## Features

- **Low-power Operation Including Special STOP Mode**
- **Frequency: 16.78 MHz at 5V ± 10% Supply and 20.97 MHz at 5V ± 5%, Software Programmable**
- **Technology: 1µ High-density Complementary Metal-Oxide Semiconductor (HCMOS), Static Design**
- **Package: 132-pin Ceramic Leaded Chip Carrier (CERQUAD) and 132-pin Ceramic Pin Grid Array (PGA)**
- **Modular Architecture in a Single Chip**
- **CPU: 32-bit 6800 Family (Upward Object-code Compatible With The 68010)**
- **New Instructions For Controller Applications**
- **Intelligent 16-bit Timer**
  - **16 Independent, Programmable Channels**
  - **Any Channel Can Perform Any Time Function (for Example Input Capture, Output Compare, Pulse Width Modulation, etc.)**
  - **Two timer Count Registers with 2-bit Programmable Prescalers**
  - **Selectable Channel Priority Levels**
  - **Reduced CPU Intervention**
  - **RISC like CPU Within the TPU**
- **Two Serial I/O Subsystems**
  - **Enhanced 68HC11-type Serial Communications Interface (SCI) Universal Asynchronous Receiver Transmitter (UART) with Parity**
  - **Enhanced 68HC11-type Serial Peripheral Interface With I/O RAM Queue (QSPI)**
- **On-chip Memory: 2-Kbytes Standby RAM**
- **On-chip, Programmable, Chip-select Logic**
  - **Up to 12 Signals for Memory and Peripheral Interface with I/O Select**
- **System Failure Protection**
  - **68HC11-type Computer Operating Properly (COP) Watchdog Timer**
  - **68HC11-type Periodic Interrupt Timer**
  - **68000 Family Spurious Interrupt, Halt, and Bus Time-out Monitors**
- **Up to 48 Discrete I/O Pins**

## Description

The TS68332 is a 32-bit microcontroller, combining high-performance data manipulation capabilities with powerful peripheral subsystems. The TS68332 is the first member of the 68300 family of modular embedded controllers featuring fully static, high-speed complementary metal-oxide semiconductor technology. Based on the powerful TS68020, the CPU32 instruction processing module provides enhanced system performance and utilizes the extensive software base of the 68000 family.

# High-performance 32-bit Integrated Microcontroller

# TS68332

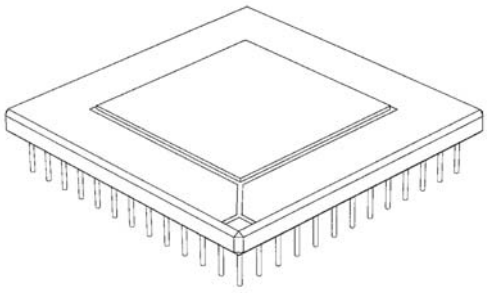
Rev. 2118A–HIREL–03/02

## Screening/Quality

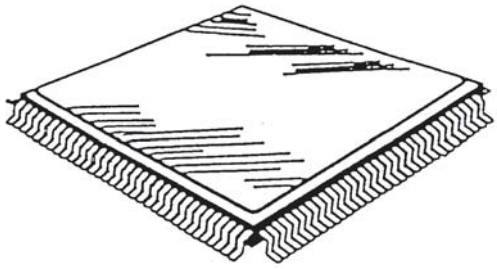
This product is manufactured in full compliance with:

- MIL-STD-883 (class B)
- DSCC 5962-91501
- Or according to Atmel-Grenoble standard

**R suffix**
**PGA 132**
Ceramic Pin Grid Array



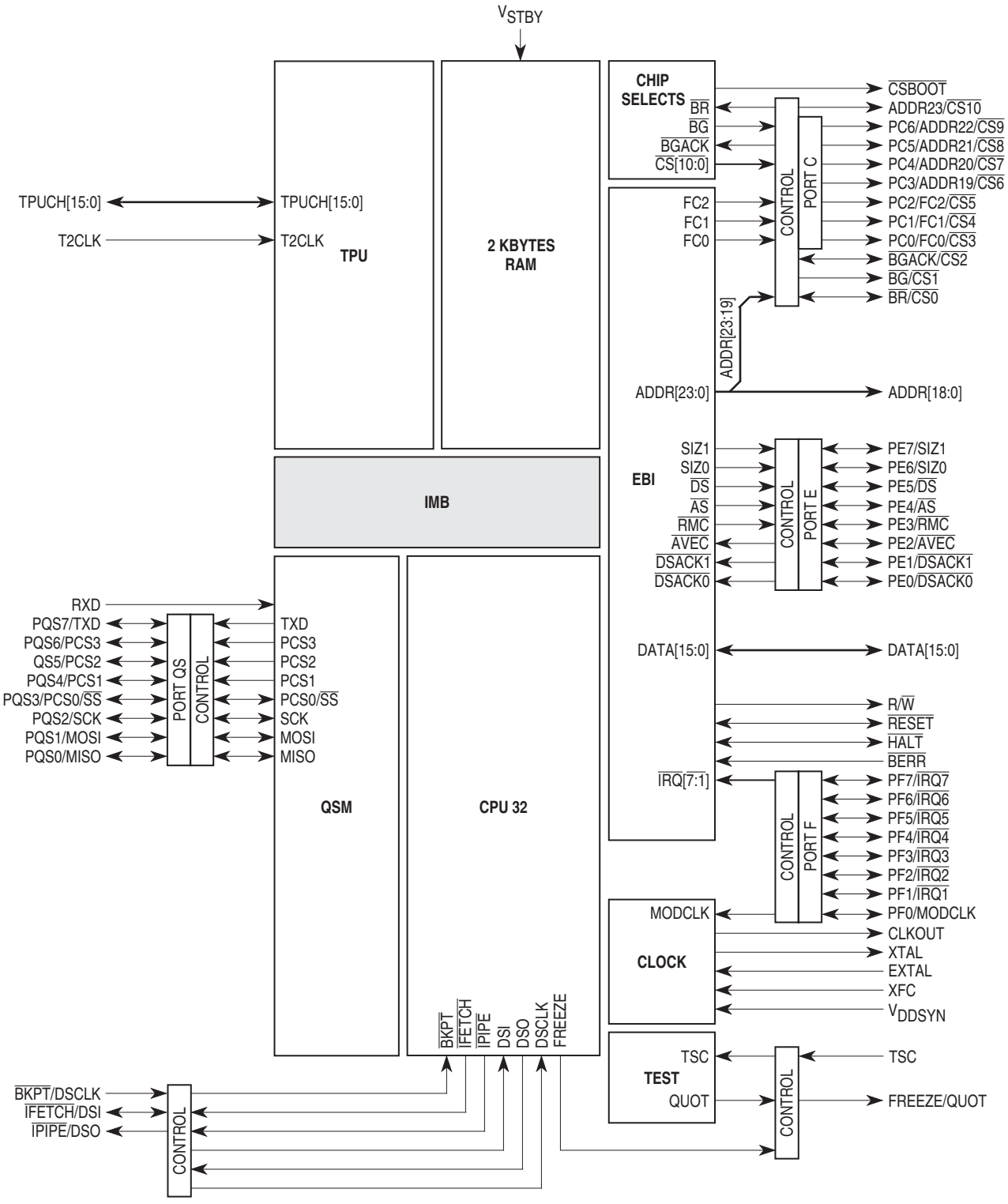
**A suffix**
**CERQUAD 132**
Ceramic Leaded Chip Carrier



## Introduction

Figure 1 is a block diagram of the TS68332 showing the major components. The pin descriptions are provided in Table 1. The TS68332 contains intelligent peripheral modules such as the Time Processor Unit (TPU), which provides 16 microcoded channels for performing time-related activities from simple input capture or output compare to complicated motor control or pulse width modulation. High-speed serial communications are provided by the Queued Serial Module (QSM) with synchronous and asynchronous protocols available. 2-Kbytes of fully static standby RAM allow fast two-cycle access for system and data stacks and variable storage with provision for battery back-up. There is a System Integration Module (SIM) which includes twelve chip selects to enhance system integration for fast external memory or peripheral access. The powerful 32-bit CPU (CPU 32) is based on the industry-standard TS68020. These modules are connected on chip via the Intermodule Bus (IMB) and provide reduced system part count, size, cost of implementation and increased reliability.

**Figure 1.** Block Diagram of TS68332



## Signal Description

Figure 1 illustrates the functional signal groups and Table 1 lists the signals and their function.

**Table 1.** Signal Index

| Signal Name | Mnemonic | Function |
|---|---|---|
| Address Bus | A23 - A0 | 24-bit address bus |
| Data Bus | D15 - D0 | 16-bit data bus used to transfer byte or word data per bus cycle |
| Data Bus Function Codes | FC2 - FC0 | Identify the processor state and the address space of the current bus cycle |
| Boot Chip Select | $\overline{\text{CSBOOT}}$ | Chip-select boot stat up ROM containing user's reset vector and initialization program |
| Chip Selects | $\overline{\text{CS10}}$ - $\overline{\text{CSO}}$ | Enables peripherals at programmed addresses |
| Bus Request | $\overline{\text{BR}}$ | Indicates that an external device requires bus mastership |
| Bus Grant | $\overline{\text{BG}}$ | Indicates that current bus cycle is complete and the TS68332 has relinquished the bus |
| Bus Grant Acknowledge | $\overline{\text{BGACK}}$ | Indicates that an external device has assumed bus mastership |
| Data and Size Acknowledgement | $\overline{\text{DSACK1}}$, $\overline{\text{DSACK0}}$ | Provides asynchronous data transfers and dynamic bus sizing |
| Autovector | $\overline{\text{AVEC}}$ | Requests an automatic vector during an interrupt acknowledge cycle |
| Read-Modify-Write Cycle | $\overline{\text{RMC}}$ | Identifies the bus cycle as part of an indivisible read-modify-write cycle |
| Address Strobe | $\overline{\text{AS}}$ | Indicates that a valid address is on the address bus |
| Data Strobe | $\overline{\text{DS}}$ | During a read cycle, DS indicates that an external device should place valid data on the data bus. During a write cycle, DS indicates that valid data is on the data bus. |
| Size | SIZ1 - SIZ0 | Indicates the number of bytes remaining to be transferred for this cycle |
| Read/Write | R/$\overline{\text{W}}$ | Indicates the direction of data transfer on the bus |
| Interrupt Request Level | $\overline{\text{IRQ7}}$ - $\overline{\text{IRQ0}}$ | Provides an interrupt priority level to the CPU |
| Reset | $\overline{\text{RESET}}$ | System reset |
| Halt | $\overline{\text{HALT}}$ | Suspend external bus activity |
| Bus Error | $\overline{\text{BERR}}$ | Indicates that an erroneous bus operation is being attempted |
| System Clockout | CLKOUT | Internal system clock |
| Crystal Oscillator | EXTAL, XTAL | Connection for an external crystal to the internal oscillator circuit |
| External Filter Capacitor | XFC | Connection pin for an external capacitor to filter the circuit of the phase-locked loop |
| Clock Mode Select | MODCK | Selects the source of the internal system clock |
| Instruction Fetch | $\overline{\text{IFETCH}}$ | Indicates when the CPU is performing an instruction word pre-fetch and when the instruction pipeline has been flushed |
| Instruction Pipe | $\overline{\text{IPIPE}}$ | Used to track movement of words through the instruction pipeline |
| Breakpoint | $\overline{\text{BKPT}}$ | Signals a hardware breakpoint to the CPU |
| Freeze | FREEZE | Indicates that the CPU has acknowledged a breakpoint |
| Quotient Out | QUOT | Serial I/O and clock for background debug mode |
| Test Mode Enable | $\overline{\text{TSTME}}$ | Hardware enable for test mode |
| Three-State Control | TSC | Places all output drivers in a high-impedance state |

**Table 1.** Signal Index (Continued)

| Signal Name | Mnemonic | Function |
|---|---|---|
| Development Serial In, Out, Clock | DSI, DSO, DSCLK | Serial I/O and clock for background debug mode |
| TPU Channels | TP15 - TP0 | TPU channel input/output Serial I/O and clock for background debug mode |
| TPU Clock In | T2CLK | External clock source to the TPU |
| SCI Receive Data | RXD | Serial input to the SCI |
| SCI Transmit Data | TXD | Serial output from the SCI |
| Peripheral Chip Select | $\overline{PCS3}$ - $\overline{PCS0}$ | QSPI peripheral chip selects |
| Slave Select | $\overline{SS}$ | Places the QSPI in slave mode |
| QSPI Serial Clock | SCK | Furnishes the clock from the QSPI in master mode or to the QSPI in slave mode |
| Master-in Slave-out | MISO | Furnishes serial input to the QSPI in master mode, and serial output from the QSPI in slave mode |
| Master-out Slave-in | MOSI | Furnishes serial output from the QSPI in master mode, and serial input to the QSPI in slave mode |
| Standby RAM | $V_{STBY}$ | Power supply for RAM |
| Synchronizer Power | $V_{DDSYN}$ | Power supply to VCO |
| System Power Supply and Return | $V_{DD}$, $V_{SS}$ | Power supply and return to the MCU |

**Figure 2.** PGA Terminal Designation

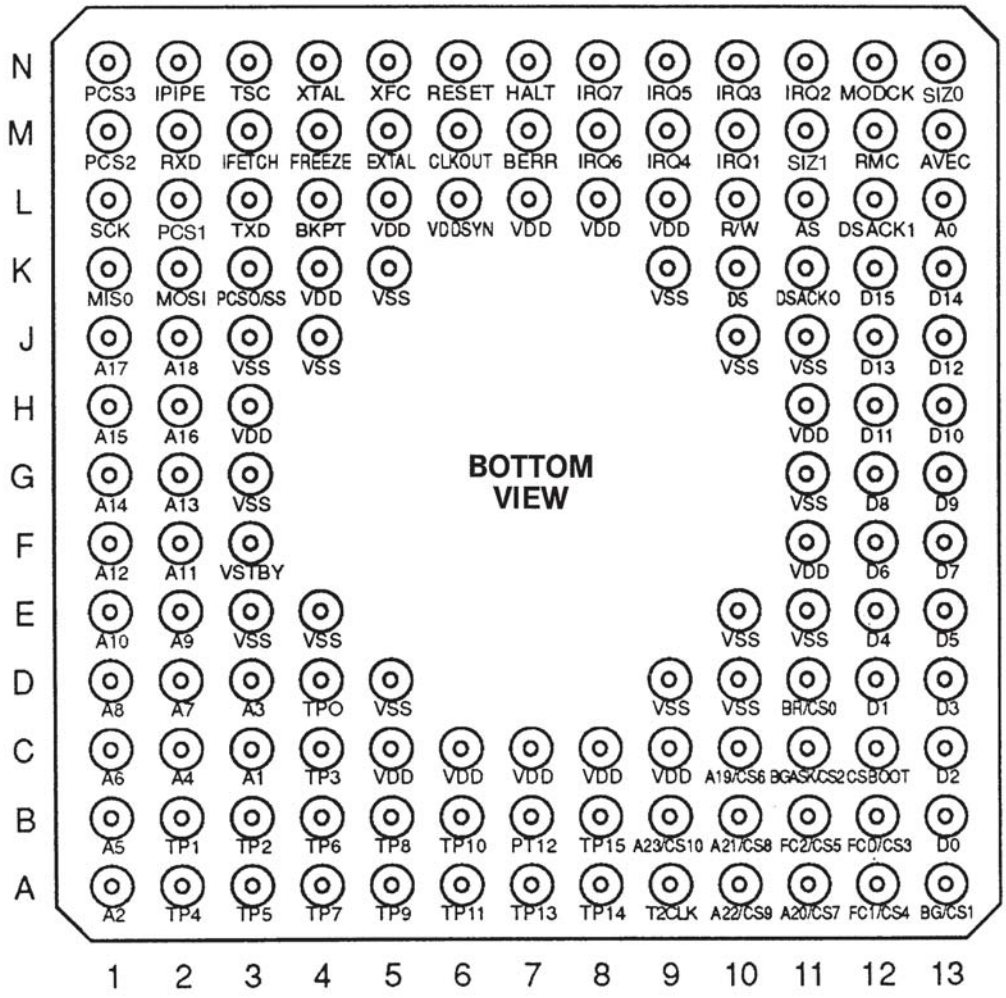
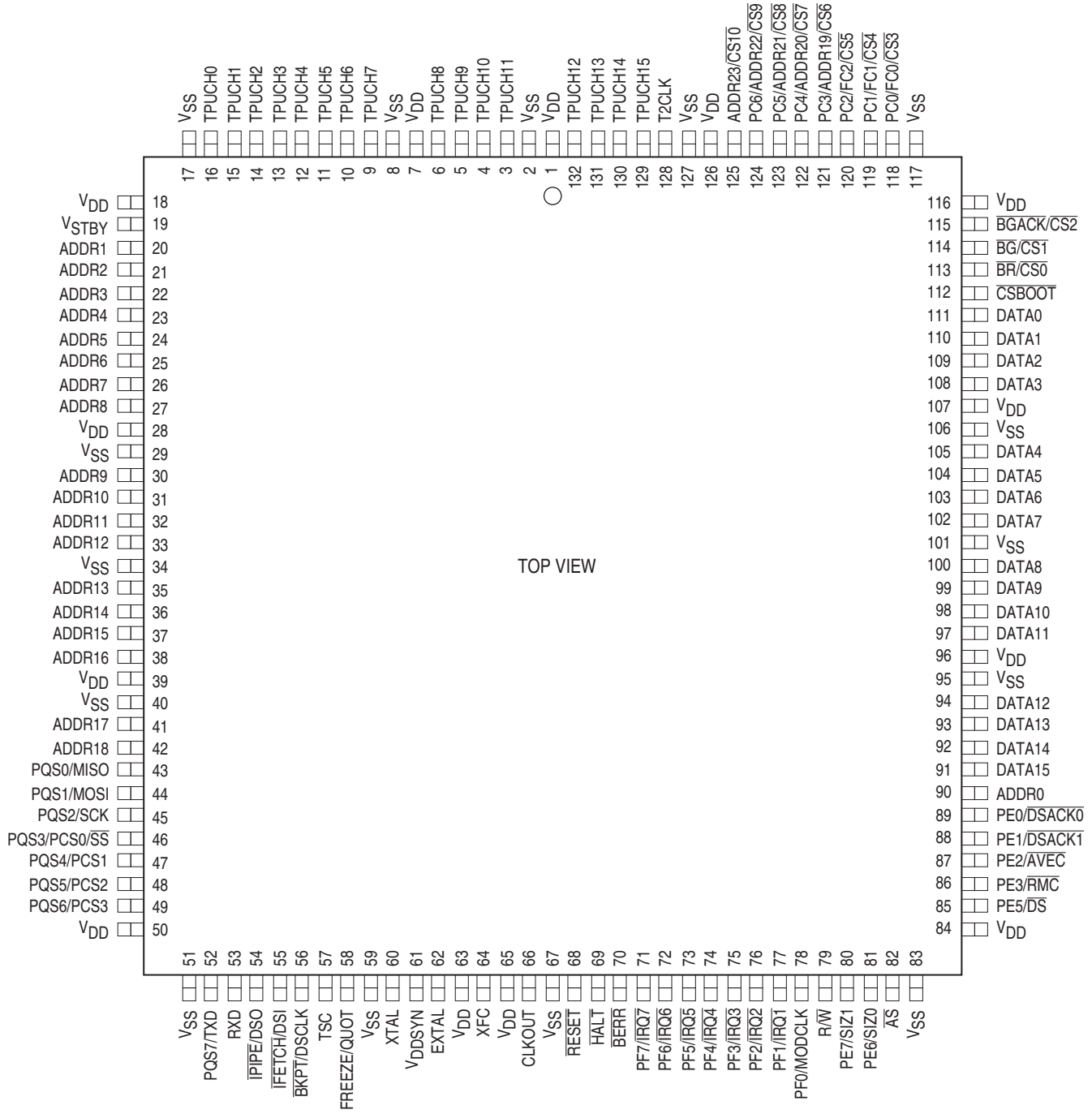


**Figure 3.** CERQUAD Terminal Designation



## Scope

This drawing describes the specific requirements for the microcontroller 68332 at 16.78 MHz and 20.97 MHz, in compliance either with MIL-STD-883 class B or Atmel-Grenoble standard.

