NXP USA Inc. - MC68332GVEH20 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	EBI/EMI, SCI, SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	15
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	132-BQFP Bumpered
Supplier Device Package	132-PQFP (24.13x24.13)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68332gveh20

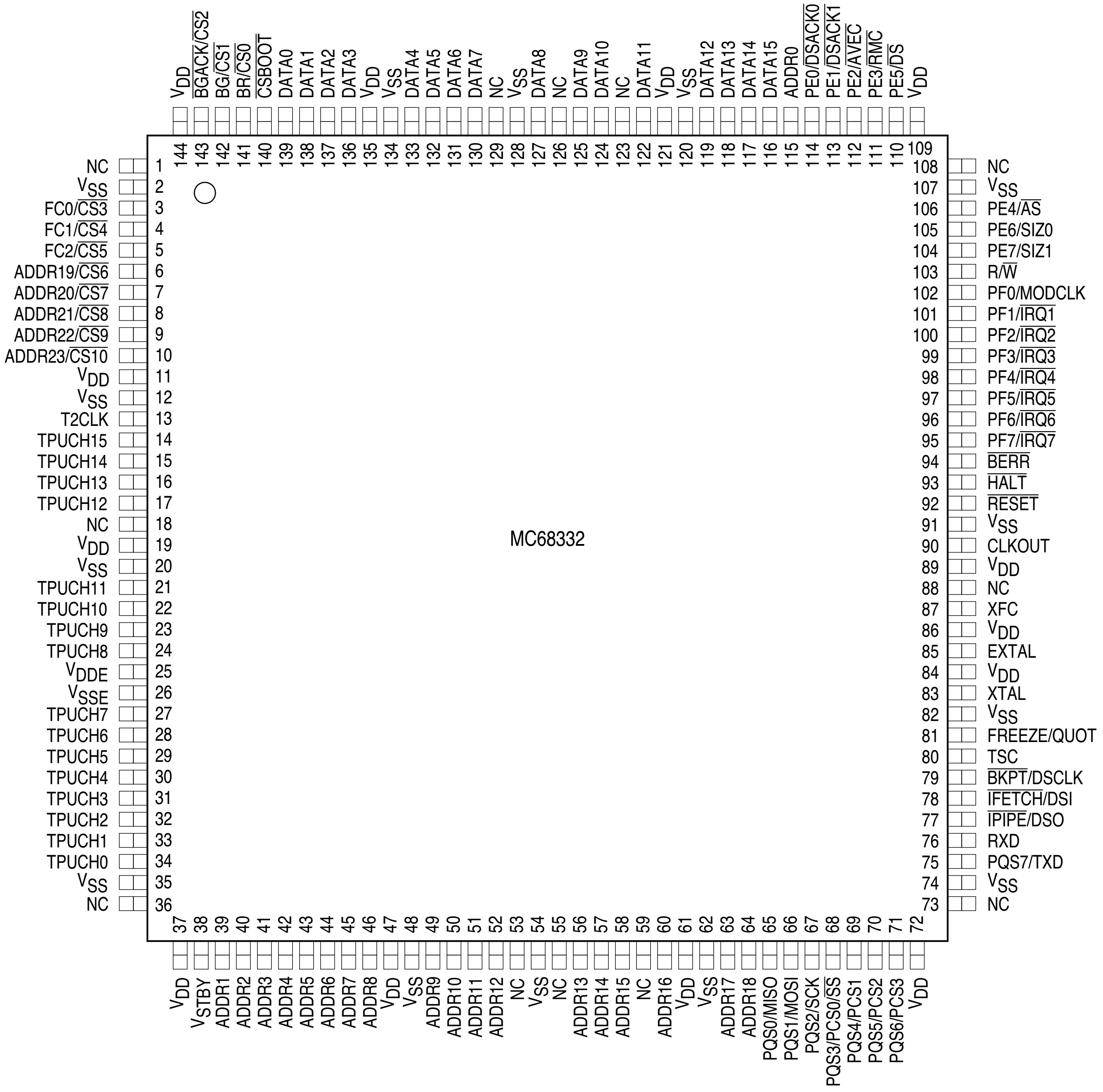


Figure 3 MC68332 144-Pin QFP Pin Assignments

2 Signal Descriptions

2.1 Pin Characteristics

The following table shows MCU pins and their characteristics. All inputs detect CMOS logic levels. All inputs can be put in a high-impedance state, but the method of doing this differs depending upon pin function. Refer to the table, MCU Driver Types, for a description of output drivers. An entry in the discrete I/O column of the MCU Pin Characteristics table indicates that a pin has an alternate I/O function. The port designation is given when it applies. Refer to the MCU Block Diagram for information about port organization.

Table 2 MCU Pin Characteristic

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
ADDR23/$\overline{\text{CS10}}$/ECLK	A	Y	N	O	—
ADDR[22:19]/$\overline{\text{CS[9:6]}}$	A	Y	N	O	PC[6:3]
ADDR[18:0]	A	Y	N	—	—
$\overline{\text{AS}}$	B	Y	N	I/O	PE5
$\overline{\text{AVEC}}$	B	Y	N	I/O	PE2
$\overline{\text{BERR}}$	B	Y	N	—	—
$\overline{\text{BG}}$/$\overline{\text{CS1}}$	B	—	—	—	—
$\overline{\text{BGACK}}$/$\overline{\text{CS2}}$	B	Y	N	—	—
$\overline{\text{BKPT}}$/DSCLK	—	Y	Y	—	—
$\overline{\text{BR}}$/$\overline{\text{CS0}}$	B	Y	N	—	—
CLKOUT	A	—	—	—	—
$\overline{\text{CSBOOT}}$	B	—	—	—	—
DATA[15:0][1]	Aw	Y	N	—	—
$\overline{\text{DS}}$	B	Y	N	I/O	PE4
$\overline{\text{DSACK1}}$	B	Y	N	I/O	PE1
$\overline{\text{DSACK0}}$	B	Y	N	I/O	PE0
DSI/$\overline{\text{IFETCH}}$	A	Y	Y	—	—
DSO/$\overline{\text{IPIPE}}$	A	—	—	—	—
EXTAL[2]	—	—	Special	—	—
FC[2:0]/$\overline{\text{CS[5:3]}}$	A	Y	N	O	PC[2:0]
FREEZE/QUOT	A	—	—	—	—
$\overline{\text{HALT}}$	Bo	Y	N	—	—
$\overline{\text{IRQ}}$[7:1]	B	Y	Y	I/O	PF[7:1]
MISO	Bo	Y	Y	I/O	PQS0
MODCLK[1]	B	Y	N	I/O	PF0
MOSI	Bo	Y	Y	I/O	PQS1
PCS0/$\overline{\text{SS}}$	Bo	Y	Y	I/O	PQS3
PCS[3:1]	Bo	Y	Y	I/O	PQS[6:4]
R/$\overline{\text{W}}$	A	Y	N	—	—
$\overline{\text{RESET}}$	Bo	Y	Y	—	—
$\overline{\text{RMC}}$	B	Y	N	I/O	PE3
RXD	—	N	N	—	—
SCK	Bo	Y	Y	I/O	PQS2
SIZ[1:0]	B	Y	N	I/O	PE[7:6]

