Technical Summary
32-Bit Modular Microcontroller

1 Introduction

The MC68334, a highly-integrated 32-bit microcontroller, combines high-performance data manipulation capabilities with powerful peripheral subsystems. The MCU is built up from standard modules that interface through a common intermodule bus (IMB). Standardization facilitates rapid development of devices tailored for specific applications.

The MCU incorporates a 32-bit CPU (CPU32), a system integration module (SIM), an 8/10-bit analog-to-digital converter (ADC), a time processor unit (TPU) and a 1-Kbyte static RAM module with TPU emulation capability (TPURAM).

The MCU can either synthesize an internal clock signal from an external reference or use an external clock input directly. Operation with a 32.768-kHz reference frequency is standard. The maximum system clock speed is 20.97 MHz. System hardware and software allow changes in clock rate during operation. Because MCU operation is fully static, register and memory contents are not affected by clock rate changes.

High-density complementary metal-oxide semiconductor (HCMOS) architecture makes the basic power consumption of the MCU low. Power consumption can be minimized by stopping the system clock. The CPU32 instruction set includes a low-power stop (LPSTOP) command that efficiently implements this capability.

This document contains information about a new product. Specifications and information herein are subject to change without notice.

Table 1 Ordering Information

Package Type	TPU Type	Temperature	Frequency	Order Number
132-Pin PQFP	Motion Control	–40 to +85 °C	16 MHz	MC68334GCFC16
			20 MHz	MC68334GCFC20
		–40 to +105 °C	16 MHz	MC68334GVFC16
			20 MHz	MC68334GVFC20
		–40 to +125 °C	16 MHz	MC68334GMFC16
			20 MHz	MC68334GMFC20

Order Quantity		
SP Prefix	No Affix	B1 Suffix
2-Piece Tray	36-Piece Tray	180 (5 Trays)

freescale™
semiconductor

TABLE OF CONTENTS

1.1 Features

- Central Processing Unit (CPU32)
  - 32-Bit Architecture
  - Virtual Memory Implementation
  - Table Lookup and Interpolate Instruction
  - Improved Exception Handling for Controller Applications
  - High-Level Language Support
  - Background Debugging Mode
  - Fully Static Operation
- System Integration Module (SIM)
  - External Bus Support
  - Programmable Chip-Select Outputs
  - System Protection Logic
  - Watchdog Timer, Clock Monitor, and Bus Monitor
  - Two 8-Bit Dual Function Input/Output Ports
  - One 7-Bit Dual Function Output Port
  - Phase-Locked Loop (PLL) Clock System
- 8/10-Bit Analog-to-Digital Converter (ADC)
  - Seven Analog/Digital Input Pins (Eighth Channel Connected to V_{SSA})
  - Eight Result Registers
  - Eight Conversion Modes
  - Three Result Alignment Modes
  - One 7-Bit Digital Input Port
- Time Processor Unit (TPU)
  - Dedicated Microengine Operating Independently of CPU32
  - 16 Independent, Programmable Channels and Pins
  - Any Channel can Perform any Microcoded Time Function
  - Two Timer Count Registers with Programmable Prescalers
  - Selectable Channel Priority Levels
- 1-Kbyte Standby RAM with TPU Emulation (TPURAM)
  - External Standby Voltage Supply Input
  - Can be Used as Standby RAM or TPU Microcode Emulation RAM

1.2 Block Diagram
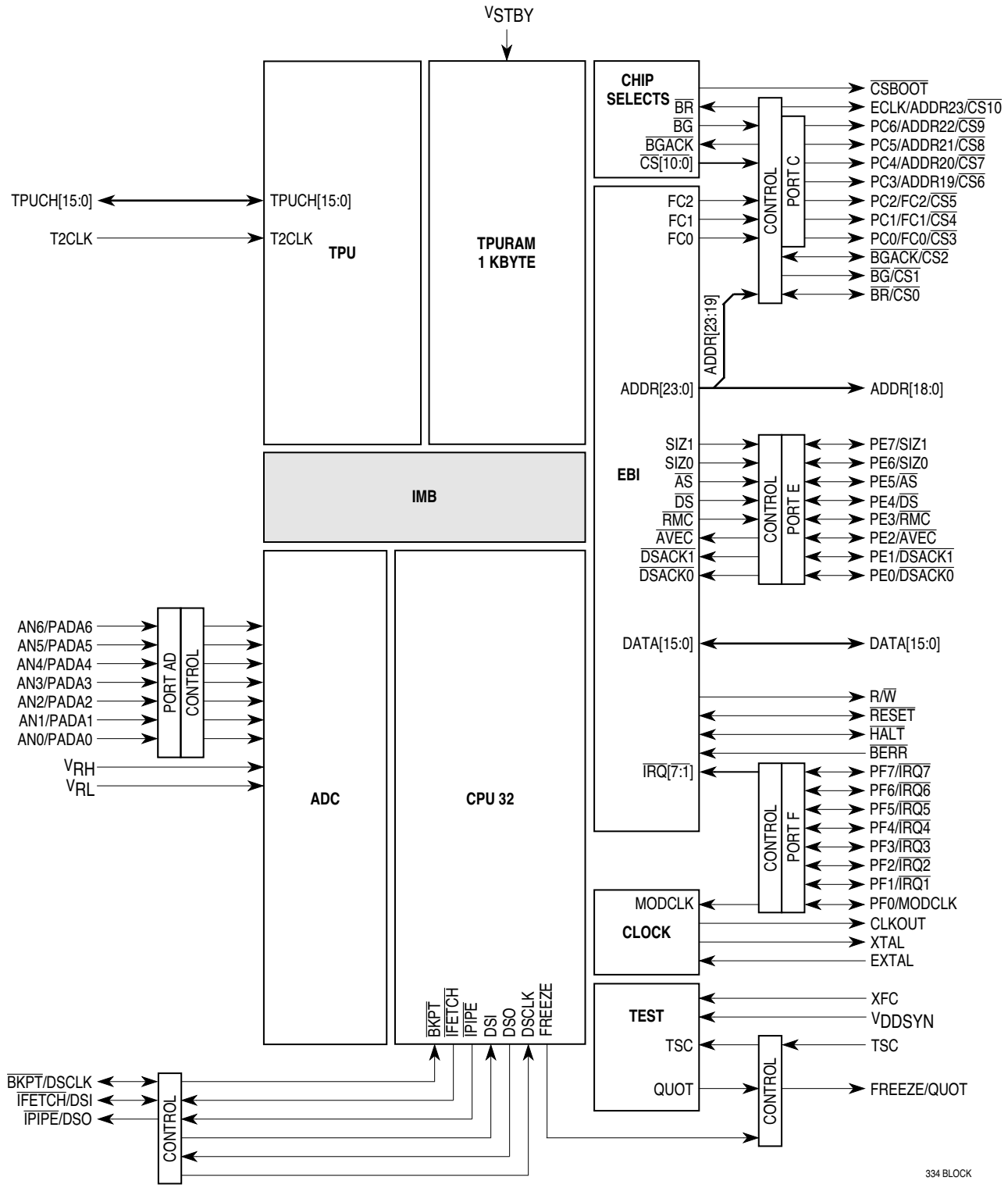
Figure 1 MC68334 Block Diagram

1.3 Pin Assignments
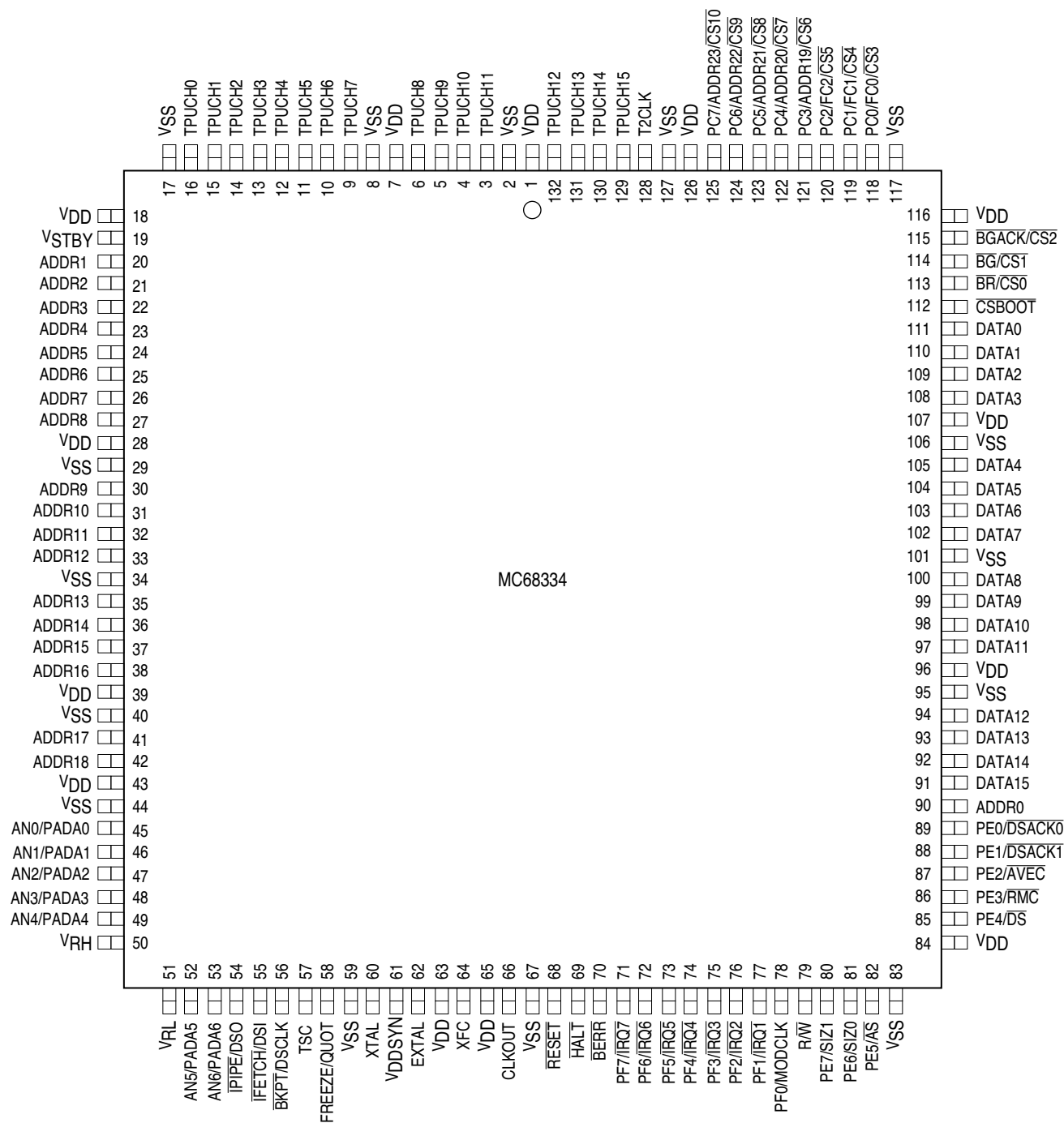
Figure 2 MC68334 132-Pin QFP Pin Assignments

1.4 Address Map

The following figure is a map of the MCU internal addresses. The RAM array is positioned by the base address registers in the associated RAM control block. Unimplemented blocks are mapped externally.
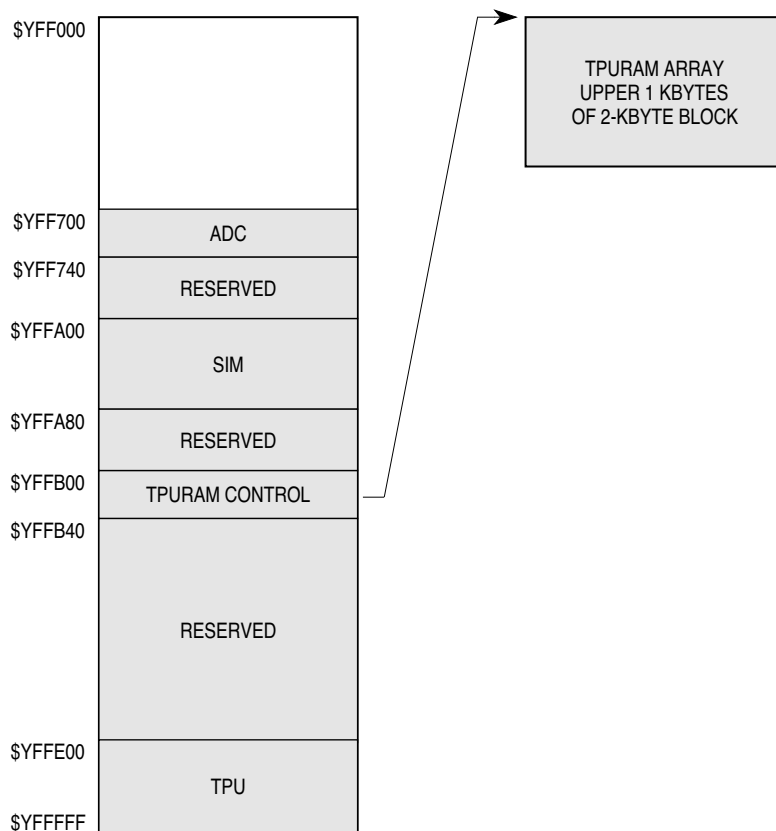
Y = M111, where M is the signal state of the module mapping (MM) bit in the SIM configuration register (Y = $7 or Y = $F).

334 ADDRESS MAP

Figure 3 MC68334 Address Map

1.5 Intermodule Bus

The intermodule bus (IMB) is a standardized bus developed to facilitate both design and operation of modular microcontrollers. It contains circuitry to support exception processing, address space partitioning, multiple interrupt levels, and vectored interrupts. The standardized modules in the MCU communicate with one another and with external components through the IMB. The IMB in the MCU uses 24 address lines and 16 data lines.

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 Characteristics

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
ADDR23/$\overline{CS10}$/ECLK	A	Y	N	—	—
ADDR[22:19]/$\overline{CS[9:6]}$	A	Y	N	O	PC[6:3]
ADDR[18:0]	A	Y	N	—	—
AN[6:0]	—	Y[1]	Y	I	PADA[6:0]
$\overline{AS}$	B	Y	N	I/O	PE5
$\overline{AVEC}$	B	Y	N	I/O	PE2
$\overline{BERR}$[2]	B	Y	N	—	—
$\overline{BG}$/$\overline{CS1}$	B	—	—	—	—
$\overline{BGACK}$/$\overline{CS2}$	B	Y	N	—	—
$\overline{BKPT}$/DSCLK	—	Y	Y	—	—
$\overline{BR}$/$\overline{CS0}$	B	Y	N	—	—
CLKOUT	A	—	—	—	—
$\overline{CSBOOT}$	B	—	—	—	—
DATA[15:0][1]	Aw	Y	N	—	—
$\overline{DS}$	B	Y	N	I/O	PE4
$\overline{DSACK1}$	B	Y	N	I/O	PE1
$\overline{DSACK0}$	B	Y	N	I/O	PE0
DSI/$\overline{IFETCH}$	A	Y	Y	—	—
DSO/$\overline{IPIPE}$	A	—	—	—	—
EXTAL[3]	—	—	—	—	—
FC[2:0]/$\overline{CS[5:3]}$	A	Y	—	O	PC[2:0]
FREEZE/QUOT	A	—	—	—	—
$\overline{HALT}$[2]	Bo	Y	N	—	—
$\overline{IRQ[7:1]}$	B	Y	Y	I/O	PF[7:1]
MODCLK[1]	B	Y	N	I/O	PF0
R/$\overline{W}$	A	Y	N	—	—
$\overline{RESET}$	Bo	Y	Y	—	—
$\overline{RMC}$	B	Y	N	I/O	PE3
SIZ[1:0]	B	Y	N	I/O	PE[7:6]
TPUCH[15:0]	A	Y	Y	—	—
TSC	—	Y	Y	—	—
T2CLK	A	Y	Y	—	—
V$_{RH}$[4]	—	—	—	—	—
V$_{RL}$[4]	—	—	—	—	—
XFC[3]	—	—	—	—	—
XTAL[3]	—	—	—	—	—
R/$\overline{W}$	A	Y	N	—	—

Table 2 MCU Pin Characteristics

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
$\overline{\text{RESET}}$	Bo	Y	Y	—	—
$\overline{\text{RMC}}$	B	Y	N	I/O	PE3
SIZ[1:0]	B	Y	N	I/O	PE[7:6]
TPUCH[15:0]	A	Y	Y	—	—
TSC	—	Y	Y	—	—
T2CLK	A	Y	Y	—	—
V_{RH}[4]	—	—	—	—	—
V_{RL}[4]	—	—	—	—	—
XFC[3]	—	—	—	—	—
XTAL[3]	—	—	—	—	—

NOTES
1. DATA[15:0] are synchronized during reset only. MODCLK and ADC pins are synchronized only when used as input port pins.
2. $\overline{\text{BERR}}$, HALT only synchronized if late $\overline{\text{BERR}}$ or HALT.
3. EXTAL, XFC, and XTAL are clock reference connections.
4. V_{RH} and V_{RL} are ADC reference voltage points.

2.2 MCU Power Connections

Table 3 MCU Power Connections

Pin	Description
V_{STBY}	Standby RAM Power/Clock Synthesizer Power
V_{DDSYN}	Clock Synthesizer Power
V_{DDA}/V_{SSA}	A/D Converter Power
V_{RH}/V_{RL}	A/D Reference Voltage
V_{SSE}/V_{DDE}	External Periphery Power (Source and Drain)
V_{SSI}/V_{DDI}	Internal Module Power (Source and Drain)

2.3 MCU Driver Types

Table 4 MCU Driver Types

Type	I/O	Description
A	O	Output-only signals that are always driven. No external pull-up required.
Aw	O	Type A output with weak P-channel pull-up during reset.
B	O	Three-state output that includes circuitry to pull up asserted output before high impedance is established, to ensure rapid rise time. An external holding resistor is required to maintain logic level while in the high-impedance state. Pins with this type of driver may only go into high-impedance state under certain conditions. The TSC signal can put all pins with this type of driver in high-impedance state.
Bo	O	Type B output that can be operated in an open-drain mode.