## Applicable Documents

### MIL-STD-883

1. MIL-STD-883: test methods and procedures for electronics.
2. MIL-I-38535: general specifications for microcircuits.
3. DSCC Drawing: 5962-91501.

## Requirements

### General

The microcircuits are in accordance with the applicable document and as specified herein.

### Design And Construction

**Terminal Connections**

Depending on the package, the terminal connections shall be as shown in Figure 2 and Figure 3.

**Lead Material and Finish**

Lead material and finish shall be any option of MIL-STD-853.

**Package**

The macrocircuits are packaged in hermetically sealed ceramic packages which conform to case outlines of MIL-M-38510 appendix C (when defined):

- 132-PIN SQ.PGA UP PAE outline
- 132-PIN Ceramic CERQUAD

### Electrical Characteristics

The following ratings define the conditions under which the device operates without damage. Sections of the device may not operate normally while being exposed to the electrical extremes and contains circuitry to protect against damage from high static voltages or electrical fields. It is advised, however, that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage (i.e., either $V_{SS}$ or $V_{DD}$).

**Table 2.** Absolute Maximum Ratings

| Symbol | Parameter | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| $V_{DD}$ | Supply Voltage | | -0.3 | +7.0 | V |
| $V_I$ | Input Voltage | | -0.3 | +7.0 | V |
| $P_{DMAX}$ | Max Power Dissipation | Low Power Operation | | 600 | mW |
| | | Stand By Mode | | 500 | mW |
| $T_{case}$ | Operating Temperature | M Suffix | -55 | +125 | °C |
| | | V Suffix | -40 | +85 | °C |
| $T_{stg}$ | Storage Temperature | | -55 | +150 | °C |
| $T_{leads}$ | Lead Temperature | Max 5 Sec. Soldering | | +270 | °C |

**Table 3.** Thermal Characteristics (at 25°C)

| Package | Symbol | Parameter | Value | Unit |
|---|---|---|---|---|
| PGA 132 | $\theta_{J-A}$ | Thermal Resistance Ceramic Junction-to-ambient | TBD | °C/W |
| | $\theta_{J-C}$ | Thermal Resistance Ceramic Junction-to-case | 10 | °C/W |
| CERQUAD 132 | $\theta_{J-A}$ | Thermal Resistance Ceramic Junction-to-ambient | TBD | °C/W |
| | $\theta_{J-C}$ | Thermal Resistance Ceramic Junction-to-case | 10 | °C/W |

**Power Considerations**

The average chip-junction temperature, $T_J$, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

$T_A$ = Ambient Temperature, °C

$\theta_{JA}$ = Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{CC} \cdot V_{CC}$, Watts - Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins - User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between $P_D$ and $T_J$ (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273) + \theta_{JA} \cdot P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part K can be determined from equation (3) by measuring $P_D$ (at equilibrium) for a known $T_A$. Using this value of K, the values of $P_D$ and $T_J$ can be obtained by solving equations (1) and (2) iteratively for any value of $T_A$.

The total thermal resistance of a package ($\theta_{JA}$) can be separated into two components, $\theta_{JC}$ and $\theta_{CA}$, representing the barrier to heat flow from the semiconductor junction to the package (case), surface ($\theta_{JC)}$ and from the case to the outside ambient ($\theta_{CA}$). These terms are related by the equation:

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \quad (4)$$

$\theta_{JC}$ is device related and cannot be influenced by the user. However, $\theta_{CA}$ is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling and thermal convection. Thus, good thermal management on the part of the user can significantly reduce $\theta_{CA}$ so that $\theta_{JA}$ approximately equals $\theta_{JC}$. Substitution of $\theta_{JC}$ for $\theta_{JA}$ in equation (1) will result in a lower semiconductor junction temperature.

### Mechanical and Environment

The microcircuits shall meet all mechanical environmental requirements of either MIL-STD-883 for class B devices or screened according to Atmel-Grenoble standards devices.

### Marking

The document where are defined the marking are identified in the related reference documents. Each microcircuit are legible and permanently marked with the following information as minimum:

- Atmel logo
- Manufacturer's part number
- Class B identification
- Date-code of inspection lot
- ESD identifier if available
- Country of manufacturing

## Quality Conformance Inspection

### DESC/MIL-STD-883

Is in accordance with MIL-M-38535 and method 5005 of MIL-STD-883. Group A and B inspections are performed on each production lot. Group C and D inspection are performed on a periodical basis.

## Electrical Characteristics

### General Requirements

All static and dynamic electrical characteristics specified and the relevant measurement conditions are given below. For inspection purpose, refer to relevant specification:

- DSCC

(last issue on request to our marketing services)

Table 4: Static electrical characteristics for all electrical variants.

Table 6: Dynamic electrical characteristics for 6832-16 (16.78 MHz).

For static characteristics, test methods refer to IEC 748-2 method number, where existing.

For dynamic characteristics, test methods refer to clause 5.4 hereafter of this specification.

## Static Characteristics

**Table 4.** DC Characteristics. $V_{DD}$ and $V_{DDSYN}$ = 5.0$V_{DC}$ ± 10% for 16.78 MHz and 5.0$V_{DC}$ ± 5% for 20.97 MHz; $V_{SS}$ = 0$V_{DC}$; $T_C$ = -55°C to +125°C or -40°C to +85°C

| Number | Symbol | Parameter | | 16.78 MHz Min | 16.78 MHz Max | 20.97 MHz Min | 20.97 MHz Max | Unit |
|---|---|---|---|---|---|---|---|---|
| 1 | $V_{IH}$ | Input High Voltage | | 0.7 ($V_{DD}$) | $V_{DD}$+0.3 | 0.7($V_{DD}$) | $V_{DD}$+0.3 | V |
| 2 | $V_{IL}$ | Input Low Voltage | | $V_{SS}$ - 0.3 | 0.2($V_{DD}$) | $V_{SS}$ - 0.3 | 0.2($V_{DD}$) | V |
| 3 | $V_{HYS}$ | Input Hysteresis(1) | | 0.5 | - | 0.5 | - | V |
| 4 | $I_{IN}$ | Input Leakage Current(2)<br>$V_{IN}$ = $V_{DD}$ or $V_{SS}$ | Input-only pins | - 2.5 | 2.5 | - 2.5 | 2.5 | µA |
| 5 | $I_{OZ}$ | High Impedance (off-state) Leakage Current (2)<br>$V_{IN}$ = $V_{DD}$ or $V_{SS\,L}$ | All input/output and output pins | - 2.5 | 2.5 | - 2.5 | 2.5 | µA |
| 6 | $V_{OH}$ | CMOS Output High Voltage(2)(3)<br>$I_{OH}$ = -10.0 µA | Group 1, 2, 4 input/output and all output pins and all output pins | $V_{DD}$ - 0.2 | - | $V_{DD}$ - 0.2 | - | V |
| 7 | $V_{OL}$ | CMOS Output High Voltage(2)<br>$I_{OH}$ = -10.0 µA | Group 1, 2, 4 input/output and all output pins | - | 0.2 | - | 0.2 | V |
| 8 | $V_{OH}$ | Output High Voltage(2)(3)<br>$I_{OH}$ = -0.8 mA | Group 1, 2, 4 input/output and all output pins | $V_{DD}$ - 0.8 | - | $V_{DD}$ - 0.8 | - | V |
| 9 | $V_{OL}$ | Output Low Voltage(2)<br>$I_{OL}$ = 1.6 mA | Group 1 I/O pins CLKOUT, FREEZE/QUOT, $\overline{\text{IPIPE}}$ | - | 0.4 | - | 0.4 | V |
| | | $I_{OL}$ = 5.3 mA | Group 2, 4 I/O pins, $\overline{\text{CSBOOT}}$, $\overline{\text{BG}}$/$\overline{\text{CS}}$ | - | 0.4 | - | 0.4 | V |
| | | $I_{OL}$ = 12 mA | Group 3 | - | 0.4 | - | 0.4 | V |
| 10 | $V_{IHTSC}$ | Three State Control Input High Voltage | | 1.6($V_{DD}$) | 9.1 | 1.6($V_{DD}$) | 9.1 | V |
| 11 | $I_{MSP}$ | Data Bus Mode Select Pull-up Current(5) | | | | | | |
| | | $V_{IN}$ = $V_{IL}$ | DATA [15:0] | - | -120 | - | -120 | µA |
| | | $V_{IN}$ = $V_{IH}$ | DATA [15:0] | -15 | - | - 15 | - | µA |
| 12 | | $V_{DD}$ supply current(5) | | | | | | |
| | $I_{DD}$ | RUN(6) | | - | 124 | - | 140 | mA |
| | $I_{DD}$ | RUN, TPU emulation mode | | - | 134 | - | 150 | mA |
| | $S_{IDD}$ | LPSTOP, 32.768 kHz crystal, VCO off (STSIM = 0) | | - | 350 | - | 350 | µA |
| | $S_{IDD}$ | LPSTOP (external clock input frequency = maximum $f_{sys}$) | | - | 5 | - | 5 | mA |

**Table 4.** DC Characteristics. $V_{DD}$ and $V_{DDSYN}$ = 5.0$V_{DC}$ ± 10% for 16.78 MHz and 5.0$V_{DC}$ ± 5% for 20.97 MHz; $V_{SS}$ = 0$V_{DC}$; $T_C$ = -55°C to +125°C or -40°C to +85°C (Continued)

| Number | Symbol | Parameter | | 16.78 MHz Min | 16.78 MHz Max | 20.97 MHz Min | 20.97 MHz Max | Unit |
|---|---|---|---|---|---|---|---|---|
| 13 | $V_{DDSYN}$ | Clock synthesizer operating voltage | | 4.5 | 5.5 | 4.75 | 5.25 | V |
| 14 | | $V_{DDSYN}$ supply current(5) | | | | | | |
| | $I_{DDSYN}$ | 32.768 kHz crystal, VCO on, maximum $f_{sys}$ | | - | 1 | - | 2 | mA |
| | $I_{DDSYN}$ | External clock, maximum $f_{sys}$ | | - | 5 | - | 6 | mA |
| | $S_{IDDSYN}$ | LPSTOP, 32.768 kHz crystal, VCO off (STSIM = 0) | | - | 150 | - | 150 | µA |
| | $I_{DDSYN}$ | 32.768 kHz crystal, $V_{DD}$ powered down | | - | 100 | - | 100 | µA |
| 15 | $V_{SB}$ | RAM standby voltage(7) | | | | | | |
| | | Specified $V_{DD}$ applied | | 0.0 | 5.5 | 0.0 | 5.25 | V |
| | | $V_{DD} = V_{SS}$ | | 3.0 | 5.5 | 3.0 | 5.25 | V |
| 16 | $I_{SB}$ | RAM standby current (5)(7)(8) | | | | | | |
| | | Normal RAM operation | $V_{DD} > V_{SB}$ - 0.5V | - | 10 | - | 10 | µA |
| | | Transient condition | $V_{SB}$ - 0.5 ≥ $V_{DD}$ ≥ $V_{SS}$ + 0.5V | - | 3 | - | 3 | mA |
| | | Standby operation | $V_{DD} < V_{SS}$ + 0.5V | - | 60 | - | 50 | µA |
| 17 | $P_D$ | Power dissipation(9) | | - | 690 | - | 766 | mW |
| 18 | $C_{in}$ | Input capacitance(2)(10) | | | | | | |
| | | | All input-only pins | - | 10 | - | 10 | pF |
| | | | All input/output pins | - | 20 | - | 20 | pF |
| 19 | $C_L$ | Load capacitance(2) | | | | | | |
| | | | Group 1 I/O pins CLKOUT, FREEZE/QUOT, IPIPE | - | 90 | - | 90 | pF |
| | | | Group 2 I/O pins and $\overline{CSBOOT}$, $\overline{BG}/\overline{CS}$ | - | 100 | - | 100 | pF |
| | | | Group 3 I/O pins | - | 130 | - | 130 | pF |
| | | | Group 4 I/O pins | - | 200 | - | 200 | pF |

Notes: 1. Applies to:
Port E [7:4] - $\overline{SIZ}$ [1:0],$\overline{AS}$, $\overline{DS}$.
Port F [7:0] -$\overline{IRQ}$ [7:1], MODCLK.
Port QS [7:0] - TXD, PCS [3:1], PCS0/$\overline{SS}$, SCK, MOSI, MISO.
$\overline{TRUCH}$ [15:0], T2CLK.
$\overline{BKPT}$/DSCLK, $\overline{IFETCH}$, $\overline{RESET}$, RXD, $\overline{TSSTME}$/TSC.
EXTAL (when PLL enabled).

2. Input- only pins: EXTAL, $\overline{TSTME}$/TSC, $\overline{BKPT}$ T2CLK, RXD.
Output-only pins: $\overline{CSBOOT}$, BG/CS, CLKOUT, FREEZE/QUOT, $\overline{I\ PIPE}$.
Input/output pins:
Group 1: DATA [15:0], IFETCH**,** TPUCH [15:01].
Group 2: Port C [6:0] - ADDR [22:19]/$\overline{CS}$ [9:6], FC[2:0]/$\overline{CS}$ [5:3].
Port E: [7:0] - $\overline{SIZ}$ [1:0], $\overline{AS}$, $\overline{DS}$, $\overline{AVEC}$, $\overline{RMC}$, $\overline{DSACK}$ [1:0]
Port F [&:0] - $\overline{IRQ}$ [7:1], MODCLK.
Port QS [7:3] - TXD, PCS [3:1], PCS0/$\overline{SS}$.
ADDR23/$\overline{CS10}$/ECLK, ADDR [18:0], R/$\overline{W}$, $\overline{BERR}$, $\overline{BR}$/$\overline{CSO}$, $\overline{BGACK}$/$\overline{CS2}$.
Group 3: $\overline{HALT}$, $\overline{RESET}$.
Group 4: MISO, MOSI, SCK.

3. Does not apply to $\overline{HALT}$ and $\overline{RESET}$ because they are open drain pins. Does not apply to Port QS [7:0], (TXD, PCS [3:1], PCS0/$\overline{SS}$, SCK, MOSI, MISO) in wired-OR mode.
4. Use of an active pull-down device is recommended.
5. Total operating current is the sum of the appropriate $I_{DD}$, $I_{DDSYN}$, and $I_{SB}$ values. $I_{DD}$ values. $I_{DD}$ values include supply currents for device modules powered by $V_{DDE}$ and $V_{DDI}$ pins
6. Current measured with system clock frequency of 16.78 MHz, all modules active.

7. The RAM module will not switch into standby mode as long as $V_{SB}$ does not exceed $V_{DD}$ by more than 0.5-volt. The RAM array cannot be accessed while the module is in standby mode.
8. When $V_{DD}$ is transitioning during power-up or power-down sequence, and $V_{SB}$ is applied, current flows between the $V_{STBY}$ and $V_{DD}$ pins, which causes standby current to increase toward the maximum transient condition specification. System noise on the $V_{DD}$ and $V_{STBY}$ pins can contribute to this condition.
9. Power dissipation measured at specified system clock frequency, all modules active. Power dissipation can be calculated using the expression:
   $P_D$ = Maximum $V_{DD}$ ($I_{DD}$ + $I_{DDSYN}$ + $I_{SB}$)
   $I_{DD}$ includes supply currents for all device modules powered by $V_{DDE}$ and $V_{DDI}$ pins.
10. This parameter is periodically sampled rather than 100% tested.

## Dynamic (Switching) Characteristics

The INTERVAL numbers refer to the timing diagram.

**Table 5.** Clock Control Timing. $V_{DD}$ and $V_{DDSYN}$ = 5.0 $V_{DC}$ ± 10% for 16.78 MHz and 5.0 $V_{DC}$ ± 5% for 20.97 MHz; $V_{SS}$ = 0 $V_{DC}$; $T_C$ = -55°C to +125°C or -40°C to +85°C

| Number | Symbol | Parameter | 16.78 | | 20.97 | | Unit |
|---|---|---|---|---|---|---|---|
| | | | Min | Max | Min | Max | |
| 1 | $f_{ref}$ | PLL reference frequency range | 25 | 50 | 25 | 50 | kHz |
| 2 | $f_{sys}$ | System frequency(1) | dc | 16.78 | dc | 20.97 | MHz |
| | | On-chip PLL system frequency | 0.131 | 16.78 | 0.131 | 20.97 | MHz |
| | | External clock operation | dc | 16.78 | dc | 20.97 | MHz |
| 3 | $f_{lpll}$ | PLL lock time(2)(3)(4)(5) | - | 20 | - | 20 | ms |
| 4 | $f_{vco}$ | VCO frequency(6) | - | 2 ($f_{sys}$ max) | - | 2 ($f_{sys}$ max) | MHz |
| 5 | $f_{LIMP}$ | Limp mode clock frequency | | | | | |
| | | SYNCR X bit = 0 | - | $f_{sys}$ max/2 | - | $f_{sys}$ max/2 | MHz |
| | | SYNCR X bit = 1 | - | $f_{sys}$ max | - | $f_{sys}$ max | MHz |
| 6 | $C_{stab}$ | CLKOUT stability(2)(3)(4)(7) | | | | | |
| | | Short term (5 µs interval) | -05 | 05 | -05 | 05 | % |
| | | Long term (500 µs interval) | -0.05 | 0.05 | -0.05 | 0.05 | % |

Notes:
1. All internal registers retain data at 0 Hz.
2. This parameter is periodically sampled rather than 100% tested.
3. Assumes that a low-leakage external filter network is used to condition clock synthesizer input voltage. Total external resistance from XFC pin due to external leakage must be greater than 15 MΩ to guarantee this specification. Filter network geometry can vary depending upon operating environment.
4. Proper layout procedures must be followed to achieve specifications.
5. Assumes that stable $V_{DDSYN}$ is applied, and that the crystal oscillator is stable. Lock time is measured from the time $V_{DD}$ and $V_{DDSYN}$ are valid until $\overline{\text{RESET}}$ is released. This specification also applies to the period required for PLL lock after changing the W and Y frequency control bits in the synthesizer control register (SYNCR) while the PLL is running, and to the period required for the clock to lock after LPSTOP.
6. Internal VCO frequency ($f_{VCO}$) is determined by SYNCR W and Y bit values. The SYNCR X bit controls a divide-by-two circuit that is not in the synthesizer feedback loop. When X = 0, the divider is enabled, and $f_{sys}$ = $f_{VCO}$: 4. When X = 1, the divider is disabled, and $f_{sys}$ = $f_{VCO}$: 2. X must equal one when operating at maximum specified $f_{sys}$.
7. Stability is the average deviation from the programmed frequency measured over the specified interval at maximum $f_{sys}$. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the PLL circuitry via $V_{DDSYN}$ and $V_{SS}$ and variation in crystal oscillator frequency increase the Cstab percentage for a given interval. When clock stability is a critical constraint on control system operation, this parameter should be measured during functional testing of the final system.