Table 5 MCU Signal Characteristics (Continued)

Signal Name	MCU Module	Signal Type	Active State
TSC	SIM	Input	—
TXD	QSM	Output	—
XFC	SIM	Input	—
XTAL	SIM	Output	—

2.5 Signal Function

Table 6 MCU Signal Function

Signal Name	Mnemonic	Function
Address Bus	ADDR[23:0]	24-bit address bus
Address Strobe	$\overline{\text{AS}}$	Indicates that a valid address is on the address bus
Autovector	$\overline{\text{AVEC}}$	Requests an automatic vector during interrupt acknowledge
Bus Error	$\overline{\text{BERR}}$	Indicates that a bus error has occurred
Bus Grant	$\overline{\text{BG}}$	Indicates that the MCU has relinquished the bus
Bus Grant Acknowledge	$\overline{\text{BGACK}}$	Indicates that an external device has assumed bus mastership
Breakpoint	$\overline{\text{BKPT}}$	Signals a hardware breakpoint to the CPU
Bus Request	$\overline{\text{BR}}$	Indicates that an external device requires bus mastership
System Clockout	CLKOUT	System clock output
Chip Selects	$\overline{\text{CS[10:0]}}$	Select external devices at programmed addresses
Boot Chip Select	$\overline{\text{CSBOOT}}$	Chip select for external boot start-up ROM
Data Bus	DATA[15:0]	16-bit data bus
Data Strobe	$\overline{\text{DS}}$	During a read cycle, indicates when it is possible for an external device to place data on the data bus. During a write cycle, indicates that valid data is on the data bus.
Data and Size Acknowledge	$\overline{\text{DSACK[1:0]}}$	Provide asynchronous data transfers and dynamic bus sizing
Development Serial In, Out, Clock	DSI, DSO, DSCLK	Serial I/O and clock for background debugging mode
Crystal Oscillator	EXTAL, XTAL	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used
Function Codes	FC[2:0]	Identify processor state and current address space
Freeze	FREEZE	Indicates that the CPU has entered background mode
Halt	$\overline{\text{HALT}}$	Suspend external bus activity
Instruction Pipeline	$\overline{\text{IFETCH}}$ $\overline{\text{IPIPE}}$	Indicate instruction pipeline activity
Interrupt Request Level	$\overline{\text{IRQ[7:1]}}$	Provides an interrupt priority level to the CPU
Master In Slave Out	MISO	Serial input to QSPI in master mode; serial output from QSPI in slave mode
Clock Mode Select	MODCLK	Selects the source and type of system clock
Master Out Slave In	MOSI	Serial output from QSPI in master mode; serial input to QSPI in slave mode
Port C	PC[6:0]	SIM digital output port signals
Peripheral Chip Select	PCS[3:0]	QSPI peripheral chip selects
Port E	PE[7:0]	SIM digital I/O port signals
Port F	PF[7:0]	SIM digital I/O port signals
Port QS	PQS[7:0]	QSM digital I/O port signals

Table 7 SIM Address Map (Continued)

Access	Address	15 8	7 0
S	$YFFA56	CHIP-SELECT OPTION 2 (CSOR2)	
S	$YFFA58	CHIP-SELECT BASE 3 (CSBAR3)	
S	$YFFA5A	CHIP-SELECT OPTION 3 (CSOR3)	
S	$YFFA5C	CHIP-SELECT BASE 4 (CSBAR4)	
S	$YFFA5E	CHIP-SELECT OPTION 4 (CSOR4)	
S	$YFFA60	CHIP-SELECT BASE 5 (CSBAR5)	
S	$YFFA62	CHIP-SELECT OPTION 5 (CSOR5)	
S	$YFFA64	CHIP-SELECT BASE 6 (CSBAR6)	
S	$YFFA66	CHIP-SELECT OPTION 6 (CSOR6)	
S	$YFFA68	CHIP-SELECT BASE 7 (CSBAR7)	
S	$YFFA6A	CHIP-SELECT OPTION 7 (CSOR7)	
S	$YFFA6C	CHIP-SELECT BASE 8 (CSBAR8)	
S	$YFFA6E	CHIP-SELECT OPTION 8 (CSOR8)	
S	$YFFA70	CHIP-SELECT BASE 9 (CSBAR9)	
S	$YFFA72	CHIP-SELECT OPTION 9 (CSOR9)	
S	$YFFA74	CHIP-SELECT BASE 10 (CSBAR10)	
S	$YFFA76	CHIP-SELECT OPTION 10 (CSOR10)	
	$YFFA78	NOT USED	NOT USED
	$YFFA7A	NOT USED	NOT USED
	$YFFA7C	NOT USED	NOT USED
	$YFFA7E	NOT USED	NOT USED

Y = M111, where M is the logic state of the module mapping (MM) bit in the SIMCR.