2.4 Signal Characteristics

Table 5 MCU Signal Characteristics

Signal Name	MCU Module	Signal Type	Active State
ADDR[23:0]	SIM	Bus	1/0
AN[6:0]	ADC	Input	—
$\overline{AS}$	SIM	Output	0
$\overline{AVEC}$	SIM	Input	0
$\overline{BERR}$	SIM	Input	0
$\overline{BG}$	SIM	Output	0
$\overline{BGACK}$	SIM	Input	0
$\overline{BKPT}$	CPU32	Input	0
$\overline{BR}$	SIM	Input	0
CLKOUT	SIM	Output	—
$\overline{CS[10:0]}$	SIM	Output	0
$\overline{CSBOOT}$	SIM	Output	0
DATA[15:0]	SIM	Bus	1/0
$\overline{DS}$	SIM	Output	0
$\overline{DSACK[1:0]}$	SIM	Input	0
DSCLK	CPU32	Input	Serial Clock
DSI	CPU32	Input	Serial Data
DSO	CPU32	Output	Serial Data
ECLK	SIM	Output	—
EXTAL	SIM	Input	—
FC[2:0]	SIM	Output	1/0
FREEZE	SIM	Output	1
$\overline{HALT}$	SIM	Input/Output	0
$\overline{IFETCH}$	CPU32	Output	0
$\overline{IPIPE}$	CPU32	Output	0
$\overline{IRQ[7:1]}$	SIM	Input	0
MODCLK	SIM	Input	—
PADA[6:0]	ADC	Input	(Port)
PC[6:0]	SIM	Output	(Port)
PE[7:0]	SIM	Input/Output	(Port)
PF[7:0]	SIM	Input/Output	(Port)
QUOT	SIM	Output	—
R/$\overline{W}$	SIM	Output	1/0
$\overline{RESET}$	SIM	Input/Output	0
$\overline{RMC}$	SIM	Input/Output	0
SIZ[1:0]	SIM	Output	1
TPUCH[15:0]	TPU	Input/Output	—
TSC	SIM	Input	1
T2CLK	TPU	Input	—
V_{RH}	ADC	Input	—
V_{RL}	ADC	Input	—
XFC	SIM	Input	—
XTAL	SIM	Output	—

2.5 Signal Function

Table 6 MCU Signal Function

Mnemonic	Signal Name	Function
ADDR[23:0]	Address Bus	24-bit address bus used by CPU32
AN[6:0]	ADC Analog Input	Inputs to ADC multiplexer
$\overline{AS}$	Address Strobe	Indicates that a valid address is on the address bus
$\overline{AVEC}$	Autovector	Requests an automatic vector during interrupt acknowledge
$\overline{BERR}$	Bus Error	Indicates that a bus error has occurred
$\overline{BG}$	Bus Grant	Indicates that the MCU has relinquished the bus
$\overline{BGACK}$	Bus Grant Acknowledge	Indicates that an external device has assumed bus mastership
$\overline{BKPT}$	Breakpoint	Signals a hardware breakpoint to the CPU
$\overline{BR}$	Bus Request	Indicates that an external device requires bus mastership
CLKOUT	System Clockout	System clock output
$\overline{CS[10:0]}$	Chip Selects	Select external devices at programmed addresses
$\overline{CSBOOT}$	Boot Chip Select	Chip select for external boot start-up ROM
DATA[15:0]	Data Bus	16-bit data bus
$\overline{DS}$	Data Strobe	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.
$\overline{DSACK[1:0]}$	Data and Size Acknowledge	Provide asynchronous data transfers and dynamic bus sizing
DSI, DSO, DSCLK	Development Serial In, Out, Clock	Serial I/O and clock for background debugging mode
ECLK	E-Clock	External M6800 bus clock output
EXTAL, XTAL	Crystal Oscillator	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used
FC[2:0]	Function Codes	Identify processor state and current address space
FREEZE	Freeze	Indicates that the CPU has entered background mode
$\overline{HALT}$	Halt	Suspend external bus activity
$\overline{IFETCH}$	Instruction Fetch	Identifies bus cycles in which operand is loaded into pipeline
$\overline{IPIPE}$	Instruction Pipeline	Indicates instruction pipeline activity
$\overline{IRQ[7:1]}$	Interrupt Request Level	Provide prioritized interrupts to the CPU
MODCLK	Clock Mode Select	Selects the source and type of system clock
PADA[6:0]	Port ADA	ADC digital input port signals
PC[6:0]	Port C	SIM digital output port signals
PE[7:0]	Port E	SIM digital I/O port signals
PF[7:0]	Port F	SIM digital I/O port signals
QUOT	Quotient Out	Provides the quotient bit of the polynomial divider
$\overline{RESET}$	Reset	System reset
$\overline{RMC}$	Read-Modify-Write Cycle	Indicates an indivisible read-modify-write instruction
R/$\overline{W}$	Read/Write	Indicates the direction of data transfer on the bus
SIZ[1:0]	Size	Indicates the number of bytes to be transferred during a bus cycle
TPUCH[15:0]	TPU I/O Channels	Bidirectional TPU channels
TSC	Three-State Control	Places all output drivers in a high-impedance state
T2CLK	TCR2 Clock	TPU clock input
$V_{RH, VRL}$	ADC Reference Voltage	Provide precise reference for A/D conversion
XFC	External Filter Capacitor	Connection for external phase-locked loop filter capacitor

3 System Integration Module

The MCU system integration module (SIM) consists of five functional blocks that control system start-up, initialization, configuration, and external bus.
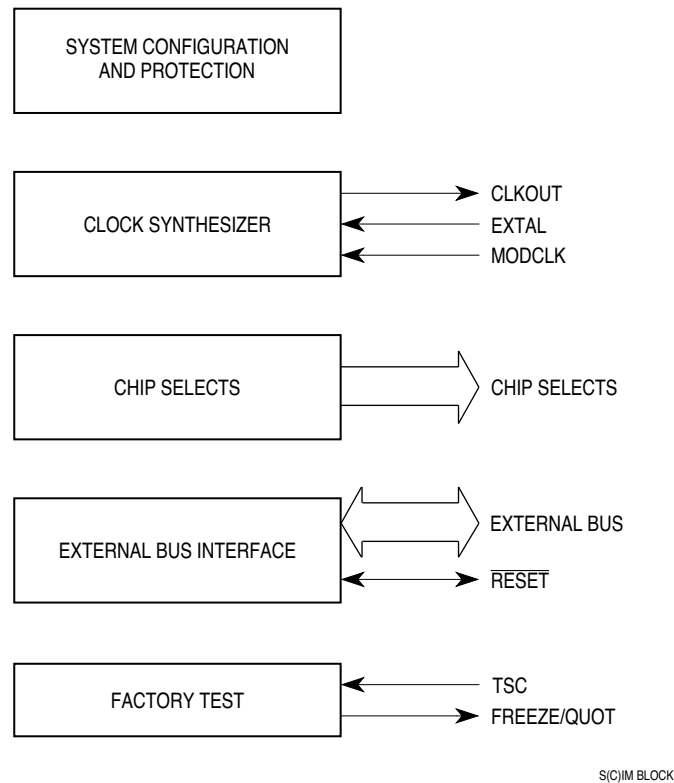
```
┌─────────────────────────┐
│  SYSTEM CONFIGURATION    │
│      AND PROTECTION      │
└─────────────────────────┘

┌─────────────────────────┐
│                         │────────▶ CLKOUT
│    CLOCK SYNTHESIZER     │◀──────── EXTAL
│                         │◀──────── MODCLK
└─────────────────────────┘

┌─────────────────────────┐
│      CHIP SELECTS        │══════▷ CHIP SELECTS
└─────────────────────────┘

┌─────────────────────────┐
│                         │◁═════▷ EXTERNAL BUS
│  EXTERNAL BUS INTERFACE  │
│                         │◀──────▶ RESET
└─────────────────────────┘

┌─────────────────────────┐
│                         │◀──────── TSC
│      FACTORY TEST        │────────▶ FREEZE/QUOT
└─────────────────────────┘

                                    S(C)IM BLOCK
```

Figure 4 SIM Block Diagram

3.1 Overview

The system configuration and protection block controls MCU configuration and operating mode. The block also provides bus and software watchdog monitors.

The system clock generates clock signals used by the SIM, other IMB modules, and external devices. In addition, a periodic interrupt generator supports execution of time-critical control routines.

The external bus interface handles the transfer of information between IMB modules and external address space.

The chip-select block provides eleven general-purpose chip-select signals and a boot ROM chip select signal. Both general-purpose and boot ROM chip-select signals have associated base address registers and option registers.

The system test block incorporates hardware necessary for testing the MCU. It is used to perform factory tests, and its use in normal applications is not supported.

The SIM control register address map occupies 128 bytes. Unused registers within the 128-byte address space return zeros when read. The "Access" column in the SIM address map below indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the SIMCR.

Table 7 SIM Address Map

Access	Address	15 8	7 0
S	$YFFA00	SIM MODULE CONFIGURATION REGISTER (SIMCR)	
S	$YFFA02	FACTORY TEST REGISTER (SIMTR)	
S	$YFFA04	CLOCK SYNTHESIZER CONTROL REGISTER (SYNCR)	
S	$YFFA06	NOT USED	RESET STATUS REGISTER (RSR)
S	$YFFA08	MODULE TEST E (SIMTRE)	
S	$YFFA0A	NOT USED	NOT USED
S	$YFFA0C	NOT USED	NOT USED
S	$YFFA0E	NOT USED	NOT USED
S/U	$YFFA10	NOT USED	PORT E DATA (PORTE0)
S/U	$YFFA12	NOT USED	PORT E DATA (PORTE1)
S/U	$YFFA14	NOT USED	PORT E DATA DIRECTION (DDRE)
S	$YFFA16	NOT USED	PORT E PIN ASSIGNMENT (PEPAR)
S/U	$YFFA18	NOT USED	PORT F DATA (PORTF0)
S/U	$YFFA1A	NOT USED	PORT F DATA (PORTF1)
S/U	$YFFA1C	NOT USED	PORT F DATA DIRECTION (DDRF)
S	$YFFA1E	NOT USED	PORT F PIN ASSIGNMENT (PFPAR)
S	$YFFA20	NOT USED	SYSTEM PROTECTION CONTROL (SYPCR)
S	$YFFA22	PERIODIC INTERRUPT CONTROL REGISTER (PICR)	
S	$YFFA24	PERIODIC INTERRUPT TIMING REGISTER (PITR)	
S	$YFFA26	NOT USED	SOFTWARE SERVICE (SWSR)
S	$YFFA28	NOT USED	NOT USED
S	$YFFA2A	NOT USED	NOT USED
S	$YFFA2C	NOT USED	NOT USED
S	$YFFA2E	NOT USED	NOT USED
S	$YFFA30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
S	$YFFA32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
S	$YFFA34	TEST MODULE SHIFT COUNT (TSTSC)	
S	$YFFA36	TEST MODULE REPETITION COUNTER (TSTRC)	
S	$YFFA38	TEST MODULE CONTROL (CREG)	
S/U	$YFFA3A	TEST MODULE DISTRIBUTED REGISTER (DREG)	
	$YFFA3C	NOT USED	NOT USED
	$YFFA3E	NOT USED	NOT USED
S/U	$YFFA40	NOT USED	PORT C DATA (PORTC)
	$YFFA42	NOT USED	NOT USED
S	$YFFA44	CHIP-SELECT PIN ASSIGNMENT (CSPAR0)	
S	$YFFA46	CHIP-SELECT PIN ASSIGNMENT (CSPAR1)	
S	$YFFA48	CHIP-SELECT BASE BOOT (CSBARBT)	
S	$YFFA4A	CHIP-SELECT OPTION BOOT (CSORBT)	
S	$YFFA4C	CHIP-SELECT BASE 0 (CSBAR0)	
S	$YFFA4E	CHIP-SELECT OPTION 0 (CSOR0)	
S	$YFFA50	CHIP-SELECT BASE 1 (CSBAR1)	
S	$YFFA52	CHIP-SELECT OPTION 1 (CSOR1)	
S	$YFFA54	CHIP-SELECT BASE 2 (CSBAR2)	
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)	

Table 7 SIM Address Map

Access	Address	15 8 7 0
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)

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

3.2 System Configuration and Protection Block

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.
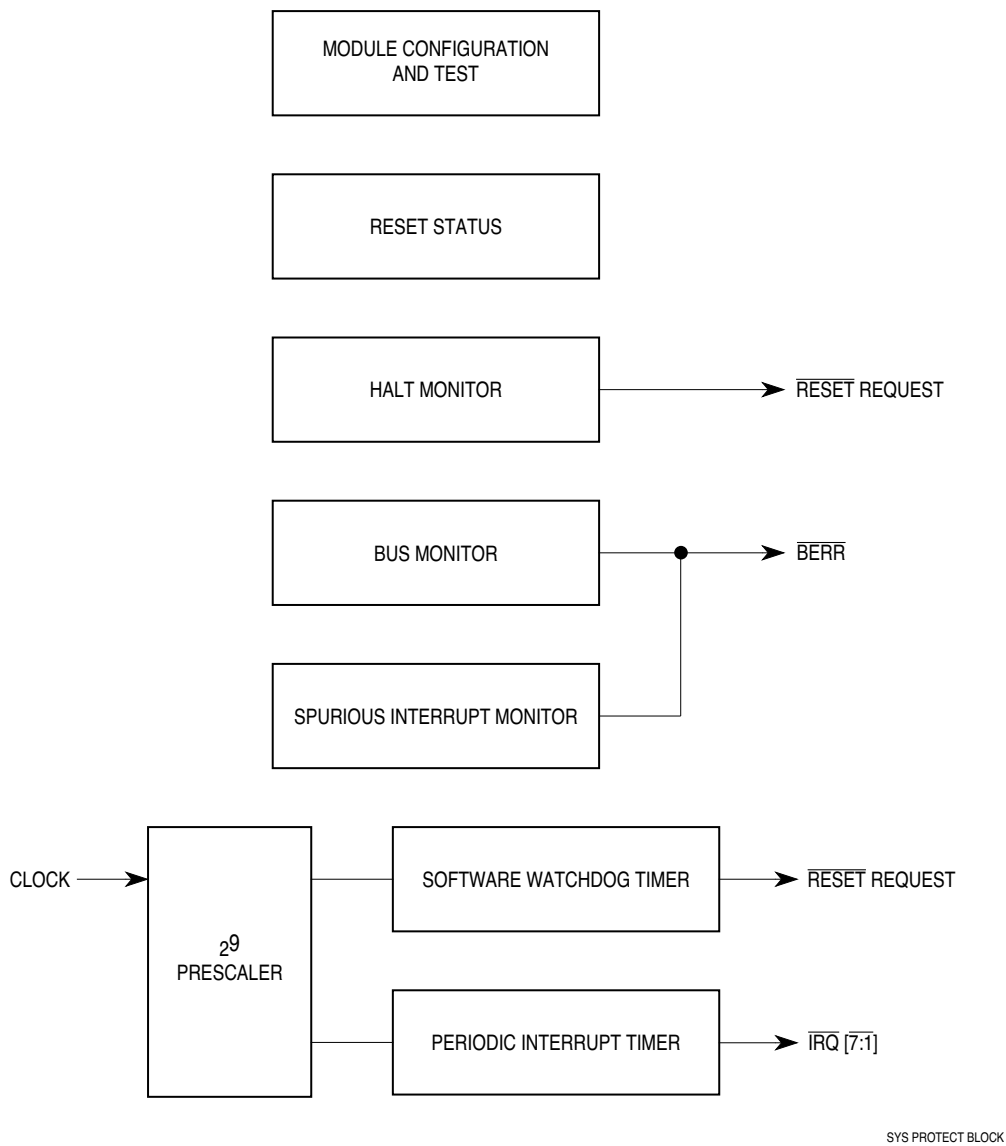
Figure 5 System Configuration and Protection Block

3.2.1 System Configuration

The SIM controls MCU configuration during normal operation and during internal testing.

SIMCR — SIM Configuration Register $YFFA00

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXOFF	FRZSW	FRZBM	0	SLVEN	0	SHEN		SUPV	MM	0	0	IARB			

RESET:

0	0	0	0	DATA11	0	0	0	1	1	0	0	1	1	1	1

The SIM configuration register controls system configuration. It can be read or written at any time, except for the module mapping (MM) bit, which can be written only once.

EXOFF — External Clock Off
>   0 = The CLKOUT pin is driven from an internal clock source.
>   1 = The CLKOUT pin is placed in a high-impedance state.

FRZSW — Freeze Software Enable
>   0 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters continue to run.
>   1 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters are disabled, preventing interrupts during software debug.

FRZBM — Freeze Bus Monitor Enable
>   0 = When FREEZE is asserted, the bus monitor continues to operate.
>   1 = When FREEZE is asserted, the bus monitor is disabled.

SLVEN — Factory Test Mode Enabled
>   This bit is a read-only status bit that reflects the state of DATA11 during reset.
>   0 = IMB is not available to an external master.
>   1 = An external bus master has direct access to the IMB.

SHEN[1:0] — Show Cycle Enable
>   This field determines what the EBI does with the external bus during internal transfer operations. A show cycle allows internal transfers to be externally monitored. The table below shows whether show cycle data is driven externally, and whether external bus arbitration can occur. To prevent bus conflict, external peripherals must not be enabled during show cycles.

SHEN	Action
00	Show cycles disabled, external arbitration enabled
01	Show cycles enabled, external arbitration disabled
10	Show cycles enabled, external arbitration enabled
11	Show cycles enabled, external arbitration enabled, internal activity is halted by a bus grant

SUPV — Supervisor/Unrestricted Data Space
>   The SUPV bit places the SIM global registers in either supervisor or user data space.
>   0 = Registers with access controlled by the SUPV bit are accessible from either the user or supervisor privilege level.
>   1 = Registers with access controlled by the SUPV bit are restricted to supervisor access only.

MM — Module Mapping
>   0 = Internal modules are addressed from $7FF000 – $7FFFFF.
>   1 = Internal modules are addressed from $FFF000 – $FFFFFF.

IARB[3:0] — Interrupt Arbitration Field
>   Each module that can generate interrupt requests has an interrupt arbitration (IARB) field. Arbitration between interrupt requests of the same priority is performed by serial contention between IARB field bit values. Contention must take place whenever an interrupt request is acknowledged, even when there is only a single pending request. An IARB field must have a nonzero value for contention to take place. If an interrupt request from a module with an IARB field value of %0000 is recognized, the CPU processes a spurious interrupt exception. Because the SIM routes external interrupt requests to the CPU, the SIM IARB field value is used for arbitration between internal and external interrupts of the same priority. The reset value of IARB for the SIM is %1111, and the reset IARB value for all other modules is %0000, which prevents SIM interrupts from being discarded during initialization.

3.2.2 System Protection Control Register

The system protection control register controls system monitor functions, software watchdog clock prescaling, and bus monitor timing. This register can be written only once following power-on or reset, but can be read at any time.

SYPCR — System Protection Control Register   **$YFFA21**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								SWE	SWP	SWT		HME	BME	BMT	

RESET:

								7	6	5	4	3	2	1	0
								1	$\overline{\text{MODCLK}}$	0	0	0	0	0	0

SWE — Software Watchdog Enable
    0 = Software watchdog disabled
    1 = Software watchdog enabled

SWP — Software Watchdog Prescale
  This bit controls the value of the software watchdog prescaler.
    0 = Software watchdog clock not prescaled
    1 = Software watchdog clock prescaled by 512

SWT[1:0] — Software Watchdog Timing
  This field selects the divide ratio used to establish software watchdog time-out period. The following table gives the ratio for each combination of SWP and SWT bits.

SWP	SWT	Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

HME — Halt Monitor Enable
    0 = Disable halt monitor function
    1 = Enable halt monitor function

BME — Bus Monitor External Enable
    0 = Disable bus monitor function for an internal to external bus cycle.
    1 = Enable bus monitor function for an internal to external bus cycle.

BMT[1:0] — Bus Monitor Timing
  This field selects a bus monitor time-out period as shown in the following table.

BMT	Bus Monitor Time-out Period
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

3.2.3 Bus Monitor

The internal bus monitor checks for excessively long $\overline{\text{DSACK}}$ response times during normal bus cycles and for excessively long $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ response times during interrupt acknowledge cycles. The monitor asserts $\overline{\text{BERR}}$ if response time is excessive.

$\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ response times are measured in clock cycles. The maximum allowable response time can be selected by setting the BMT field.

The monitor does not check $\overline{\text{DSACK}}$ response on the external bus unless the CPU initiates the bus cycle. The BME bit in the 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.

3.2.4 Halt Monitor

The halt monitor responds to an assertion of $\overline{\text{HALT}}$ on the internal bus. A flag in the reset status register (RSR) indicates that the last reset was caused by the halt monitor. The halt monitor reset can be inhibited by the HME bit in the SYPCR.

3.2.5 Spurious Interrupt Monitor

The spurious interrupt monitor issues $\overline{\text{BERR}}$ if no interrupt arbitration occurs during an interrupt-acknowledge cycle.

3.2.6 Software Watchdog

The software watchdog is controlled by SWE in the SYPCR. Once enabled, the watchdog requires that a service sequence be written to SWSR on a periodic basis. If servicing does not take place, the watchdog times out and issues a reset. This register can be written at any time, but returns zeros when read.

SWSR — Software Service Register $YFFA27

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	0	0	0	0	0	0	0

RESET:

| | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Register shown with read value

Perform a software watchdog service sequence as follows:

- Write $55 to SWSR.
- Write $AA to SWSR.

Both writes must occur before time-out in the order listed, but any number of instructions can be executed between the two writes.

The watchdog clock rate is affected by SWP and SWT in SYPCR. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new time-out period takes effect.

The reset value of SWP is affected by the state of the MODCLK pin on the rising edge of reset, as shown in the following table.

MODCLK	SWP
0	1
1	0

3.2.7 Periodic Interrupt Timer

The periodic interrupt timer (PIT) generates interrupts of specified priorities at specified intervals. Timing for the PIT is provided by a programmable prescaler driven by the system clock.

PICR — Periodic Interrupt Control Register $YFFA22

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	PIRQL			PIV							

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

This register contains information concerning periodic interrupt priority and vectoring. Bits [10:0] can be read or written at any time. Bits [15:11] are unimplemented and always return zero.

PIRQL[2:0] — Periodic Interrupt Request Level
The following table shows what interrupt request level is asserted when a periodic interrupt is generated. If a PIT interrupt and an external $\overline{IRQ}$ signal of the same priority occur simultaneously, the PIT interrupt is serviced first. The periodic timer continues to run when the interrupt is disabled.

PIRQL	Interrupt Request Level
000	Periodic Interrupt Disabled
001	Interrupt Request Level 1
010	Interrupt Request Level 2
011	Interrupt Request Level 3
100	Interrupt Request Level 4
101	Interrupt Request Level 5
110	Interrupt Request Level 6
111	Interrupt Request Level 7

PIV[7:0] — Periodic Interrupt Vector
The bits of this field contain the vector generated in response to an interrupt from the periodic timer. When the SIM responds, the periodic interrupt vector is placed on the bus.

PITR — Periodic Interrupt Timer Register $YFFA24

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM							

RESET:

0	0	0	0	0	0	0	$\overline{MODCLK}$	0	0	0	0	0	0	0	0

The PITR contains the count value for the periodic timer. A zero value turns off the periodic timer. This register can be read or written at any time.