**Table 6.** AC Timing. $V_{DD}$ and $V_{DDSYN}$ = 5.0 $V_{DC}$ ± 10% for 16.78 MHz and 5.0 $V_{DC}$ ± 5% for 20.97 MHz; $V_{SS}$ = 0 $V_{DC}$; $T_C$ = -55°C to +125°C or -40°C to +85°C [1]

| | | | 16.78 MHz | | 20.97 MHz | | |
|---|---|---|---|---|---|---|---|
| **Number** | **Symbol** | **Parameter** | **Min** | **Max** | **Min** | **Max** | **Unit** |
| F1 | f | Frequency of operation (32.768 kHz crystal)[2] | 0.13 | 16.78 | 0.13 | 20.97 | MHz |
| 1 | $t_{CYC}$ | Clock period | 59.6 | - | 47.7 | - | ns |
| 1A | $t_{Ecyc}$ | ECLK period | 476 | - | 381 | - | ns |
| 1B | $t_{Xcyc}$ | External clock input period[3] | 59.6 | - | 47.7 | - | ns |
| 2, 3 | $t_{CW}$ | Clock pulse width | 24 | - | 18.8 | - | ns |
| 2A, 3A | $t_{ECW}$ | ECLK pulse width | 236 | - | 183 | - | ns |
| 2B, 3B | $t_{XCHL}$ | External clock input high/low time[3] | 29.8 | - | 23.8 | - | ns |
| 4, 5 | $t_{Crf}$ | Clock rise and fall time | - | 5 | - | 5 | ns |
| 4A, 5A | $t_{rt}$ | Rise and fall time - All outputs except CLKOUT | - | 8 | - | 8 | ns |
| 4B, 5B | $T_{XCrf}$ | External clock rise and fall time[4] | - | 5 | - | 5 | ns |
| 6 | $t_{CHAV}$ | Clock high to address, FC, SIZE, $\overline{RMC}$ valid | 0 | 29 | 0 | 23 | ns |
| 7 | $t_{CHAZx}$ | Clock high to address, Data, FC, SIZE, $\overline{RMC}$ high impedance | 0 | 59 | 0 | 47 | ns |
| 8 | $t_{CHAZn}$ | Clock high to address, FC, SIZE, RMC invalid | 0 | - | 0 | - | ns |
| 9 | $t_{CLSA}$ | Clock low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$, asserted | 2 | 25 | 0 | 23 | ns |
| 9A | $t_{STSA}$ | $\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$, asserted (read)[5] | -15 | 15 | -10 | 10 | ns |
| 9C | $t_{CLIA}$ | Clock low to $\overline{IFETCH}$, $\overline{IPIPE}$ asserted | 2 | 22 | 2 | 22 | ns |
| 11 | $t_{AVSA}$ | Address, FC, SIZE, $\overline{RMC}$ valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ read) asserted | 15 | - | 10 | - | ns |
| 12 | $t_{CLSN}$ | Clock low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ negated | 2 | 29 | 2 | 23 | ns |
| 12A | $t_{CLIN}$ | Clock low to $\overline{IFETCH}$, $\overline{IPIPE}$ negated | 2 | 22 | 2 | 22 | ns |
| 13 | $t_{SNAI}$ | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ negated to address, FC, SIZE invalid (address hold) | 15 | - | 10 | - | ns |
| 14 | $t_{SWA}$ | $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ read) width asserted | 100 | - | 80 | - | ns |
| 14A | $t_{SWAW}$ | $\overline{DS}$, $\overline{CS}$, width asserted (write) | 45 | - | 36 | - | ns |
| 14B | $t_{SWDW}$ | $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ read) width asserted (fast write cycle) | 40 | - | 32 | - | ns |
| 15 | $t_{SN}$ | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ width negated[6] | 40 | - | 32 | - | ns |
| 16 | $t_{CHSZ}$ | Clock high to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ high impedance | - | 59 | - | 47 | ns |
| 17 | $t_{SNRN}$ | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ negated to R/$\overline{W}$ negated | 15 | - | 10 | - | ns |
| 18 | $t_{CHRH}$ | Clock high to R/$\overline{W}$ high | 0 | 29 | 0 | 23 | ns |
| 20 | $t_{CHRL}$ | Clock high to R/$\overline{W}$ low | 0 | 29 | 0 | 23 | ns |
| 21 | $t_{RAAA}$ | R/$\overline{W}$ asserted to $\overline{AS}$, $\overline{CS}$ asserted | 15 | - | 10 | - | ns |
| 22 | $t_{RASA}$ | R/$\overline{W}$ low to $\overline{DS}$, $\overline{CS}$ asserted (write) | 70 | - | 54 | - | ns |
| 23 | $t_{CHDO}$ | Clock high to data out valid | - | 29 | - | 23 | ns |

**Table 6.** AC Timing. $V_{DD}$ and $V_{DDSYN}$ = 5.0 $V_{DC}$ ± 10% for 16.78 MHz and 5.0 $V_{DC}$ ± 5% for 20.97 MHz; $V_{SS}$ = 0 $V_{DC}$; $T_C$ = -55°C to +125°C or -40°C to +85°C (Continued)[1]

| Number | Symbol | Parameter | 16.78 MHz | | 20.97 MHz | | Unit |
|---|---|---|---|---|---|---|---|
| | | | Min | Max | Min | Max | |
| 24 | $t_{DVASN}$ | Data out valid to negating edge of $\overline{AS}$, $\overline{CS}$ (fast write cycle) | 15 | - | 10 | - | ns |
| 25 | $t_{SNDOI}$ | $\overline{DS}$, $\overline{CS}$ negated to data out invalid (data out hold) | 15 | - | 10 | - | ns |
| 26 | $t_{DVSA}$ | Data out valid to $\overline{DS}$, $\overline{CS}$ asserted (write) | 15 | - | 10 | - | ns |
| 27 | $t_{DICL}$ | Data in valid to clock low (data setup) | 5 | - | 5 | - | ns |
| 27A | $t_{BELCL}$ | Late $\overline{BERR}$, $\overline{HALT}$ asserted to clock low (setup time) | 20 | - | 15 | - | ns |
| 28 | $t_{SNDN}$ | $\overline{AS}$, $\overline{DS}$ negated to $\overline{DSACK}$ [1:0], $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ negated | 0 | 80 | 0 | 60 | ns |
| 29 | $t_{SNDI}$ | $\overline{DS}$, $\overline{CS}$ negated to data in invalid (data in hold)[7] | 0 | - | 0 | | ns |
| 29A | $t_{SHDI}$ | $\overline{DS}$, $\overline{CS}$ negated to date in high impedance[7][8] | - | 55 | - | 48 | ns |
| 30 | $t_{CLDI}$ | CLKOUT low to data in invalid (fast cycle hold)[7] | 15 | - | 10 | - | ns |
| 30A | $t_{CLDH}$ | CLKOUT low to data in high impedance[7] | - | 90 | - | 72 | ns |
| 31 | $t_{DADI}$ | $\overline{DSACK}$ [1:0] asserted to data in valid[9] | - | 50 | - | 46 | ns |
| 33 | $t_{CLBAN}$ | Clock low to $\overline{BG}$ asserted/negated | - | 29 | - | 23 | ns |
| 35 | $t_{BRAGA}$ | $\overline{BR}$ asserted to $\overline{BG}$ asserted ($\overline{RMC}$ not asserted)[10] | 1 | - | 1 | - | $t_{CYC}$ |
| 37 | $t_{GAGN}$ | $\overline{BGACK}$ asserted to $\overline{BG}$ negated | 1 | 2 | 1 | 2 | $t_{CYC}$ |
| 39 | $t_{GH}$ | $\overline{BG}$ width negated | 2 | - | 2 | - | $t_{CYC}$ |
| 39A | $t_{GA}$ | $\overline{BG}$ width asserted | 1 | - | 1 | - | $t_{CYC}$ |
| 46 | $t_{RWA}$ | R/$\overline{W}$ width asserted (write or read) | 150 | - | 115 | - | ns |
| 46A | $t_{RWAS}$ | R/$\overline{W}$ width asserted (fast write or read cycle) | 90 | - | 70 | - | ns |
| 47A | $t_{AIST}$ | Asynchronous input setup time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$ [1:0], $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$ | 5 | - | 5 | - | ns |
| 47B | $t_{AIHT}$ | Asynchronous input hold time | 15 | - | 12 | - | ns |
| 48 | $t_{DABA}$ | $\overline{DSACK}$ [1:0], asserted to $\overline{BERR}$, $\overline{HALT}$ asserted[11] | - | 30 | - | 30 | ns |
| 53 | $t_{DOCH}$ | Data out hold from clock high | 0 | - | 0 | - | ns |
| 54 | $t_{CHDH}$ | Clock high to data out high impedance | - | 28 | - | 23 | ns |
| 55 | $t_{RADC}$ | R/$\overline{W}$ asserted to data bus impedance change | 40 | - | 32 | - | ns |
| 56 | $t_{HRPW}$ | $\overline{RESET}$ pulse width (reset instruction) | 512 | - | 512 | - | $t_{CYC}$ |
| 57 | $t_{BNHN}$ | $\overline{BERR}$ negated to $\overline{HALT}$ negated (rerun) | 0 | - | 0 | - | ns |
| 70 | $t_{SCLDD}$ | Clock low to data bus driven (show) | 0 | 29 | 0 | 23 | ns |
| 71 | $t_{SCLDS}$ | Data setup time to clock low (show) | 15 | - | 10 | - | ns |
| 72 | $t_{SCLDH}$ | Data hold from clock low (show) | 10 | - | 10 | - | ns |
| 73 | $t_{BKST}$ | $\overline{BKPT}$ input setup time | 15 | - | 10 | - | ns |
| 74 | $t_{BKHT}$ | $\overline{BKPT}$ input hold time | 10 | - | 10 | - | ns |
| 75 | $t_{MSS}$ | Mode select setup time | 20 | - | 20 | - | $t_{CYC}$ |

**Table 6.** AC Timing. $V_{DD}$ and $V_{DDSYN}$ = 5.0 $V_{DC}$ ± 10% for 16.78 MHz and 5.0 $V_{DC}$ ± 5% for 20.97 MHz; $V_{SS}$ = 0 $V_{DC}$; $T_C$ = -55°C to +125°C or -40°C to +85°C (Continued)[1]

| Number | Symbol | Parameter | 16.78 MHz | | 20.97 MHz | | Unit |
|---|---|---|---|---|---|---|---|
| | | | Min | Max | Min | Max | |
| 76 | $t_{MSH}$ | Mode select hold time | 0 | - | 0 | - | ns |
| 77 | $t_{RSTA}$ | $\overline{RESET}$ assertion time[12] | 4 | - | 4 | - | $t_{CYC}$ |
| 78 | $t_{RSTR}$ | $\overline{RESET}$ rise time[13][14] | - | 10 | - | 10 | $t_{CYC}$ |

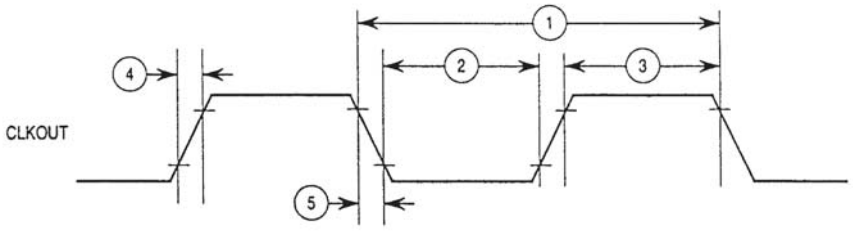
Notes:
1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
2. Minimum system clock frequency is four times the crystal frequency, subject to specified limits.
3. When an external clock is used, minimum high and low times are based on a 50% duty cycle. The minimum allowable $t_{Xcyc}$ period is reduced when the duty cycle of the external clock signal varies. The relationship between external clock input duty cycle and minimum $t_{Xcyc}$ is expressed:
   Minimum $t_{Xcyc}$ period = minimum $t_{XCHL}$/(50% - external input duty cycle tolerance)
4. Parameters for an external clock signal applied while the internal PLL is disabled (MODCLK pin held low during reset). Does not pertain to an external VCO reference applied while the PLL is enabled (MODCLK pin held high during reset). When the PLL is enabled, the clock synthesizer detects successive transitions of the reference signal. If transitions occur within the correct clock period, rise/fall times and duty cycle are at critical.
5. Specification 9A is the worst-case skew between $\overline{AS}$ and $\overline{DS}$ or $\overline{CS}$. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause $\overline{AS}$ and $\overline{DS}$ to fall outside the limits shown in specification 9.
6. If multiple chip selects are used, $\overline{CS}$ width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The $\overline{CS}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
7. Hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
8. Maximum value is equal to ($t_{cyc}$/2) + 25 ns.
9. If the asynchronous setup time (specification 47A) requirements are satisfied, the $\overline{DSACK\ [1:0]}$ low to data setup time (specification 31) and $\overline{DSACK\ [1:0]}$ low to $\overline{BERR}$ low setup time (specification 48) can be ignored. The data must only satisfy the data-in to clock low setup time (specification 27) for the following clock cycle. $\overline{BERR}$ must satisfy only the late $\overline{BERR}$ low to clock low setup time (specification 27A) for the following clock cycle.
10. To ensure coherency during every operand transfer, $\overline{BG}$ will not be asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete and $\overline{RMC}$ is negated.
11. In the absence of $\overline{DSACK\ [1:0]}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time (specification 47A).
12. After external $\overline{RESET}$ negation is detected, a short transition period (approximately 2 $t_{cyc}$) elapses, then the SIM drives $\overline{RESET}$ low for 512 $t_{cyc.}$
13. External assertion of the $\overline{RESET}$ input can overlap internally-generated resets. To insure that an external reset is recognized in all cases, $\overline{RESET}$ must be asserted for at least 590 CLKOUT cycles.
14. External logic must pull $\overline{RESET}$ high during this period in order for normal MCU operation to begin.
15. Address access time = (2.5 + WS) $t_{cyc}$ - $t_{CHAV}$ - $t_{DICL.}$ Chip select access time = (2 + WS) $t_{cyc}$ - $t_{CLSA}$ - $t_{DICL}$.
    Where: WS = number of wait states. When fast termination is used (2 clock bus) WS = - 1.

## Test Conditions Specific to the Device

### Time Definitions

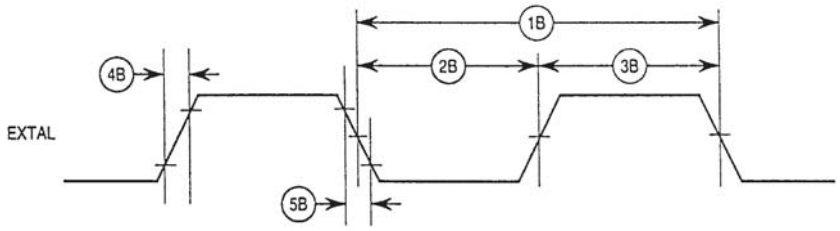
The times specified in Table 6 as dynamic characteristics are defined in Figure 4 to Figure 15 below, by a reference number given the column “NUM” of the tables together with the relevant figure number.

**Figure 4.** Clkout Output Timing Diagram



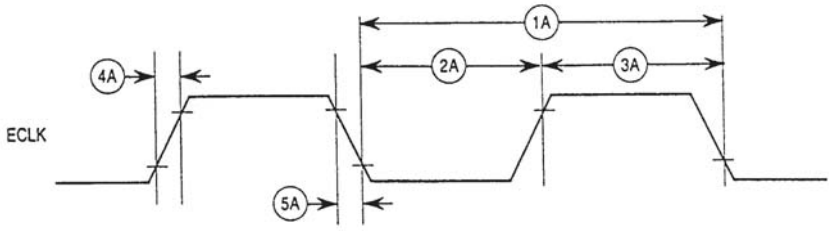

Note: Timing shown with respect to 20% and 70% $V_{DD}$.

**Figure 5.** External Input Timing Diagram




Note: Timing shown with respect to 20% and 70% $V_{DD}$. Pulse width shown with respect to 50% $V_{DD.}$

**Figure 6.** ECLK Output Timing Diagram




Note: Timing Shown With Respect To 20% And 7% $V_{DD}$.

