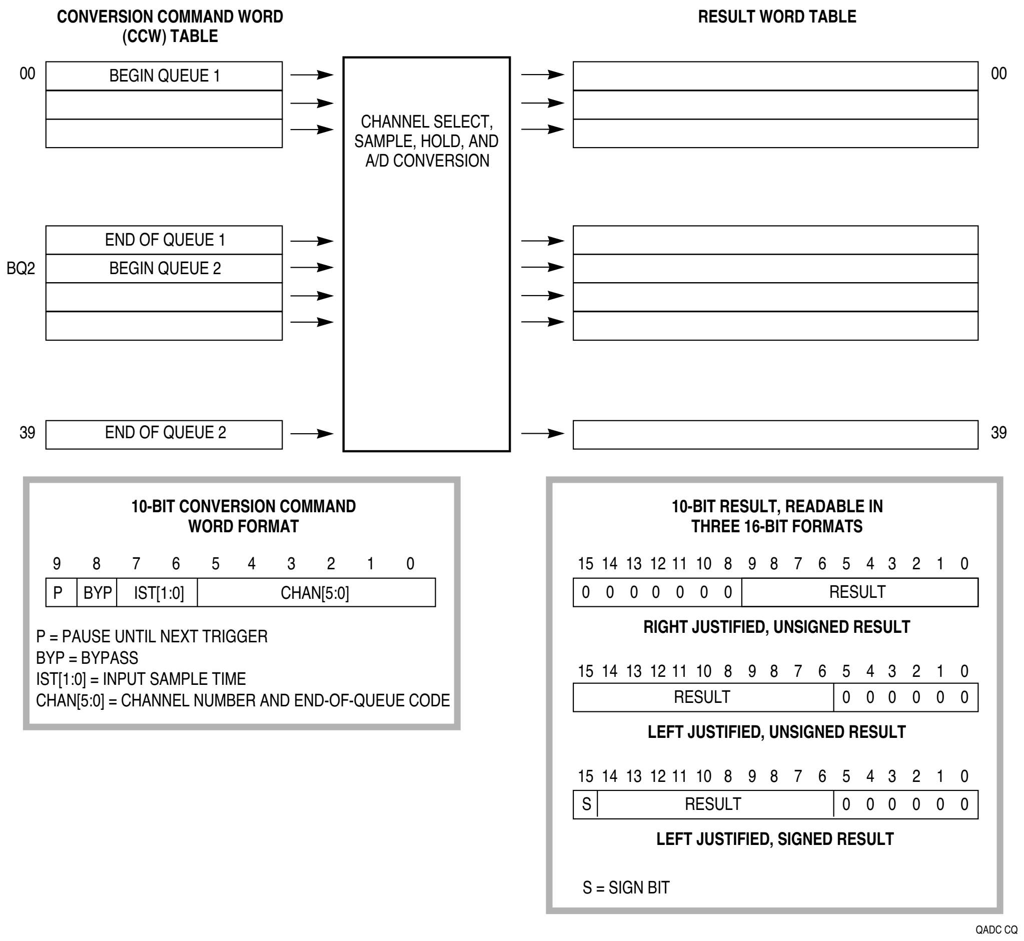
When the CONT bit in a command RAM byte is set, PCS pins are continuously driven in specified states during and between transfers. If the chip-select pattern changes during or between transfers, the original pattern is driven until execution of the following transfer begins. When CONT is cleared, the data in register PORTQS is driven between transfers. The data in PORTQS must match the inactive states of SCK and any peripheral chip-selects used.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wrap-around mode is enabled.

The ten implemented bits of the CCW word can be read and written. Unimplemented bits are read as zeros, and write operations have no effect. Each location in the CCW table corresponds to a location in the result word table. When a conversion is completed for a CCW entry, the 10-bit result is written in the corresponding result word entry.

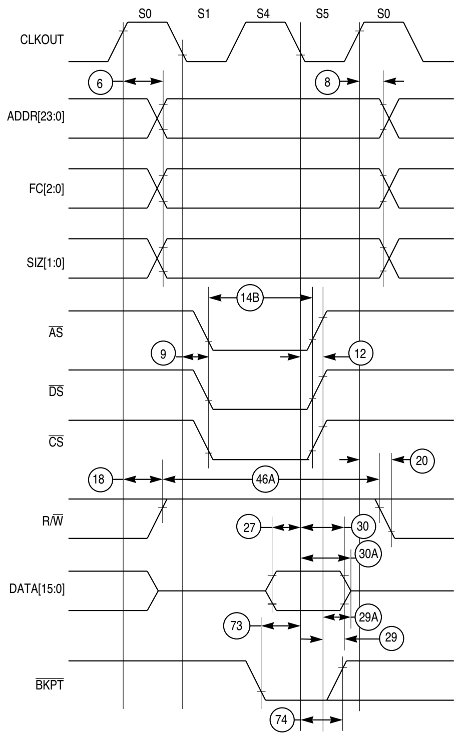
The beginning of queue 1 is the first location in the CCW table. The first location of queue 2 is specified by the beginning of queue 2 pointer BQ2 in QACR2. To dedicate the entire CCW table to queue 1, queue 2 is disabled, and BQ2 is programmed to any value greater than 39. To dedicate the entire CCW table to queue 2, queue 1 is disabled, and BQ2 is specified as the first location in the CCW table.