PTP — Periodic Timer Prescaler Control
    0 = Periodic timer clock not prescaled
    1 = Periodic timer clock prescaled by a value of 512
The reset state of PTP is the complement of the state of the MODCLK signal during reset.

PITM[7:0] — Periodic Interrupt Timing Modulus Field
This is an 8-bit timing modulus. The period of the timer can be calculated as follows:

$$PIT\ Period\ =\ \frac{(PITM)(Prescale)(4)}{EXTAL\ Frequency}$$

where

PIT Period = Periodic interrupt timer period
PITM = Periodic interrupt timer register modulus (PITR[7:0])
EXTAL Frequency = Crystal frequency
Prescale = 512 or 1 depending on the state of the PTP bit in the PITR

3.3 System Clock

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

The system clock signal can be generated in one of three ways. An internal phase-locked loop can synthesize the clock from either an internal reference or an external reference, or the clock signal can be input from an external frequency source. Keep these clock sources in mind while reading the rest of this section. The figure below is a block diagram of the system clock.
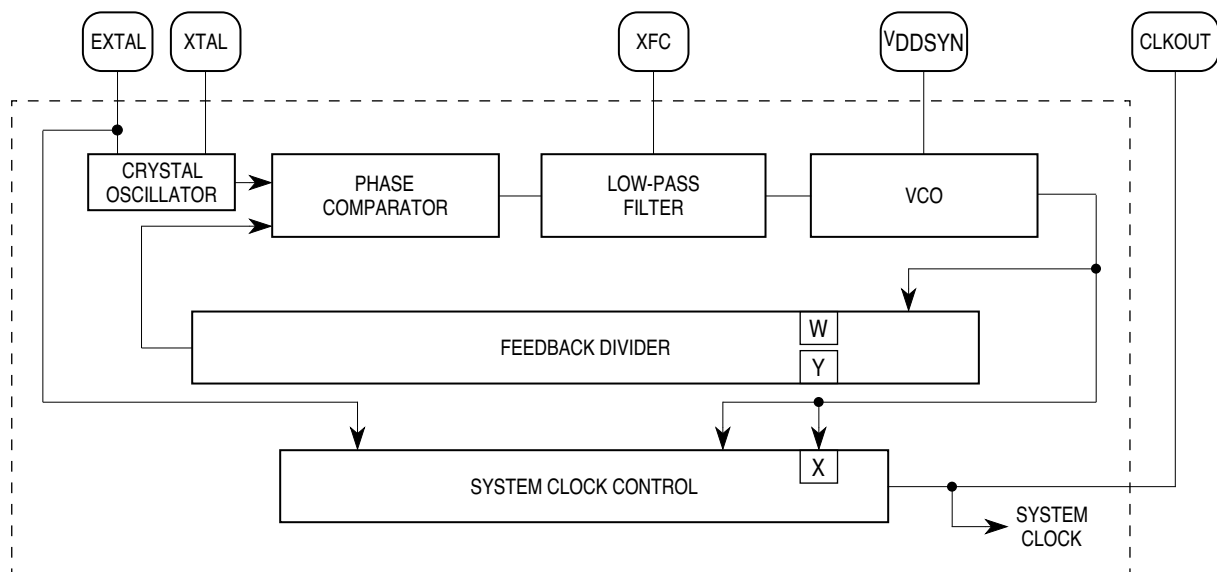

32 PLL BLOCK

Figure 6 System Clock Block Diagram

3.3.1 Clock Sources

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

To generate a reference frequency using the internal oscillator a reference crystal must be connected between the EXTAL and XTAL pins. The figure below shows a recommended circuit.
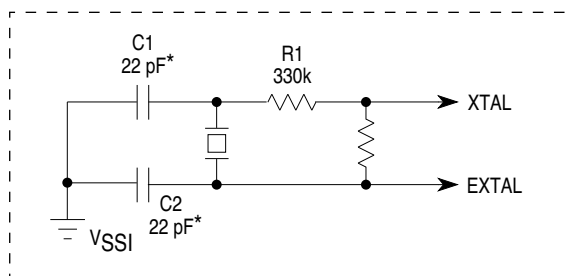
* RESISTANCE AND CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A DAISHINKU DMX-38 32.768-kHz CRYSTAL.
SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.

32 OSCILLATOR

Figure 7 System Clock Oscillator Circuit

If an external reference signal or an external system clock signal is applied via 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.

When an external system clock signal is applied (PLL disabled, MODCLK = 0 during reset), the duty cycle of the input is critical, especially at operating frequencies close to maximum. The relationship between clock signal duty cycle and clock signal period is expressed:

Minimum External Clock Period =

$$\frac{\text{Minimum External Clock High § Low Time}}{50\% - \text{Percentage Variation of External Clock Input Duty Cycle}}$$

3.3.2 Clock Synthesizer Operation

V_{DDSYN} is used to power the clock circuits when either an internal or an external reference frequency is applied. A separate power source increases MCU noise immunity and can be used to run the clock when the MCU is powered down. A quiet power supply must be used as the V_{DDSYN} source. Adequate external bypass capacitors should be placed as close as possible to the V_{DDSYN} pin to assure stable operating frequency. When an external system clock signal is applied and the PLL is disabled, V_{DDSYN} should be connected to the V_{DD} supply. Refer to the *SIM Reference Manual* (SIMRM/AD) for more information regarding system clock power supply conditioning.

A voltage controlled oscillator (VCO) generates the system clock signal. To maintain a 50% clock duty cycle, VCO frequency is either two or four times system clock frequency, depending on the state of the X bit in SYNCR. A portion of the clock signal is fed back to a divider/counter. The divider controls the frequency of one input to a phase comparator. The other phase comparator input is a reference signal, either from the crystal oscillator or from an external source. The comparator generates a control signal proportional to the difference in phase between the two inputs. The signal is low-pass filtered and used to correct VCO output frequency.

Filter geometry can vary, depending upon the external environment and required clock stability. The figure below shows two recommended filters. XFC pin leakage must be kept within specified limits to maintain optimum stability and PLL performance.

An external filter network connected to the XFC pin is not required when an external system clock signal is applied and the PLL is disabled. The XFC pin must be left floating in this case.

NORMAL OPERATING ENVIRONMENT   **HIGH-STABILITY OPERATING ENVIRONMENT**
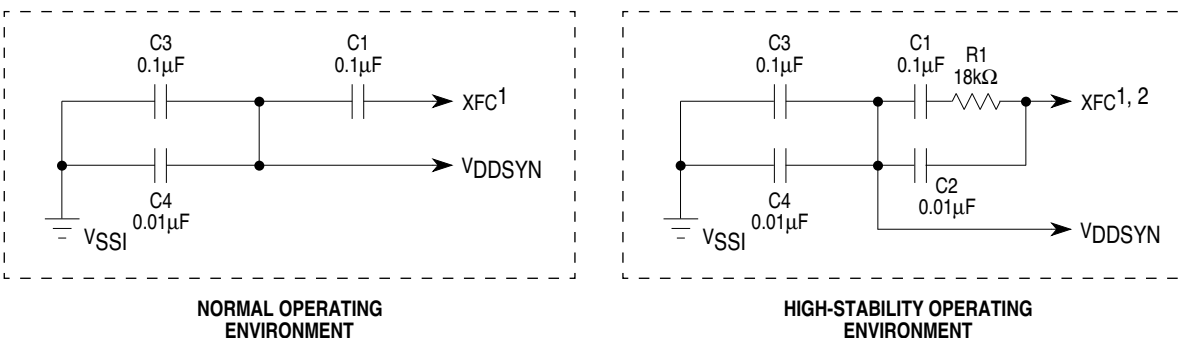
1. MAINTAIN LOW LEAKAGE ON THE XFC NODE.
2. RECOMMENDED LOOP FILTER FOR REDUCED SENSITIVITY TO LOW-FREQUENCY NOISE.

16/32 XFC CONN

Figure 8 System Clock Filter Networks

The synthesizer locks when VCO frequency is equal to EXTAL frequency. Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever comparator input changes, the synthesizer must relock. Lock status is shown by the SLOCK bit in SYNCR. During power-up, the MCU does not come out of reset state until the synthesizer locks. Crystal type, characteristic frequency, and layout of external oscillator circuitry affect lock time.

When the clock synthesizer is used, control register SYNCR determines operating frequency and various modes of operation. The SYNCR W bit controls a three-bit prescaler in the feedback divider. Setting W increases VCO speed by a factor of four. The SYNCR Y field determines the count modulus for a modulo 64 down counter, causing it to divide by a value of Y + 1. When W or Y values change, VCO frequency changes, and there is a VCO relock delay. The SYNCR X bit controls a divide-by-two circuit that is not in the synthesizer feedback loop. When X = 0 (reset state), the divider is enabled, and system clock frequency is one-fourth VCO frequency; setting X disables the divider, doubling clock speed without changing VCO speed. There is no relock delay when clock speed is changed by the X bit.

Clock frequency is determined by SYNCR bit settings as follows:

$$F_{SYSTEM} = F_{REFERENCE}[4(Y + 1)(2^{2W + X})]$$

The reset state of SYNCR ($3F00) produces a modulus-64 count.

For the device to perform correctly, system clock and VCO frequencies selected by the W, X, and Y bits must be within the limits specified for the MCU. Do not use a combination of bit values that selects either an operating frequency or a VCO frequency greater than the maximum specified values.

3.3.3 External Bus Clock

The state of the external clock division bit (EDIV) in SYNCR determines clock rate for the external bus clock signal (ECLK) available on pin ADDR23. ECLK is a bus clock for MC6800 devices and peripherals. ECLK frequency can be set to system clock frequency divided by eight or system clock frequency divided by sixteen. The clock is enabled by the $\overline{CS10}$ field in chip select pin assignment register 1 (CSPAR1). ECLK operation during low-power stop is described in the following paragraph. Refer to **3.5 Chip Selects** for more information about the external bus clock.

3.3.4 Low-Power Operation

Low-power operation is initiated by the CPU32. To reduce power consumption selectively, the CPU can set the STOP bits in each module configuration register. To minimize overall microcontroller power consumption, the CPU can execute the LPSTOP instruction, which causes the SIM to turn off the system clock.

When individual module STOP bits are set, clock signals inside each module are turned off, but module registers are still accessible.

When the CPU executes LPSTOP, a special CPU space bus cycle writes a copy of the current interrupt mask into the clock control logic. The SIM brings the MCU out of low-power operation when either an interrupt of higher priority than the stored mask or a reset occurs.

During a low-power stop, unless the system clock signal is supplied by an external source and that source is removed, the SIM clock control logic and the SIM clock signal (SIMCLK) continue to operate. The periodic interrupt timer and input logic for the $\overline{RESET}$ and $\overline{IRQ}$ pins are clocked by SIMCLK. The SIM can also continue to generate the CLKOUT signal while in low-power mode.

The stop mode system integration module clock (STSIM) and stop mode external clock (STEXT) bits in SYNCR determine clock operation during low-power stop. The table below summarizes the effects of STSIM and STEXT. MODCLK value is the logic level on the MODCLK pin during the last reset before LPSTOP execution. Any clock in the off state is held low. If the synthesizer VCO is turned off during LPSTOP, there is a PLL relock delay after the VCO is turned back on.

Table 8 Clock Control

Mode	Pins		SYNCR Bits		Clock Status		
LPSTOP	MODCLK	EXTAL	STSIM	STEXT	SIMCLK	CLKOUT	ECLK
No	0	External Clock	X	X	External Clock	External Clock	External Clock
Yes	0	External Clock	0	0	External Clock	Off	Off
Yes	0	External Clock	0	1	External Clock	External Clock	External Clock
Yes	0	External Clock	1	0	External Clock	Off	Off
Yes	0	External Clock	1	1	External Clock	External Clock	External Clock
No	1	Crystal or Reference	X	X	VCO	VCO	VCO
Yes	1	Crystal or Reference	0	0	Crystal or Reference	Off	Off
Yes	1	Crystal or Reference	0	1	Crystal or Reference	Crystal/ Reference	Off
Yes	1	Crystal or Reference	1	0	VCO	Off	Off
Yes	1	Crystal or Reference	1	1	VCO	VCO	VCO

3.3.5 Loss of Reference Signal

The state of the reset enable (RSTEN) bit in SYNCR determines what happens when clock logic detects a reference failure.

When RSTEN is cleared (default state out of reset), the clock synthesizer is forced into an operating condition referred to as limp mode. Limp mode frequency varies from device to device, but maxi-

mum limp frequency does not exceed one half maximum system clock when X = 0, or maximum system clock frequency when X = 1.

When RSTEN is set, the SIM resets the MCU.

The limp status bit (SLIMP) in SYNCR indicates whether the synthesizer has a reference signal. It is set when a reference failure is detected.

3.3.6 Clock Control

The clock control circuits determine system clock frequency and clock operation under special circumstances, such as following loss of synthesizer reference or during low-power operation. Clock source is determined by the logic state of the MODCLK pin during reset.

SYNCR — Clock Synthesizer Control Register $YFFA04

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	X				Y			EDIV	0	0	SLIMP	SLOCK	RSTEN	STSIM	STEXT

RESET:

| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | U | U | 0 | 0 | 0 |

When the on-chip clock synthesizer is used, system clock frequency is controlled by the bits in the upper byte of SYNCR. Bits in the lower byte show status of or control operation of internal and external clocks. The SYNCR can be read or written only when the CPU is operating at the supervisor privilege level.

W — Frequency Control (VCO)
This bit controls a prescaler tap in the synthesizer feedback loop. Setting the bit increases the VCO speed by a factor of four. VCO relock delay is required.

X — Frequency Control Bit (Prescale)
This bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting the bit doubles clock speed without changing the VCO speed. There is no VCO relock delay.

Y[5:0] — Frequency Control (Counter)
The Y field controls the modulus down counter in the synthesizer feedback loop, causing it to divide by a value of Y + 1. Values range from 0 to 63. VCO relock delay is required.

EDIV — E Clock Divide Rate
    0 = ECLK frequency is system clock divided by 8.
    1 = ECLK frequency is system clock divided by 16.
ECLK is an external M6800 bus clock available on pin ADDR23. Refer to **3.5 Chip Selects** for more information.

SLIMP — Limp Mode Flag
    0 = External crystal is VCO reference.
    1 = Loss of crystal reference.
When the on-chip synthesizer is used, loss of reference frequency causes SLIMP to be set. The VCO continues to run using the base control voltage. Maximum limp frequency is maximum specified system clock frequency. X-bit state affects limp frequency.

SLOCK — Synthesizer Lock Flag
    0 = VCO is enabled, but has not locked.
    1 = VCO has locked on the desired frequency (or system clock is external).
The MCU maintains reset state until the synthesizer locks, but SLOCK does not indicate synthesizer lock status until after the user writes to SYNCR.

RSTEN — Reset Enable
        0 = Loss of crystal causes the MCU to operate in limp mode.
        1 = Loss of crystal causes system reset.

STSIM — Stop Mode SIM Clock
        0 = When LPSTOP is executed, the SIM clock is driven from the crystal oscillator and the VCO is
            turned off to conserve power.
        1 = When LPSTOP is executed, the SIM clock is driven from the VCO.

STEXT — Stop Mode External Clock
        0 = When LPSTOP is executed, the CLKOUT signal is held negated to conserve power.
        1 = When LPSTOP is executed, the CLKOUT signal is driven from the SIM clock, as determined by
            the state of the STSIM bit.

3.4 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external de-
vices. The external bus has 24 address lines and 16 data lines.

The EBI provides dynamic sizing between 8-bit and 16-bit data accesses. It supports byte, word, and
long-word transfers. Ports are accessed through the use of asynchronous cycles controlled by the data
transfer (SIZ1 and SIZ0) and data size acknowledge pins ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$). Multiple bus cycles
may be required for a transfer to or from an 8-bit port.

Port width is the maximum number of bits accepted or provided during a bus transfer. External devices
must follow the handshake protocol described below. Control signals indicate the beginning of the cycle,
the address space, the size of the transfer, and the type of cycle. The selected device controls the length
of the cycle. Strobe signals, one for the address bus and another for the data bus, indicate the validity
of an address and provide timing information for data. The EBI operates in an asynchronous mode for
any port width.

To add flexibility and minimize the necessity for external logic, MCU chip-select logic can be synchro-
nized with EBI transfers. Chip-select logic can also provide internally-generated bus control signals for
these accesses. Refer to **3.5 Chip Selects** for more information.

3.4.1 Bus Control Signals

The CPU initiates a bus cycle by driving the address, size, function code, and read/write outputs. At the
beginning of the cycle, size signals SIZ0 and SIZ1 are driven along with the function code signals. The
size signals indicate the number of bytes remaining to be transferred during an operand cycle. They are
valid while the address strobe ($\overline{\text{AS}}$) is asserted. The following table shows SIZ0 and SIZ1 encoding. The
read/write (R/$\overline{\text{W}}$) signal determines the direction of the transfer during a bus cycle. This signal changes
state, when required, at the beginning of a bus cycle, and is valid while $\overline{\text{AS}}$ is asserted. R/$\overline{\text{W}}$ only chang-
es state when a write cycle is preceded by a read cycle or vice versa. The signal can remain low for two
consecutive write cycles.

Table 9 Size Signal Encoding

SIZ1	SIZ0	Transfer Size
0	1	Byte
1	0	Word
1	1	3 Byte
0	0	Long Word

3.4.2 Function Codes

The CPU32 automatically generates function code signals FC[2:0]. The function codes can be considered address extensions that automatically select one of eight address spaces to which an address applies. These spaces are designated as either user or supervisor, and program or data spaces. Address space seven is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid while $\overline{AS}$ is asserted.

Table 10 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 $\overline{AS}$ is asserted.

3.4.4 Address Strobe

$\overline{AS}$ 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, nonmultiplexed 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 $\overline{AS}$ is asserted in a write cycle.

3.4.6 Data Strobe

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

3.4.7 Bus Cycle Termination Signals

During bus cycles, external devices assert the data transfer and size acknowledge signals ($\overline{DSACK1}$ and $\overline{DSACK0}$). 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 ($\overline{BERR}$) signal is also a bus cycle termination indicator and can be used in the absence of $\overline{DSACK1}$ and $\overline{DSACK0}$ 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 $\overline{BERR}$ signal for internal and internal-to-external transfers. When $\overline{BERR}$ and $\overline{HALT}$ 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.

3.4.8 Data Transfer Mechanism

The MCU architecture supports byte, word, and long-word operands, allowing access to 8- and 16-bit data ports through the use of asynchronous cycles controlled by the data transfer and size acknowledge inputs ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$).

3.4.9 Dynamic Bus Sizing

The MCU dynamically interprets the port size of the addressed device during each bus cycle, allowing operand transfers to or from 8- and 16-bit ports. During an operand transfer cycle, the slave device signals its port size and indicates completion of the bus cycle to the MCU through the use of the $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ inputs, as shown in the following table.

Table 11 Effect of $\overline{\text{DSACK}}$ Signals

$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	Result
1	1	Insert Wait States in Current Bus Cycle
1	0	Complete Cycle — Data Bus Port Size is 8 Bits
0	1	Complete Cycle — Data Bus Port Size is 16 Bits
0	0	Reserved

For example, if the MCU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the 16 bits of valid data and then runs another bus cycle to obtain the other 16 bits. The operation for an 8-bit port is similar, but requires four read cycles. The addressed device uses the $\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$ signals to indicate the port width. For instance, a 16-bit device always returns $\overline{\text{DSACK0}}$ = 1 and $\overline{\text{DSACK1}}$ = 0 for a 16-bit port, regardless of whether the bus cycle is a byte or word operation.

Dynamic bus sizing requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 16-bit port must reside on data bus bits [15:0] and an 8-bit port must reside on data bus bits [15:8]. This minimizes the number of bus cycles needed to transfer data and ensures that the MCU transfers valid data.

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins. Operand bytes are designated as shown in the following figure. OP0 is the most significant byte of a long-word operand, and OP3 is the least significant byte. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

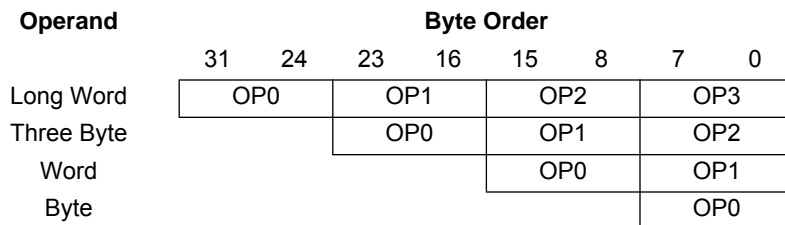| Operand	Byte Order			
	31 24	23 16	15 8	7 0
Long Word	OP0	OP1	OP2	OP3
Three Byte		OP0	OP1	OP2
Word			OP0	OP1
Byte				OP0

Figure 9 Operand Byte Order

3.4.10 Operand Alignment

The 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 remaining number of bytes 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, and ADDR0 indicates the byte offset from the base.