**Figure 7.** Read Cycle Timing Diagram

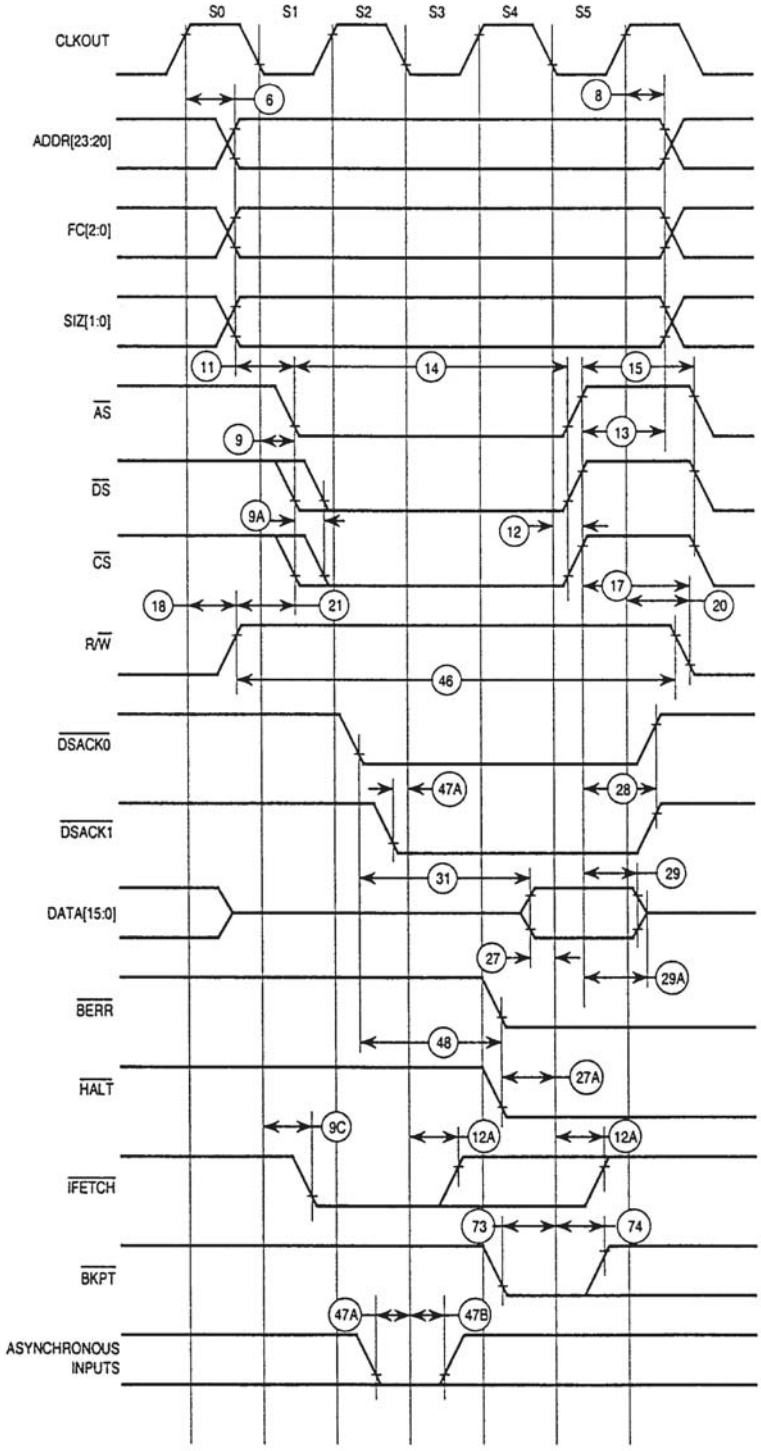


**Figure 8.** Write Cycle Timing Diagram

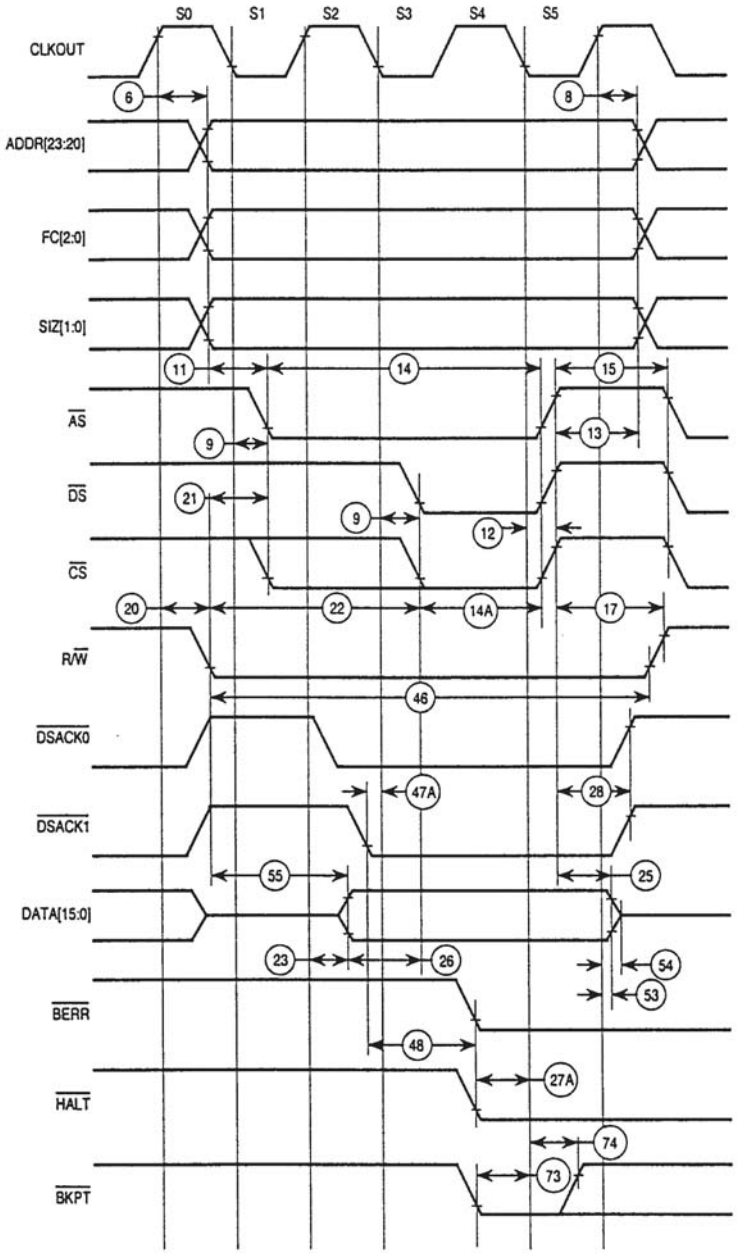


**Figure 9.** Fast Termination Read Cycle Timing Diagram

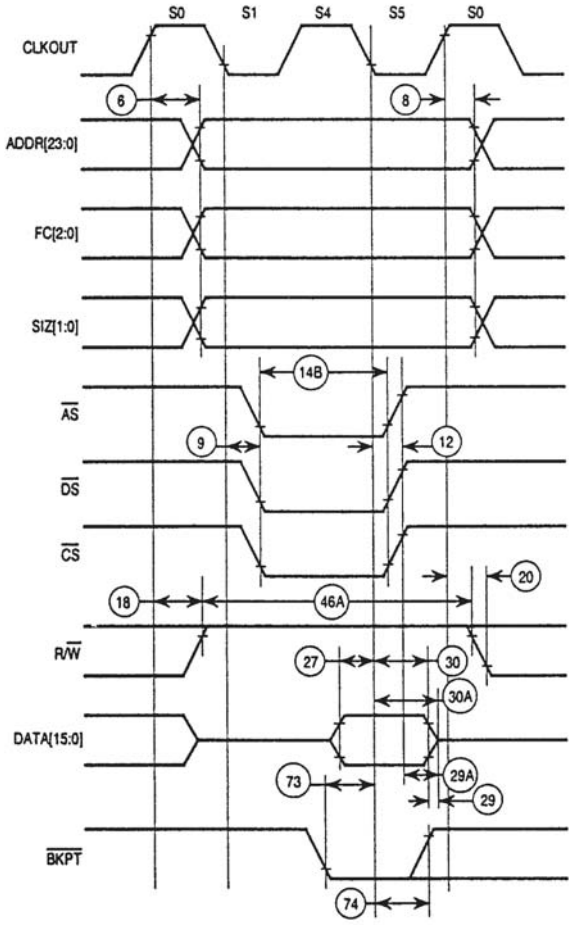


**Figure 10.** Fast Termination Write Cycle Timing Diagram

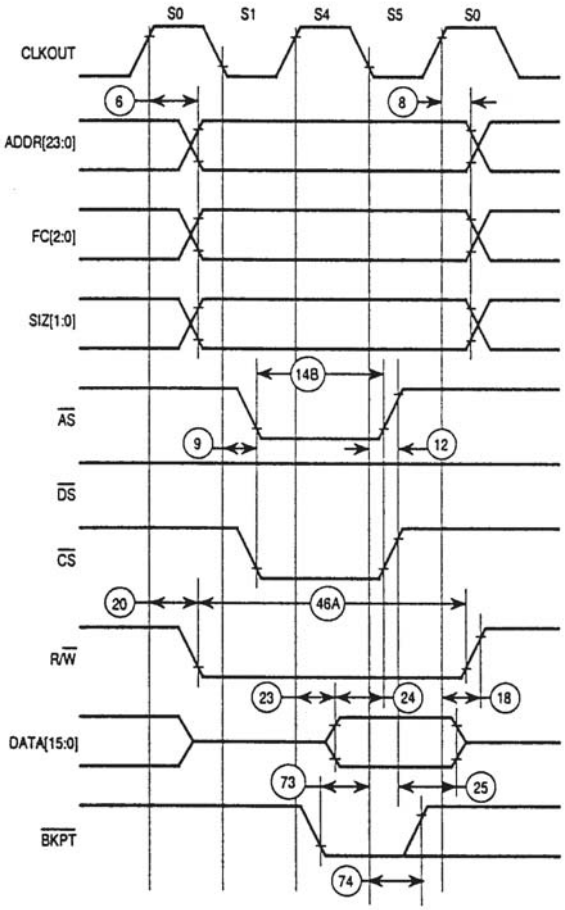


**Figure 11.** Bus Arbitration Timing Diagram – Active Bus Case

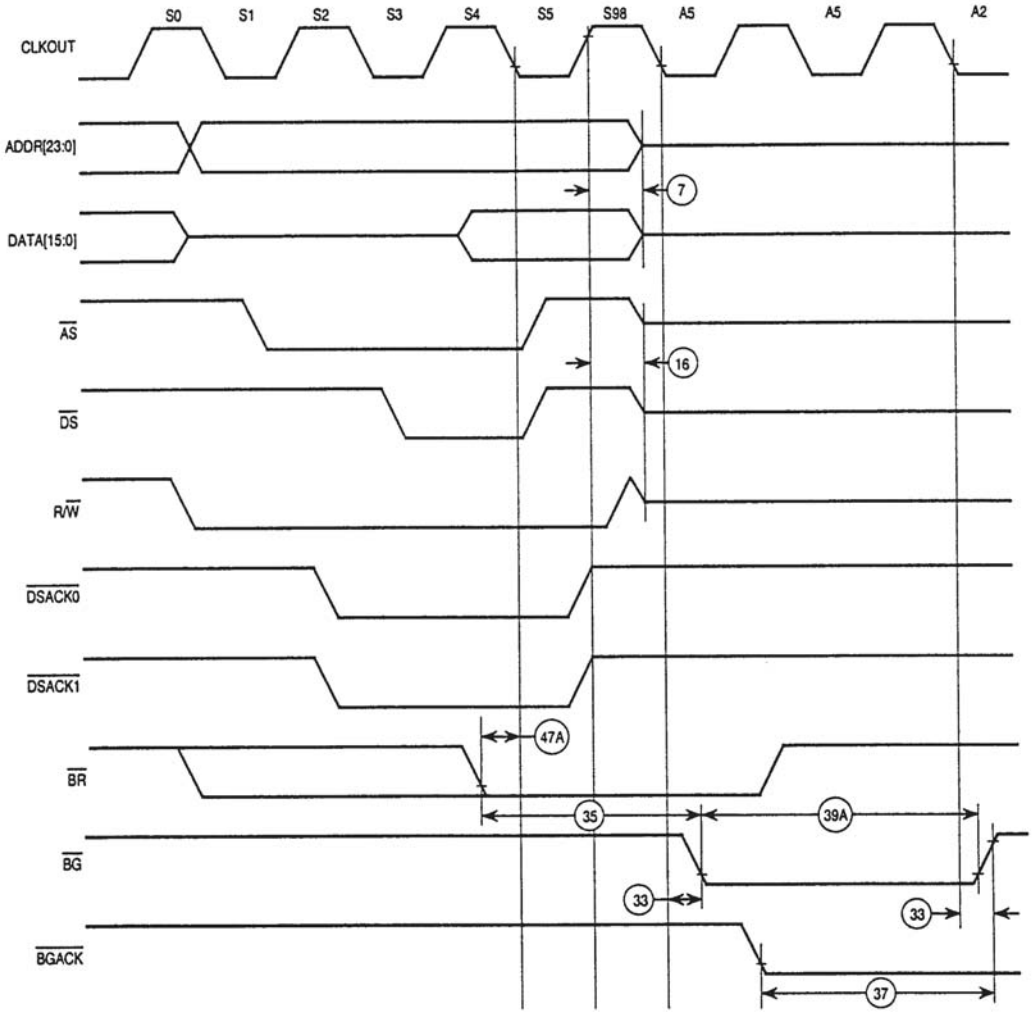


**Figure 12.** Bus Arbitration Timing Diagram – Idle Bus Case

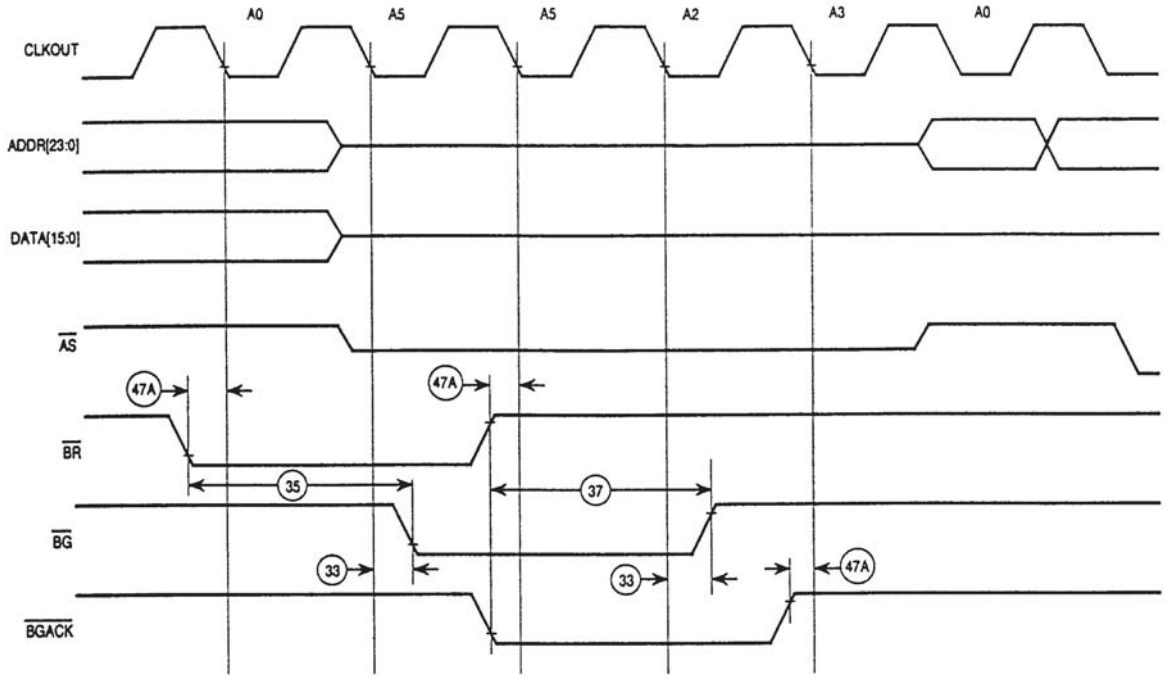
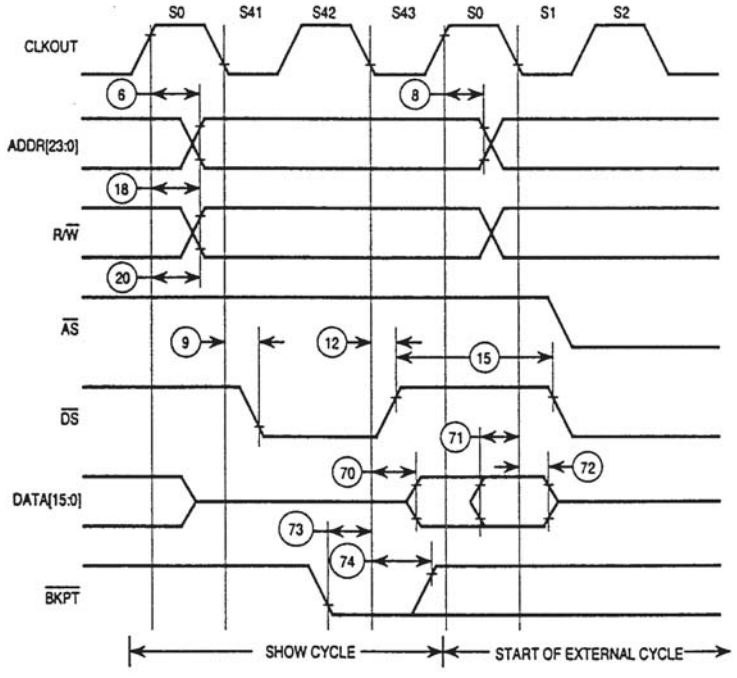
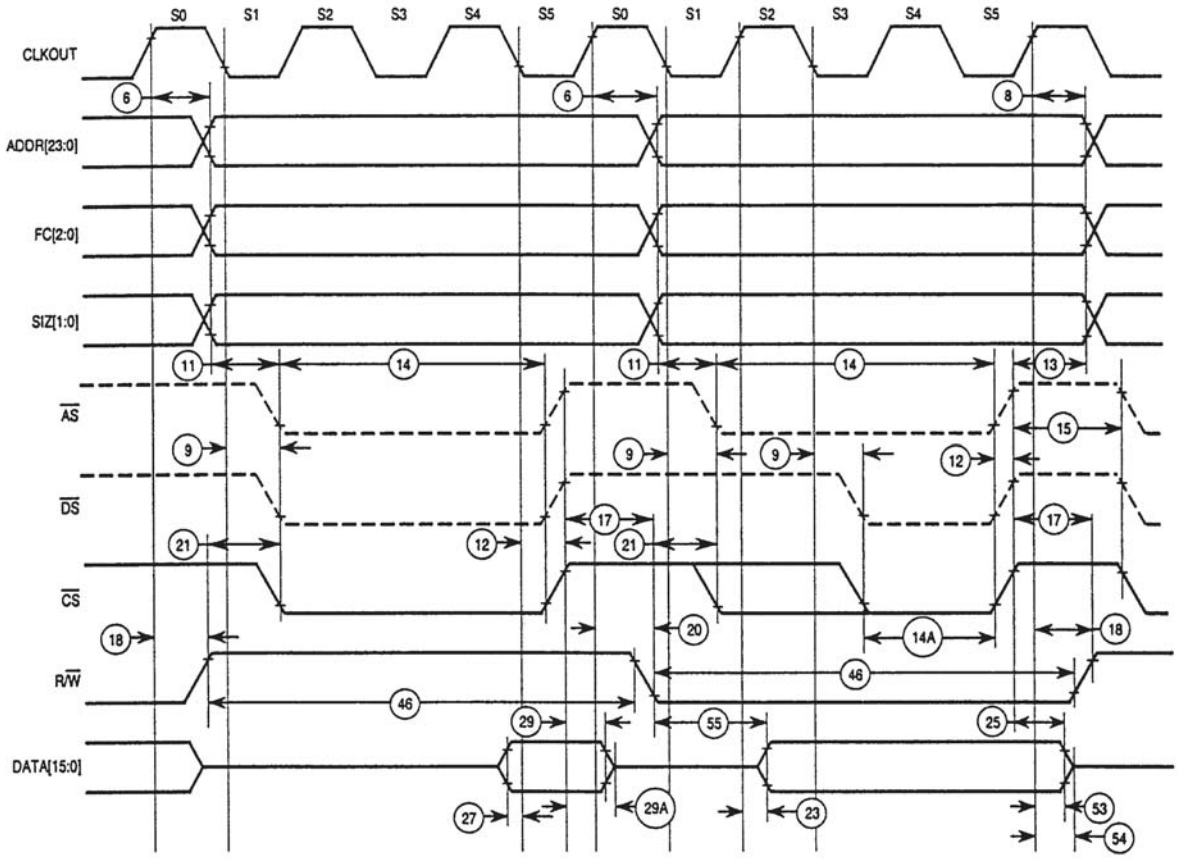



**Figure 13.** Show Cycle Timing Diagram



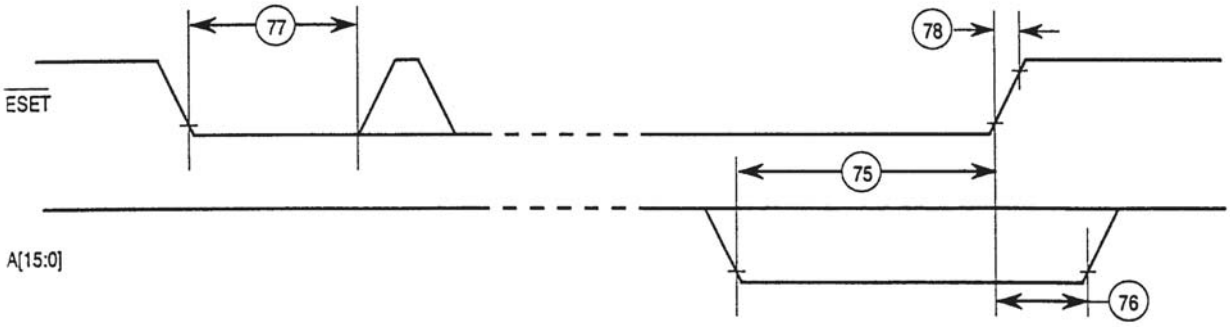

Note: Show cycles can stretch during S42 when bus accesses take longer than two cycles due to IMB module wait-state insertion.

**Figure 14.** Chip Select Timing Diagram




Note: AS and DS timing shown for reference only.

**Figure 15.** Reset and Mode Select Timing Diagram

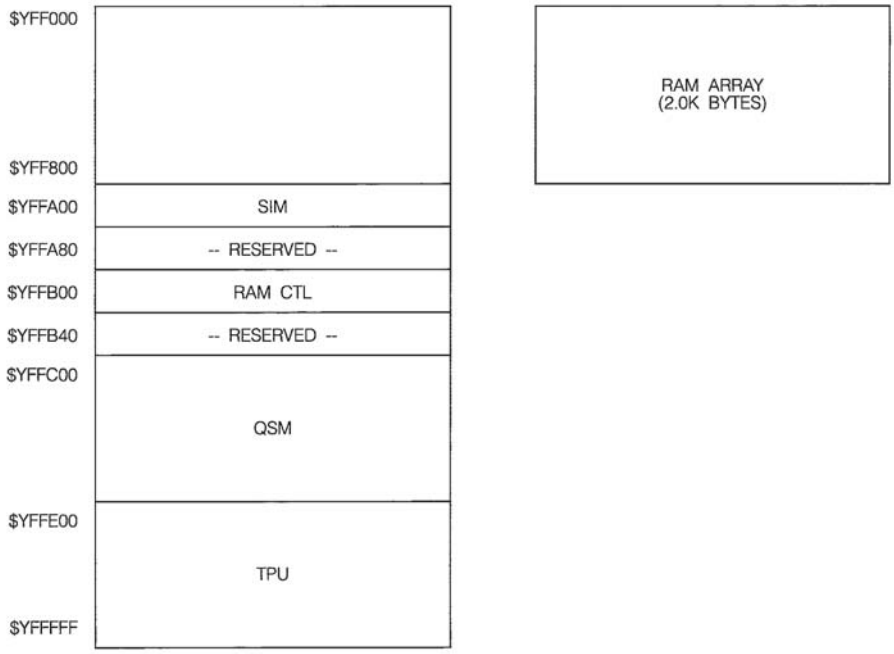


# Functional Description

## Module Memory Map

The RAM array is positioned by the base address register in the RAM CTRL block. Reset forces the RAM array to be disabled. Unimplemented blocks are mapped externally.