3.2 System Configuration and Protection

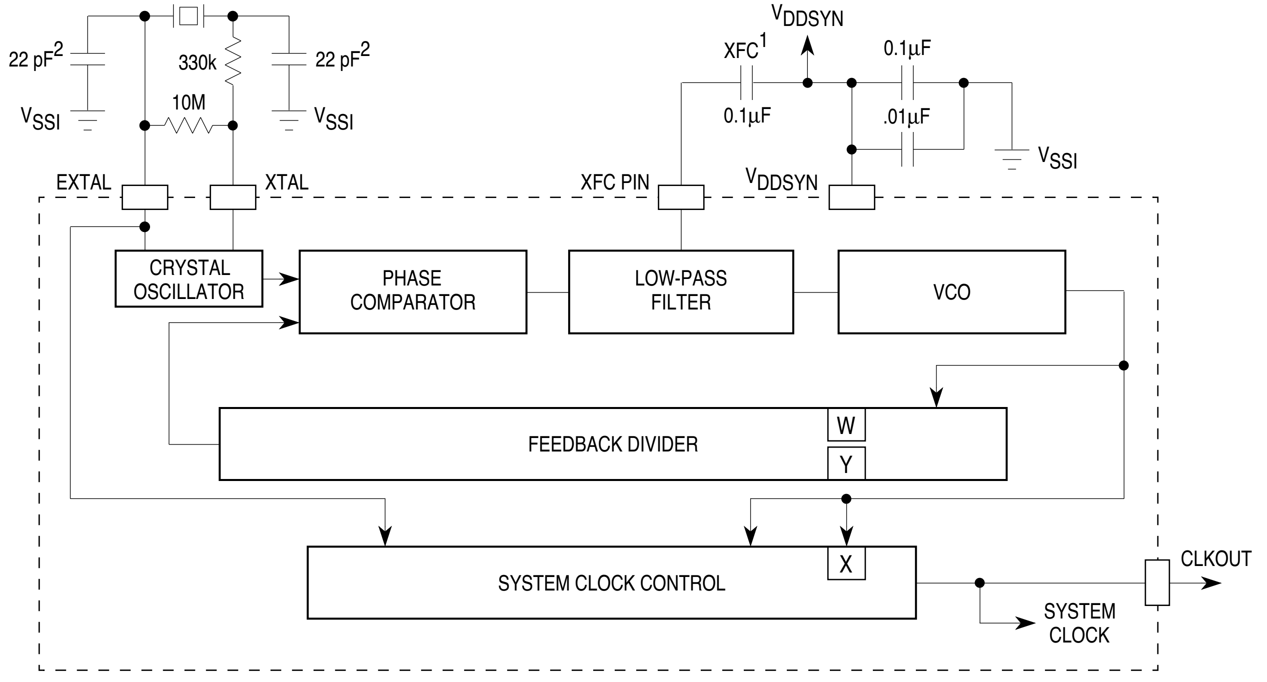
This functional block provides configuration control for the entire MCU. It also performs interrupt arbitration, bus monitoring, and system test functions. MCU system protection includes a bus monitor, a HALT monitor, a spurious interrupt monitor, and a software watchdog timer. These functions have been made integral to the microcontroller to reduce the number of external components in a complete control system.

3.3 System Clock

The system clock in the SIM provides timing signals for the IMB modules and for an external peripheral bus. Because MCU operation is fully static, register and memory contents are not affected when the clock rate changes. System hardware and software support changes in the clock rate during operation.

The system clock signal can be generated in three ways. An internal phase-locked loop can synthesize the clock from an internal or external frequency source, or the clock signal can be input from an external source.

Following is a block diagram of the clock submodule.



1. MUST BE LOW-LEAKAGE CAPACITOR (INSULATION RESISTANCE 30,000 MΩ OR GREATER).

2. RESISTANCE AND CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A DAISHINKU DMX-38 32.768-kHz CRYSTAL. SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.

SYS CLOCK BLOCK 32KHZ

Figure 7 System Clock Block Diagram

3.3.1 Clock Sources

The state of the clock mode (MODCLK) pin during reset determines the clock source. When MODCLK is held high during reset, the clock synthesizer generates a clock signal from either a crystal oscillator or an external reference input. Clock synthesizer control register SYNCR determines operating frequency and various modes of operation. When MODCLK is held low during reset, the clock synthesizer is disabled, and an external system clock signal must be applied. When the synthesizer is disabled, SYNCR control bits have no effect.

A reference crystal must be connected between the EXTAL and XTAL pins to use the internal oscillator. Use of a 32.768-kHz crystal is recommended. These crystals are inexpensive and readily available. If an external reference signal or an external system clock signal is applied through the EXTAL pin, the XTAL pin must be left floating. External reference signal frequency must be less than or equal to maximum specified reference frequency. External system clock signal frequency must be less than or equal to maximum specified system clock frequency.

Table 9 CPU32 Address Space Encoding

FC2	FC1	FC0	Address Space
0	0	0	Reserved
0	0	1	User Data Space
0	1	0	User Program Space
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

3.4.3 Address Bus

Address bus signals ADDR[23:0] define the address of the most significant byte to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{\text{AS}}$ is asserted.

3.4.4 Address Strobe

$\overline{\text{AS}}$ is a timing signal that indicates the validity of an address on the address bus and the validity of many control signals. It is asserted one-half clock after the beginning of a bus cycle.

3.4.5 Data Bus

Data bus signals DATA[15:0] make up a bidirectional, non-multiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer 8 or 16 bits of data in one bus cycle. During a read cycle, the data is latched by the MCU on the last falling edge of the clock for that bus cycle. For a write cycle, all 16 bits of the data bus are driven, regardless of the port width or operand size. The MCU places the data on the data bus one-half clock cycle after $\overline{\text{AS}}$ is asserted in a write cycle.

3.4.6 Data Strobe

Data strobe ($\overline{\text{DS}}$) is a timing signal. For a read cycle, the MCU asserts $\overline{\text{DS}}$ to signal an external device to place data on the bus. $\overline{\text{DS}}$ is asserted at the same time as $\overline{\text{AS}}$ during a read cycle. For a write cycle, $\overline{\text{DS}}$ signals an external device that data on the bus is valid. The MCU asserts $\overline{\text{DS}}$ one full clock cycle after the assertion of $\overline{\text{AS}}$ during a write cycle.

3.4.7 Bus Cycle Termination Signals

During bus cycles, external devices assert the data transfer and size acknowledge signals ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$). During a read cycle, the signals tell the MCU to terminate the bus cycle and to latch data. During a write cycle, the signals indicate that an external device has successfully stored data and that the cycle can end. These signals also indicate to the MCU the size of the port for the bus cycle just completed. (Refer to 3.4.9 Dynamic Bus Sizing.)

The bus error ($\overline{\text{BERR}}$) signal is also a bus cycle termination indicator and can be used in the absence of $\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$ to indicate a bus error condition. It can also be asserted in conjunction with these signals, provided it meets the appropriate timing requirements. The internal bus monitor can be used to generate the $\overline{\text{BERR}}$ signal for internal and internal-to-external transfers. When $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ are asserted simultaneously, the CPU takes a bus error exception.