3.4.11 Misaligned Operands

CPU32 processor architecture 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 MCU does not support misaligned operand 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 via a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word on a following bus cycle.

3.4.12 Operand Transfer Cases

The following table summarizes how operands are aligned for various types of transfers. OPn entries are portions of a requested operand that are read or written during a bus cycle and are defined by SIZ1, SIZ0, and ADDR0 for that bus cycle.

Table 12 Operand Alignment

Transfer Case	SIZ1	SIZ0	ADDR0	$\overline{DSACK1}$	$\overline{DSACK0}$	DATA [15:8]	DATA [7:0]
Byte to 8-Bit Port (Even/Odd)	0	1	X	1	0	OP0	(OP0)[1]
Byte to 16-Bit Port (Even)	0	1	0	0	X	OP0	(OP0)
Byte to 16-Bit Port (Odd)	0	1	1	0	X	(OP0)	OP0
Word to 8-Bit Port (Aligned)	1	0	0	1	0	OP0	(OP1)
Word to 8-Bit Port (Misaligned)[2]	1	0	1	1	0	OP0	(OP0)
Word to 16-Bit Port (Aligned)	1	0	0	0	X	OP0	OP1
Word to 16-Bit Port (Misaligned)[2]	1	0	1	0	X	(OP0)	OP0
3 Byte to 8-Bit Port (Aligned)[3]	1	1	0	1	0	OP0	(OP1)
3 Byte to 8-Bit Port (Misaligned)[2, 3]	1	1	1	1	0	OP0	(OP0)
3 Byte to 16-Bit Port (Aligned)[3]	1	1	0	0	X	OP0	OP1
3 Byte to 16-Bit Port (Misaligned)[2, 3]	1	1	1	0	X	(OP0)	OP0
Long Word to 8-Bit Port (Aligned)	0	0	0	1	0	OP0	(OP1)
Long Word to 8-Bit Port (Misaligned)[2]	1	0	1	1	0	OP0	(OP0)
Long Word to 16-Bit Port (Aligned)	0	0	0	0	X	OP0	OP1
Long Word to 16-Bit Port (Misaligned)[2]	1	0	1	0	X	(OP0)	OP0

1. Operands in parentheses are ignored by the CPU32 during read cycles.
2. The CPU32 does not support misaligned word or long-word transfers.
3. Three-byte transfer cases occur only as a result of a long word to byte transfer.

3.5 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select and address decode signals. The MCU includes 12 programmable chip-select circuits that can provide 2- to 20-clock-cycle access to external memory and peripherals. Address block sizes of two Kbytes to one Mbyte can be selected. The figure below is a functional diagram of a chip-select circuit.

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobe, or interrupt acknowledge signals. Chip select logic can also generate $\overline{DSACK}$ and $\overline{AVEC}$ signals internally. Each signal can also be synchronized with the ECLK signal available on ADDR23.
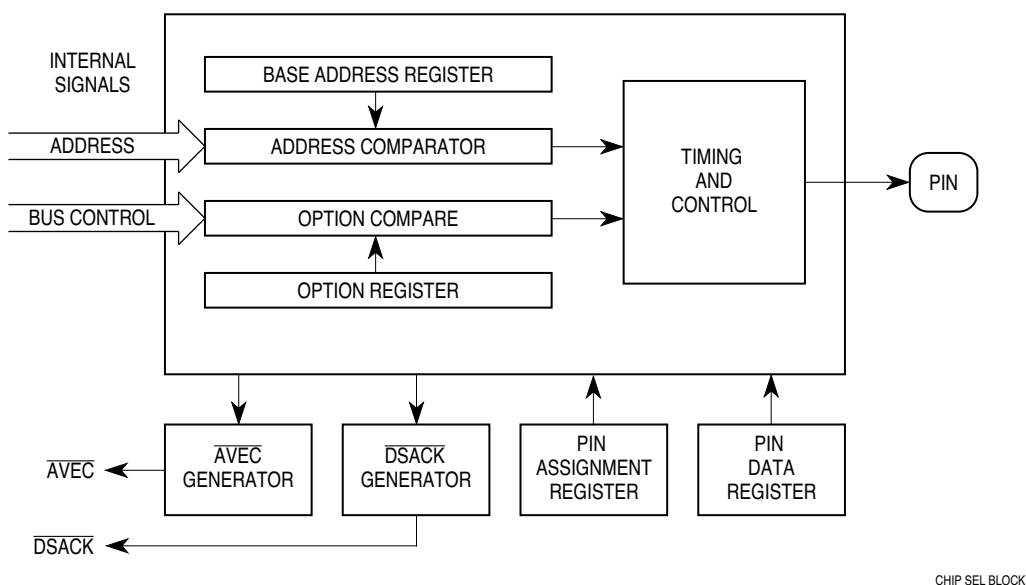
Figure 10 Chip-Select Circuit Block Diagram

When a memory access occurs, chip-select logic compares address space type, address, type of access, transfer size, and interrupt priority (in the case of interrupt acknowledge) to parameters stored in chip-select registers. If all parameters match, the appropriate chip-select signal is asserted. Select signals are active low. If a chip-select function is given the same address as a microcontroller module or an internal memory array, an access to that address goes to the module or array, and the chip-select signal is not asserted. The external address and data buses do not reflect the internal access.

All chip-select circuits are configured for operation out of reset. However, all chip-select signals except $\overline{CSBOOT}$ are disabled, and cannot be asserted until the BYTE field in the corresponding option register is programmed to a nonzero value, selecting a transfer size. The chip-select option register must not be written until a base address has been written to a proper base address register. $\overline{CSBOOT}$ is automatically asserted out of reset. Alternate functions for chip-select pins are enabled if appropriate data bus pins are held low at the release of the reset signal.

The following table lists allocation of chip-selects and discrete outputs on the pins of the MCU.

Table 13 Chip Select Allocation

Pin	Chip Select	Discrete Outputs
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—
$\overline{\text{BR}}$	$\overline{\text{CS0}}$	—
$\overline{\text{BG}}$	$\overline{\text{CS1}}$	—
$\overline{\text{BGACK}}$	$\overline{\text{CS2}}$	—
FC0	$\overline{\text{CS3}}$	PC0
FC1	$\overline{\text{CS4}}$	PC1
FC2	$\overline{\text{CS5}}$	PC2
ADDR19	$\overline{\text{CS6}}$	PC3
ADDR20	$\overline{\text{CS7}}$	PC4
ADDR21	$\overline{\text{CS8}}$	PC5
ADDR22	$\overline{\text{CS9}}$	PC6
ADDR23	$\overline{\text{CS10}}$	ECLK

3.5.1 Chip-Select Registers

Each chip-select pin can have one or more functions. Chip-select pin assignment registers (CS-PAR[0:1]) determine functions of the pins. Pin assignment registers also determine port size (8- or 16-bit) for dynamic bus allocation. A pin data register (PORTC) latches data for chip-select pins that are used for discrete output.

Blocks of addresses are assigned to each chip-select function. Block sizes of two Kbytes to one Mbyte can be selected by writing values to the appropriate base address register (CSBAR[0:10], CSBARBT). Address blocks for separate chip-select functions can overlap.

Chip select option registers (CSOR[0:10], CSORBT) determine timing of and conditions for assertion of chip-select signals. Eight parameters, including operating mode, access size, synchronization, and wait state insertion can be specified.

Initialization software usually resides in a peripheral memory device controlled by the chip-select circuits. A set of special chip-select functions and registers (CSORBT, CSBARBT) is provided to support bootstrap operation.

3.5.2 Pin Assignment Registers

The pin assignment registers contain twelve 2-bit fields ($\overline{\text{CS[10:0]}}$ and $\overline{\text{CSBOOT}}$) that determine functions of the chip-select pins. Each pin has two or three possible functions, as shown below.

Table 14 Chip-Select Pin Functions

Assignment Register	16-Bit Chip Select	8-Bit Chip Select	Alternate Function	Discrete Output
CSPAR0	$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—
	$\overline{\text{CS0}}$	$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
	$\overline{\text{CS1}}$	$\overline{\text{CS1}}$	$\overline{\text{BG}}$	—
	$\overline{\text{CS2}}$	$\overline{\text{CS2}}$	$\overline{\text{BGACK}}$	—
	$\overline{\text{CS3}}$	$\overline{\text{CS3}}$	FC0	PC0
	$\overline{\text{CS4}}$	$\overline{\text{CS4}}$	FC1	PC1
	$\overline{\text{CS5}}$	$\overline{\text{CS5}}$	FC2	PC2
CSPAR1	$\overline{\text{CS6}}$	$\overline{\text{CS6}}$	ADDR19	PC3
	$\overline{\text{CS7}}$	$\overline{\text{CS7}}$	ADDR20	PC4
	$\overline{\text{CS8}}$	$\overline{\text{CS8}}$	ADDR21	PC5
	$\overline{\text{CS9}}$	$\overline{\text{CS9}}$	ADDR22	PC6
	$\overline{\text{CS10}}$	$\overline{\text{CS10}}$	ADDR23	ECLK

The table below shows pin assignment field encoding. Pins that have no discrete output function do not use the %00 encoding.

Table 15 Pin Assignment Encodings

Bit Field	Description
00	Discrete Output
01	Alternate Function
10	Chip Select (8-Bit Port)
11	Chip Select (16-Bit Port)

CSPAR0 — Chip Select Pin Assignment Register 0   **$YFFA44**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	CSPA0[6]		CSPA0[5]		CSPA0[4]		CSPA0[3]		CSPA0[2]		CSPA0[1]		$\overline{\text{CSBOOT}}$	

RESET:

0	0	DATA2	1	DATA2	1	DATA2	1	DATA1	1	DATA1	1	DATA1	1	1	DATA0

CSPAR0 contains seven 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR0[15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect.

Table 16 CSPAR0 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA0[6]	$\overline{\text{CS5}}$	FC2	PC2
CSPA0[5]	$\overline{\text{CS4}}$	FC1	PC1
CSPA0[4]	$\overline{\text{CS3}}$	FC0	PC0
CSPA0[3]	$\overline{\text{CS2}}$	$\overline{\text{BGACK}}$	—
CSPA0[2]	$\overline{\text{CS1}}$	$\overline{\text{BG}}$	—
CSPA0[1]	$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—	—

CSPAR1 — Chip Select Pin Assignment Register 1   **$YFFA46**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	CSPA1[4]		CSPA1[3]		CSPA1[2]		CSPA1[1]		CSPA1[0]	

RESET:

0	0	0	0	0	0	DATA7	1	DATA[7:6]	1	DATA[7:5]	1	DATA[7:4]	1	DATA[7:3]	1

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; writes have no effect.

Table 17 CSPAR1 Pin Assignments

CSPAR1 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA1[4]	$\overline{\text{CS10}}$	ADDR23	ECLK
CSPA1[3]	$\overline{\text{CS9}}$	ADDR22	PC6
CSPA1[2]	$\overline{\text{CS8}}$	ADDR21	PC5
CSPA1[1]	$\overline{\text{CS7}}$	ADDR20	PC4
CSPA1[0]	$\overline{\text{CS6}}$	ADDR19	PC3

Port size determines the way in which bus transfers to an external address are allocated. Port size of eight bits or sixteen bits can be selected when a pin is assigned as a chip select. Port size and transfer size affect how the chip-select signal is asserted. Refer to **3.5.4 Option Registers** for more information.

Out of reset, chip-select pin function is determined by the logic level on a corresponding data bus pin. These pins have weak internal pull-up drivers, but can be held low by external devices. Either 16-bit chip-select function (%11) or alternate function (%01) can be selected during reset. All pins except the boot ROM select pin ($\overline{\text{CSBOOT}}$) are disabled out of reset. There are twelve chip-select functions and only eight associated data bus pins. There is not a one-to-one correspondence. The table below shows reset states.

Table 18 Reset Pin Function of $\overline{\text{CS}}$[10:6]

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{\text{CS10}}$/ ADDR23	$\overline{\text{CS9}}$/ ADDR22	$\overline{\text{CS8}}$/ ADDR21	$\overline{\text{CS7}}$/ ADDR20	$\overline{\text{CS6}}$/ ADDR19
1	1	1	1	1	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	$\overline{\text{CS7}}$	$\overline{\text{CS6}}$
1	1	1	1	0	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	$\overline{\text{CS7}}$	ADDR19
1	1	1	0	X	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	ADDR20	ADDR19
1	1	0	X	X	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{\text{CS10}}$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

The $\overline{\text{CSBOOT}}$ signal is normally enabled out of reset. The state of the DATA0 line during reset determines what port width $\overline{\text{CSBOOT}}$ uses. If DATA0 is held high (either by the weak internal pull-up driver or by an external pull-up device), 16-bit width is selected. If DATA0 is held low, 8-bit port size is selected.

A pin programmed as a discrete output drives an external signal to the value specified in the pin data register. No discrete output function is available on pins $\overline{\text{CSBOOT}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$, or $\overline{\text{BGACK}}$. ADDR23 provides ECLK output rather than a discrete output signal.

When a pin is programmed for discrete output or alternate function, internal chip-select logic still functions and can be used to generate $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ internally on an address and control signal match.

3.5.3 Base Address Registers

Each chip select has an associated base address register. A base address is the lowest address in the block of addresses enabled by a chip select. Block size is the extent of the address block above the base address. Block size is determined by the value contained in a BLKSZ field. Block addresses for different chip selects can overlap.

The BLKSZ field determines which bits in the base address field are compared to corresponding bits on the address bus during an access. Provided other constraints determined by option register fields are also satisfied, when a match occurs, the associated chip-select signal is asserted.

The chip-select address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be a multiple of block size. Base address register diagrams show how base register bits correspond to address lines.

After reset, the MCU fetches the initialization routine from the address contained in the reset vector, located beginning at address $000000 of program space. To support bootstrap operation from reset, the base address field in chip-select base address register boot (CSBARBT) has a reset value of all zeros. A memory device containing the reset vector and initialization routine can be automatically enabled by $\overline{\text{CSBOOT}}$ after a reset. The block size field in CSBARBT has a reset value of 512 Kbytes.

CSBARBT — Chip-Select Base Address Register Boot ROM   **$YFFA48**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

CSBAR[10:0] — Chip-Select Base Address Registers   **$YFFA4C–$YFFA74**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BLKSZ — Block Size Field
    This field determines the size of the block that must be enabled by the chip select. The following table shows bit encoding for the base address registers block size field.

Block Size Field	Block Size	Address Lines Compared
000	2 Kbytes	ADDR[23:11]
001	8 Kbytes	ADDR[23:13]
010	16 Kbytes	ADDR[23:14]
011	64 Kbytes	ADDR[23:16]
100	128 Kbytes	ADDR[23:17]
101	256 Kbytes	ADDR[23:18]
110	512 Kbytes	ADDR[23:19]
111	1 Mbyte	ADDR[23:20]

ADDR[23:11] — Base Address Field
    This field sets the starting address of a particular address space. The address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be a multiple of block size. Base address register diagrams show how base register bits correspond to address lines.

3.5.4 Option Registers

    The option registers contain eight fields that determine timing of and conditions for assertion of chip-select signals. To assert a chip-select signal, and to provide $\overline{DSACK}$ or autovector support, other constraints set by fields in the option register and in the base address register must also be satisfied.

CSORBT — Chip-Select Option Register Boot ROM   **$YFFA4A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{W}$		STRB	$\overline{DSACK}$				SPACE		IPL			$\overline{AVEC}$

RESET:

0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0

CSOR[10:0] — Chip-Select Option Registers   **$YFFA4E–$YFFA76**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{W}$		STRB	$\overline{DSACK}$				SPACE		IPL			$\overline{AVEC}$

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSORBT, the option register for $\overline{CSBOOT}$, contains special reset values that support bootstrap operations from peripheral memory devices.

The following bit descriptions apply to both CSORBT and CSOR[10:0] option registers.

MODE — Asynchronous/Synchronous Mode
    0 = Asynchronous mode selected (chip-select assertion determined by internal or external bus control signals)
    1 = Synchronous mode selected (chip-select assertion synchronized with ECLK signal)
In asynchronous mode, the chip select is asserted synchronized with $\overline{AS}$ or $\overline{DS}$.

The $\overline{DSACK}$ field is not used in synchronous mode because a bus cycle is only performed as a synchronous operation. When a match condition occurs on a chip select programmed for synchronous operation, the chip select signals the EBI that an ECLK cycle is pending.

BYTE — Upper/Lower Byte Option
    This field is used only when the chip-select 16-bit port option is selected in the pin assignment register. The following table lists upper/lower byte options.

Byte	Description
00	Disable
01	Lower Byte
10	Upper Byte
11	Both Bytes

R/$\overline{W}$ — Read/Write
    This field causes a chip select to be asserted only for a read, only for a write, or for both read and write. Refer to the following table for options available.

R/$\overline{W}$	Description
00	Reserved
01	Read Only
10	Write Only
11	Read/Write

STRB — Address Strobe/Data Strobe
    0 = Address strobe
    1 = Data strobe
This bit controls the timing for assertion of a chip select in asynchronous mode. Selecting address strobe causes chip select to be asserted synchronized with address strobe. Selecting data strobe causes chip select to be asserted synchronized with data strobe.

$\overline{DSACK}$ — Data and Size Acknowledge
    This field specifies the source of $\overline{DSACK}$ in asynchronous mode. It also allows the user to adjust bus timing with internal $\overline{DSACK}$ generation by controlling the number of wait states that are inserted to optimize bus speed in a particular application. The following table shows the $\overline{DSACK}$ field encoding. The fast termination encoding (1110) is used for two-cycle access to external memory.

$\overline{\text{DSACK}}$	Description
0000	No Wait States
0001	1 Wait State
0010	2 Wait States
0011	3 Wait States
0100	4 Wait States
0101	5 Wait States
0110	6 Wait States
0111	7 Wait States
1000	8 Wait States
1001	9 Wait States
1010	10 Wait States
1011	11 Wait States
1100	12 Wait States
1101	13 Wait States
1110	Fast Termination
1111	External $\overline{\text{DSACK}}$

SPACE — Address Space

Use this option field to select an address space for the chip-select logic. The CPU32 normally operates in supervisor or user space, but interrupt acknowledge cycles must take place in CPU space.

Space	Address Space
00	CPU Space
01	User Space
10	Supervisor Space
11	Supervisor/User Space

IPL — Interrupt Priority Level

If the space field is set for CPU space (00), chip-select logic can be used for interrupt acknowledge. During an interrupt acknowledge cycle, the priority level on address lines ADDR[3:1] is compared to the value in the IPL field. If the values are the same, a chip select is asserted, provided that other option register conditions are met. The following table shows IPL field encoding.

IPL	Description
000	Any Level
001	IPL1
010	IPL2
011	IPL3
100	IPL4
101	IPL5
110	IPL6
111	IPL7

This field only affects the response of chip selects and does not affect interrupt recognition by the CPU. Any level means that chip select is asserted regardless of the level of the interrupt acknowledge cycle.

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

    0 = External interrupt vector enabled
    1 = Autovector enabled

This field selects one of two methods of acquiring an interrupt vector number during an external interrupt acknowledge cycle.

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 circuit generates an internal $\overline{\text{AVEC}}$ signal in response to an external interrupt cycle, and the SIM supplies an automatic vector number. Otherwise, the vector number must be supplied by the requesting device. An internal autovector is generated only in response to interrupt requests from the SIM $\overline{\text{IRQ}}$ pins — interrupt requests from other IMB modules are ignored.

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	14	13	12	11	10	9	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	14	13	12	11	10	9	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	14	13	12	11	10	9	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.

PEPAR — Port E Pin Assignment Register $YFFA17

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			NOT USED					PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0

RESET:

| | | | | | | | | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 |

The bits in this register control the function of each port E pin. Any bit set to one configures the corresponding pin as a bus control signal, with the function shown in the following table. Any bit cleared to zero defines the corresponding pin to be an I/O pin, controlled by PORTE and DDRE.

Data bus bit 8 controls the state of this register following reset. If DATA8 is set to one during reset, the register is set to $FF, which defines all port E pins as bus control signals. If DATA8 is cleared to zero during reset, this register is set to $00, configuring all port E pins as I/O pins.

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 a bus control signal.