Figure 8-10 illustrates the operation of the queue structure.



CONVERSION COMMAND WORD (CCW) TABLE
RESULT WORD TABLE
00
BEGIN QUEUE 1
CHANNEL SELECT, SAMPLE, HOLD, AND A/D CONVERSION
END OF QUEUE 1
BQ2
BEGIN QUEUE 2
39
END OF QUEUE 2
10-BIT CONVERSION COMMAND WORD FORMAT
9 8 7 6 5 4 3 2 1 0
P BYP IST[1:0] CHAN[5:0]
P = PAUSE UNTIL NEXT TRIGGER
BYP = BYPASS
IST[1:0] = INPUT SAMPLE TIME
CHAN[5:0] = CHANNEL NUMBER AND END-OF-QUEUE CODE
10-BIT RESULT, READABLE IN THREE 16-BIT FORMATS
15 14 13 12 11 10 8 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 RESULT
RIGHT JUSTIFIED, UNSIGNED RESULT
RESULT 0 0 0 0 0 0
LEFT JUSTIFIED, UNSIGNED RESULT
S RESULT 0 0 0 0 0 0
LEFT JUSTIFIED, SIGNED RESULT
S = SIGN BIT
QADC CQ

Figure 8-10 QADC Conversion Queue Operation



68300 FAST RD CYC TIM

Figure A-6 Fast Termination Read Cycle Timing Diagram

FRZ — FREEZE Assertion Response

The FRZ bit determines whether or not the QADC responds to assertion of the IMB FREEZE signal.

0 = QADC ignores the IMB FREEZE signal.
1 = QADC finishes any current conversion, then freezes.

SUPV — Supervisor/Unrestricted Data Space

The SUPV bit designates the assignable space as supervisor or unrestricted.

0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted.
1 = All QADC registers and tables are designated as supervisor-only data space.

IARB[3:0] — Interrupt Arbitration ID

The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value.

D.5.2 QADC Test Register

QADCTEST — QADC Test Register **$YFF202**

Used for factory test only.

D.5.3 QADC Interrupt Register

QADCINT — QADC Interrupt Register **$YFF204**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	IRLQ1[2:0]			RSVD	IRLQ2[2:0]			IVB[7:2]						IVB[1:0][1]	
RESET:															
	0	0	0		0	0	0	0	0	0	0	1	1	1	1

NOTES:
1. Bits 1 and 0 are supplied by the QADC.

IRLQ1[2:0] — Queue 1 Interrupt Level

When queue 1 generates an interrupt request, IRLQ1[2:0] determines which of the interrupt request signals is asserted. When a request is acknowledged, the QADC compares IRLQ1[2:0] to a mask value supplied by the CPU32 to determine whether to respond. IRLQ1[2:0] must have a value in the range of $0 (interrupts disabled) to $7 (highest priority).

IRLQ2[2:0] — Queue 2 Interrupt Level

When queue 2 generates an interrupt request, IRLQ2[2:0] determines which of the interrupt request signals is asserted. When a request is acknowledged, the QADC compares IRLQ2[2:0] to a mask value supplied by the CPU32 to determine whether to respond. IRLQ2[2:0] must have a value in the range of $0 (interrupts disabled) to $7 (highest priority).

D.5.6 QADC Control Registers

QACR0 — QADC Control Register 0 **$YFF20A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX	RESERVED						PSH[4:0]					PSA	PSL[2:0]		
RESET:															
0							0	0	0	1	1	0	0	1	1

MUX — Externally Multiplexed Mode

The MUX bit configures the QADC for externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA[2:0] pins to be outputs.

0 = Internally multiplexed, 16 possible channels.

1 = Externally multiplexed, 44 possible channels.