Autovector signal ($\overline{\text{AVEC}}$) can terminate external $\overline{\text{IRQ}}$ pin interrupt acknowledge cycles. $\overline{\text{AVEC}}$ indicates that the MCU will internally generate a vector number to locate an interrupt handler routine. If it is continuously asserted, autovectors will be generated for all external interrupt requests. $\overline{\text{AVEC}}$ is ignored during all other bus cycles.

$\overline{\text{AVEC}}$ — Autovector Enable

0 = External interrupt vector enabled
1 = Autovector enabled

This field selects one of two methods of acquiring the interrupt vector during the interrupt acknowledge cycle. It is not usually used in conjunction with a chip-select pin.

If the chip select is configured to trigger on an interrupt acknowledge cycle (SPACE = 00) and the $\overline{\text{AVEC}}$ field is set to one, the chip select automatically generates an $\overline{\text{AVEC}}$ in response to the interrupt cycle. Otherwise, the vector must be supplied by the requesting device.

The $\overline{\text{AVEC}}$ bit must not be used in synchronous mode, as autovector response timing can vary because of ECLK synchronization.

3.5.5 Port C Data Register

Bit values in port C determine the state of chip-select pins used for discrete output. When a pin is assigned as a discrete output, the value in this register appears at the output. This is a read/write register. Bit 7 is not used. Writing to this bit has no effect, and it always returns zero when read.

PORTC — Port C Data Register **$YFFA41**

	15	8	7	6	5	4	3	2	1	0
	NOT USED		0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:			0	1	1	1	1	1	1	1

3.6 General-Purpose Input/Output

SIM pins can be configured as two general-purpose I/O ports, E and F. The following paragraphs describe registers that control the ports.

PORTE0, PORTE1 —Port E Data Register **$YFFA11, $YFFA13**

	15	8	7	6	5	4	3	2	1	0
	NOT USED		PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:			U	U	U	U	U	U	U	U

A write to the port E data register is stored in the internal data latch and, if any port E pin is configured as an output, the value stored for that bit is driven on the pin. A read of the port E data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register.

The port E data register is a single register that can be accessed in two locations. When accessed at $YFFA11, the register is referred to as PORTE0; when accessed at $YFFA13, the register is referred to as PORTE1. The register can be read or written at any time. It is unaffected by reset.

DDRE — Port E Data Direction Register **$YFFA15**

	15	8	7	6	5	4	3	2	1	0
	NOT USED		DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:			0	0	0	0	0	0	0	0

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. This register can be read or written at any time.

PFPAR — Port F Pin Assignment Register **$YFFA1F**

15	8	7	6	5	4	3	2	1	0
NOT USED		PFPA7	PFPA6	PFPA5	PFPA4	PFPA3	PFPA2	PFPA1	PFPA0
RESET:									
		DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9

The bits in this register control the function of each port F pin. Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be an interrupt request signal or MODCLK. The MODCLK signal has no function after reset.

Table 17 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA7	PF7	$\overline{\text{IRQ7}}$
PFPA6	PF6	$\overline{\text{IRQ6}}$
PFPA5	PF5	$\overline{\text{IRQ5}}$
PFPA4	PF4	$\overline{\text{IRQ4}}$
PFPA3	PF3	$\overline{\text{IRQ3}}$
PFPA2	PF2	$\overline{\text{IRQ2}}$
PFPA1	PF1	$\overline{\text{IRQ1}}$
PFPA0	PF0	MODCLK

Data bus pin 9 controls the state of this register following reset. If DATA9 is set to one during reset, the register is set to $FF, which defines all port F pins as interrupt request inputs. If DATA9 is cleared to zero during reset, this register is set to $00, defining all port F pins as I/O pins.

3.7 Resets

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The system integration module determines whether a reset is valid, asserts control signals, performs basic system configuration based on hardware mode-select inputs, then passes control to the CPU.

Reset occurs when an active low logic level on the $\overline{\text{RESET}}$ pin is clocked into the SIM. Resets are gated by the CLKOUT signal. Asynchronous resets are assumed to be catastrophic. An asynchronous reset can occur on any clock edge. Synchronous resets are timed to occur at the end of bus cycles. If there is no clock when $\overline{\text{RESET}}$ is asserted, reset does not occur until the clock starts. Resets are clocked in order to allow completion of write cycles in progress at the time $\overline{\text{RESET}}$ is asserted.

Reset is the highest-priority CPU32 exception. Any processing in progress is aborted by the reset exception, and cannot be restarted. Only essential tasks are performed during reset exception processing. Other initialization tasks must be accomplished by the exception handler routine.

3.7.1 SIM Reset Mode Selection

The logic states of certain data bus pins during reset determine SIM operating configuration. In addition, the state of the MODCLK pin determines system clock source and the state of the $\overline{\text{BKPT}}$ pin determines what happens during subsequent breakpoint assertions. The following table is a summary of reset mode selection options.

Table 18 Reset Mode Selection

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)

When TSC assertion takes effect, internal signals are forced to values that can cause inadvertent mode selection. Once the output drivers change state, the MCU must be powered down and restarted before normal operation can resume.

3.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the central processing unit, the system integration module, and a device or module requesting interrupt service.

The CPU32 provides for eight levels of interrupt priority (0–7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in the status register. The CPU32 handles interrupts as a type of asynchronous exception.

Interrupt recognition is based on the states of interrupt request signals $\overline{\text{iIRQ[7:1]}}$ and the IP mask value. Each of the signals corresponds to an interrupt priority. $\overline{\text{IRQ1}}$ has the lowest priority, and $\overline{\text{IRQ7}}$ has the highest priority.

The IP field consists of three bits. Binary values %000 to %111 provide eight priority masks. Masks prevent an interrupt request of a priority less than or equal to the mask value (except for $\overline{\text{IRQ7}}$) from being recognized and processed. When IP contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

Interrupt request signals can be asserted by external devices or by microcontroller modules. Request lines are connected internally by means of a wired NOR — simultaneous requests of differing priority can be made. Internal assertion of an interrupt request signal does not affect the logic state of the corresponding MCU pin.

External interrupt requests are routed to the CPU via the external bus interface and SIM interrupt control logic. The CPU treats external interrupt requests as though they come from the SIM.

External $\overline{\text{IRQ[6:1]}}$ are active-low level-sensitive inputs. External $\overline{\text{IRQ7}}$ is an active-low transition-sensitive input. $\overline{\text{IRQ7}}$ requires both an edge and a voltage level for validity.