**Figure 16.** Module Memory Map




Note: Y - M111, where M is the modmap signal state on the IMB which reflects the state of the modmap bit in the module configuration register of the system integration module (Y = $7 or $F).

## CPU32 Overview

The CPU32, the instruction processing module of the 68300 family, is based on the industry-standard TS68000 core processor with many features of the 68010 and TS68020 as well as unique features suited for high-performance controller applications. The CPU32 is designed to provide a significant increase in performance over existing microcontroller CPUs to meet the demand for higher performance requirements for the 1990s, while maintaining source code and binary code compatibility with the 68000 family.

Ease of programming is an important consideration in using a microcontroller. An instruction format implementing a register-memory interaction philosophy predominates in the design, and all data resources are available to all operations requiring those resources.

All capabilities and functions of this module are detailed fully in the CPU32 reference manual.

**Block Diagram**

The major clocks depicted operate in a highly independent fashion that maximizes concurrency of operation while managing the essential synchronization of instruction execution and bus operation. The bus controller loads instructions from the data bus into the decode unit.

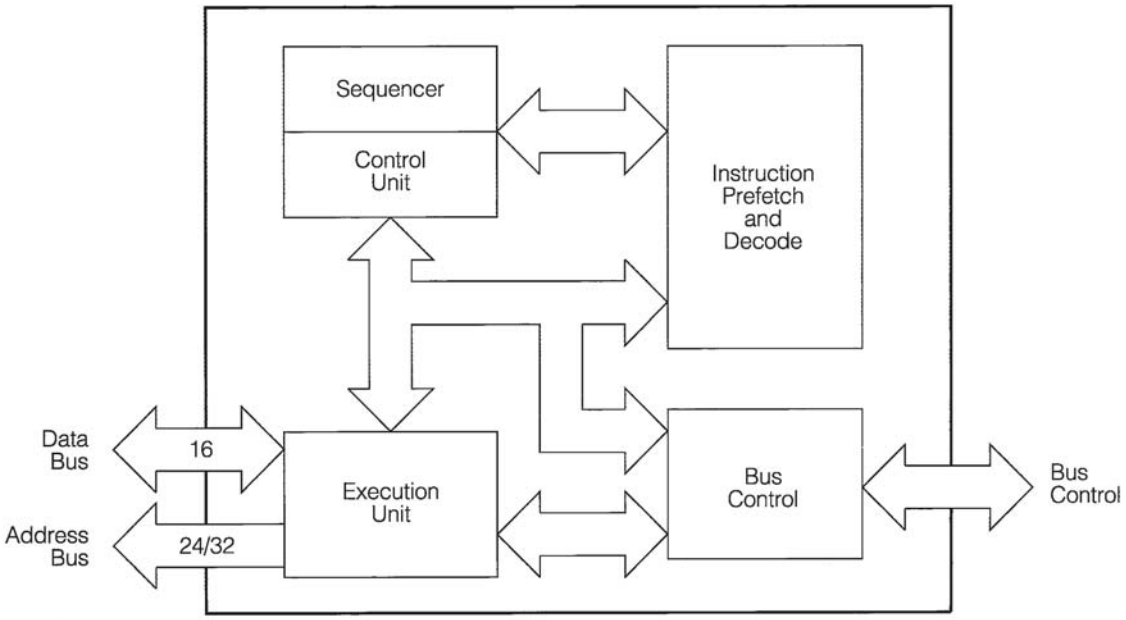
The sequencer and control unit provide overall chip control, managing the internal buses, registers, and functions of the execution unit.

**Architecture Summary**

The CPU32 architecture includes several important features that provide both power and versatility to the user. The CPU32 is source and object code compatible with the TS68000 and 68010. All user-state programs can be executed unchanged. The major CPU32 features are as follows:

- 32-bit internal data path and arithmetic hardware
- 32-bit internal address bus, 24-bit external address bus
- eight 32-bit general-purpose data registers
- seven 32-bit general-purpose address registers
- separate user and supervisor stack pointers and address spaces
- separate program and data address spaces
- full interrupt processing
- fully upward object code compatible with 68000 family
- virtual memory implementation, loop mode of instruction execution,
- fast multiply, divide, and shift instructions
- fast bus interface with dynamic bus port sizing
- improved execution handling for controller applications
- enhanced addressing modes:
  - scaled index
  - address register indirect with base displacement and index
  - expanded PC relative modes 32-bit branch displacements breakpoint instruction.
- instruction set enhancements:
  - high precision multiply and divide
  - trap on condition codes
  - upper and lower bounds checking
  - enhanced breakpoint instruction
- trace on change of flow
- table lookup and interpolate instruction
- low power stop instruction
- hardware breakpoint signal, background mode
- 16.78 MHz and 20.97 MHz operating frequency at -55°C to +125°C
- fully static implementation

**Figure 17.** CPU32 Block Diagram




**Programmer's Model**

The programming model of the CPU32 consists of two groups of registers: user model and supervisor model, which correspond to the user and supervisor privilege levels. Executing at the user privilege level, user programs can only use the registers of the user model. Executing at the supervisor level, system software uses the control registers of the supervisor level to perform supervisor functions.

The supervisor level has higher privileges than the user level. Not all instructions are permitted to execute in the lower privileged user level, but all instructions are available at the supervisor level. This scheme allows a separation of supervisor and user levels, and so the supervisor can protect system resources from uncontrolled access. The processor uses the privilege level indicated by the S bit in the status register to select either the user or supervisor privilege level and either the USP or SSP for stack operations.

The user programming model remains unchanged from previous 68000 family microprocessors. The supervisor programming model, which supplements the user programming model is used exclusively by the CPU32 system programmers who utilize the supervisor privilege level to implement sensitive operating system functions. The supervisor programming model contains all the controls to access and enable the special features of the CPU32. All application software, written to run at the non privileged user level, migrates to the CPU32 from any 68000 platform without modification. The programming models are shown in Figure 18 and Figure 19.

**Registers**

Registers D7-D0 are used as data registers and readily support 8-bit (byte), 16-bit (word) and 32-bit (long word) operand lengths for all operations. Registers A6-A0 and the user and supervisor stack pointers are address registers that may be used as software stack pointers of base address registers. Register A7 is a register that applies to the user stack pointer in the user privilege level and to the supervisor stack pointer in the user privilege level. In addition, the address registers may be used for word and long-word operations. All of the 16 general-purpose registers (D7-D0, A7-A0) may be used as index registers.

The PC contains the address of the next instruction to be executed by the CPU32.

The status register (SR) stores the processor status. It contains the condition codes that reflect the results of a previous operation and can be used for conditional instruction execution in a program.

The vector base register (VBR) contains the base address of the exception vector table in memory. The displacement of an exception vector is added to the value in this register to access the vector table.

Alternate function code registers (SFC and DFC) contain 3-bit function codes. Function codes can be considered extensions of the 24-bit linear address that optionally provide as many as eight 16-Mbyte address spaces. These address spaces are designated as either user or supervisor space and as either program or data space. There is a CPU space to allow the CPU to acquire specific control information not usually associated with read or write bus cycles. The function code signals FC2-FC0 select the appropriate address space.

**Figure 18.** User Programming Model

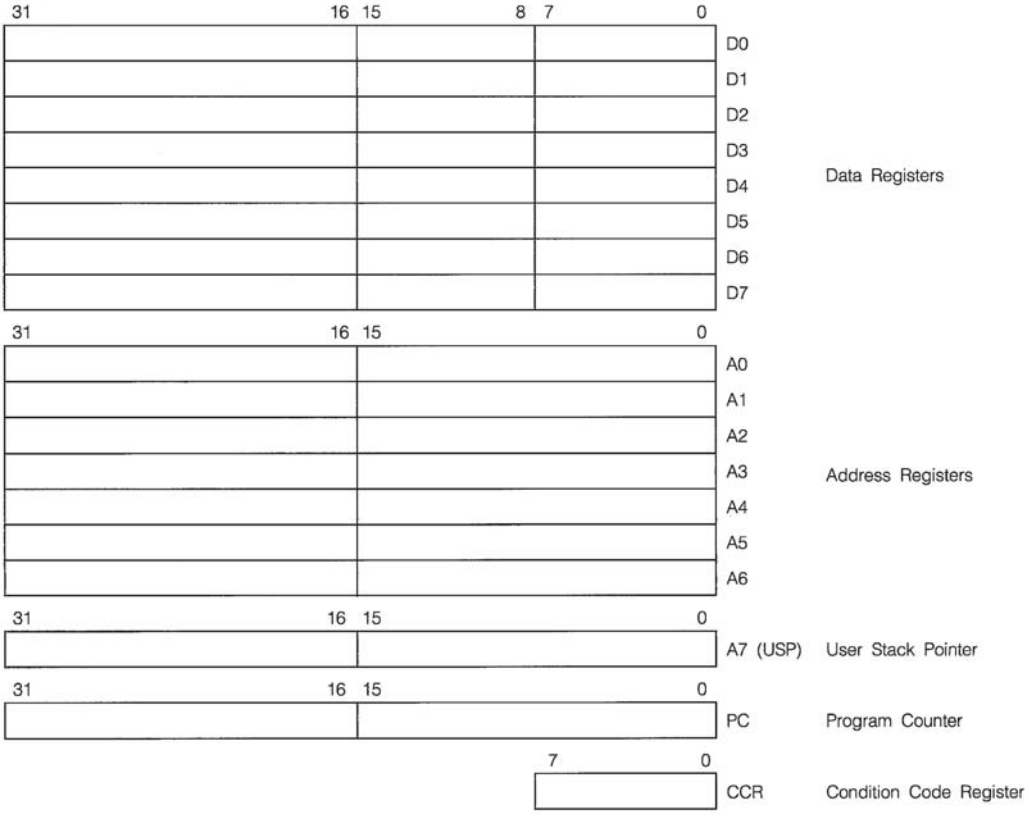
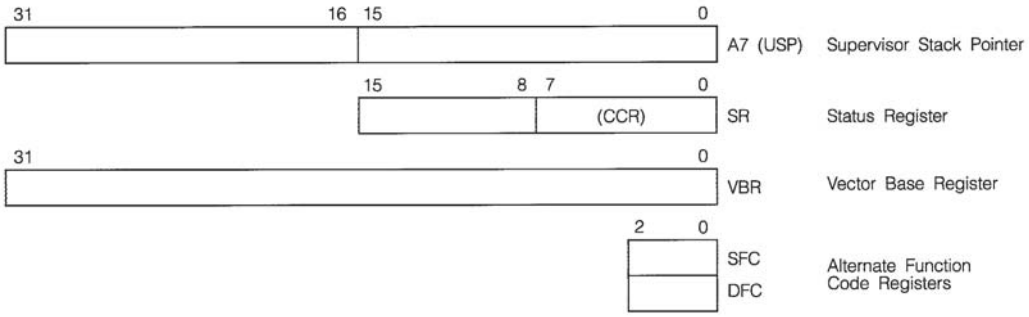


**Figure 19.** Supervisor Programming Model Supplement




**Data Types**

Six basic data types are supported:

- bits
- packaged binary-coded decimal digits
- byte integers (8 bits)
- word integers (16 bits)
- long-word integers (32 bits)
- quad-word integers (64 bits)

*Organization In Registers*

The eight data registers can store data operands of 1, 8, 16, 32 and 64 bits and addresses of 16 or 32 bits. The seven address registers and the two stack pointers are used for address operands of 16 or 32 bits. The PC is 32 bits wide.

**System Features**

The CPU32 includes a number of features to aid system implementation. These include a privilege mechanism, separation of address spaces, multilevel priority interrupts, trap instructions, and a trace facility.

The privilege mechanism provides user and supervisor privilege states, privileged instructions, and external distinction of user and supervisor state references. The processor separates references between program and data space. This permits sharing of code segments that access separate data segments.

The CPU32 supports seven priority levels for 199 memory vectored interrupts. For each interrupt, the vector location can be provided externally or generated internally. The seventh level provides a non-maskable interrupt capability.

To simplify system development, instructions are provided to check internal processor conditions and allow software traps. The trace facility allows instruction-by-instruction tracing of program execution without alteration of the program or special hardware.

**Virtual Memory**

The full addressing range of the CPU32 on the TS68332 is 16-Mbyte in each of eight address spaces. Even though most systems implement a smaller physical memory, the system can be made to appear to have a full 16-Mbyte of memory available to each user program by using virtual memory techniques.

**Loop Mode Instruction Execution**

The CPU32 has several features that provide efficient execution of program loops. One of these features is the DBcc looping primitive instruction. To increase the performance of the CPU32, a loop mode has been added to the processor. The loop mode is used by a single word instruction that does not change program flow. Loop mode is implemented in conjunction with the DBcc instruction. Once in loop mode, the processor performs only the data cycles associated with the instruction and suppresses all instruction fetches.

**Vector Base Register**

The VBR contains the base address of the 1024-byte exception vector table, consisting of 256 exception vectors. Exception vectors contain memory addresses of routines that begin execution at the completion of exception processing, i.e. an interrupt routine.

**Processing States**

The processor is always in one of four processing states: normal, exception, halted or background. The normal processing state is that associated with instruction execution; the bus is used to fetch instructions and operands and to store results. The exception processing state is associated with interrupts, trap instructions, tracing, and other exception conditions. The exception may be internally generated explicitly by an instruction or by an unusual condition arising during the execution of an instruction. Externally, exception processing can be forced by an interrupt, a bus error, or a reset. The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. The background processing state is initiated by breakpoints, execution of special instructions, or a double bus fault. Background processing allows interactive debugging of the system via a simple serial interface.

**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 seven basic addressing modes are as follows:

- 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, pre-decrement, 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, and/or program counter.

## Instructions

*68000 Family Compatibility*

It is the philosophy of the 68000 family that all user-code programs can execute unchanged on a more advanced processor, and supervisor-mode programs and exception handlers should require only minimal alteration.

The CPU32 can be thought of as an intermediate member of the 68000 family. Object code from an TS68000 or 68010 may be executed on the CPU32, and many of the instruction and addressing mode extensions of the TS68020 are also supported. Refer to the CPU32 reference manual for a detailed comparison of the CPU32 and TS68020 instruction set (see also Table 7).

*New Instructions*

Two new instructions have been added to the TS68000 instruction set for use in controller applications. They are low power stop (LPSTOP) and table lookup and interpolate (TBL).

*Low Power Stop (LPSTOP):* In applications where power consumption is a consideration, the CPU32 forces the device into a low-power standby mode when immediate processing is not required. The low-power stop mode is entered by executing the LPSTOP instruction.

The processor will remain in this mode until a user-specification (or higher) interrupt level or reset occurs.

*Table Lookup and Interpolate (TBL):* To maximize throughput for real-time applications, reference data is often “pre-calculated” and stored in memory for quick access. The storage of each data point would require an inordinate amount of memory. The table instruction requires only a sample of data points stored in the array, reducing memory requirements. This single instruction allows intermediate values to be recovered by linear interpolation, thus significantly increasing CPU throughput compared with earlier interpolation methods which used several instructions. The results are optionally rounded with the round-to-nearest algorithm.

## Development Support

The following features have been implemented on the CPU32 to enhance the instrumentation and development environment:

- 68000 family development support,
- background debug mode,
- deterministic opcode tracking,
- hardware breakpoints.

*68000 Family Development Support*

All 68000 family members include features to facilitate applications development. These features include the following:

*Trace On Instruction:* 68000 family processors include an instruction-by-instruction tracing facility as an aid to program development. The CPU32 also allows the user to trace only those instructions causing a change in program flow.

*Breakpoint Instruction:* An emulator may insert software breakpoints into the target code to indicate when a breakpoint has occurred. On the CPU32, this function is provided via illegal instructions, $4848-$484F, to serve as breakpoint instructions.

*Unimplemented Instruction Emulation:* During instruction execution, when an attempt is made to execute an illegal instruction, an illegal instruction exception occurs. Unimplemented instructions (F-line, A-line,...) utilize separate exception vectors to permit efficient emulation of unimplemented instructions in software.

*Background Debug Mode*

Microcomputer systems generally provide a debugger, implemented in software, for system analysis at the lowest level. The background debug mode in the CPU32 is unique in that the debugger has been implemented in CPU microcode. Registers can be viewed and/or altered, memory can be read or written to, and test features can be invoked. Incorporating these capabilities on-chip simplifies the environment in which the in-circuit emulator operates.

*Deterministic Opcode Tracking*

CPU 32 function code outputs are augmented by two supplementary signals to monitor the instruction pipeline. The instruction pipe (PIPE) output indicates the start of each new instruction and each mid-instruction pipeline advance. The instruction fetch (FETCH) output identifies the bus cycles in which the operand is loaded into the instruction pipeline. Pipeline flushes are also signaled with IFETCH. Monitoring these two signals allows a bus analyzer to synchronize itself to the instruction stream and monitor its activity.

*On-chip Breakpoint Hardware*

An external breakpoint trap on any memory access.

**Table 7.** Instruction Set Summary

| Mnemonic | Description |
|---|---|
| ABCD | Add Decimal with Extend |
| ADD | Add |
| ADDA | Add Address |
| ADDI | Add Immediate |
| ADDQ | Add Quick |
| ADDX | Add with Extend |
| AND | Logical AND |
| ANDI | Logical AND Immediate |
| ASL, ASR | Arithmetic Shift Left and Right |
| Bcc | Branch Conditionally |
| BCHG | Test Bit and Change |
| BCLR | Test Bit and Clear |
| BGND | Background |
| BKPT | Breakpoint |
| BRA | Branch |
| BSET | Test Bit and Set |
| BSR | Branch to Subroutine |
| BTST | Test Bit |
| CHK, CHK2 | Check Register Against Upper and Lower Bounds |
| CLR | Clear |
| CMP | Compare |
| CMPA | Compare Address |
| CMPI | Compare Immediate |
| CMPM | Compare Memory to Memory |
| CMP2 | Compare Register Against Upper and Lower Bounds |
| DBcc | Test Condition, Decrement and Branch |
| DIVS, DIVSL | Signed Divide |
| DIVU, DIVUL | Unsigned Divide |

**Table 7.** Instruction Set Summary (Continued)

| Mnemonic | Description |
|---|---|
| EOR | Logical Exclusive OR |
| EORI | Logical Exclusive OR Immediate |
| EXG | Exchange Registers |
| EXT, EXTB | Sign Extend |
| ILLEGAL | Take Illegal Instruction Trap |
| JMP | Jump |
| JSR | Jump to Subroutine |
| LEA | Load Effective Address |
| LINK | Link and Allocate |
| LPSTOP | Low Power Stop |
| LSL, LSR | Logical Shift Left and Right |
| MOVE | Move |
| MOVE CCR | Move Condition Code Register |
| MOVE SR | Move Status Register |
| MOVE USP | Move User Stack Pointer |
| MOVEA | Move Address |
| MOVEC | Move Control Register |
| MOVEM | Move Multiple Registers |
| MOVEP | Move Peripheral |
| MOVEQ | Move Quick |
| MOVES | Move Alternate Addree Space |
| MULS, MULS.L | Signed Multiply |
| MULU, MULU.L | Unsigned Multiply |
| NBCD | Negate Decimal with Extend |
| NEG | Negate |
| NEGX | Negate with Extend |
| NOP | No Operation |
| OR | Logical Inclusive OR |
| ORI | Logical Inclusive OR Immediate |
| PEA | Push Effective Address |
| RESET | Reset External Devices |
| ROL, ROR | Rotate Left and Right |
| ROXL, ROXR | Rotate with Extend Left and Right |
| RTD | Return and De-allocate |
| RTE | Return from Exception |
| RTR | Return and Restore Codes |
| RTS | Return from Subroutine |

**Table 7.** Instruction Set Summary (Continued)

| Mnemonic | Description |
|---|---|
| SBCD | Subtract Decimal with Extend |
| Scc | Set Conditionally |
| STOP | Stop |
| SUB | Subtract |
| SUBA | Subtract Address |
| SUBI | Subtract Immediate |
| SUBQ | Subtract Quick |
| SUBX | Subtract with Extend |
| SWAP | Swap Register Words |
| TBLS, TBLSN | Signed/Unsigned Table Lookup |
| TBLU, TBLUN | and Interpolate |
| TAS | Test Operand and Set |
| TRAP | Trap |
| TRAPcc | Trap Conditionally |
| TRAPV | Trap on Overflow |
| TST | Test Operand |
| UNLK | Unlink |

## Bus Operation

This section provides a functional description of the bus and the signals that control it. Operation of the bus is the same whether the MCU or an external device is the bus master; the names and description of bus cycles are from the point of view of the bus master. The MCU architecture supports byte, word, and long-word operands, allowing access to 8-bit and 16-bit data ports through use of asynchronous cycles controlled by the data transfer (SIZ1 and SIZ0) and data size acknowledge pins ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$).