Table 19 Port E Pin Assignments

PEPAR Bit	Port E Signal	Bus Control Signal
PEPA7	PE7	SIZ1
PEPA6	PE6	SIZ0
PEPA5	PE5	$\overline{AS}$
PEPA4	PE4	$\overline{DS}$
PEPA3	PE3	$\overline{RMC}$
PEPA2	PE2	$\overline{AVEC}$
PEPA1	PE1	$\overline{DSACK1}$
PEPA0	PE0	$\overline{DSACK0}$

PORTF0, PORTF1 — Port F Data Register $YFFA19, $YFFA1B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			NOT USED					PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0

RESET:

| | | | | | | | | U | U | U | U | U | U | U | U |

The write to the port F data register is stored in the internal data latch, and if any port F pin is configured as an output, the value stored for that bit is driven onto the pin. A read of the port F 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 F data register is a single register that can be accessed in two locations. When accessed at $YFFA19, the register is referred to as PORTF0; when accessed at $YFFA1B, the register is referred to as PORTF1. The register can be read or written at any time. It is unaffected by reset.

DDRF — Port F Data Direction Register $YFFA1D

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			NOT USED					DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0

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 for 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.

15	14	13	12	11	10	9	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 20 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA7	PF7	$\overline{IRQ7}$
PFPA6	PF6	$\overline{IRQ6}$
PFPA5	PF5	$\overline{IRQ5}$
PFPA4	PF4	$\overline{IRQ4}$
PFPA3	PF3	$\overline{IRQ3}$
PFPA2	PF2	$\overline{IRQ2}$
PFPA1	PF1	$\overline{IRQ1}$
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 $\overline{RESET}$ 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 $\overline{RESET}$ 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 $\overline{RESET}$ 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 $\overline{BKPT}$ pin determines what happens during subsequent breakpoint assertions. The following table is a summary of reset mode selection options.

Table 21 Reset Mode Selection

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
DATA1	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$	$\overline{\text{BR}}$ $\overline{\text{BG}}$ $\overline{\text{BGACK}}$
DATA2	$\overline{\text{CS3}}$ $\overline{\text{CS4}}$ $\overline{\text{CS5}}$	FC0 FC1 FC2
DATA3 DATA4 DATA5 DATA6 DATA7	$\overline{\text{CS6}}$ $\overline{\text{CS[7:6]}}$ $\overline{\text{CS[8:6]}}$ $\overline{\text{CS[9:6]}}$ $\overline{\text{CS[10:6]}}$	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
DATA8	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, $\overline{\text{SIZ[1:0]}}$	PORTE
DATA9	$\overline{\text{IRQ[7:1]}}$ MODCLK	PORTF
DATA11	Test Mode Disabled	Test Mode Enabled
MODCLK	VCO = System Clock	EXTAL = System Clock
$\overline{\text{BKPT}}$	Background Mode Disabled	Background Mode Enabled

Data lines have weak internal pull-up drivers. External bus loading can overcome the weak internal pull-up drivers on data bus lines, and hold pins low during reset.   Use an active device to hold data bus lines low. Data bus configuration logic must release the bus before the first bus cycle after reset to prevent conflict with external memory devices. The first bus cycle occurs ten CLKOUT cycles after $\overline{\text{RESET}}$ is released. If external mode selection logic causes a conflict of this type, an isolation resistor on the driven lines may be required.

3.7.2 Functions of Pins for Other Modules During Reset

Generally, pins associated with modules other than the SIM default to port functions, and input/output ports are set to input state. This is accomplished by disabling pin functions in the appropriate control registers, and by clearing the appropriate port data direction registers. Refer to individual module sections in this manual for more information. The following table is a summary of module pin function out of reset.

Table 22 Module Pin Functions

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
TPU	TPUCH[15:0]	TPU Input
	T2CLK	TCR2 Clock
ADC	AN[6:0]/PADA[6:0]	Analog Input

3.7.3 Reset Timing

The $\overline{\text{RESET}}$ input must be asserted for a specified minimum period in order for reset to occur. External $\overline{\text{RESET}}$ assertion can be delayed internally for a period equal to the longest bus cycle time (or the bus monitor time-out period) in order to protect write cycles from being aborted by reset. While $\overline{\text{RESET}}$ is asserted, SIM pins are either in a disabled high-impedance state or are driven to their inactive states.

When an external device asserts $\overline{\text{RESET}}$ for the proper period, reset control logic clocks the signal into an internal latch. The control logic drives the $\overline{\text{RESET}}$ pin low for an additional 512 CLKOUT cycles after it detects that the $\overline{\text{RESET}}$ signal is no longer being externally driven, to guarantee this length of reset to the entire system.

If an internal source asserts a reset signal, the reset control logic asserts $\overline{\text{RESET}}$ for a minimum of 512 cycles. If the reset signal is still asserted at the end of 512 cycles, the control logic continues to assert $\overline{\text{RESET}}$ until the internal reset signal is negated.

After 512 cycles have elapsed, the reset input pin goes to an inactive, high-impedance state for 10 cycles. At the end of this 10-cycle period, the reset input is tested. When the input is at logic level one, reset exception processing begins. If, however, the reset input is at logic level zero, the reset control logic drives the pin low for another 512 cycles. At the end of this period, the pin again goes to high-impedance state for ten cycles, then it is tested again. The process repeats until $\overline{\text{RESET}}$ is released.

3.7.4 Power-On Reset

When the SIM clock synthesizer is used to generate the system clock, power-on reset involves special circumstances related to application of system and clock synthesizer power. Regardless of clock source, voltage must be applied to clock synthesizer power input pin V_{DDSYN} in order for the MCU to operate. The following discussion assumes that V_{DDSYN} is applied before and during reset. This minimizes crystal start-up time. When V_{DDSYN} is applied at power-on, start-up time is affected by specific crystal parameters and by oscillator circuit design. V_{DD} ramp-up time also affects pin state during reset.

During power-on reset, an internal circuit in the SIM drives the internal IMB and external reset lines. The circuit releases the internal reset line as V_{DD} ramps up to the minimum specified value, and SIM pins are initialized. As V_{DD} reaches specified minimum value, the clock synthesizer VCO begins operation and clock frequency ramps up to specified limp mode frequency. The external $\overline{\text{RESET}}$ line remains asserted until the clock synthesizer PLL locks and 512 CLKOUT cycles elapse.

The SIM clock synthesizer provides clock signals to the other MCU modules. After the clock is running and the internal reset signal is asserted for four clock cycles, these modules reset. V_{DD} ramp time and VCO frequency ramp time determine how long these four cycles take. Worst case is approximately 15 milliseconds. During this period, module port pins may be in an indeterminate state. While input-only pins can be put in a known state by means of external pull-up resistors, external logic on input/output or output-only pins must condition the lines during this time. Active drivers require high-impedance buffers or isolation resistors to prevent conflict.

3.7.5 Use of Three State Control Pin

Asserting the three-state control (TSC) input causes the MCU to put all output drivers in an inactive, high-impedance state. The signal must remain asserted for ten clock cycles in order for drivers to change state. There are certain constraints on use of TSC during power-on reset:

When the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the ten cycles take. Worst case is approximately 20 ms from TSC assertion.

When an external clock signal is applied (MODCLK held low during reset), pins go to high-impedance state as soon after TSC assertion as ten clock pulses have been applied to the EXTAL pin.

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 eight levels of interrupt priority. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in status register.

There are seven interrupt request signals ($\overline{\text{IRQ}}$[7:1]). These signals are used internally on the IMB, and there are corresponding pins for external interrupt service requests. The CPU treats all interrupt requests as though they come from internal modules — external interrupt requests are treated as interrupt service requests from the SIM. Each of the interrupt request signals corresponds to an interrupt priority level. $\overline{\text{IRQ1}}$ has the lowest priority and $\overline{\text{IRQ7}}$ the highest.

Interrupt recognition is determined by interrupt priority level and interrupt priority mask value. The interrupt priority mask consists of three bits in the CPU32 status register. 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 from being recognized and processed. $\overline{\text{IRQ7}}$, however, is always recognized, even if the mask value is %111.

$\overline{\text{IRQ}}$[7:1] are active-low level-sensitive inputs. The low on the pin must remain asserted until an interrupt acknowledge cycle corresponding to that level is detected.

$\overline{\text{IRQ7}}$ is transition-sensitive as well as level-sensitive: a level-7 interrupt is not detected unless a falling edge transition is detected on the $\overline{\text{IRQ7}}$ line. This prevents redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted as well as 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 with a priority equal to or lower than the current IP mask value is made, the CPU32 does not recognize the occurrence of the request. If simultaneous interrupt requests of different priorities are made, and both have a priority greater than the mask value, the CPU32 recognizes the higher-level request.

3.8.1 Interrupt Acknowledge and Arbitration

When the CPU32 detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it places the interrupt request level on the address bus and initiates a CPU space read cycle. The request level serves two purposes: it is decoded by modules or external devices that have requested interrupt service, to determine whether the current interrupt acknowledge cycle pertains to them, and it is latched into the interrupt priority mask field in the CPU32 status register, to preclude further interrupts of lower priority during interrupt service.

Modules or external devices that have requested interrupt service must decode the interrupt priority mask value placed on the address bus during the interrupt acknowledge cycle and respond if the priority of the service request corresponds to the mask value. However, before modules or external devices respond, interrupt arbitration takes place.

Arbitration is performed by means of serial contention between values stored in individual module interrupt arbitration (IARB) fields. Each module that can make an interrupt service request, including the SIM, has an IARB field in its configuration register. IARB fields can be assigned values from %0000 to

%1111. In order to implement an arbitration scheme, each module that can initiate an interrupt service request must be assigned a unique, non-zero IARB field value during system initialization. Arbitration priorities range from %0001 (lowest) to %1111 (highest) — if the CPU recognizes an interrupt service request from a source that has an IARB field value of %0000, a spurious interrupt exception is processed.

WARNING

Do not assign the same arbitration priority to more than one module. When two or more IARB fields have the same nonzero value, the CPU32 interprets multiple vector numbers at the same time, with unpredictable consequences.

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.

Although arbitration is intended to deal with simultaneous requests of the same priority, it always takes place, even when a single source is requesting service. This is important for two reasons: the EBI does not transfer the interrupt acknowledge read cycle to the external bus unless the SIM wins contention, and failure to contend causes the interrupt acknowledge bus cycle to be terminated early, by a bus error.

When arbitration is complete, the module with the highest arbitration priority must terminate the bus cycle. Internal modules place an interrupt vector number on the data bus and generate appropriate internal cycle termination signals. In the case of an external interrupt request, after the interrupt acknowledge cycle is transferred to the external bus, the appropriate external device must decode the mask value and respond with a vector number, then generate data and size acknowledge ($\overline{\text{DSACK}}$) termination signals, or it must assert the autovector ($\overline{\text{AVEC}}$) request signal. If the device does not respond in time, the EBI bus monitor asserts the bus error signal $\overline{\text{BERR}}$, and a spurious interrupt exception is taken.

Chip-select logic can also be used to generate internal $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signals in response to interrupt requests from external devices. Chip-select address match logic functions only after the EBI transfers an interrupt acknowledge cycle to the external bus following IARB contention. If a module makes an interrupt request of a certain priority, and the appropriate chip-select registers are programmed to generate $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signals in response to an interrupt acknowledge cycle for that priority level, chip-select logic does not respond to the interrupt acknowledge cycle, and the internal module supplies a vector number and generates internal cycle termination signals.

For periodic timer interrupts, the PIRQ field in the periodic interrupt control register (PICR) determines PIT priority level. A PIRQ value of %000 means that PIT interrupts are inactive. By hardware convention, when the CPU32 receives simultaneous interrupt requests of the same level from more than one SIM source (including external devices), the periodic interrupt timer is given the highest priority, followed by the $\overline{\text{IRQ}}$ pins. 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. The processor state is stacked. The S bit in the status register is set, establishing supervisor access level, and bits T1 and T0 are cleared, disabling tracing.
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 priority of the interrupt request being acknowledged; and ADDR0 = %1.

3. The request level is latched from the address bus into the interrupt priority mask field in the status or condition code register.
D. Modules that have requested interrupt service decode the priority value in ADDR[3:1]. If request priority is the same as acknowledged priority, arbitration by IARB contention takes place.
E. After arbitration, the interrupt acknowledge cycle is completed in one of the following ways:
1. When there is no contention (IARB = %0000), the spurious interrupt monitor asserts $\overline{BERR}$, and the CPU generates the spurious interrupt vector number.
2. The dominant interrupt source supplies a vector number and $\overline{DSACK}$ signals appropriate to the access. The CPU acquires the vector number.
3. The $\overline{AVEC}$ signal is asserted (the signal can be asserted by the dominant interrupt source or the pin can be tied low), and the CPU generates an autovector number corresponding to interrupt priority.
4. The bus monitor asserts $\overline{BERR}$ 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 Freescale 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**

4 Central Processor Unit

Based on the powerful MC68020, the CPU32 processing module provides enhanced system performance and also uses the extensive software base for the Freescale M68000 family.

4.1 Overview

The CPU32 is fully object code compatible with the M68000 Family, which excels at processing calculation-intensive algorithms and supporting high-level languages. The CPU32 supports all of the MC68010 and most of the MC68020 enhancements, such as virtual memory support, loop mode operation, instruction pipeline, and 32-bit mathematical operations. Powerful addressing modes provide compatibility with existing software programs and increase the efficiency of high-level language compilers. Special instructions, such as table lookup and interpolate and low-power stop, support the specific requirements of controller applications. Also included is the background debugging mode, an alternate operating mode that suspends normal operation and allows the CPU to accept debugging commands from the development system.

Ease of programming is an important consideration in using a microcontroller. The CPU32 instruction set is optimized for high performance. The eight 32-bit general-purpose data registers readily support 8-bit (byte), 16-bit (word), and 32-bit (long word) operations. Ease of program checking and diagnosis is further enhanced by trace and trap capabilities at the instruction level.

Use of high-level languages is increasing as controller applications become more complex and control programs become larger. High-level languages aid rapid development of software, with less error, and are readily portable. The CPU32 instruction set supports high-level languages.

4.2 Programming Model

The CPU32 has sixteen 32-bit general registers, a 32-bit program counter, one 32-bit supervisor stack pointer, a 16-bit status register, two alternate function code registers, and a 32-bit vector base register.

The programming model of the CPU32 consists of a user model and supervisor model, corresponding to the user and supervisor privilege levels. Some instructions available at the supervisor level are not available at the user level, allowing the supervisor to protect system resources from uncontrolled access. Bit S in the status register determines the privilege level.

The user programming model remains unchanged from previous M68000 Family microprocessors. Application software written to run at the non-privileged user level migrates without modification to the CPU32 from any M68000 platform. The move from SR instruction, however, is privileged in the CPU32. It is not privileged in the M68000.

31	16	15	8	7	0		

```
                                                        D0
                                                        D1
                                                        D2
                                                        D3        Data Registers
                                                        D4
                                                        D5
                                                        D6
                                                        D7
```

31   16 15   0

```
                                                        A0
                                                        A1
                                                        A2
                                                        A3        Address Registers
                                                        A4
                                                        A5
                                                        A6
```

31   16 15   0

A7 (USP)   User Stack Pointer

31   0

PC   Program Counter

7   0

CCR   Condition Code Register
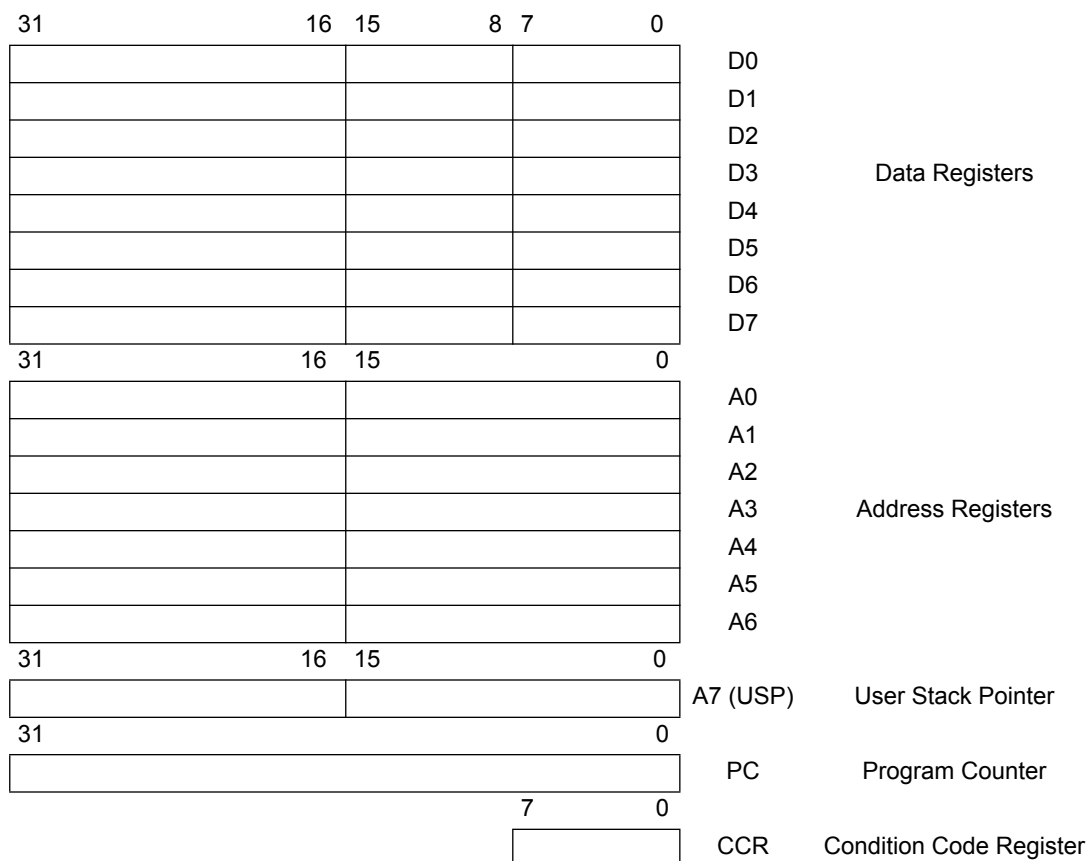
Supervisor Programming Model Supplement

31   16 15   0

A7' (SSP)   Supervisor Stack Pointer

15   8 7   0

(CCR)   SR   Status Register

31   0

VBR   Vector Base Register

2   0

SFC   Alternate Function
DFC   Code Registers
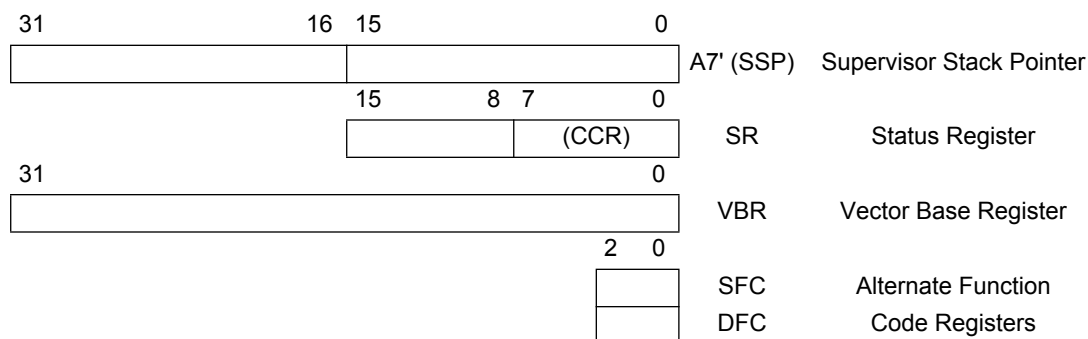
4.3 Status Register

The status register contains the condition codes that reflect the results of a previous operation and can be used for conditional instruction execution in a program. The lower byte containing the condition codes is the only portion of the register available at the user privilege level; it is referenced as the condition code register (CCR) in user programs. At the supervisor privilege level, software can access the full status register, including the interrupt priority mask and additional control bits.

SR — Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T1	T0	S	0	0		IP		0	0	0	X	N	Z	V	C

RESET:

0	0	1	0	0	1	1	1	0	0	0	U	U	U	U	U

System Byte
  T[1:0] — Trace Enable
  S — Supervisor/User State
  Bits [12:11] — Unimplemented
  IP[2:0] — Interrupt Priority Mask

User Byte (Condition Code Register)
  Bits [7:5] — Unimplemented
  X — Extend
  N — Negative
  Z — Zero
  V — Overflow
  C — Carry

4.4 Data Types

Six basic data types are supported:

- Bits
- Packed Binary Coded Decimal Digits
- Byte Integers (8 bits)
- Word Integers (16 bits)
- Long-Word Integers (32 bits)
- Quad-Word Integers (64 bits)

4.5 Addressing Modes

Addressing in the CPU32 is register-oriented. Most instructions allow the results of the specified operation to be placed either in a register or directly in memory. This flexibility eliminates the need for extra instructions to store register contents in memory. The CPU32 supports seven basic addressing modes:

- Register direct
- Register indirect
- Register indirect with index
- Program counter indirect with displacement
- Program counter indirect with index
- Absolute
- Immediate

Included in the register indirect addressing modes are the capabilities to post-increment, predecrement, and offset. The program counter relative mode also has index and offset capabilities. In addition to these addressing modes, many instructions implicitly specify the use of the status register, stack pointer, or program counter.