$\overline{\text{IRQ[6:1]}}$ are maskable. $\overline{\text{IRQ7}}$ is nonmaskable. The $\overline{\text{IRQ7}}$ input is transition-sensitive in order to prevent redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted, and each time the priority mask changes from %111 to a lower number while$\overline{\text{IRQ7}}$ is asserted.

Interrupt requests are sampled on consecutive falling edges of the system clock. Interrupt request input circuitry has hysteresis. To be valid, a request signal must be asserted for at least two consecutive clock periods. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when exception processing of higher-priority exceptions is complete.

The CPU32 does not latch the priority of a pending interrupt request. If an interrupt source of higher priority makes a service request while a lower priority request is pending, the higher priority request is serviced. If an interrupt request of equal or lower priority than the current IP mask value is made, the CPU does not recognize the occurrence of the request in any way.

3.8.1 Interrupt Acknowledge and Arbitration

Interrupt acknowledge bus cycles are generated during exception processing. When the CPU detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it performs a CPU space read from address $FFFFF : [IP] : 1.

The CPU space read cycle performs two functions: it places a mask value corresponding to the highest priority interrupt request on the address bus, and it acquires an exception vector number from the interrupt source. The mask value also serves two purposes: it is latched into the CCR IP field in order to mask lower-priority interrupts during exception processing, and it is decoded by modules that have requested interrupt service to determine whether the current interrupt acknowledge cycle pertains to them.

Modules that have requested interrupt service decode the IP value placed on the address bus at the beginning of the interrupt acknowledge cycle, and if their requests are at the specified IP level, respond to the cycle. Arbitration between simultaneous requests of the same priority is performed by means of serial contention between module interrupt arbitration (IARB) field bit values.

Each module that can make an interrupt service request, including the SIM, has an IARB field in its configuration register. An IARB field can be assigned a value from %0001 (lowest priority) to %1111 (highest priority). A value of %0000 in an IARB field causes the CPU to process a spurious interrupt exception when an interrupt from that module is recognized.

Because the EBI manages external interrupt requests, the SIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SIM is %1111, and the reset IARB value for all other modules is %0000. Initialization software must assign different IARB values in order to implement an arbitration scheme.

Each module must have a unique IARB value. When two or more IARB fields have the same nonzero value, the CPU interprets multiple vector numbers simultaneously, with unpredictable consequences.

Arbitration must always take place, even when a single source requests service. This point is important for two reasons: the CPU interrupt acknowledge cycle is not driven on the external bus unless the SIM wins contention, and failure to contend causes an interrupt acknowledge bus cycle to be terminated by a bus error, which causes a spurious interrupt exception to be taken.

When arbitration is complete, the dominant module must place an interrupt vector number on the data bus and terminate the bus cycle. In the case of an external interrupt request, because the interrupt acknowledge cycle is transferred to the external bus, an external device must decode the mask value and respond with a vector number, then generate bus cycle termination signals. If the device does not respond in time, a spurious interrupt exception is taken.

The periodic interrupt timer (PIT) in the SIM can generate internal interrupt requests of specific priority at predetermined intervals. By hardware convention, PIT interrupts are serviced before external interrupt service requests of the same priority. Refer to 3.2.7 Periodic Interrupt Timer for more information.

3.8.2 Interrupt Processing Summary

A summary of the interrupt processing sequence follows. When the sequence begins, a valid interrupt service request has been detected and is pending.

A. The CPU finishes higher priority exception processing or reaches an instruction boundary.
B. Processor state is stacked. The contents of the status register and program counter are saved.
C. The interrupt acknowledge cycle begins:
   1. FC[2:0] are driven to %111 (CPU space) encoding.
   2. The address bus is driven as follows. ADDR[23:20] = %1111; ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; ADDR[3:1] = the level of the interrupt request being acknowledged; and ADDR0 = %1.
   3. Request priority level is latched into the IP field in the status register from the address bus.
D. Modules or external peripherals that have requested interrupt service decode the request level in ADDR[3:1]. If the request level of at least one interrupting module or device is the same as the value in ADDR[3:1], interrupt arbitration contention takes place. When there is no contention, the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and a spurious interrupt exception is processed.
E. After arbitration, the interrupt acknowledge cycle can be completed in one of three ways:

1. The dominant interrupt source supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU32 acquires the vector number.
2. The $\overline{\text{AVEC}}$ signal is asserted (the signal can be asserted by the dominant interrupt source or the pin can be tied low), and the CPU32 generates an autovector number corresponding to interrupt priority.
3. The bus monitor asserts $\overline{\text{BERR}}$ and the CPU32 generates the spurious interrupt vector number.

F. The vector number is converted to a vector address.

G. The content of the vector address is loaded into the PC, and the processor transfers control to the exception handler routine.

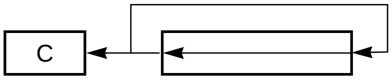
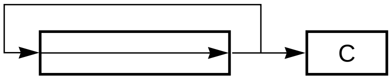
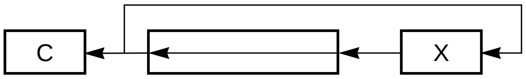
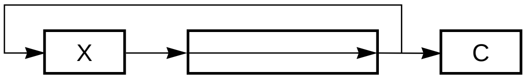
3.9 Factory Test Block

The test submodule supports scan-based testing of the various MCU modules. It is integrated into the SIM to support production testing.

Test submodule registers are intended for Motorola use. Register names and addresses are provided to indicate that these addresses are occupied.

SIMTR —System Integration Test Register	**$YFFA02**
SIMTRE —System Integration Test Register (E Clock)	**$YFFA08**
TSTMSRA —Master Shift Register A	**$YFFA30**
TSTMSRB —Master Shift Register B	**$YFFA32**
TSTSC —Test Module Shift Count	**$YFFA34**
TSTRC —Test Module Repetition Count	**$YFFA36**
CREG —Test Module Control Register	**$YFFA38**
DREG —Test Module Distributed Register	**$YFFA3A**

Table 20 Instruction Set Summary(Continued)