### Function Codes

The function code signals (FC2 - FC0) select one of eight 16-Mbyte address space to which the address applies.

### Address Bus

The address bus signals (A23 - A0) define the address of the byte (or the most significant byte) to be transferred during a bus cycle. The address is valid while $\overline{\text{AS}}$ asserted.

### Address Strobe

The Address Strobe ($\overline{\text{AS}}$) is a timing signal that indicates the validity of an address on the address bus and of many control signals.

### Data Bus

The data signals (D15 - D0) comprise a bi-directional, non-multiplexed parallel bus that contains the data being transferred to or from the MCU. A read or write operation may transfer 8 or 16 bits of data (1 or 2 bytes) in one bus cycle.

### Data Strobe

The Data Strobe ($\overline{\text{DS}}$) is a timing signal that applies to the data bus. For a read cycle, the MCU asserts $\overline{\text{DS}}$ to signal the external device to place data on the bus. For a write cycle, $\overline{\text{DS}}$ signals to the external devices that the data to be written is valid on the bus.

### Bus Control Signals

The MCU initiates a bus cycle by driving the address, size, function, code, and read/write outputs. At the beginning of a bus cycle, the size signals (SIZ1, SIZ0) are driven along with the function code signals. SIZ1 and SIZ0 indicate the number of bytes remaining to be transferred during an operand cycle (consisting of one or more bus cycles). Table 8 shows the encoding of SIZ1 and SIZ0. The read/write (R/$\overline{W}$) signal determines the direction of the transfer during a bus cycle. The read-modify-write cycle signal ($\overline{RMC}$) is asserted at the beginning of the first bus cycle of a read-modify-write operation, and remains asserted until completion of the final bus cycle of the operation.

**Table 8.** Size Signal Encoding

| SIZ1 | SIZ2 | Transfer Size |
|---|---|---|
| 0 | 1 | Byte |
| 1 | 0 | Word |
| 1 | 1 | 3 Byte |
| 0 | 0 | Long Word |

### Bus Cycle Termination Signals

During bus cycles, external devices assert the data transfer and size acknowledge signals $\overline{DSACK1}$ and/or $\overline{DSACK0}$ as part of the bus protocol. During a read cycle, this signals the MCU to terminate the bus cycle and to latch the data. During a write cycle, this indicates that the external device has successfully stored the data and that the cycle may terminate. These signals also indicate to the MCU the size of the port for the bus cycle just completed.

The bus error ($\overline{BERR}$) signal is also a bus cycle termination indicator and can be used in the absence of $\overline{DSACKx}$ to indicate a bus error condition. It can also be asserted in conjunction with $\overline{DSACKx}$ to indicate a bus error condition, provided it meets the appropriate timing. Additionally, the $\overline{BERR}$ and $\overline{HALT}$ signals can be asserted simultaneously, in lieu of, or in conjunction with, the $\overline{DSACKx}$ signals.

The internal bus monitor can be used to generate the $\overline{BERR}$ signal for internal and internal-to-external transfers. An external bus master must provide its own $\overline{BERR}$ generation and drive the $\overline{BERR}$ pin, since the internal $\overline{BERR}$ monitor has no information about transfers initiated by an external bus master.

Finally, the autovector ($\overline{AVEC}$) signal can be used to terminate interrupt acknowledge cycles, indicating that the MCU should internally generate a vector number to locate an interrupt handler routine. $\overline{AVEC}$ is ignored during all other bus cycles.

### Dynamic Bus Sizing

The MCU dynamically interrupts the port size of the addressed device during each bus signal, allowing operand transfers to or from 8- and 16-bit ports. During an operand transfer cycle, the slave device signals its port size (byte or word) and indicates completion of the bus cycle to the MCU through the use of the $\overline{DSACKx}$ encodings and assertion results. Refer to Table 9 for $\overline{DSACKx}$ encodings and assertion results. 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 runs another bus cycle to obtain the other 16 bits.

Dynamic bus sizing requires that the portion of the data bus for a transfer to or from a particular port size be fixed. For example an 8-bit port must reside on data bus bits 15 - 8.

The $\overline{SIZx}$ signals also form part of the bus sizing protocol. These outputs indicate the remaining number or bytes to be transferred during the current bus cycle.

**Table 9.** DSACK Codes and Results

| $\overline{\text{DSACK1}}$ | $\overline{\text{DSACK0}}$ | Result |
|---|---|---|
| 1 (Negated) | 1 (Negated) | Insert wait states in current bus cycle |
| 1 (Negated) | 0 (Asserted) | Complete cycle – Data bus port size is 8-bits |
| 0 (Asserted) | 1 (Negated) | Complete cycle – Data bus port size is 16-bits |
| 0 (Asserted) | 0 (Asserted) | Reserved |

**Bus Operation**

The MCU bus is used in an asynchronous manner. The external devices connected to the bus can operate at clock frequencies different from the clock for the MCU. Bus operation uses the handshake lines ($\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{DSACK1}}$, $\overline{\text{DSACK0}}$, $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$) to control data transfers. Decoding the size outputs and lower address line A0 provides strobes that select the active portion of the data bus. The slave device (memory or peripheral) then responds by placing the requested data on the correct portion of the data bus for a read cycle or latching the data on a write cycle, and asserting the $\overline{\text{DSACK1}}$/$\overline{\text{DSACK0}}$ combination that corresponds to the port size to end the cycle. If no slave responds or the access is invalid, external control logic asserts the $\overline{\text{BERR}}$, or $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$) signal(s) to abort or retry the bus cycle, respectively.

**Fast Termination Cycles**

With an external device that has a fast access time, the chip-select circuit fast-termination option can provide a two-cycle external bus transfer. Since the chip select circuits are driven from the system clock, the bus cycle termination is inherently synchronized with the system clock.

**Bus Exception Control Cycles**

The bus architecture requires assertion of $\overline{\text{DSACKx}}$ from an external device to signal that a bus cycle is complete. $\overline{\text{DSACKx}}$ or $\overline{\text{AVEC}}$ is not asserted in these cases:

- The external device does not respond
- No interrupt vector is provided
- Various other application-dependent errors occur

This MCU has a bus error input ($\overline{\text{BERR}}$) when no device responds by asserting $\overline{\text{DSACKx}}$ or within an appropriate period of time after the MCU asserts the $\overline{\text{AVEC}}$. This allows the cycle to terminate and the MCU to enter exception processing for the error condition. Another signal that is used for bus exception control is the halt signal ($\overline{\text{HALT}}$).This signal can be asserted by an external device for debugging purposes to cause single bus operation or (in combination with $\overline{\text{BERR}}$) a retry of a bus cycle in error.

**Bus Arbitration**

The bus design of the MCU provides for a single bus master at any one time: either the MCU or an external device. One or more of the external devices on the bus can have the capability of becoming bus master. Bus arbitration is the protocol by which an external device becomes bus master; the bus controller in the MCU manages the bus arbitration signals so that the MCU has the lowest priority. External devices that need to obtain the bus must assert the bus arbitration signals in a certain sequence. Systems that include several devices that can become bus master require external circuitry to assign priorities to the devices, so that when two or more external devices attempt to become bus master at the same time, the one having the highest priority becomes the bus master first. The protocol is explained fully in the SIM manual, however here is the basic sequence of events:

- An external device asserts the bus request signal ($\overline{BR}$),
- The MCU asserts the bus grant signal to indicate that the bus is available ($\overline{BG}$),
- The external device asserts the bus grant acknowledge signal ($\overline{BGACK}$) to indicate that it has assumed bus mastership.

Bus arbitration requests are recognized during normal processing, HALT assertion, when the CPU has halted due to a double bus fault.

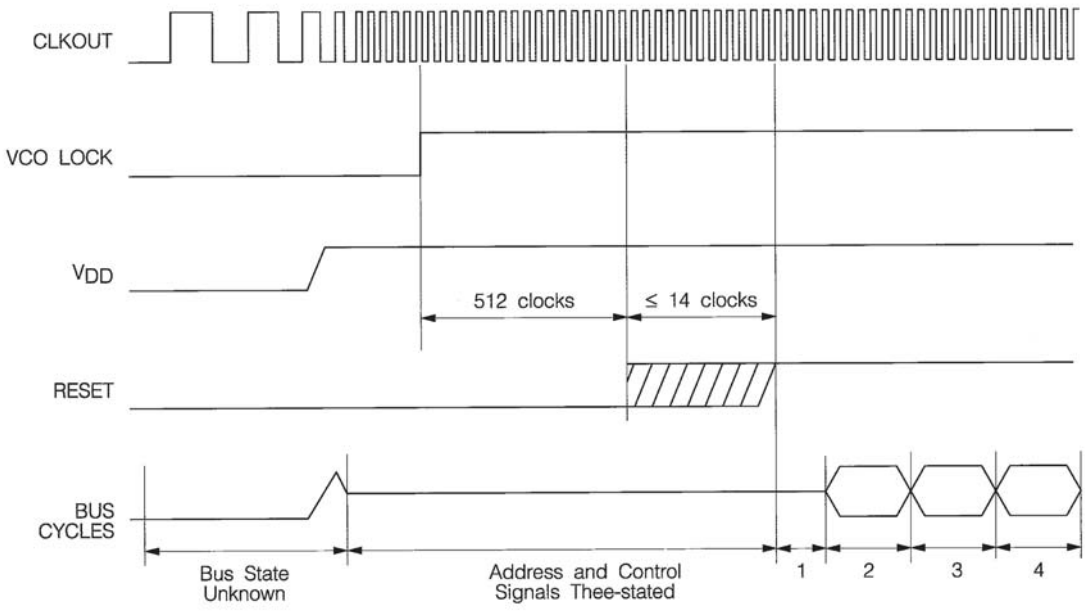
**Reset Operation**

The MCU has reset control logic to determine the cause of reset and synchronize it if necessary. If an external device drives the $\overline{RESET}$ pin low, the reset control logic holds $\overline{RESET}$ asserted internally until the external $\overline{RESET}$ is released. When the reset control logic detects that the external $\overline{RESET}$ is no longer being driven, it drives $\overline{RESET}$low for an additional 512 cycles to guarantee this length of reset to the entire system. If $\overline{RESET}$ is asserted from any other source, the reset control logic asserts $\overline{RESET}$ for a minimum of 512 cycles and until the source of reset is negated. Figure 20 is a timing diagram of the power-up reset operation, showing the relationship between $\overline{RESET}$, $V_{DD}$, and bus signals. During the reset period, the entire bus (except for non-tri-statable signals, which are driven to their inactive state three-states. Once $\overline{RESET}$ negates, all control signals are driven to their inactive state, the data bus is in read mode, and the address bus is driven. After this, the first bus cycle for $\overline{RESET}$ exception processing begins.

$\overline{RESET}$ should be asserted for at least 590 clock periods to ensure that the MCU resets. Resetting the MCU causes any bus cycle in progress to terminate as if $\overline{DSACKx}$ or $\overline{BERR}$ has been asserted. In addition, the MCU initializes registers appropriately for a reset exception.

For further information refer to the System Integration Module Manual.

**Figure 20.** Initial Reset Operation Timing




Notes:
1. Internal startup time
2. SSP read here
3. PC read here
4. First instruction fetched here

## System Integration Module

The TS68332 system integration module (SIM) consists of five sudmodules that control the microcontroller unit (MCU) system start-up, initialization, configuration, and external bus with a minimum of external devices. The fives submodules that make up the SIM, shown in Figure 21, are as follows:

- System Configuration and Protection
- Clock Synthesizer
- Chip Selects
- External Bus Interface
- System Test

### System Configuration and Protection Submodule

The SIM module allows the user to control some features of system configuration by writing bits in the Module Configuration Register. This register also contains read-only status bits that show the state of some of the SIM features.

This MCU is designed with the concept of providing maximum system safe-guards. Many of the functions that normally must be provided in external circuits are incorporated in this MCU. The features provided in the system configuration and protection submodule are as follows:

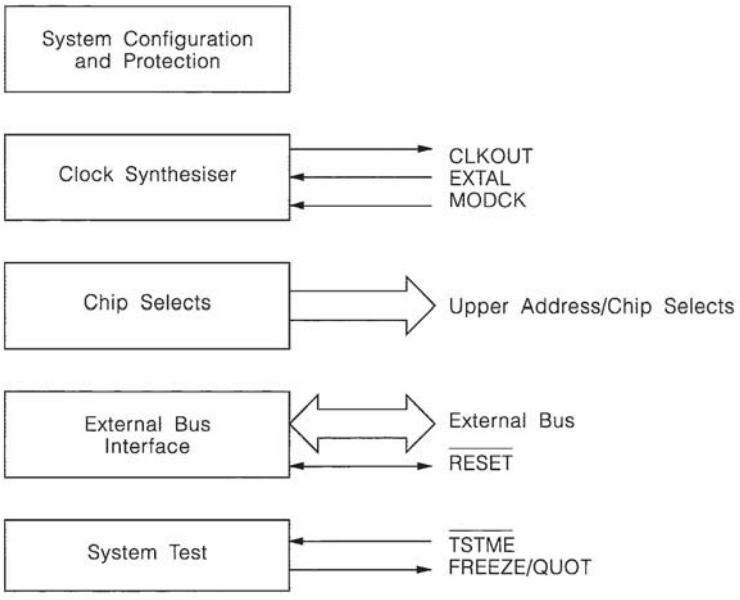
*System Configuration*

The module configuration register allows the user to configure the system according to the particular system requirements.

***Internal Bus Monitor***

The MCU provides an internal bus monitor to monitor the $\overline{\text{DSACKx}}$ response time for all internal bus accesses. An option allows the monitoring of internal to external bus accesses. There are four selectable response times that allow for the response speed of peripherals used in the system. A bus error ($\overline{\text{BERR}}$) signal is asserted internally if the $\overline{\text{DSACKx}}$ response time is exceeded. When operating as a bus master, the $\overline{\text{BERR}}$ signal is not asserted externally.

**Figure 21.** System Integration Module Block Diagram




*Halt Monitor*

A halt monitor causes a reset to occur if the internal halt ($\overline{\text{HALT}}$) is asserted by the CPU.

*Spurious Interrupt Monitor*

If no interrupt arbitration occurs during an interrupt acknowledge (IACK) cycle, the $\overline{\text{BERR}}$ signal is asserted internally.

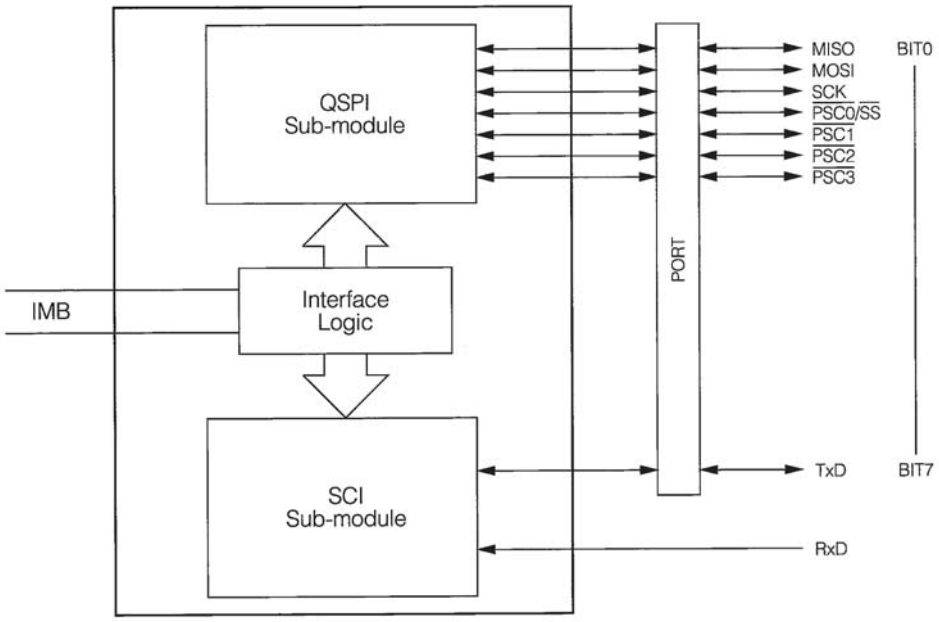
*Software Watchdog*

The watchdog asserts $\overline{\text{RESET}}$ if the software fails to service the software watchdog for a designated period of time (presumably because it is trapped in a loop or lost). There are four selectable time-out periods, and a prescaler may be used for long time-out periods.

***Periodic Interrupt Timer***

The MCU provides a timer to generate periodic interrupts. The periodic interrupt time period can vary from 122 µs - 15.94 µs (with a 32.768 kHz crystal used to generate the system clock).

**Figure 22.** Clock Submodule Block Diagram




Note: Must be low leakage capacitor.

**Clock Synthesizer**

The clock synthesizer (Figure 22) can operate from an on-chip phase locked loop (PLL) using an external crystal connected between the EXTAL and XTAL pins as a reference frequency source. A 32.768 kHz watch crystal provides an inexpensive reference, but the reference crystal frequency can be any frequency from 25 - 50 kHz. Outside the 25 - 50 kHz range, an external oscillator can be used with the on-chip synthesizer and VCO, or the frequency can be driven directly into the EXTAL pin (the XTAL pin should be left floating for this case).

The system clock frequency is programmable from 131 kHz to the maximum clock frequency with a resolution of 131 kHz. A separate power pin ($V_{DDSYN}$) is used to allow the clock circuits to run with the rest of the MCU powered down and to provide increased noise immunity for the clock circuits. If for some reason the external signal is removed from the device then the clock synthesizer will generate its own internal clock signal to allow the device to enter some kind of error recovery routine. This is known as LIMP mode. The clock frequency generated will not have an associated timing spec but should be around 9 MHz.

**Chip-select Submodule**

Typical microcomputer systems require external hardware to provide select signals to external peripherals. This MCU integrates these functions on-chip in order to provide the cost, speed, and reliability benefits of a higher level of integration. The chip-select signals can also be programmed as output enable, read or write strobe, or IACK signals.

Since initialization software would probably reside in a peripheral memory device controlled by the chip-select circuits, a CSBOOT register provides default reset values to support bootstrap operation.