PSH[4:0] — Prescaler Clock High Time

The PSH field selects the QCLK high time in the prescaler. To keep QCLK within the specified range, PSH[4:0] must be programmed to guarantee the minimum acceptable time for parameter t_{PSH} (refer to **Table A-13** for more information). The following equation relates t_{PSH} to PSH[4:0]:

$$t_{PSH} = \frac{PSH[4:0] + 1}{f_{sys}}$$

PSA — Prescaler Add a Tick

The PSA bit modifies the QCLK duty cycle by adding one system clock tick to the high time and subtracting one system clock tick from the low time.

0 = QCLK high and low times are not modified.

1 = Add one system clock tick to the high time of QCLK and subtract one system clock tick from the low time.

PSL[2:0] — Prescaler Clock Low Time

The PSL field selects the QCLK low time in the prescaler. To keep QCLK within the specified range, PSL[2:0] must be programmed to guarantee the minimum acceptable time for parameter t_{PSL} (refer to **Table A-13** for more information). The following equation relates t_{PSL} to PSL[2:0]:

$$t_{PSL} = \frac{PSL[2:0] + 1}{f_{sys}}$$

TCR1P[1:0] — Timer Count Register 1 Prescaler Control

TCR1 is clocked from the output of a prescaler. The prescaler's input is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK bit. The prescaler divides this input by 1, 2, 4, or 8. Channels using TCR1 have the capability to resolve down to the TPU system clock divided by four. **Table D-52** is a summary of prescaler output.

Table D-52 TCR1 Prescaler Control Bits

TCR1P[1:0]	Prescaler Divide By	TCR1 Clock Input	
		PSCK = 0	PSCK = 1
00	1	$f_{sys} \div 32$	$f_{sys} \div 4$
01	2	$f_{sys} \div 64$	$f_{sys} \div 8$
10	4	$f_{sys} \div 128$	$f_{sys} \div 16$
11	8	$f_{sys} \div 256$	$f_{sys} \div 32$

TCR2P[1:0] — Timer Count Register 2 Prescaler Control

TCR2 is clocked from the output of a prescaler. If T2CG = 0, the input to the TCR2 prescaler is the external TCR2 clock source. If T2CG = 1, the input is the TPU system clock divided by eight. The TCR2P field specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to resolve down to the TPU system clock divided by eight. **Table D-53** is a summary of prescaler output.

Table D-53 TCR2 Prescaler Control Bits

TCR2P[1:0]	Prescaler Divide By	Internal Clock Divided By	External Clock Divided By
00	1	8	1
01	2	16	2
10	4	32	4
11	8	64	8

EMU — Emulation Control

In emulation mode, the TPU executes microinstructions from TPURAM exclusively. Access to the TPURAM module via the IMB is blocked, and the TPURAM module is dedicated for use by the TPU. After reset, this bit can be written only once.

0 = TPU and TPURAM operate normally.
1 = TPU and TPURAM operate in emulation mode.

T2CG — TCR2 Clock/Gate Control

When T2CG is set, the external TCR2 pin functions as a gate of the DIV8 clock (the TPU system clock divided by eight). In this case, when the external TCR2 pin is low, the DIV8 clock is blocked, preventing it from incrementing TCR2. When the external TCR2 pin is high, TCR2 is incremented at the frequency of the DIV8 clock. When T2CG is cleared, an external clock input from the TCR2 pin, which has been synchronized and fed through a digital filter, increments TCR2.

0 = TCR2 pin used as clock source for TCR2.
1 = TCR2 pin used as gate of DIV8 clock for TCR2.

CH[15:0] — Encoded Channel Priority Levels

Table D-56 shows channel priority levels.

Table D-56 Channel Priorities

CHx[1:0]	Service	Guaranteed Time Slots
00	Disabled	—
01	Low	1 out of 7
10	Middle	2 out of 7
11	High	4 out of 7

D.8.11 Channel Interrupt Status Register

CISR — Channel Interrupt Status Register **$YFFE20**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CH[15:0] — Channel Interrupt Status

0 = Channel interrupt not asserted.

1 = Channel interrupt asserted.

D.8.12 Link Register

LR — Link Register **$YFFE22**

Used for factory test only.

D.8.13 Service Grant Latch Register

SGLR — Service Grant Latch Register **$YFFE24**

Used for factory test only.

D.8.14 Decoded Channel Number Register

DCNR — Decoded Channel Number Register **$YFFE26**

Used for factory test only.

D.8.15 TPU Parameter RAM

The channel parameter registers are organized as one hundred 16-bit words of RAM. Channels 0 to 13 have six parameters. Channels 14 and 15 each have eight parameters. The parameter registers constitute a shared work space for communication between the CPU32 and the TPU. Refer to **Table D-57**.