Instruction	Syntax	Operand Size	Operation
MOVES[1]	Rn, <ea> <ea>, Rn	8, 16, 32	Rn ⇒ Destination using DFC Source using SFC ⇒ Rn
MULS/MULU	<ea>, Dn <ea>, Dl <ea>, Dh : Dl	16 ∗ 16 ⇒ 32 32 ∗ 32 ⇒ 32 32 ∗ 32 ⇒ 64	Source ∗ Destination ⇒ Destination (signed or unsigned)
NBCD	Í	8 8	0 – Destination_{10} – X ⇒ Destination
NEG	Í	8, 16, 32	0 – Destination ⇒ Destination
NEGX	Í	8, 16, 32	0 – Destination – X ⇒ Destination
NOP	none	none	PC + 2 ⇒ PC
NOT	Í	8, 16, 32	$\overline{\text{Destination}}$ ⇒ Destination
OR	<ea>, Dn Dn, <ea>	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ORI	#<data>, <ea>	8, 16, 32	Data + Destination ⇒ Destination
ORI to CCR	#<data>, CCR	16	Source + CCR ⇒ SR
ORI to SR[1]	#<data>, SR	16	Source ; SR ⇒ SR
PEA	Í	32	SP – 4 ⇒ SP; <ea> ⇒ SP
RESET[1]	none	none	Assert $\overline{\text{RESET}}$ line
ROL	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
ROR	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
ROXL	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
ROXR	Dn, Dn #<data>, Dn Í	8, 16, 32 8, 16, 32 16	
RTD	#d	16	(SP) ⇒ PC; SP + 4 + d ⇒ SP

RTE[1]	none	none	(SP) ⇒ SR; SP + 2 ⇒ SP; (SP) ⇒ PC;
			SP + 4 ⇒ SP;
			Restore stack according to format

RTR	none	none	(SP) ⇒ CCR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP
RTS	none	none	(SP) ⇒ PC; SP + 4 ⇒ SP
SBCD	Dn, Dn – (An), – (An)	8 8	Destination10 – Source10 – X ⇒ Destination
Scc	Í	8	If condition true, then destination bits are set to 1; else, destination bits are cleared to 0
STOP[1]	#<data>	16	Data ⇒ SR; STOP
SUB	<ea>, Dn Dn, <ea>	8, 16, 32	Destination – Source ⇒ Destination
SUBA	<ea>, An	16, 32	Destination – Source ⇒ Destination
SUBI	#<data>, <ea>	8, 16, 32	Destination – Data ⇒ Destination
SUBQ	#<data>, <ea>	8, 16, 32	Destination – Data ⇒ Destination
SUBX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Destination – Source – X ⇒ Destination

5.1.7 Period Measurement with Missing Transition Detect (PMM)

Period measurement with missing transition detect allows a special-purpose 23-bit period measurement. It detects the occurrence of a missing transition (caused by a missing tooth on the sensed wheel), indicated by a period measurement that is greater than a programmable ratio of the previous period measurement. Once detected, this condition can be counted and compared to a programmable number of additional transitions detected before TCR2 is reset to $FFFF. In addition, one byte at an address specified by a channel parameter can be read and used as a flag. A nonzero value of the flag indicates that TCR2 is to be reset to $FFFF once the next missing transition is detected.

5.1.8 Position-Synchronized Pulse Generator (PSP)

Any channel of the TPU can generate an output transition or pulse, which is a projection in time based on a reference period previously calculated on another channel. Both TCRs are used in this algorithm: TCR1 is internally clocked, and TCR2 is clocked by a position indicator in the user's device. An example of a TCR2 clock source is a sensor that detects special teeth on the flywheel of an automobile using PMA or PMM. The teeth are placed at known degrees of engine rotation; hence, TCR2 is a coarse representation of engine degrees, i.e., each count represents some number of degrees.

Up to 15 position-synchronized pulse generator function channels can operate with a single input reference channel executing a PMA or PMM input function. The input channel measures and stores the time period between the flywheel teeth and resets TCR2 when the engine reaches a reference position. The output channel uses the period calculated by the input channel to project output transitions at specific engine degrees. Because the flywheel teeth might be 30 or more degrees apart, a fractional multiplication operation resolves down to the desired degrees. Two modes of operation allow pulse length to be determined either by angular position or by time.

5.1.9 Stepper Motor (SM)

The stepper motor control algorithm provides for linear acceleration and deceleration control of a stepper motor with a programmable number of step rates of up to 14. Any group of channels, up to eight, can be programmed to generate the control logic necessary to drive a stepper motor.

The time period between steps (P) is defined as:

$$P(r) = K1 - K2 * r$$

where r is the current step rate (1–14), and K1 and K2 are supplied as parameters.

After providing the desired step position in a 16-bit parameter, the CPU issues a step request. Next, the TPU steps the motor to the desired position through an acceleration/deceleration profile defined by parameters. The parameter indicating the desired position can be changed by the CPU while the TPU is stepping the motor. This algorithm changes the control state every time a new step command is received.

A 16-bit parameter initialized by the CPU for each channel defines the output state of the associated pin. The bit pattern written by the CPU defines the method of stepping, such as full stepping or half stepping. With each transition, the 16-bit parameter rotates one bit. The period of each transition is defined by the programmed step rate.

5.1.10 Period/Pulse-Width Accumulator (PPWA)

The period/pulse-width accumulator algorithm accumulates a 16-bit or 24-bit sum of either the period or the pulse width of an input signal over a programmable number of periods or pulses (from 1 to 255). After an accumulation period, the algorithm can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and number of channels within the block. Generation of links depends on the mode of operation. Any channel can be used to measure an accumulated number of periods of an input signal. A maximum of 24 bits can be used for the accumu-

EMU — Emulation Control

In emulation mode, the TPU executes microinstructions from MCU TPURAM exclusively. Access to the TPURAM module through the IMB by a host 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 not in emulation mode
1 = TPU and TPURAM in emulation mode

T2CG — TCR2 Clock/Gate Control

When the T2CG bit is set, the external TCR2 pin functions as a gate of the DIV8 clock (the TPU system clock divided by 8). 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 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

STF — Stop Flag

0 = TPU operating
1 = TPU stopped (STOP bit has been asserted)

SUPV — Supervisor Data Space

0 = Assignable registers are unrestricted (FC2 is ignored)
1 = Assignable registers are restricted (FC2 is decoded)

PSCK — Prescaler Clock

0 = System clock/32 is input to TCR1 prescaler
1 = System clock/4 is input to TCR1 prescaler

IARB — Interrupt Arbitration Identification Number

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. Refer to the **3.8 Interrupts** for more information.