4.6 Instruction Set Summary

Table 23 Instruction Set Summary

Instruction	Syntax	Operand Size	Operation
ABCD	Dn, Dn – (An), – (An)	8 8	$Source_{10}$ + $Destination_{10}$ + X $\Rightarrow$ Destination
ADD	Dn, ⟨ea⟩ ⟨ea⟩, Dn	8, 16, 32 8, 16, 32	Source + Destination $\Rightarrow$ Destination
ADDA	⟨ea⟩, An	16, 32	Source + Destination $\Rightarrow$ Destination
ADDI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Immediate data + Destination $\Rightarrow$ Destination
ADDQ	#⟨data⟩, ⟨ea⟩	8, 16, 32	Immediate data + Destination $\Rightarrow$ Destination
ADDX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Source + Destination + X $\Rightarrow$ Destination
AND	⟨ea⟩, Dn Dn, ⟨ea⟩	8, 16, 32 8, 16, 32	Source $*$ Destination $\Rightarrow$ Destination
ANDI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Data $*$ Destination $\Rightarrow$ Destination
ANDI to CCR	#⟨data⟩, CCR	8	Source $*$ CCR $\Rightarrow$ CCR
ANDI to SR[1]	#⟨data⟩, SR	16	Source $*$ SR $\Rightarrow$ SR
ASL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
Bcc	⟨label⟩	8, 16, 32	If condition true, then PC + d $\Rightarrow$ PC
BCHG	Dn, ⟨ea⟩ #⟨data⟩, ⟨ea⟩	8, 32 8, 32	(⟨bit number⟩ of destination) $\Rightarrow$ Z $\Rightarrow$ bit of destination
BCLR	Dn, ⟨ea⟩ #⟨data⟩, ⟨ea⟩	8, 32 8, 32	(⟨bit number⟩ of destination) $\Rightarrow$ Z; 0 $\Rightarrow$ bit of destination
BGND	none	none	If background mode enabled, then enter background mode, else format/vector offset $\Rightarrow$ – (SSP); PC $\Rightarrow$ – (SSP); SR $\Rightarrow$ – (SSP); (vector) $\Rightarrow$ PC
BKPT	#⟨data⟩	none	If breakpoint cycle acknowledged, then execute returned operation word, else trap as illegal instruction.
BRA	⟨label⟩	8, 16, 32	PC + d $\Rightarrow$ PC
BSET	Dn, ⟨ea⟩ #⟨data⟩, ⟨ea⟩	8, 32 8, 32	(⟨bit number⟩ of destination) $\Rightarrow$ Z; 1 $\Rightarrow$ bit of destination
BSR	⟨label⟩	8, 16, 32	SP – 4 $\Rightarrow$ SP; PC $\Rightarrow$ (SP); PC + d $\Rightarrow$ PC
BTST	Dn, ⟨ea⟩ #⟨data⟩, ⟨ea⟩	8, 32 8, 32	(⟨bit number⟩ of destination) $\Rightarrow$ Z
CHK	⟨ea⟩, Dn	16, 32	If Dn < 0 or Dn < (ea), then CHK exception
CHK2	⟨ea⟩, Rn	8, 16, 32	If Rn < lower bound or Rn > upper bound, then CHK exception
CLR	⟨ea⟩	8, 16, 32	0 $\Rightarrow$ Destination
CMP	⟨ea⟩, Dn	8, 16, 32	(Destination – Source), CCR shows results
CMPA	⟨ea⟩, An	16, 32	(Destination – Source), CCR shows results
CMPI	#⟨data⟩, ⟨ea⟩	8, 16, 32	(Destination – Data), CCR shows results
CMPM	(An) +, (An) +	8, 16, 32	(Destination – Source), CCR shows results
CMP2	⟨ea⟩, Rn	8, 16, 32	Lower bound $\leq$ Rn $\leq$ Upper bound, CCR shows result

Table 23 Instruction Set Summary

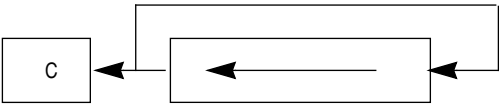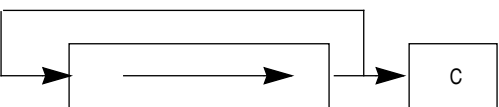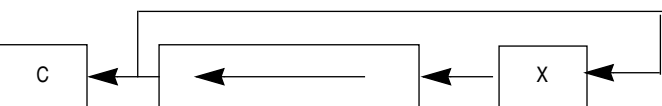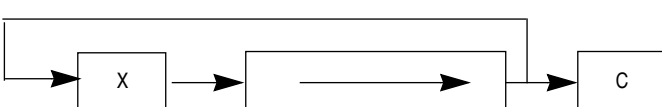| Instruction	Syntax	Operand Size	Operation
DBcc	Dn, ⟨label⟩	16	If condition false, then Dn – 1 ⇒ PC; if Dn ≠ (– 1), then PC + d ⇒ PC
DIVS/DIVU	⟨ea⟩, Dn	32/16 ⇒ 16: 16	Destination / Source ⇒ Destination (signed or unsigned)
DIVSL/DIVUL	⟨ea⟩, Dr: Dq ⟨ea⟩, Dq ⟨ea⟩, Dr : Dq	64/32 ⇒ 32 : 32 32/32 ⇒ 32 32/32 ⇒ 32 : 32	Destination / Source ⇒ Destination (signed or unsigned)
EOR	Dn, ⟨ea⟩	8, 16, 32	Source ⊕ Destination ⇒ Destination
EORI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Data ⊕ Destination ⇒ Destination
EORI to CCR	#⟨data⟩, CCR	8	Source ⊕ CCR ⇒ CCR
EORI to SR[1]	#⟨data⟩, SR	16	Source ⊕ SR ⇒ SR
EXG	Rn, Rn	32	Rn ⇒ Rn
EXT	Dn Dn	8 ⇒ 16 16 ⇒ 32	Sign extended Destination ⇒ Destination
EXTB	Dn	8 ⇒ 32	Sign extended Destination ⇒ Destination
ILLEGAL	none	none	SSP – 2 ⇒ SSP; vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SSP – 2 ⇒ SSP; SR ⇒ (SSP); illegal instruction vector address ⇒ PC
JMP	⟨ea⟩	none	Destination ⇒ PC
JSR	⟨ea⟩	none	SP – 4 ⇒ SP; PC ⇒ (SP); destination ⇒ PC
LEA	⟨ea⟩, An	32	⟨ea⟩ ⇒ An
LINK	An, #⟨d⟩	16, 32	SP – 4 ⇒ SP, An ⇒ (SP); SP ⇒ An, SP + d ⇒ SP
LPSTOP[1]	#⟨data⟩	none	Data ⇒ SR; interrupt mask ⇒ EBI; STOP
LSL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
MOVE	⟨ea⟩, ⟨ea⟩	8, 16, 32	Source ⇒ Destination
MOVEA	⟨ea⟩, An	16, 32 ⇒ 32	Source ⇒ Destination
MOVEA[1]	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVE from CCR	CCR, ⟨ea⟩	16	CCR ⇒ Destination
MOVE to CCR	⟨ea⟩, CCR	16	Source ⇒ CCR
MOVE from SR[1]	SR, ⟨ea⟩	16	SR ⇒ Destination
MOVE to SR[1]	⟨εα⟩, SR	16	Source ⇒ SR
MOVE USP[1]	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVEC[1]	Rc, Rn Rn, Rc	32 32	Rc ⇒ Rn Rn ⇒ Rc
MOVEM	list, ⟨ea⟩ ⟨ea⟩, list	16, 32 16, 32 ⇒ 32	Listed registers ⇒ Destination Source ⇒ Listed registers
MOVEP	Dn, (d_{16}, An) (d_{16}, An), Dn	16, 32	Dn [31:24] ⇒ (An + d); Dn [23:16] ⇒ (An + d + 2); Dn [15:8] ⇒ (An + d + 4); Dn [7:0] ⇒ (An + d + 6) (An + d) ⇒ Dn [31:24]; (An + d + 2) ⇒ Dn [23:16]; (An + d + 4) ⇒ Dn [15:8]; (An + d + 6) ⇒ Dn [7:0]

Table 23 Instruction Set Summary

Instruction	Syntax	Operand Size	Operation
MOVEQ	#⟨data⟩, Dn	$8 \Rightarrow 32$	Immediate data $\Rightarrow$ Destination
MOVES[1]	Rn, ⟨ea⟩ ⟨ea⟩, Rn	8, 16, 32	Rn $\Rightarrow$ Destination using DFC Source using SFC $\Rightarrow$ Rn
MULS/MULU	⟨ea⟩, Dn ⟨ea⟩, Dl ⟨ea⟩, Dh:Dl	$16 * 16 \Rightarrow 32$ $32 * 32 \Rightarrow 32$ $32 * 32 \Rightarrow 64$	Source $*$ Destination $\Rightarrow$ Destination (signed or unsigned)
NBCD	⟨ea⟩	8 8	$0 - \text{Destination}_{10} - X \Rightarrow$ Destination
NEG	⟨ea⟩	8, 16, 32	$0 - \text{Destination} \Rightarrow$ Destination
NEGX	⟨ea⟩	8, 16, 32	$0 - \text{Destination} - X \Rightarrow$ Destination
NOP	none	none	$PC + 2 \Rightarrow PC$
NOT	⟨ea⟩	8, 16, 32	$\overline{\text{Destination}} \Rightarrow$ Destination
OR	⟨ea⟩, Dn Dn, ⟨ea⟩	8, 16, 32 8, 16, 32	Source; Destination $\Rightarrow$ Destination
ORI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Data; Destination $\Rightarrow$ Destination
ORI to CCR	#⟨data⟩, CCR	16	Source; CCR $\Rightarrow$ SR
ORI to SR[1]	#⟨data⟩, SR	16	Source; SR $\Rightarrow$ SR
PEA	⟨ea⟩	32	$SP - 4 \Rightarrow SP$; ⟨ea⟩ $\Rightarrow$ SP
RESET[1]	none	none	Assert $\overline{\text{RESET}}$ line
ROL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	C ← ← (rotate left)
ROR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	→ → C (rotate right)
ROXL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	C ← ← X (rotate left with extend)
ROXR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	X → → C (rotate right with extend)
RTD	#⟨d⟩	16	$(SP) \Rightarrow PC$; $SP + 4 + d \Rightarrow SP$
RTE[1]	none	none	$(SP) \Rightarrow SR$; $SP + 2 \Rightarrow SP$; $(SP) \Rightarrow PC$; $SP + 4 \Rightarrow SP$; restore stack according to format
RTR	none	none	$(SP) \Rightarrow CCR$; $SP + 2 \Rightarrow SP$; $(SP) \Rightarrow PC$; $SP + 4 \Rightarrow SP$
RTS	none	none	$(SP) \Rightarrow PC$; $SP + 4 \Rightarrow SP$
SBCD	Dn, Dn – (An), – (An)	8 8	$\text{Destination}_{10} - \text{Source}_{10} - X \Rightarrow$ Destination
Scc	⟨ea⟩	8	If condition true, then destination bits are set to one; else, destination bits are cleared to zero
STOP[1]	#⟨data⟩	16	Data $\Rightarrow$ SR; STOP
SUB	⟨ea⟩, Dn Dn, ⟨ea⟩	8, 16, 32	Destination – Source $\Rightarrow$ Destination

Table 23 Instruction Set Summary

Instruction	Syntax	Operand Size	Operation
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
SWAP	Dn	16	MSW ⟷ LSW
TBLS/TBLU	⟨ea⟩, Dn Dym : Dyn, Dn	8, 16, 32	Dyn – Dym ⇒ Temp (Temp ∗ Dn [7:0]) ⇒ Temp (Dym ∗ 256) + Temp ⇒ Dn
TBLSN/TBLUN	⟨ea⟩, Dn Dym : Dyn, Dn	8, 16, 32	Dyn – Dym ⇒ Temp (Temp ∗ Dn [7:0]) / 256 ⇒ Temp Dym + Temp ⇒ Dn
TRAP	#⟨data⟩	none	SSP – 2 ⇒ SSP; format/vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SR ⇒ (SSP); vector address ⇒ PC
TRAPcc	none #⟨data⟩	none 16, 32	If cc true, then TRAP exception
TRAPV	none	none	If V set, then overflow TRAP exception
TST	⟨ea⟩	8, 16, 32	Source – 0, to set condition codes
UNLK	An	32	An ⇒ SP; (SP) ⇒ An, SP + 4 ⇒ SP

NOTE:
1. Privileged instruction

4.7 Background Debugging Mode

The background debugger on the CPU32 is implemented in CPU microcode. The background debugging commands are summarized below.

Table 24 Background Debugging Command Summary

Command	Mnemonic	Description
Read D/A Register	RDREG/RAREG	Read the selected address or data register and return the results through the serial interface.
Write D/A Register	WDREG/WAREG	The data operand is written to the specified address or data register.
Read System Register	RSREG	The specified system control register is read. All registers that can be read in supervisor mode can be read in background mode.
Write System Register	WSREG	The operand data is written into the specified system control register.
Read Memory Location	READ	Read the sized data at the memory location specified by the long-word address. The source function code register (SFC) determines the address space accessed.
Write Memory Location	WRITE	Write the operand data to the memory location specified by the long-word address. The destination function code (DFC) register determines the address space accessed.
Dump Memory Block	DUMP	Used in conjunction with the READ command to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and retrieve the first result. Subsequent operands are retrieved with the DUMP command.
Fill Memory Block	FILL	Used in conjunction with the WRITE command to fill large blocks of memory. Initially, a WRITE is executed to set up the starting address of the block and supply the first operand. The FILL command writes subsequent operands.
Resume Execution	GO	The pipe is flushed and refilled before resuming instruction execution at the current PC.
Patch User Code	CALL	Current program counter is stacked at the location of the current stack pointer. Instruction execution begins at user patch code.
Reset Peripherals	RST	Asserts RESET for 512 clock cycles. The CPU is not reset by this command. Synonymous with the CPU RESET instruction.
No Operation	NOP	NOP performs no operation and can be used as a null command.

5 Time Processor Unit

The time processor unit (TPU) is an intelligent, semi-autonomous microcontroller designed for timing control. The TPU operates simultaneously with the CPU; it processes ROM instructions, schedules tasks, performs input and output, and accesses shared data without CPU intervention. Consequently, setup and service time for each timer event are minimized. The figure below is a simplified block diagram of the TPU.
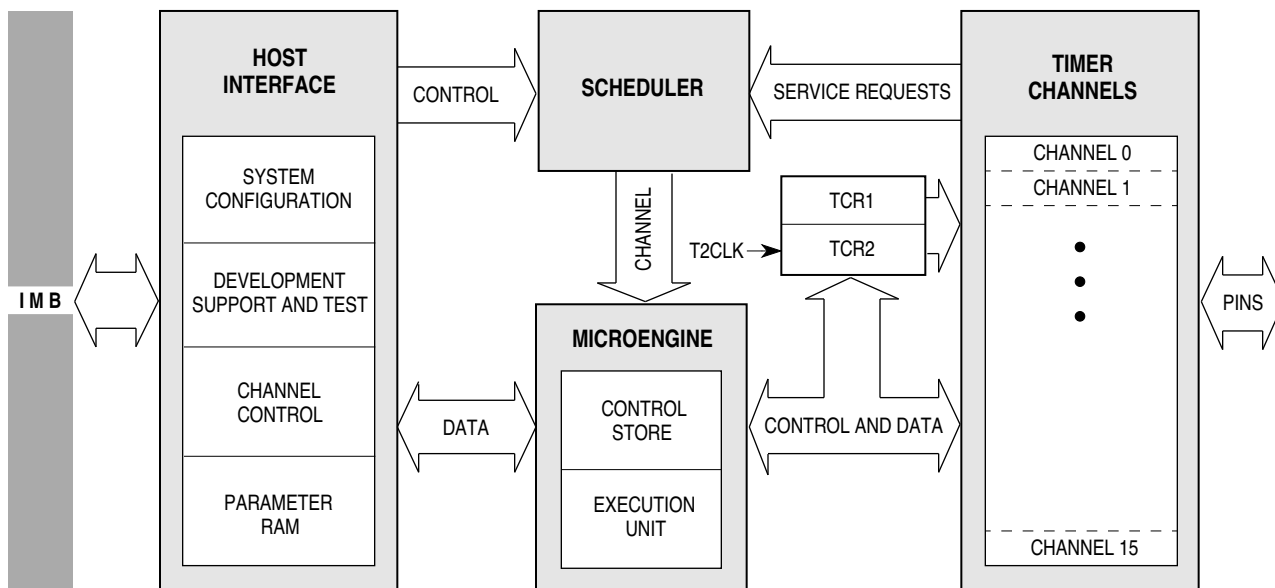
TPU BLOCK

Figure 11 TPU Block Diagram

5.1 Overview

The TPU can be viewed as a special-purpose microcomputer that performs a programmable series of two operations, match and capture. Each occurrence of either operation is called an event. A programmed series of events is called a function. TPU functions replace software functions that would require host CPU interrupt service. Two sets of mask-programmed functions are currently available. Refer to **5.6 Time Functions** for more information.

5.2 Programmer's Model

The TPU control register address map occupies 512 bytes. The "Access" column in the TPU address map below indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the TPUMCR.

Table 25 TPU Address Map

Access	Address	15 8	7 0
S	$YFFE00	TPU MODULE CONFIGURATION REGISTER (TPUMCR)	
S	$YFFE02	TEST CONFIGURATION REGISTER (TCR)	
S	$YFFE04	DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR)	
S	$YFFE06	DEVELOPMENT SUPPORT STATUS REGISTER (DSSR)	
S	$YFFE08	TPU INTERRUPT CONFIGURATION REGISTER (TICR)	
S	$YFFE0A	CHANNEL INTERRUPT ENABLE REGISTER (CIER)	
S	$YFFE0C	CHANNEL FUNCTION SELECTION REGISTER 0 (CFSR0)	
S	$YFFE0E	CHANNEL FUNCTION SELECTION REGISTER 1 (CFSR1)	
S	$YFFE10	CHANNEL FUNCTION SELECTION REGISTER 2 (CFSR2)	
S	$YFFE12	CHANNEL FUNCTION SELECTION REGISTER 3 (CFSR3)	
S/U	$YFFE14	HOST SEQUENCE REGISTER 0 (HSQR0)	
S/U	$YFFE16	HOST SEQUENCE REGISTER 1 (HSQR1)	
S/U	$YFFE18	HOST SERVICE REQUEST REGISTER 0 (HSRR0)	
S/U	$YFFE1A	HOST SERVICE REQUEST REGISTER 1 (HSRR1)	
S	$YFFE1C	CHANNEL PRIORITY REGISTER 0 (CPR0)	
S	$YFFE1E	CHANNEL PRIORITY REGISTER 1 (CPR1)	
S	$YFFE20	CHANNEL INTERRUPT STATUS REGISTER (CISR)	
S	$YFFE22	LINK REGISTER (LR)	
S	$YFFE24	SERVICE GRANT LATCH REGISTER (SGLR)	
S	$YFFE26	DECODED CHANNEL NUMBER REGISTER (DCNR)	

Y = M111, where M represents the logic state of the MM bit in the SIMCR.

5.3 TPU Components

The TPU module consists of two 16-bit time bases, sixteen independent timer channels, a task scheduler, a microengine, and a host interface. In addition, a dual-port parameter RAM is used to pass parameters between the module and the host CPU.

5.3.1 Time Bases

Two 16-bit counters provide reference time bases for all output compare and input capture events. Prescalers for both time bases are controlled by the host CPU via bit fields in the TPU module configuration register (TPUMCR). Timer count registers TCR1 and TCR2 provide access to current counter values. TCR1 and TCR2 can be read/write accessed in microcode, but are not directly available to the host CPU. The TCR1 clock is derived from the system clock. The TCR2 clock can be derived from the system clock or from an external clock input via the T2CLK pin.

5.3.2 Timer Channels

The TPU has 16 independent channels, each connected to an MCU pin. The channels have identical hardware. Each channel consists of an event register and pin control logic. The event register contains a 16-bit capture register, a 16-bit compare/match register, and a 16-bit greater-than-or-equal-to comparator. The direction of each pin, either output or input, is determined by the TPU microengine. Each channel can either use the same time base for match and capture, or can use one time base for match and the other for capture.

5.3.3 Scheduler

When a service request is received, the scheduler determines which TPU channel is serviced by the microengine. A channel can request service for one of four reasons: for host service, for a link to another channel, for a match event, or for a capture event. The host system assigns each active channel one of three priorities: high, middle, or low. When multiple service requests are received simultaneously, a priority-scheduling mechanism grants service based on channel number and assigned priority.