The chip-select submodule supports the following programmable features:

| | |
|---|---|
| *Twelve Programmable Chip-select Circuits* | Twelve chip select signals are available ($\overline{\text{CSBOOT}}$) and $\overline{\text{CS10}}$ tro $\overline{\text{CSO}}$). These signals use the $\overline{\text{CSBOOT}}$ pin, bus arbitration pins $\overline{\text{BR}}$, $\overline{\text{BG}}$, and $\overline{\text{BGACK}}$, function code pins FC2-FC0, and address pins A23-A19. The ($\overline{\text{CSBOOT}}$) pin is dedicated to a single function because it must function after a reset with no initialization, the other chip select circuits share functions on their output pins. All 12 chip select circuits are independently programmable from the same list of selectable features. Each chip select circuit has an individual base register and option register which contain the programmable characteristics of that chip select. Using these address lines as chip select signals does not restrict the large linear address space of the MCU since the chip select logic always uses the internal address lines. |
| *Variable Block Sizes* | The block size starting from the specified base address can be programmed as 2K, 8K, 16K, 64K, 128K, 256K, 512Kbytes or 1-Mbyte. |
| *Both 8-bit and 16-bit Ports Supported* | Eight-bit ports are accessible on both odd and even addresses when connected to data bus bits 15-8. Sixteen-bit ports can be accessed as odd bytes, even bytes, or words. |
| *Read Only, Write Only, or Read/write Capability* | Chip selects can be asserted synchronized with read, write, or both read and write. |
| *Address Strobe and Data Strobe Timing Option* | Chip-select signals can be synchronized with either address strobe or data strobe, so that control signals such as output enable or write enable can be easily generated. |
| ***Internal DSACK Generation with Wait States*** | The port programmed in the pin assignment register can be referenced for generating $\overline{\text{DSACK}}$ and the proper number of wait states for a particular device programmed by the user. |
| ***Address Space Checking*** | Supervisor, user, and CPU space accesses can be optionally checked. |
| ***Interrupt Priority Level Checking*** | In the IACK cycle, the acknowledged interrupt level can be compared with the user-specified level programmed in the option field. If autovector option is selected, $\overline{\text{AVEC}}$ is internally asserted. |
| *Discrete Output* | Port C pins A22-A19 and FC2-FC0 can be programmed for discrete output, with data stored in the pin data register (CSPDR). |
| *68000-type Peripheral Support* | 68000-type peripherals that require an E clock for synchronization can be supported. Chip select is asserted, synchronized with the E clock on pin A23, providing correct data bus timing for the MCU. |

### Test Submodule

The test submodule is a primary tool to support all types of testing, such as production test and user self-test, that is integrated into the MCU. The submodule supports scan-based testing of various modules in the MCU. The scan test employed here consists of the test submodule performing the following steps:

- serially shifting stimulus data to an idle module under test (MUT)
- activating the module under test
- serially shifting response data back from the module under test
- latching the response data for interrogation by the bus master

The further information to the System Integration Module Manual.

## QSM Queued Serial Module

The queued serial module (QSM) provides the microcontroller unit (MCU) with two serial communication interfaces divided into two submodules: the queued serial peripheral interface (QSP) and the serial communications interface (SCI). The QSPI is a full-duplex, synchronous serial interface for communicating with peripherals and other MCUs. It is enhanced by the addition of a RAM queue for receive and transmit data. The SCI is a full-duplex universal asynchronous receiver transmitter (UART) serial interface. These submodules operate independently (see Figure 23).

### QSM Pins

The QSM has nine external pins. Eight of these pins can be used as general-purpose I/O pins. If the pin is not being user for its submodule function. The ninth pin, RXD, is an input-only pin used exclusively by the SCI submodule. The pins are identified as follows:

MISO – Master In Slave Out

MOSI – Master Out Slave In

SCK – Serial Clock

PCS0/SS – Peripheral Chip-Select 0/Slave Select

PCS3-PCS1 – Peripheral Chip Selects 3-1

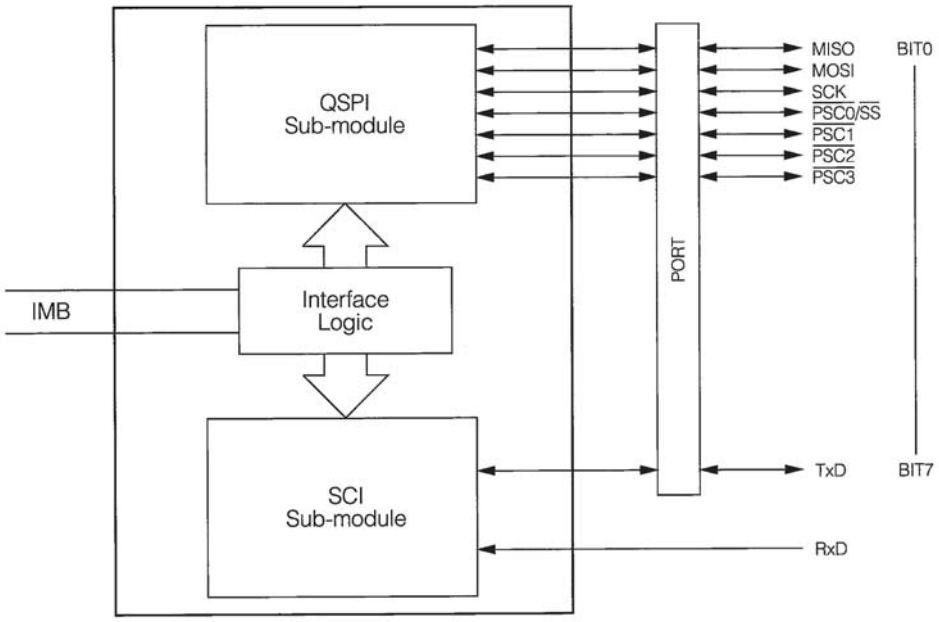
TXD – Transmit Data

RXD – Receive Data

### QSPI Submodule

The QSPI submodule communicates with external peripherals and other MCUs via a synchronous serial bus. The QSPI is fully compatible with the Serial Peripheral Interface (SPI) systems found on other Atmel-Grenoble devices such as the 68HC11 and 68HC05 families. It has all of the capabilities of the standard SPI system as well as several new features. The following paragraphs describe the main feature of the QSPI.

**Figure 23.** QSM Block Diagram




**QSPI Features**

Standard SPI features are listed below, followed by a list of the additional features offered on the QSPI:

- full duplex, three-wire synchronous transfers,
- half-duplex, two-wire synchronous transfers,
- master or slave operation,
- programmable master bit rates,
- programmable clock polarity and phase,
- end-of-transmission interrupt flag,
- master-master mode fault flag,
- easily interfaces to simple expansion parts (A/D converters, EEPROMs, display drivers, etc.).

**QSPI Enhanced Features**

A programmable queue allows the QSPI to perform up to 16 serial transfers without CPU intervention. Each transfer corresponds to a queue entry containing all the information needed by the QSPI to independently complete one serial transfer. This unique feature greatly reduces CPU/QSPI interaction, resulting in increased CPU and system throughput.

Once the CPU has set up the queue of QSPI commands and enables the QSPI, the QSPI operates independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating that it has finished, and then either interrupts the CPU or waits for CPU intervention.

*Programmable Peripheral Chip Selects:* Four peripheral chip-select pins allow the QSPI to access up to 16 independent peripherals by decoding the four peripheral chip-select signals. Up to four independent peripherals can be selected by direct connection to a chip-select pin. The peripheral chip selects simplify interfacing to two or more serial peripherals by providing dedicated peripheral chip-select signals and thus alleviating the need for CPU intervention.

*Wraparound Transfer Mode:* Wraparound transfer mode allows for automatic, continuous re-execution of the preprogrammed queue entries. Newly transferred data replaces previously transferred data. Wraparound simplifies interfacing with A/D converters by automatically providing the CPU with the latest conversions in the QSPI RAM. Consequently, serial peripherals appear as memory-mapped parallel devices to the CPU.

*Programmable Transfer Length:* The number of bits in a serial transfer is programmable from 8 to 16 bits, inclusive. For example, 10-bits could be used for communicating with an external 10-bits A/D convertor. Likewise, a vacuum fluorescent display driver might require a 12-bits serial transfer. The programmable length simplifies interfacing to serial peripherals that require different data lengths.

*Programmable Transfer Delay:* An inter-transfer delay may be programmed from approximately 1 to 500 µs (using a 16.78 MHz system clock). For example, an A/D convertor may require time between transfers to complete a new conversion. The default delay is 1 µs. The programmable length of delay simplifies interfacing to serial peripherals that require delay time between data transfers.

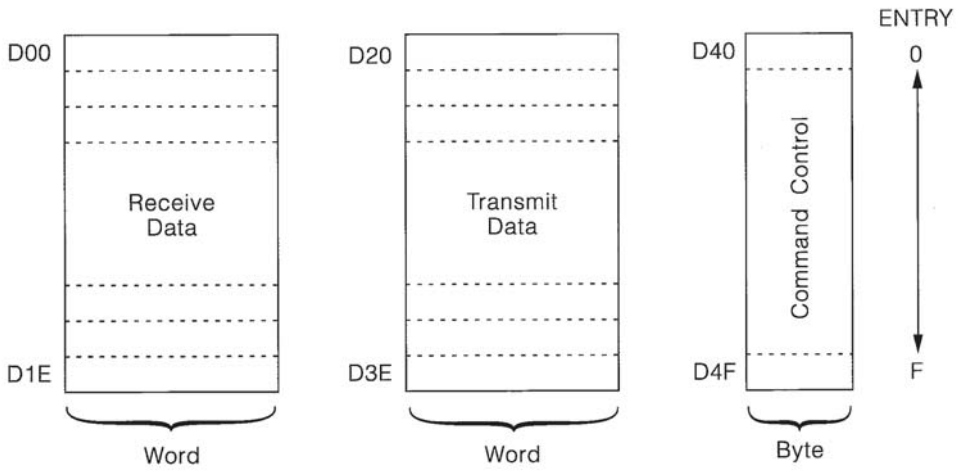
*Programmable Queue Pointer:* The QSPI has a pointer that points to the queue location containing the data for the next serial transfer. The CPU can switch from one task to another in the QSPI by writing to the queue pointer, changing the location in the queue that is to be transferred next. Otherwise, the pointer increments after each serial transfer. By segmenting the queue, multiple-task support can be provided by the QSPI.

*Continuous Transfer Mode:* The continuous transfer mode allows the user to exchange an uninterrupted bit stream with a peripheral. A minimum of 8-bits and a maximum of 256-bits may be transferred in a single burst without CPU intervention. Longer transfers are possible; however, minimal CPU intervention is required to prevent loss of data. A 1microsecond pause (using a 16.78 MHz system clock) is inserted between each entry transfer.

*QSPI RAM:* The QSPI uses an 80-byte block of dual-access static RAM that can be accessed by both the QSPI and the CPU. Because of sharing, the length of time taken by the CPU to access the QSPI RAM, when the QSPI is enabled, may be longer than when the QSPI is disabled. From one to four CPU wait states may be inserted by the QSPI in the process of reading or writing.

The RAM is divided into three segments: receive data, transmit data, and command control. Receive data is information received from a serial device external to the MCU. Transmit data is information stored by the CPU for transmission to an external peripheral chip. Command control contains all the information needed by the QSPI to perform the transfer. Figure 24 illustrates the organization of the RAM.

**Figure 24.** Organization of the QSPI RAM




## SCI Submodule

The SCI submodule is used to communicate with external devices and other MCUs via an asynchronous serial bus. The SCI is fully compatible with the SCI systems found on other Atmel MCUs such as the 68HC11 and 68HC05 families. It has all of the capabilities of previous SCI systems as well as several significant new features.

- **Features**

Standard SCI features are listed below, followed by a list of additional features offered:

Standard SCI Two-wire Systems Feature:

- Standard Non Return to Zero (NRZ) Mark/space Format
- Advanced Error Detection Mechanism (Detects Noise Duration Up To 1/16 of A Bit-time)
- Full-duplex Operation
- Software Selectable Word Length (8- or 9-bits Words)
- Separate Transmitter And Receiver Enable Bits
- May Be Interrupt Driven
- Four Separate Interrupt Enable Bits

Standard SCI Receiver Features:

- Receiver Wake Up Function (Idle or Address Mark Bit)
- Idle-line Detect
- Framing Error Detect
- Noise Detect
- Overrun Detect
- Receive Data Register Full Flag

Standard SCI Transmitter Features:

- Transmit Data Register Empty Flag
- Transmit Complete Flag
- Send Break

QSM-enhanced SCI Two-wire Systems Features:

- 13-bits Programmable Baud Rate Modulus Counter
- Even/odd Parity Generation And Detection

QSM-enhanced SCI Receiver Features

- Two Idle-line Detect Modes
- Receiver Active Flag

*13-bit Programmable Baud Rate Modulus Counter:* A baud rate modulus counter has been added to provide the user with more flexibility in choosing the crystal frequency for the system clock. The modulus counter allows the SCI baud rate generator to produce standard transmission frequencies for a wide range of system clocks. The user is no longer constrained to select crystal frequencies based on the desired serial baud rate. This counter baud rates from 64 baud to 524 baud with a 16.78 MHz system clock.

*Even/odd Parity Generation and Detection:* The user now has the choice either of seven or eight data bits plus one parity bit, or of eight or nine data bits with no parity bit. Even or odd parity is available. The transmitter automatically generates the parity bit for a transmitted byte. The receiver detects when a parity error has occurred on a received byte and sets a parity error flag.

*Two Idle-line Detect Modes:* Standard Atmel-Grenoble SCI systems detect an idle line when 10 or 11 consecutive bit-times are all ones. Used with the receiver wake up mode, the receiver can be awakened prematurely if the message preceding the start of the idle line contained ones in advance of its stop bit. The new (second) idle-line detect mode only starts counting idle time after a valid stop bit is received, which ensures correct idle-line detection.

*Receiver Active Flag (RAF):* Receiver Active Flag (RAF) indicates the status of the receiver. It is set when a possible start bit is detected and is cleared when an idle line is detected. RAF is also cleared if the start bit is determined to be line noise. This flag can be used to prevent collisions in systems with multiple masters.

For further information refer to the System Integration Module Manual.

## Standby RAM (with TPU emulation)

The TS68332 contains 2-Kbytes of standby RAM. This section describes the operation and control of the RAM module.

### Overview

The Ram module contains 2048 bytes of fully static RAM, powered by $V_{DD}$ in normal operation. The entire array may be used as standby RAM if power is supplied to the $V_{STBY}$ pin. Switching between $V_{DD}$ and $V_{STBY}$ occurs automatically.

The RAM may be used as general-purpose memory for the MCU, providing fast, two-clock accesses to the CPU. Typically, the RAM is used for program control stacks and frequently modified data variables. The CPU may read or write byte, word, or long-word data.

The RAM may also be used as microcode control memory for the Time Processor Unit (TPU). The TPU must be placed in emulation mode to use the RAM in this manner which allows users to develop their own microcode primitives.

### RAM Array Addressing

The RAM array can be placed anywhere in the address map of the array base address (RAMBAR), provided that it is on a 2-Kbytes boundary and does not overlap the three RAM module control registers used for control and testing. RAMBAR can be written only once after reset. This prevents the RAM array being accidentally remapped by software.

### TPU Emulation Mode Operation

The RAM array may be used as the microcode control store for the TPU module. This mode of operation is selected from within the TPU. See Development support in the TPU manual for a complete description.

The TPU is connected to the RAM via a dedicated bus. While in emulation mode, the access timing of the RAM module matches the timing of the TPU microinstruction ROM to ensure accurate emulation. Normal accesses via the IMB are inhibited and the control register have to effect, allowing external RAM to emulate the 2K RAM array at the same addresses.

The further information refer to the System Integration Module Manual.

## TPU Overview

The TPU performs simple as well as complex timing tasks, independently from the CPU, making it the latest advance in timer systems. Viewed as a special purpose microcomputer, this processor performs two operations, match and capture, on one operand: TIME. Every occurrence of either action is called an event. The servicing of these events by the TPU replaces the servicing of interrupts by the host Central Processing Unit (CPU). The timing functions currently synthesized are the following:

- Discrete Input/output
- Input Capture /input Transition Counter
- Output Compare
- Pulse Width Modulation
- Synchronized Pulse Width Modulation
- Period Measurement With Additional Transition Defect
- Period Measurement With Missing Transition Detect
- Position-synchronized Pulse Generator
- Stepper Motor
- Period/pulse-width Accumulator

The previous pre-programmed functions are related to the TPU Rom mask set A, currently in use for the TS68332 MCU, as the “standard” TPU maskset.

The advanced TPU affords for the first time high-resolution timing and multiple time function capability (flexibility) in the timer system pins.

### High-resolution Timing

High-resolution timing is limited by CPU overhead required for servicing timing tasks such as period measurement, pulse measurement, pulse-width modulated waveform generation, etc. On the TPU, high-resolution timing is achieved by two main capabilities:

- reduced latency,
- reduced service time, which free the CPU to focus on other responsibilities.

The TPU provides a higher resolution than the CPU could achieve, and creates no CPU overhead for servicing timing tasks.

*Latency*

Latency is the interval of time from an even to the start of event servicing. The ability of the TPU to service its own interrupts or events reduces latency and the CPU is not required to service each input transition capture that occurs on a pin, or to determine each match time required for waveform synthesis. Once configured by the host CPU, the self-contained TPU performs complex time functions requiring high resolution with little or no CPU intervention.

*Service Time*

Service is the time expended servicing an event. In older microcontroller unit (MCU) timer functions, the service time is constrained because the MCU instruction set is not optimized for time function synthesis. The TPU instruction set is optimized, and time functions are synthesized with fewer instructions than the CPU. Instructions execute faster and service time is reduced. Instructions executed by the TPU are not user software, but firmware, special-purpose microcode written by Atmel-Grenoble to perform as set time functions. Microcode is placed into the TPU control store (ROM) when the device is manufactured.

**Features**

- 16 channels; each channel associated with a pin
- Each channel can perform any time function
- Each time function may be assigned to more than one channel at a given time
- Each channel has an event register comprised of the following:
  - 16-bits capture register
  - 16-bits compare/match register
  - 16-bits greater-than or equal-to comparator
- Each channel can be synchronized to one or both of the two 16-bits free-running timer count registers (TCR1 and TCR2)
- 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. The four settings of the prescaler are divide by 1, 2, 4 and 8. Channels using TRC1 have the capability to resolve down to the TPU system clock divided by four.
- TCR2 is clocked from the output of a prescaler. The prescaler's input is the external TCR2 pin. The four settings of the prescaler are divide by 1, 2, 4 and 8. Channels using the TCR1 have the capability to resolve down to the PRU system clock divided by 8.
- TCR2 may be used as a hardware pulse accumulator clocked from the external TCR2 pin, or as a gated pulse accumulator or the clock that increments TCR1.
- All channels have at least six 16-bits parameter registers. Channels 14 and 15 each have eight 16-bits parameter registers. All parameter registers are contained in a dual-port RAM, accessible from both the TPU and CPU.
- A scheduler with three priority levels segregates high, middle, and low-priority time functions. Any channel may be assigned to one of these three priority levels.
- All time functions are microcoded.
- Emulation and development support is provided for all time function features such as breakpoint, freeze and single step, giving internal register accessibility.
- Coherent transfer capability for two parameter is provided in hardware.
- Coherent transfer capability for N parameters may be performed as a TPU microcode function. (Refer to Development support in the TPU reference manual for further details on this feature).

**General Concept**

The TPU is an intelligent, semi-autonomous peripheral dedicated to timing control. Its intelligence enables the servicing of timing events without CPU intervention. This device uses a private microengine for a processor, a scheduler, input/output channels, ROM instructions, and shared-access data RAM to operate independently and simultaneously with the CPU (see Figure 25). Consequently, the setup and service time for each timer event is minimized.