TICR — TPU Interrupt Configuration Register **$YFFE08**

<table>
<tr><td>15</td><td></td><td></td><td></td><td>11</td><td>10</td><td></td><td>8</td><td>7</td><td></td><td></td><td>4</td><td>3</td><td></td><td></td><td>0</td></tr>
<tr><td colspan="5">NOT USED</td><td colspan="3">CIRL</td><td colspan="4">CIBV</td><td colspan="4">NOT USED</td></tr>
<tr><td colspan="16">RESET:</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td></td><td></td><td></td><td></td></tr>
</table>

CIRL — Channel Interrupt Request Level

The interrupt request level for all channels is specified by this 3-bit encoded field. Level seven for this field indicates a nonmaskable interrupt; level zero indicates that all channel interrupts are disabled.

CIBV — Channel Interrupt Base Vector

The TPU is assigned 16 unique interrupt vector numbers, one vector number for each channel. The CIBV field specifies the most significant nibble of all 16 TPU channel interrupt vector numbers. The lower nibble of the TPU interrupt vector number is determined by the channel number on which the interrupt occurs.

Table 26 Effect of DDRQS on QSM Pin Function

QSM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQ0	0	Serial Data Input to QSPI
			1	Disables Data Input
	Slave		0	Disables Data Output
			1	Serial Data Output from QSPI
MOSI	Master	DDQ1	0	Disables Data Output
			1	Serial Data Output from QSPI
	Slave		0	Serial Data Input to QSPI
			1	Disables Data Input
SCK[1]	Master	DDQ2	0	Disables Clock Output
			1	Clock Output from QSPI
	Slave		0	Clock Input to QSPI
			1	Disables Clock Input
PCS0/$\overline{\text{SS}}$	Master	DDQ3	0	Assertion Causes Mode Fault
			1	Chip-Select Output
	Slave		0	QSPI Slave Select Input
			1	Disables Select Input
PCS[3:1]	Master	DDQ[4:6]	0	Disables Chip-Select Output
			1	Chip-Select Output
	Slave		0	Inactive
			1	Inactive
TXD[2]	Transmit	DDQ7	X	Serial Data Output from SCI
RXD	Receive	None	NA	Serial Data Input to SCI

NOTES:

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 = 1), in which case it becomes SCI serial output TXD.

DDRQS determines the direction of the TXD pin only when the SCI transmitter is disabled. When the SCI transmitter is enabled, the TXD pin is an output.

Pin Names	Mnemonics	Mode	Function
Master In Slave Out	MISO	Master Slave	Serial Data Input to QSPI Serial Data Output from QSPI
Master Out Slave In	MOSI	Master Slave	Serial Data Output from QSPI Serial Data Input to QSPI
Serial Clock	SCK	Master Slave	Clock Output from QSPI Clock Input to QSPI
Peripheral Chip Selects	PCS[3:1]	Master	Select Peripherals
Peripheral Chip Select Slave Select	PCS0 $\overline{SS}$	Master Master Slave	Selects Peripheral Causes Mode Fault Initiates Serial Transfer

6.5.2 QSPI Registers

The programmer's model for the QSPI submodule consists of the QSM global and pin control registers, four QSPI control registers, one status register, and the 80-byte QSPI RAM.

The CPU can read and write to registers and RAM. The four control registers must be initialized before the QSPI is enabled to ensure defined operation. SPCR1 should be written last because it contains QSPI enable bit SPE. Asserting this bit starts the QSPI. The QSPI control registers are reset to a defined state and can then be changed by the CPU. Reset values are shown below each register.

Refer to the following memory map of the QSPI.

Address	Name	Usage
$YFFC18	SPCR0	QSPI Control Register 0
$YFFC1A	SPCR1	QSPI Control Register 1
$YFFC1C	SPCR2	QSPI Control Register 2
$YFFC1E	SPCR3	QSPI Control Register 3
$YFFC1F	SPSR	QSPI Status Register
$YFFD00	RAM	QSPI Receive Data (16 Words)
$YFFD20	RAM	QSPI Transmit Data (16 Words)
$YFFD40	RAM	QSPI Command Control (8 Words)

Writing a different value into any control register except SPCR2 while the QSPI is enabled disrupts operation. SPCR2 is buffered to prevent disruption of the current serial transfer. After completion of the current serial transfer, the new SPCR2 values become effective.

Writing the same value into any control register except SPCR2 while the QSPI is enabled has no effect on QSPI operation. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location.

SPCR0 — QSPI Control Register 0 **$YFFC18**

15	14	13			10	9	8	7							0
MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

SPCR0 contains parameters for configuring the QSPI before it is enabled. The CPU can read and write this register. The QSM has read-only access.

SPCR2 — QSPI Control Register 2 **$YFFC1C**

15	14	13	12	11			8	7	6	5	4	3			0
SPIFIE	WREN	WRTO	0	ENDQP				0	0	0	0	NEWQP			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPCR2 contains QSPI configuration parameters. The CPU can read and write this register; the QSM has read access only. Writes to SPCR2 are buffered. A write to SPCR2 that changes a bit value while the QSPI is operating is ineffective on the current serial transfer, but becomes effective on the next serial transfer. Reads of SPCR2 return the current value of the register, not of the buffer.

SPIFIE — SPI Finished Interrupt Enable
0 = QSPI interrupts disabled
1 = QSPI interrupts enabled
SPIFIE enables the QSPI to generate a CPU interrupt upon assertion of the status flag SPIF.

WREN — Wrap Enable
0 = Wraparound mode disabled
1 = Wraparound mode enabled
WREN enables or disables wraparound mode.

WRTO — Wrap To
When wraparound mode is enabled, after the end of queue has been reached, WRTO determines which address the QSPI executes.

Bit 12 — Not Implemented

ENDQP — Ending Queue Pointer
This field contains the last QSPI queue address.

Bits [7:4] — Not Implemented

NEWQP — New Queue Pointer Value
This field contains the first QSPI queue address.

SPCR3 — QSPI Control Register 3 **$YFFC1E**

15	14	13	12	11	10	9	8	7							0
0	0	0	0	0	LOOPQ	HMIE	HALT	SPSR							
RESET:															
0	0	0	0	0	0	0	0								

SPCR3 contains QSPI configuration parameters. The CPU can read and write SPCR3, but the QSM has read-only access.

Bits [15:11] — Not Implemented

LOOPQ — QSPI Loop Mode
0 = Feedback path disabled
1 = Feedback path enabled
LOOPQ controls feedback on the data serializer for testing.

HMIE — HALTA and MODF Interrupt Enable
0 = HALTA and MODF interrupts disabled
1 = HALTA and MODF interrupts enabled
HMIE controls CPU interrupts caused by the HALTA status flag or the MODF status flag in SPSR.