5.3.4 Microengine

The microengine is composed of a control store and an execution unit. Control-store ROM holds the microcode for each factory-masked time function. When assigned to a channel by the scheduler, the execution unit executes microcode for a function assigned to that channel by the host CPU. Microcode can also be executed from the TPURAM module instead of the control store. The TPURAM module allows emulation and development of custom TPU microcode without the generation of a microcode ROM mask. Refer to **5.5 Emulation Support** for more information.

5.3.5 Host Interface

Host interface registers allow communication between the host CPU and the TPU, both before and during execution of a time function. The registers are accessible from the IMB through the TPU bus interface unit.

5.3.6 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the system address map. Channel parameters are organized as 128 16-bit words. Although all parameter word locations in RAM can be accessed by all channels, only 100 are normally used: channels 0 to 13 use six parameter words, while channels 14 and 15 each use eight parameter words. The parameter RAM address map shows how parameter words are organized in memory.

The host CPU specifies function parameters by writing the appropriate RAM address. The TPU reads the RAM to determine channel operation. The TPU can also store information to be read by the CPU in RAM. Detailed descriptions of the parameters required by each time function are beyond the scope of this technical summary. Refer to the *TPU Reference Manual* (TPURM/AD) for more information.

For pre-programmed functions, one of the parameter words associated with each channel contains three channel control fields. These fields perform the following functions:

PSC — Forces the output level of the pin.

PAC — For input capture, PAC specifies the edge transition to be detected. For output comparison, PAC specifies the logic level to be output when a match occurs.

TBS — Specifies channel direction (input or output) and assigns a time base to the input capture and output compare functions of the channel.

Table 26 TPU Parameter RAM Address Map

Channel Number	Base Address	Parameter Address							
		0	1	2	3	4	5	6	7
0	$YFFFF##	00	02	04	06	08	0A	—	—
1	$YFFFF##	10	12	14	16	18	1A	—	—
2	$YFFFF##	20	22	24	26	28	2A	—	—
3	$YFFFF##	30	32	34	36	38	3A	—	—
4	$YFFFF##	40	42	44	46	48	4A	—	—
5	$YFFFF##	50	52	54	56	58	5A	—	—
6	$YFFFF##	60	62	64	66	68	6A	—	—
7	$YFFFF##	70	72	74	76	78	7A	—	—
8	$YFFFF##	80	82	84	86	88	8A	—	—
9	$YFFFF##	90	92	94	96	98	9A	—	—
10	$YFFFF##	A0	A2	A4	A6	A8	AA	—	—
11	$YFFFF##	B0	B2	B4	B6	B8	BA	—	—
12	$YFFFF##	C0	C2	C4	C6	C8	CA	—	—
13	$YFFFF##	D0	D2	D4	D6	D8	DA	—	—
14	$YFFFF##	E0	E2	E4	E6	E8	EA	EC	EE
15	$YFFFF##	F0	F2	F4	F6	F8	FA	FC	FE

— = Not Implemented

Y = M111, where M represents the logic state of the MM bit in the SIMCR.

5.4 TPU Operation

All TPU functions are related to one of the two 16-bit free-running timebases. Functions are synthesized by combining sequences of match events and capture events. Because the primitives are implemented in hardware, the TPU can determine precisely when a match or capture event occurs, and respond rapidly. An event register for each channel provides for simultaneity of match/capture event occurrences on all channels.

When a match or input capture event requiring service occurs, the affected channel generates a service request to the scheduler. The scheduler determines the priority of the request and assigns the channel to the microengine at the first available time. The microengine performs the function defined by the content of the control store or emulation RAM, using parameters from the parameter RAM.

5.4.1 Event Timing

Match and capture events are handled by independent channel hardware. This provides an event accuracy of one time-base clock period, regardless of the number of channels that are active. An event normally causes a channel to request service. However, before an event can be serviced, any pending previous requests must be serviced. The time needed to respond to and service an event is determined by the number of channels requesting service, the relative priorities of the channels requesting service, and the microcode execution time of the active functions. Worst-case event service time (latency) determines TPU performance in a given application. Latency can be closely estimated — see Freescale *TPU Reference Manual* (TPURM/AD) for more information.

5.4.2 Channel Orthogonality

Most timer systems are limited by the fixed number of functions assigned to each pin. All TPU channels contain identical hardware and are functionally equivalent in operation, so that any channel can be configured to perform any time function. Any function can operate on the calling channel, and, under program control, on another channel determined by the program or by a parameter. The user controls the combination of time functions.

5.4.3 Interchannel Communication

The autonomy of the TPU is enhanced by the ability of a channel to affect the operation of one or more other channels without CPU intervention. Interchannel communication can be accomplished by issuing a link service request to another channel, by controlling another channel directly, or by accessing the parameter RAM of another channel.

5.4.4 Programmable Channel Service Priority

The TPU provides a programmable service priority level to each channel. Three priority levels are available. When more than one channel of a given priority requests service at the same time, arbitration is accomplished according to channel number. To prevent a single high-priority channel from permanently blocking other functions, other service requests of the same priority are performed in channel order after the lowest-numbered, highest-priority channel is serviced.

5.4.5 Coherency

For data to be coherent, all available portions of it must be identical in age, or must be logically related. As an example, consider a 32-bit counter value that is read and written as two 16-bit words. The 32-bit value is read-coherent only if both 16-bit portions are updated at the same time, and write-coherent only if both portions take effect at the same time. Parameter RAM hardware supports coherent access of two adjacent 16-bit parameters. The host CPU must use a long-word operation to guarantee coherency.

5.5 Emulation Support

Although factory-programmed time functions can perform a wide variety of control tasks, they may not be ideal for all applications. The TPU provides emulation capability that allows the user to develop new time functions. Emulation mode is entered by setting the EMU bit in the TPUMCR. In emulation mode, an auxiliary bus connection is made between TPURAM and the TPU module, and access to TPURAM via the intermodule bus is disabled. A 9-bit address bus, a 32-bit data bus, and control lines transfer information between the modules. To ensure exact emulation, RAM module access timing remains consistent with access timing of the TPU ROM control store.

To support changing TPU application requirements, Freescale has established a TPU function library. The function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. Refer to Freescale Programming Note TPUPN00/D, *Using the TPU Function Library and TPU Emulation Mode* for information about developing custom functions and accessing the TPU function library. Refer to the *TPU Reference Manual* (TPURM/AD) for more information about specific functions.

5.6 Time Functions

The following paragraphs describe factory-programmed time functions implemented in the TPU microcode ROM. A complete description of the functions is beyond the scope of this summary. Refer to the *TPU Reference Manual* (TPURM/AD) for additional information.

5.6.1 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in parameter RAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing the user to fully define the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table.

5.6.2 Input Capture/Transition Counter (ITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the bus master. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels.

5.6.3 Queued Output Match (QOM)

QOM can generate single or multiple output match events from a table of offsets in parameter RAM. Loop modes allow complex pulse trains to be generated once, a specified number of times, or continuously. The function can be triggered by a link from another TPU channel. In addition, the reference time for the sequence of matches can be obtained from another channel. QOM can generate pulse-width modulated waveforms, including waveforms with high times of 0% or 100%. QOM also allows a TPU channel to be used as a discrete output pin.

5.6.4 Programmable Time Accumulator (PTA)

PTA accumulates a 32-bit sum of the total high time, low time, or period of an input signal over a programmable number of periods or pulses. The accumulation can start on a rising or falling edge. After the specified number of periods or pulses, the PTA generates an interrupt request and optionally generates links to other channels.

From 1 to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, providing instantaneous or average frequency measurement capability, and the latest complete accumulation (over the programmed number of periods).

5.6.5 Multichannel Pulse-Width Modulation (MCPWM)

MCPWM generates pulse-width modulated outputs with full 0% to 100% duty cycle range independent of other TPU activity. This capability requires two TPU channels plus an external gate for one PWM channel. (A simple one-channel PWM capability is supported by the QOM function.)

Multiple PWMs generated by MCPWM have two types of high time alignment: edge aligned and center aligned. Edge aligned mode uses n + 1 TPU channels for n PWMs; center aligned mode uses 2n + 1 channels. Center aligned mode allows a user defined 'dead time' to be specified so that two PWMs can be used to drive an H-bridge without destructive current spikes. This feature is important for motor control applications.

5.6.6 Fast Quadrature Decode (FQD)

FQD is a position feedback function for motor control. It decodes the two signals from a slotted encoder to provide the CPU with a 16-bit free running position counter. FQD incorporates a "speed switch" which disables one of the channels at high speed, allowing faster signals to be decoded. A time stamp is provided on every counter update to allow position interpolation and better velocity determination at low speed or when low resolution encoders are used. The third index channel provided by some encoders is handled by the ITC function.

5.6.7 Universal Asynchronous Receiver/Transmitter (UART)

The UART function uses one or two TPU channels to provide asynchronous communications. Data word length is programmable from 1 to 14 bits. The function supports detection or generation of even, odd, and no parity. Baud rate is freely programmable and can be higher than 100 Kbaud. Eight bidirectional UART channels running in excess of 9600 baud could be implemented on the TPU.

5.6.8 Brushless Motor Commutation (COMM)

This function generates the phase commutation signals for a variety of brushless motors, including three-phase brushless direct current. It derives the commutation state directly from the position decoded in FQD, thus eliminating the need for hall effect sensors.

The state sequence is implemented as a user-configurable state machine, thus providing a flexible approach with other general applications. A CPU offset parameter is provided to allow all the switching angles to be advanced or retarded on the fly by the CPU. This feature is useful for torque maintenance at high speeds.

5.6.9 Frequency Measurement (FQM)

FQM counts the number of input pulses to a TPU channel during a user-defined window period. The function has single shot and continuous modes. No pulses are lost between sample windows in continuous mode. The user selects whether to detect pulses on the rising or falling edge. This function is intended for high speed measurement; measurement of slow pulses with noise rejection can be made with PTA.

5.6.10 Hall Effect Decode (HALLD)

This function decodes the sensor signals from a brushless motor, along with a direction input from the CPU, into a state number. The function supports two- or three-sensor decoding. The decoded state number is written into a COMM channel, which outputs the required commutation drive signals. In addition to brushless motor applications, the function can have more general applications, such as decoding "option" switches.

5.7 TPU Registers

The TPU memory map contains three groups of registers:

> System Configuration Registers
> Channel Control and Status Registers
> Development Support and Test Verification Registers

5.7.1 System Configuration Registers

TPUMCR — TPU Module Configuration Register$\qquad$**$YFFE00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	0	0	IARB			

RESET:

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

STOP — Stop Bit
> 0 = TPU operating normally
> 1 = Internal clocks shut down

TCR1P — 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 4.
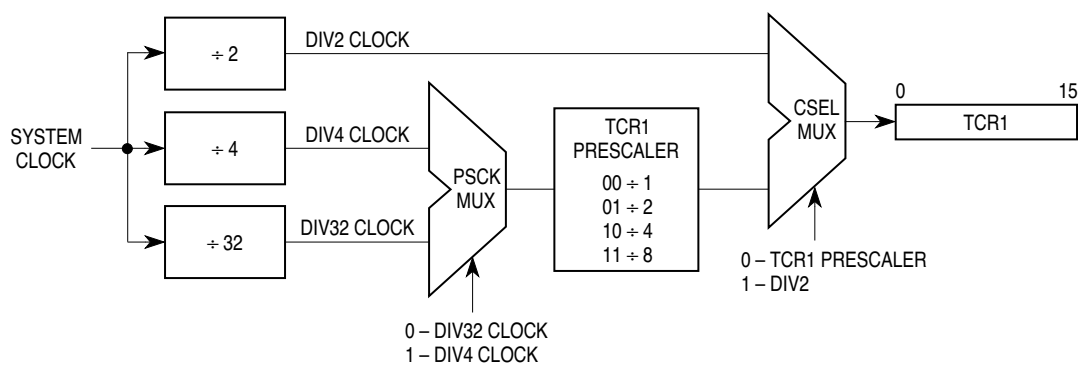
Figure 12 Prescaler Control 1

TCR1 Prescaler	Divide By	PSCK = 0		PSCK = 1	
		Number of Clocks	Rate at 16 MHz	Number of Clocks	Rate at 16 MHz
00	1	32	2 ms	4	250 ns
01	2	64	4 ms	8	500 ns
10	4	128	8 ms	16	1 ms
11	8	256	16 ms	32	2 ms

TCR2P — 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 8. The following table is a summary of prescaler output.

TCR2 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 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
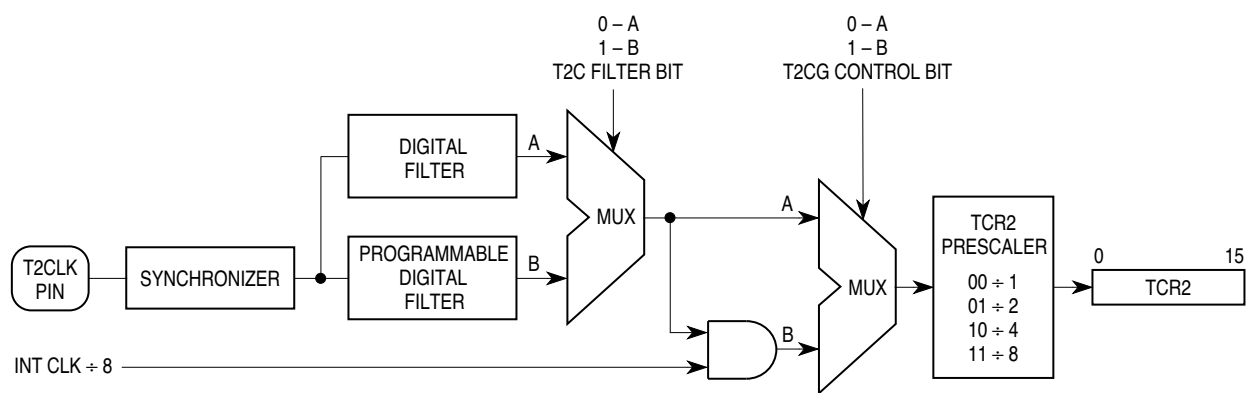
Figure 13 Prescaler Control 2

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, nonzero IARB field value.   Refer to **3.8 Interrupts** for more information.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED					CIRL			CIBV				NOT USED			

RESET:

| | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |

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.

5.7.2 Channel Control Registers

CIER — Channel Interrupt Enable Register $YFFE0A

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 Enable/Disable
    0 = Channel interrupts disabled
    1 = Channel interrupts enabled

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 Bit
    0 = Channel interrupt not asserted
    1 = Channel interrupt asserted

CFSR0 — Channel Function Select Register 0 $YFFE0C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL15				CHANNEL14				CHANNEL13				CHANNEL12			

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR1 — Channel Function Select Register 1 $YFFE0E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL11				CHANNEL10				CHANNEL9				CHANNEL8			

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR2 — Channel Function Select Register 2 $YFFE10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL7				CHANNEL6				CHANNEL5				CHANNEL4			

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR3 — Channel Function Select Register 3 $YFFE12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHANNEL3				CHANNEL2				CHANNEL1				CHANNEL0			

RESET:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CHANNEL[15:0] — Encoded Time Function for each Channel
    Encoded 4-bit fields in the channel function select registers specify one of 16 time functions to be executed on the corresponding channel.

HSQR0 — Host Sequence Register 0   **$YFFE14**

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	

RESET:

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

HSQR1 — Host Sequence Register 1   **$YFFE16**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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] — Encoded Host Sequence

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified.

HSRR0 — Host Service Request Register 0   **$YFFE18**

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	

RESET:

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

HSRR1 — Host Service Request Register 1   **$YFFE1A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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] — Encoded Type of Host Service

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits depends on the time function specified.

A host service request field cleared to %00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three nonzero states. The CPU should monitor the host service request register until the TPU clears the service request to %00 before the CPU changes any parameters or issues a new service request to the channel.

CPR0 — Channel Priority Register 0   **$YFFE1C**

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	

RESET:

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

CPR1 — Channel Priority Register 1   **$YFFE1E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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] — Encoded One of Three Channel Priority Levels

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

5.7.3 Development Support and Test Registers

These registers are used for custom microcode development or for factory test. Describing the use of the registers is beyond the scope of this technical summary. Register names and addresses are given for reference only. Refer to the *TPU Reference Manual* (TPURM/AD) for more information.

DSCR — Development Support Control Register   **$YFFE04**

DSSR — Development Support Status Register   **$YFFE06**

LR — Link Register   **$YFFE22**

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

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

TCR — Test Configuration Register   **$YFFE02**

6 Analog-to-Digital Converter Module

The analog-to-digital converter (ADC) is a unipolar, successive-approximation converter with eight modes of operation. It has selectable 8- or 10-bit resolution. Accuracy is ±1 count (1 LSB) in 8-bit mode and ±4 counts (2 LSB) in 10-bit mode. Monotonicity is guaranteed in both modes. The ADC can perform an 8-bit single conversion (4-clock sample) in 10 microseconds and a 10-bit single conversion in 11 microseconds.

The ADC consists of an analog and a digital subsystem. A block diagram of the converter appears on the following page.

6.1 Analog Subsystem

The analog front end consists of a multiplexer, a buffer amplifier, a resistor-capacitor array, and a high-gain comparator. The multiplexer selects one of eight internal or seven external signal sources for conversion. The resistor capacitor digital-to-analog converter (RC DAC) array performs two functions: it acts as a sample/hold circuit, and it provides the digital-to-analog comparison output necessary for successive approximation conversion. The comparator indicates whether each successive output of the RC DAC array is higher or lower than the sampled input.

6.2 Digital Control Subsystem

The digital control section includes conversion sequence control logic, channel and reference select logic, a successive approximation register, eight result registers, a port data register, and control/status registers. It controls the multiplexer and the output of the RC DAC array during the sample and conversion periods, stores the results of comparison in the successive-approximation register, and transfers the result to a result register.

The ADC bus interface unit (ABIU) contains logic necessary to interface the ADC to the intermodule bus. The ADC is designed to act as a slave device on the bus. The interface must respond with appropriate bus cycle termination signals and supply appropriate interface timing to the other submodules.
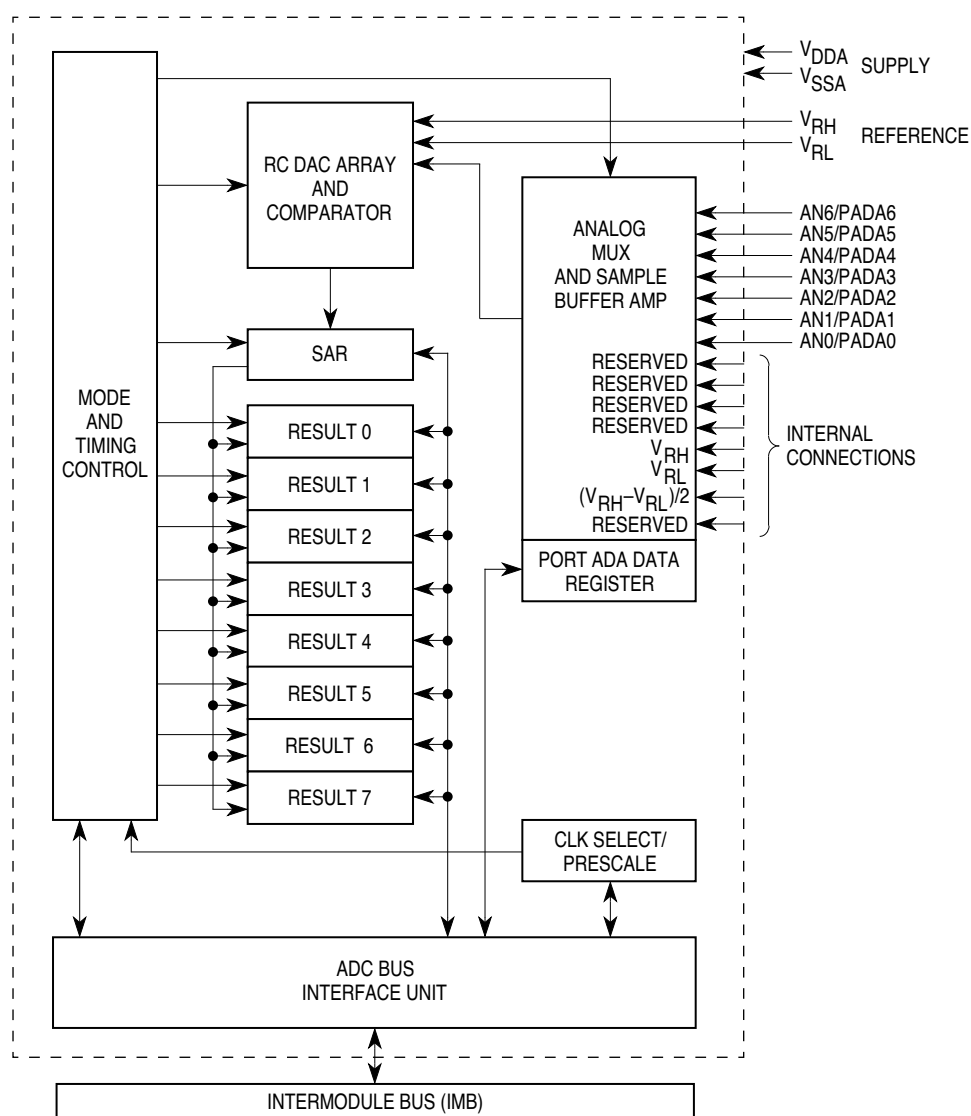
Figure 14 Analog-to-Digital Converter Block Diagram

6.3 ADC Address Map

The ADC control register address map occupies 64 bytes. The "Access" column in the ADC address map below indicates which registers are accessible only at the supervisor privilege level and which can be assigned to either the supervisor or user privilege level, according to the value of the SUPV bit in the ADCMCR.

Table 27 ADC Module Address Map

Access	Address	15 8	7 0
S	$YFF700	ADC MODULE CONFIGURATION REGISTER (ADCMCR)	
S	$YFF702	ADC FACTORY TEST REGISTER (ADTEST)	
S	$YFF704	(RESERVED)	
S/U	$YFF706	PORT ADA DATA (PORTADA)	
S/U	$YFF708	(RESERVED)	
S/U	$YFF70A	ADC CONTROL 0 (ADCTL0)	
S/U	$YFF70C	ADC CONTROL 1 (ADCTL1)	
S/U	$YFF70E	ADC STATUS (ADSTAT)	
S/U	$YFF710	RIGHT-JUSTIFIED UNSIGNED RESULT 0 (RJURR0)	
S/U	$YFF712	RIGHT-JUSTIFIED UNSIGNED RESULT 1 (RJURR1)	
S/U	$YFF714	RIGHT-JUSTIFIED UNSIGNED RESULT 2 (RJURR2)	
S/U	$YFF716	RIGHT-JUSTIFIED UNSIGNED RESULT 3 (RJURR3)	
S/U	$YFF718	RIGHT-JUSTIFIED UNSIGNED RESULT 4 (RJURR4)	
S/U	$YFF71A	RIGHT-JUSTIFIED UNSIGNED RESULT 5 (RJURR5)	
S/U	$YFF71C	RIGHT-JUSTIFIED UNSIGNED RESULT 6 (RJURR6)	
S/U	$YFF71E	RIGHT-JUSTIFIED UNSIGNED RESULT 7 (RJURR7)	
S/U	$YFF720	LEFT-JUSTIFIED SIGNED RESULT 0 (LJSRR0)	
S/U	$YFF722	LEFT-JUSTIFIED SIGNED RESULT 1 (LJSRR1)	
S/U	$YFF724	LEFT-JUSTIFIED SIGNED RESULT 2 (LJSRR2)	
S/U	$YFF726	LEFT-JUSTIFIED SIGNED RESULT 3 (LJSRR3)	
S/U	$YFF728	LEFT-JUSTIFIED SIGNED RESULT 4 (LJSRR4)	
S/U	$YFF72A	LEFT-JUSTIFIED SIGNED RESULT 5 (LJSRR5)	
S/U	$YFF72C	LEFT-JUSTIFIED SIGNED RESULT 6 (LJSRR6)	
S/U	$YFF72E	LEFT-JUSTIFIED SIGNED RESULT 7 (LJSRR7)	
S/U	$YFF730	LEFT-JUSTIFIED UNSIGNED RESULT 0 (LJURR0)	
S/U	$YFF732	LEFT-JUSTIFIED UNSIGNED RESULT 1 (LJURR1)	
S/U	$YFF734	LEFT-JUSTIFIED UNSIGNED RESULT 2 (LJURR2)	
S/U	$YFF736	LEFT-JUSTIFIED UNSIGNED RESULT 3 (LJURR3)	
S/U	$YFF738	LEFT-JUSTIFIED UNSIGNED RESULT 4 (LJURR4)	
S/U	$YFF73A	LEFT-JUSTIFIED UNSIGNED RESULT 5 (LJURR5)	
S/U	$YFF73C	LEFT-JUSTIFIED UNSIGNED RESULT 6 (LJURR6)	
S/U	$YFF73E	LEFT-JUSTIFIED UNSIGNED RESULT 7 (LJURR7)	

Y = M111, where M is the logic state of the MM bit in the SIMCR

6.4 ADC Registers

ADCMCR — ADC Module Configuration Register $YFF700

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ		NOT USED					SUPV	NOT USED						

RESET:

1   0   0   0

     The ADCMCR is used to initialize the ADC.

STOP — STOP Mode
     0 = Normal operation
     1 = Low-power operation
     STOP places the ADC in low-power state by disabling the ADC clock and powering down the analog circuitry. Setting STOP aborts any conversion in progress. STOP is set to logic level one at reset, and may be cleared to logic level zero by the CPU.

     Clearing STOP enables normal ADC operation. However, because analog circuitry bias current has been turned off, there is a period of recovery before output stabilization.

FRZ[1:0] — Freeze 1
     The FRZ field is used to determine ADC response to assertion of the FREEZE signal. The following table shows possible responses.

FRZ	Response
00	Ignore FREEZE
01	Reserved
10	Finish conversion, then freeze
11	Freeze immediately

SUPV — Supervisor/Unrestricted
     0 = Unrestricted access
     1 = Supervisor access
     SUPV defines access to assignable ADC registers.

ADTEST — ADC Test Register $YFF702

     ADTEST is used during factory test of the ADC.

PORTADA — Port ADA Data Register $YFF706

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PORT ADA DATA							

RESET:

Reflects state of the input pins

     Port ADA is an input port that shares pins with the A/D converter inputs. The port ADA data register resides in the lower half of the PDR. Bit 7 always returns zero when read.

PDR[6:0] — Port ADA Data
     A read of PDR[6:0] will return the logic level of the port A pins. If the input is outside the defined voltage range, the result of the read will be indeterminate. Use of a port A pin for digital input does not preclude use as an analog input.

ADCTL0 — A/D Control Register 0   **$YFF70A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				NOT USED				RES10	STS		PRS				

RESET:

| | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

ADCTL0 is used to select ADC clock source and to set up prescaling. Writes to it have immediate effect.

RES10 — 10-Bit Resolution
>   0 = 8-bit resolution
>   1 = 10-bit resolution
Conversion results are appropriately aligned in result registers to reflect conversion status.

STS[1:0] — Sample Time Select
>   Total conversion time depends on initial sample time, transfer time, final sample time, and resolution time. Initial sample time is fixed at two clocks. Transfer time is fixed at two clocks. Resolution time is fixed at ten ADC clock cycles for an 8-bit conversion and twelve ADC clock cycles for a 10-bit conversion. Final sample time depends on the STS field, as shown below.

STS[1:0]	Sample Time
00	2 A/D Clock Periods
01	4 A/D Clock Periods
10	8 A/D Clock Periods
11	16 A/D Clock Periods

PRS[4:0] — Prescaler Rate Selection Field
>   ADC clock is generated from system clock using a modulo counter and a divide-by-two circuit. The binary value of this field is the counter modulus. System clock is divided by the PRS value plus one, then sent to the divide-by-two circuit, as shown in the following table. Maximum ADC clock rate is 2 MHz. Reset value of PRS is a divisor value of eight, resulting in a nominal 2-MHz ADC clock.

PRS[4:0]	Divisor Value
00000	4
00001	4
00010	6
...	...
11101	60
11110	62
11111	64

ADCTL1 — A/D Control Register 1   **$YFF70C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				NOT USED					SCAN	MULT	S8CM	CD	CC	CB	CA

RESET:

| | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ADCTL1 is used to initiate an A/D conversion and to select conversion modes and a conversion channel. It can be written or read at any time. A write to ADCTL1 initiates a conversion sequence. If a conversion sequence is already in progress, a write to ADCTL1 aborts it and resets the SCF and CCF flags in the ADC status register.

SCAN — Scan Mode Selection Bit
      0 = Single conversion sequence
      1 = Continuous conversion
Length of conversion sequence(s) is determined by S8CM.

MULT — Multichannel Conversion Bit
      0 = Conversion sequence(s) run on single channel (channel selected via [CD:CA])
      1 = Sequential conversion of a block of four or eight channels (block selected via [CD:CA])
Length of conversion sequence(s) is determined by S8CM.

S8CM — Select Eight-Conversion Sequence Mode
      0 = Four-conversion sequence
      1 = Eight-conversion sequence
This bit determines the number of conversions in a conversion sequence.

CD:CA — Channel Selection Field
    The bits in this field are used to select an input or block of inputs for A/D conversion.

    The following table summarizes the operation of S8CM and CD:CA when MULT is cleared (single channel mode). Number of conversions per channel is determined by SCAN.

S8CM	CD	CC	CB	CA	Input	Result Register
0	0	0	0	0	AN0	RSLT0 – RSLT3
0	0	0	0	1	AN1	RSLT0 – RSLT3
0	0	0	1	0	AN2	RSLT0 – RSLT3
0	0	0	1	1	AN3	RSLT0 – RSLT3
0	0	1	0	0	AN4	RSLT0 – RSLT3
0	0	1	0	1	AN5	RSLT0 – RSLT3
0	0	1	1	0	AN6	RSLT0 – RSLT3
0	0	1	1	1	AN7*	RSLT0 – RSLT3
0	1	0	0	0	Reserved	RSLT0 – RSLT3
0	1	0	0	1	Reserved	RSLT0 – RSLT3
0	1	0	1	0	Reserved	RSLT0 – RSLT3
0	1	0	1	1	Reserved	RSLT0 – RSLT3
0	1	1	0	0	V_{RH}	RSLT0 – RSLT3
0	1	1	0	1	V_{RL}	RSLT0 – RSLT3
0	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT0 – RSLT3
0	1	1	1	1	Test/Reserved	RSLT0 – RSLT3
1	0	0	0	0	AN0	RSLT0 – RSLT7
1	0	0	0	1	AN1	RSLT0 – RSLT7
1	0	0	1	0	AN2	RSLT0 – RSLT7
1	0	0	1	1	AN3	RSLT0 – RSLT7
1	0	1	0	0	AN4	RSLT0 – RSLT7
1	0	1	0	1	AN5	RSLT0 – RSLT7
1	0	1	1	0	AN6	RSLT0 – RSLT7
1	0	1	1	1	AN7*	RSLT0 – RSLT7
1	1	0	0	0	Reserved	RSLT0 – RSLT7
1	1	0	0	1	Reserved	RSLT0 – RSLT7
1	1	0	1	0	Reserved	RSLT0 – RSLT7
1	1	0	1	1	Reserved	RSLT0 – RSLT7
1	1	1	0	0	V_{RH}	RSLT0 – RSLT7
1	1	1	0	1	V_{RL}	RSLT0 – RSLT7
1	1	1	1	0	$(V_{RH} - V_{RL}) / 2$	RSLT0 – RSLT7
1	1	1	1	1	Test/Reserved	RSLT0 – RSLT7

* Since the ADC in the MCU has only seven external analog inputs, AN7 is connected to V_{SSA}.

The following table summarizes the operation of S8CM and CD:CA when MULT is set (multichannel mode). Number of conversions per channel is determined by SCAN. Channel numbers are given in order of conversion.

S8CM	CD	CC	CB	CA	Input	Result Register
0	0	0	X	X	AN0	RSLT0
					AN1	RSLT1
					AN2	RSLT2
					AN3	RSLT3
0	0	1	X	X	AN4	RSLT0
					AN5	RSLT1
					AN6	RSLT2
					AN7*	RSLT3
0	1	0	X	X	Reserved	RSLT0
					Reserved	RSLT1
					Reserved	RSLT2
					Reserved	RSLT3
0	1	1	X	X	V_{RH}	RSLT0
					V_{RL}	RSLT1
					$(V_{RH} - V_{RL}) / 2$	RSLT2
					Test/Reserved	RSLT3
1	0	X	X	X	AN0	RSLT0
					AN1	RSLT1
					AN2	RSLT2
					AN3	RSLT3
					AN4	RSLT4
					AN5	RSLT5
					AN6	RSLT6
					AN7*	RSLT7
1	1	X	X	X	Reserved	RSLT0
					Reserved	RSLT1
					Reserved	RSLT2
					Reserved	RSLT3
					V_{RH}	RSLT4
					V_{RL}	RSLT5
					$(V_{RH} - V_{RL}) / 2$	RSLT6
					Test/Reserved	RSLT7

* Since the ADC in the MCU has only seven external analog inputs, AN7 is connected to V_{SSA}.

ADSTAT — ADC Status Register $YFF70E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCF	NOT USED				CCTR			CCF							

RESET:

| 0 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ADSTAT contains information related to the status of a conversion sequence.

SCF — Sequence Complete Flag
    0 = Sequence not complete
    1 = Sequence complete
SCF is set at the end of the conversion sequence when SCAN is cleared, and at the end of the first conversion sequence when SCAN is set. SCF is cleared when ADCTL1 is written and a new conversion sequence begins.

CCTR[2:0] — Conversion Counter Field
This field reflects the contents of the conversion counter pointer in either four or eight count conversion sequence. The value corresponds to the number of the next result register to be written, and thus indicates which channel is being converted.

CCF[7:0] — Conversion Complete Field
Each bit in this field corresponds to an A/D result register (CCF7 to RSLT7, etc.). A bit is set when conversion for the corresponding channel is complete, and remains set until the result register is read.

RSLT[7:0] — A/D Result Registers $YFF710–$YFF73E

The result registers store data after conversion is complete. Each register can be read from three different addresses in the register block. Data format depends on the address from which the result register is read.

RJURR — Unsigned Right-Justified Format $YFF710–$YFF71F

Conversion result is unsigned right-justified data. Bits [9:0] are used for 10-bit resolution, bits [7:0] are used for 8-bit resolution (bits [9:8] are zero). Bits [15:10] always return zero when read.

LJSRR — Signed Left-Justified Format $YFF720–$YFF72F

Conversion result is signed left-justified data. Bits [15:6] are used for 10-bit resolution, bits [15:8] are used for 8-bit resolution (bits [7:6] are zero). Although the ADC is unipolar, it is assumed that the zero point is halfway between low and high reference when this format is used. For positive input, bit 15 = 0, for negative input, bit 15 = 1. Bits [5:0] always return zero when read.

LJURR — Unsigned Left-Justified Format $YFF730–$YFF73F

Conversion result is unsigned left-justified data. Bits [15:6] are used for 10-bit resolution, bits [15:8] are used for 8-bit resolution (bits [7:6] are zero). Bits [5:0] always return zero when read.

7 Standby RAM with TPU Emulation

The TPURAM module contains a 1-Kbyte array of fast (two bus cycle) static RAM, which is especially useful for system stacks and variable storage. Alternately, it can be used by the TPU as emulation RAM for new timer algorithms.

7.1 Overview

The 1-Kbyte TPURAM array occupies the top 1024 bytes of a 2-Kbyte block (addresses $400 to $7FF when the ADDR11 bit in TRAMBAR = 0; addresses $0C00 to $0FFF when the ADDR11 bit in TRAM-BAR = 1). The block can be mapped to any 2-Kbyte boundary in the address map, but must not overlap the module control registers (overlap makes the registers inaccessible). Data can be read or written in bytes, word, or long words. TPURAM responds to both program and data space accesses. Data can be read or written in bytes, word, or long words. The TPURAM is powered by V_{DD} in normal operation. During power-down, the TPURAM contents are maintained by power on standby voltage pin V_{STBY}. Power switching between sources is automatic.

Access to the TPURAM array is controlled by the RASP field in TRAMMCR. This field can be encoded so that TPURAM responds to both program and data space accesses. This allows code to be executed from TPURAM, and permits the use of program counter relative addressing mode for operand fetches from the array.

An address map of the TPURAM control registers follows. All TPURAM control registers are located in supervisor data space.

Table 28 TPURAM Control Register Address Map

Access	Address	15 8	7 0
S	$YFFB00	TPURAM MODULE CONFIGURATION REGISTER (TRAMMCR)	
S	$YFFB02	TPURAM TEST REGISTER (TRAMTST)	
S	$YFFB04	TPURAM BASE ADDRESS REGISTER (TRAMBAR)	
	$YFFB06–$YFFB3F	NOT USED	

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

7.2 TPURAM Register Block

There are three TPURAM control registers: the TPURAM module configuration register (TRAMMCR), the TPURAM test register (TRAMTST), and the TPURAM array base address registers (TRAMBAR).

There is an 8-byte minimum register block size for the module. Unimplemented register addresses are read as zeros, and writes have no effect.

7.3 TPURAM Registers

TRAMMCR — TPURAM Module Configuration Register   **$YFFB00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	0	0	0	0	0	0	RASP	NOT USED							

RESET:

0	0	0	0	0	0	0	1

STOP — Stop Control

    0 = TPURAM array operates normally.

    1 = TPURAM array enters low-power stop mode.

This bit controls whether the TPURAM array is in stop mode or normal operation. Reset state is zero, for normal operation. In stop mode, the array retains its contents, but cannot be read or written by the CPU.

RASP — RAM Array Space Field

    0 = TPURAM array is placed in unrestricted space

    1 = TPURAM array is placed in supervisor space

TRAMTST — TPURAM Test Register   **$YFFB02**

    TRAMTST is used for factory testing of the TPURAM module.

TRAMBAR — TPURAM Base Address and Status Register   **$YFFB04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	NOT USED		RAMDS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0			0

ADDR[23:11] — RAM Array Base Address

This field specifies ADDR[23:11] of the TPURAM array base address when the array is enabled. ADDR11 determines the 2-Kbyte boundary that the block of addresses containing the array is mapped to. The array occupies only the upper 1024 bytes of the 2-Kbyte block of addresses. When ADDR11 bit is set, valid array addresses range from $0C00 to%0FFFF; when the ADDR11 bit is cleared, valid array addresses range from $0400 to $07FF.

RAMDS — RAM Array Disable

    0 = TPURAM array is enabled

    1 = TPURAM array is disabled

The TPURAM array is disabled by internal logic after a master reset. Writing a valid base address to the TPURAM array base address field (bits [15:3]) automatically clears RAMDS, enabling the TPURAM array.

7.4 TPURAM Operation

There are six TPURAM operating modes, as follows:

- The TPURAM module is in normal mode when powered by V_{DD}. The array can be accessed by byte, word, or long word. A byte or aligned word (high-order byte is at an even address) access only takes one bus cycle or two system clocks. A long word or misaligned word access requires two bus cycles.
- Standby mode is intended to preserve TPURAM contents when V_{DD} is removed. TPURAM contents are maintained by V_{STBY}. Circuitry within the TPURAM module switches to the higher of V_{DD} or V_{STBY} with no loss of data. When TPURAM is powered by V_{STBY}, access to the array is not guaranteed.
- Reset mode allows the CPU to complete the current bus cycle before resetting. When a synchronous reset occurs while a byte or word TPURAM access is in progress, the access will be completed. If reset occurs during the first word access of a long-word operation, only the first word access will be completed. If reset occurs during the second word access of a long word operation, the entire access will be completed. Data being read from or written to the RAM may be corrupted by asynchronous reset.

- Test mode functions in conjunction with the SIM test functions. Test mode is used during factory test of the MCU.
- Writing the STOP bit of TRAMMCR causes the TPURAM module to enter stop mode. The TPURAM array is disabled (which allows external logic to decode TPURAM addresses, if necessary), but all data is retained. If V_{DD} falls below V_{STBY} during stop mode, internal circuitry switches to V_{STBY}, as in standby mode. Stop mode is exited by clearing the STOP bit.
- The TPURAM array may be used to emulate the microcode ROM in the TPU module. This provides a means of developing custom TPU code. The TPU selects TPU emulation mode. While in TPU emulation mode, the access timing of the TPURAM module matches the timing of the TPU microinstruction ROM to ensure accurate emulation. Normal accesses via the IMB are inhibited and the control registers have no effect, allowing external RAM to emulate the TPURAM at the same addresses.

8 Summary of Changes

This is a partial revision. Most of the publication remains the same, but the following changes were made to improve it. Typographical errors that do not affect content are not annotated.

Page 2	Ordering Information revised to reflect new quantities, TPU mask options.
Page 7	Address map revised to reflect TPURAM array mapping.
Pages 11-42	Expanded and revised SIM section. Added new information to **3.3 System Clock**, **3.5 Chip Selects**, **3.7 Resets**, and **3.8 Interrupts**.
Page 72-74	Revised Standby RAM with TPU Emulation section to include details of array mapping to upper half of 2-Kbyte address block.

freescale™
semiconductor