HALT — Halt

0 = Halt not enabled

1 = Halt enabled

When HALT is asserted, the QSPI stops on a queue boundary. It is in a defined state from which it can later be restarted.

SPSR — QSPI Status Register **$YFFC1F**

15	8	7	6	5	4	3			0
SPCR3		SPIF	MODF	HALTA	0	CPTQP			
RESET:									
		0	0	0	0	0	0	0	0

SPSR contains QSPI status information. Only the QSPI can assert the bits in this register. The CPU reads this register to obtain status information and writes it to clear status flags.

SPIF — QSPI Finished Flag

0 = QSPI not finished

1 = QSPI finished

SPIF is set after execution of the command at the address in ENDQP.

MODF — Mode Fault Flag

0 = Normal operation

1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode ($\overline{SS}$ input taken low).

The QSPI asserts MODF when the QSPI is the serial master (MSTR = 1) and the $\overline{SS}$ input pin is negated by an external driver.

HALTA — Halt Acknowledge Flag

0 = QSPI not halted

1 = QSPI halted

HALTA is asserted when the QSPI halts in response to CPU assertion of HALT.

Bit 4 — Not Implemented

CPTQP — Completed Queue Pointer

CPTQP points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP contains either the reset value ($0) or a pointer to the last command completed in the previous queue.

6.5.3 QSPI RAM

The QSPI contains an 80-byte block of dual-access static RAM that is used by both the QSPI and the CPU. The RAM is divided into three segments: receive data, transmit data, and command control data. Receive data is information received from a serial device external to the MCU. Transmit data is information stored by the CPU for transmission to an external peripheral. Command control data is used to perform the transfer.

Refer to the following illustration of the organization of the RAM.

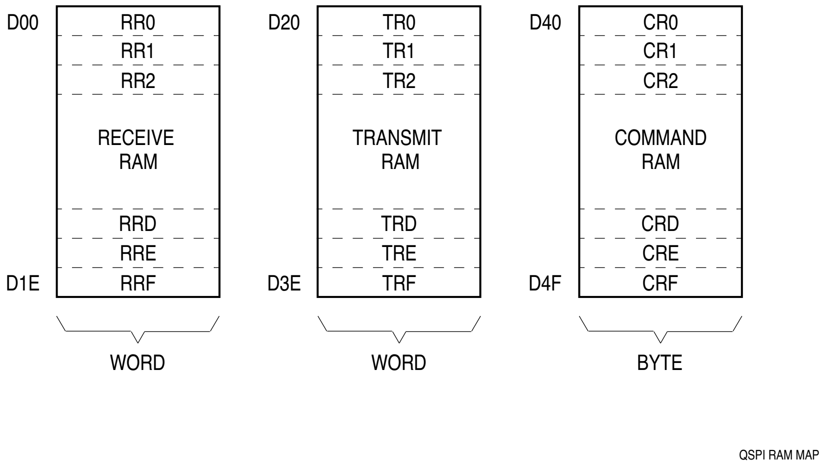


Figure 15 QSPI RAM

Once the CPU has set up the queue of QSPI commands and enabled the QSPI, the QSPI can operate independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating that it is finished, and then either interrupts the CPU or waits for CPU intervention. It is possible to execute a queue of commands repeatedly without CPU intervention.

RR[0:F] — Receive Data RAM **$YFFD00**

Data received by the QSPI is stored in this segment. The CPU reads this segment to retrieve data from the QSPI. Data stored in receive RAM is right-justified. Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, word, or long-word addressing.

The CPTQP value in SPSR shows which queue entries have been executed. The CPU uses this information to determine which locations in receive RAM contain valid data before reading them.

TR[0:F] — Transmit Data RAM **$YFFD20**

Data that is to be transmitted by the QSPI is stored in this segment. The CPU usually writes one word of data into this segment for each queue command to be executed.

Information to be transmitted must be written to transmit data RAM in a right-justified format. The QSPI cannot modify information in the transmit data RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

CR[0:F] — Command Ram **$YFFD40**

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*
—	—	—	—	—	—	—	—
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0*

COMMAND CONTROL | PERIPHERAL CHIP SELECT

*The PCS0 bit represents the dual-function PCS0/$\overline{\text{SS}}$.

IDLE — Idle-Line Detected Flag

0 = SCI receiver did not detect an idle-line condition.
1 = SCI receiver detected an idle-line condition.

IDLE is disabled when RWU in SCCR1 is set. IDLE is set when the SCI receiver detects the idle-line condition specified by ILT in SCCR1. If cleared, IDLE will not set again until after RDRF is set. RDRF is set when a break is received, so that a subsequent idle line can be detected.

OR — Overrun Error Flag

0 = RDRF is cleared before new data arrives.
1 = RDRF is not cleared before new data arrives.

OR is set when a new byte is ready to be transferred from the receive serial shifter to the RDR, and RDRF is still set. Data transfer is inhibited until OR is cleared. Previous data in RDR remains valid, but data received during overrun condition (including the byte that set OR) is lost.

NF — Noise Error Flag

0 = No noise detected on the received data
1 = Noise occurred on the received data

NF is set when the SCI receiver detects noise on a valid start bit, on any data bit, or on a stop bit. It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times. If none of the three samples are the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until an entire frame is received and RDRF is set.

FE — Framing Error Flag

0 = No framing error on the received data.
1 = Framing error or break occurred on the received data.

FE is set when the SCI receiver detects a zero where a stop bit was to have occurred. FE is not set until the entire frame is received and RDRF is set. A break can also cause FE to be set. It is possible to miss a framing error if RXD happens to be at logic level one at the time the stop bit is expected.

PF — Parity Error Flag

0 = No parity error on the received data
1 = Parity error occurred on the received data

PF is set when the SCI receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set.

SCDR — SCI Data Register **$YFFC0E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:															
0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

SCDR contains two data registers at the same address. Receive data register (RDR) is a read-only register that contains data received by the SCI. The data comes into the receive serial shifter and is transferred to RDR. Transmit data register (TDR) is a write-only register that contains data to be transmitted. The data is first written to TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission. R[7:0]/T[7:0] contain either the first eight data bits received when SCDR is read, or the first eight data bits to be transmitted when SCDR is written. R8/T8 are used when the SCI is configured for 9-bit operation. When it is configured for 8-bit operation, they have no meaning or effect.