A “time-of-delay” approach is used where all time functions are related to one of two 16-bits free-running TCRs. Time functions are synthesized by combining the two time primitives, match and capture events. By performing these time primitives in hardware, the TPU can precisely determine the time when a match event is to occur and then specify the state of the output pin accordingly. The TPU can also accurately record the time at which an input transition occurs and can perform calculations based on the time of the occurrence. An event register for each channel provides for simultaneity of match/capture-event occurrences on all channels.

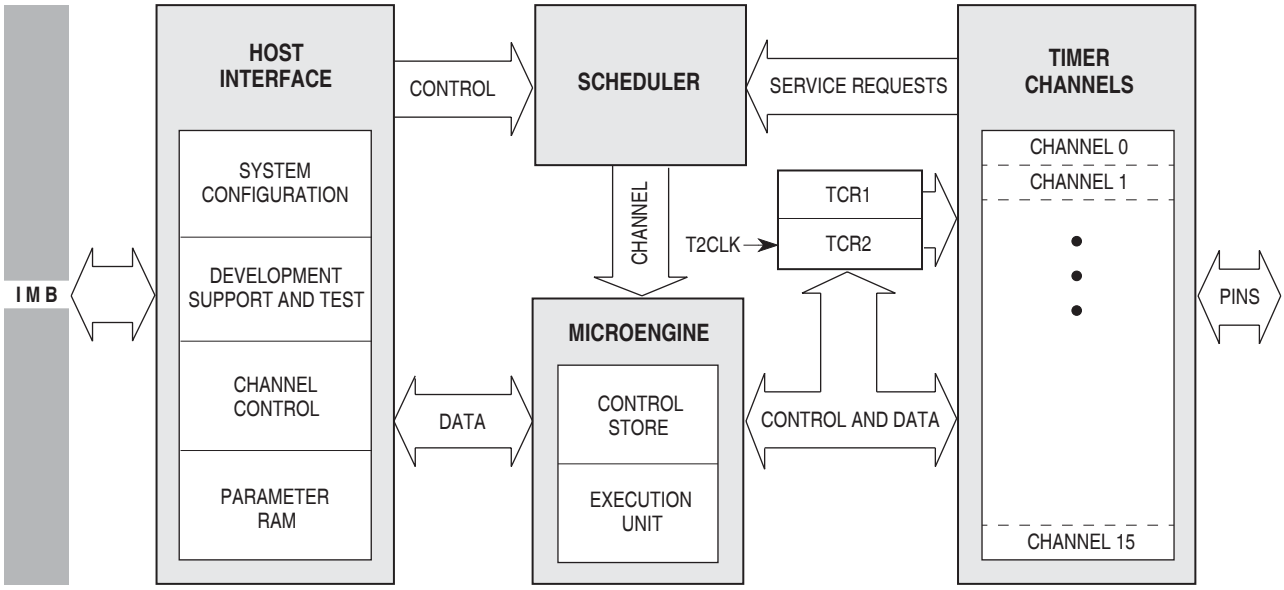
When a match or input capture event requiring service occurs on a channel, the channel generates a service request to the scheduler. The scheduler prioritizes the request with other pending service requests. When the microengine is idle, the scheduler causes the microengine to execute a microcode sequence. When the microengine is busy, the new sequence begins when the code being executed ends. The microengine performs the function, which is defined by the content of the control store, using parameters from the parameter RAM and from the event registers, etc., as needed. The following is an example.

Channel X is generating a periodic waveform and presently the output is high. When the value of the TCR used by that channel increments to match the value of the event register of channel X, a match event occurs. The event switches the output to low and generates a new service request to the scheduler. The scheduler than schedules and initiates service of channel X by the microengine.

When execution of the sequence begins, the microengine uses the execution unit:

- To obtain (from the parameter RAM) the value representing the duration of counts for which channel X should remain low, and
- To add to this value the value from the content of the event register of channel X. The content of the event register is then replaced by this sum; the channel control is set for a match event on the same TCR; and the pin control is set to cause the output pin for channel X to switch high when the event occurs. A channel interrupt, which signals the end of service to the CPU, may be asserted (if the time function provides for it and the interrupt is enabled). The microengine is then free to service the next event determined by the scheduler.

**Figure 25.** TPU Simplified Block Diagram




**Flexibility**

The TPU has the flexibility to be configured to directly solve the user's timer requirements. This flexibility is attained through five capabilities:

- channel orthogonality
- inter-channel communication
- programmable channel service priority
- selection of timing functions
- emulation capability

*Channel Orthogonality*

Traditionally, timer systems have been limited by the specific functions of channel pins dedicated to perform time functions such as input capture, output compare, or pulse accumulation. All channels of the TPU contain identical hardware and are functionally equivalent in operation, such that any channel can be configured to perform any time function. The user controls the combination of time functions; the only constraint is the number of pins available for timing functions.

*Inter-channel Communication*

The TPU's ability to service itself requires a continuous flow of direct and indirect communication. Direct communication is accomplished through a "change channel" feature in which any channel of the TPU can operate another channel to affect its state. Indirect communication is provided by a link feature in which any channel can lonk to one more channels, including itself, to signal a need for future service. As a result, the user can reference the operation of one channel to the occurrence of a specific action on another channel.

*Programmable Channel Service Priority*

Applications may require different priorities of event service. The channel service priority may be programmed to one of three levels: high, middle, and low. The scheduler allows calculation of worst-case latency for event servicing and ensures servicing of all channels by preventing permanent blockage.

*Selection of Timing Functions*

The available timing functions can be programmed to operate on any channel. Parameter registers associated with each channel are used as general-purpose time operands.

*Emulation Capability*

The TPU cannot resolve all timer problems using predefined time functions alone; therefore, development of user-defined time functions is allowed in emulation mode. Using the RAM module of the MCU as a “writable control store” provides TPU emulation. In TPU emulation mode, an auxiliary bus connection is made between the RAM module and the TPU module, and access to the RAM module via the intermodule bus is disabled. A 9-bits address bus, a 32-bits data bus, and control lines transfer information between the modules. To ensure exact emulation, the access timing of the RAM module remains consistent with the TPU ROM control store.

**Applications**

The TPU’s high speed, versatile architecture, and time functions facilitate its use in many control applications, such as stepper motors and angle-based engine control. Control of a stepper motor or an angle-based automotive engine usually requires high CPU overhead. These applications show how the SM, PMA/PPM, and time functions minimize the overhead associated with these applications, and provide sophistication and flexibility for a wide variety of applications.

Further detailed information on the TPU is found in the TPU reference manual.

# Preparation For Delivery

## Packaging

Microcircuit are prepared for delivery in accordance with MIL-M-38510.

## Certificate of Compliance

Atmel-Grenoble offers a certificate of compliance with each shipment of parts, affirming the products are in compliance either with MIL-STD-883 or Atmel-Grenoble standard and guaranteeing the parameters are tested at extreme temperatures for the entire temperature range.

# Handling

MOS devices must be handled with certain precautions to avoid damage due to accumulation of static charge. Input protection devices have been designed in the chip to minimize the effect of this static buildup. However, the following handling practices are recommended:

a) Device should be handled on benches with conductive and grounded surface

b) Ground test equipment, tools and operator

c) Do not handle devices by the leads

d) Store in conductive foam or carriers

e) Avoid use of plastic, rubber, or silk in MOS areas

f) Maintain relative humidity above 50%, if practical

## Packaging Information

**Figure 26.** 132–ball–Ceramic Pin–Grid Array (PGA)

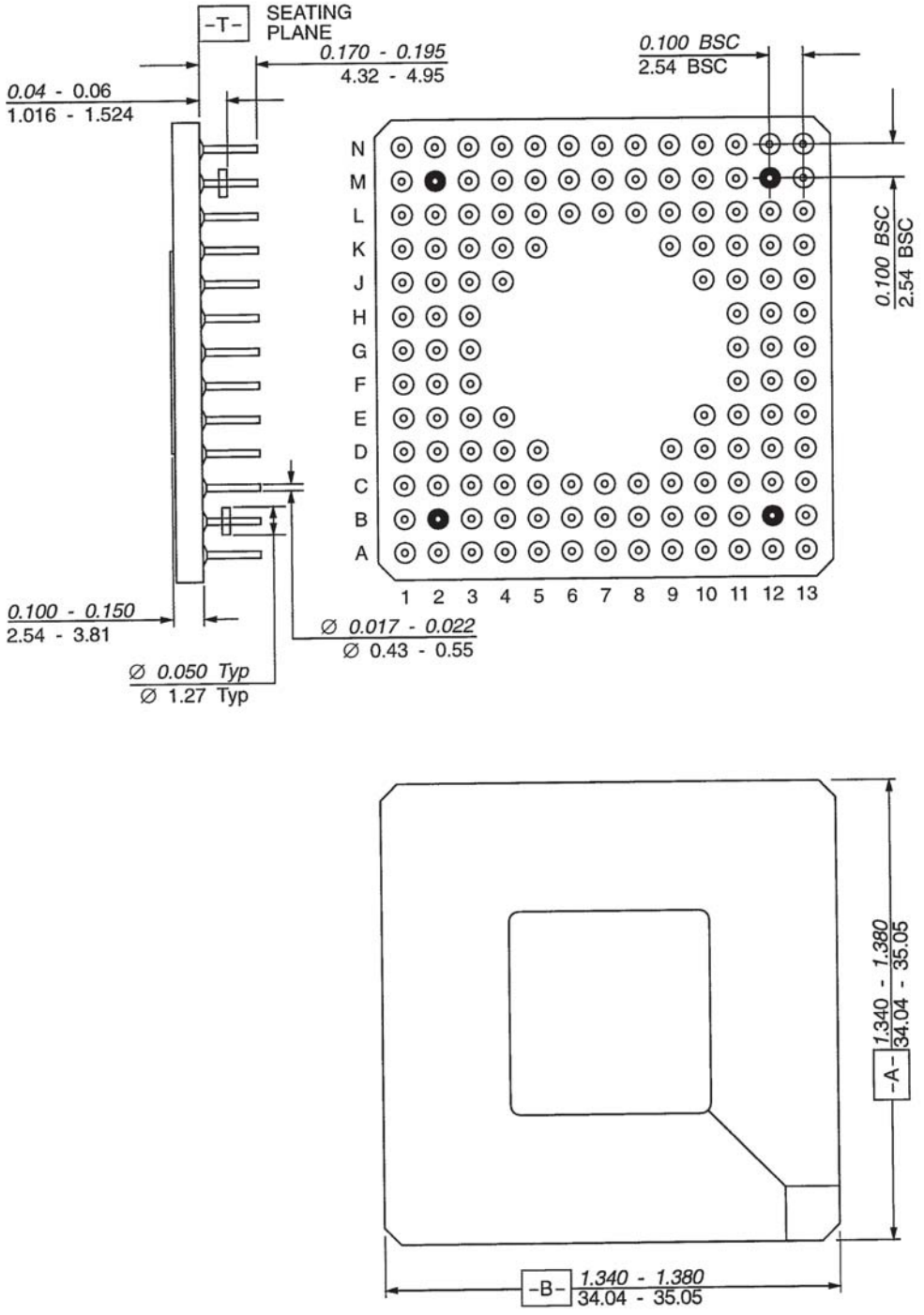
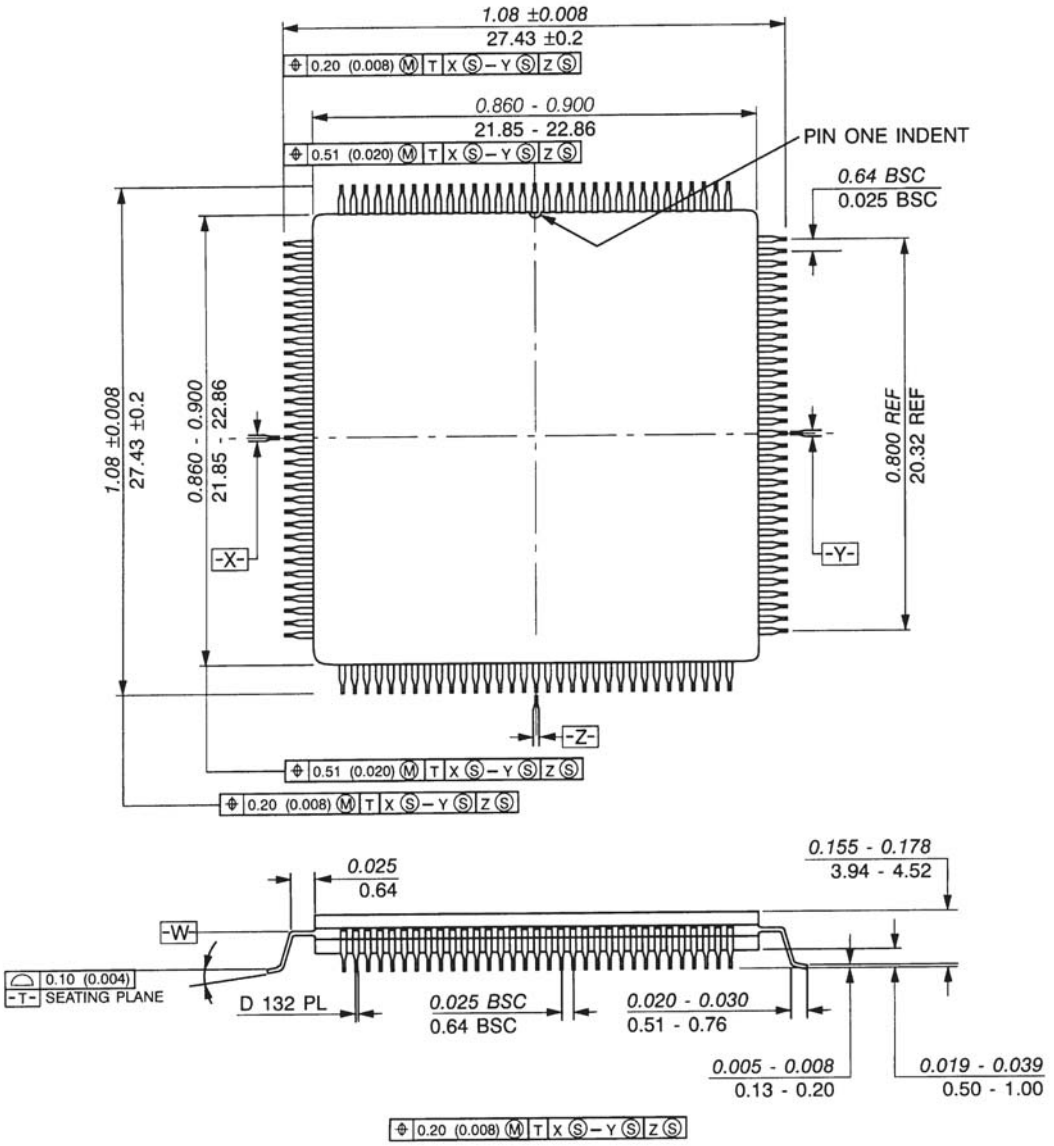


**Figure 27.** 132–lead CERQUAD




Notes:
1. Dimensioning and tolerancing per ansi Y14.5M, 1982.
2. Controlling dimensions: inch.
3. Dim A and B define maximum ceramic body dimensions including glass protrusion and mismatch of ceramic body top and bottom.
4. Datum plane - W - is located at the underside of leads where leads exit package body.
5. Datums X-Y and Z to be determined where center leads exit package body at datum - W -.
6. Dim S and V to be determined at seating plane, datum - T -.
7. Dim A and B to be determined at datum plane - T -.

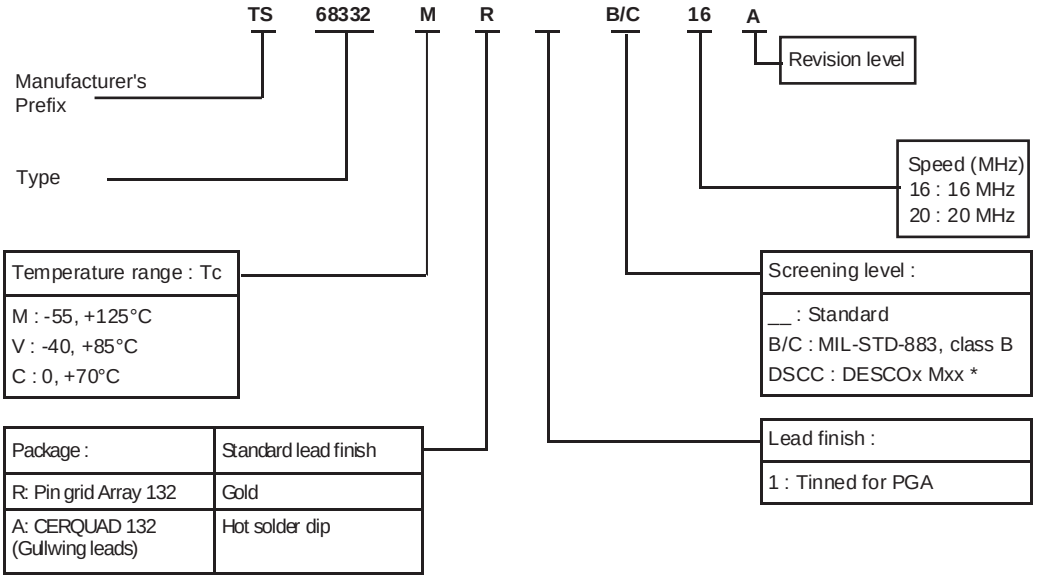
# Ordering Information

## Hi-Rel Product

| Commercial Atmel Part-Number | Norms | Package | Temperature Range Tc (°C) | Frequency (MHz) | Drawing Number |
|---|---|---|---|---|---|
| TS68332MRB/C16 | MIL-STD-883 | PGA 132 | -55/+125 | 16.78 | Atmel-Grenoble datasheet |
| TS68332MR1B/C16 | MIL-STD-883 | PGA 132 tin | -55/+125 | 16.78 | Atmel-Grenoble datasheet |
| TS68332MAB/C16 | MIL-STD-883 | CERQUAD 132 | -55/+125 | 16.78 | Atmel-Grenoble datasheet |
| TS68332MRB/C20 | MIL-STD-883 | PGA 132 | -55/+125 | 20.97 | Atmel-Grenoble datasheet |
| TS68332MR1B/C20 | MIL-STD-883 | PGA 132 tin | -55/+125 | 20.97 | Atmel-Grenoble datasheet |
| TS68332MAB/C20 | MIL-STD-883 | CERQUAD 132 | -55/+125 | 20.97 | Atmel-Grenoble datasheet |
| TS68332DESC01ZA | MIL-STD-883 | PGA 132 tin | -55/+125 | 16.78 | 5962-9150101MZA |
| TS68332DESC01ZC | MIL-STD-883 | PGA 132 | -55/+125 | 16.78 | 5962-9150101MZC |
| TS68332DESC02ZC | MIL-STD-883 | PGA 132 | -55/+125 | 20.97 | 5962-9150102MZC |
| TS68332DESC01XA | MIL-STD-883 | CERQUAD 132 | -55/+125 | 16.78 | 5962-9150101MXA |
| TS68332DESC02XA | MIL-STD-883 | CERQIAD 132 | -55/+125 | 20.97 | 5962-9150102MXA |

## Standard Product

| Commercial Atmel Part-Number | Norms | Package | Temperature Range Tc (°C) | Frequency (MHz) | Drawing Number |
|---|---|---|---|---|---|
| TS68332VR16 | Atmel-Grenoble Standard | PGS 132 | -40/+85 | 16.78 | Atmel-Grenoble datasheet |
| TS68332MR16 | Atmel-Grenoble Standard | PGS 132 | -55/+125 | 16.78 | Atmel-Grenoble datasheet |
| TS68332VA16 | Atmel-Grenoble Standard | CERQUAD 132 | -40/+85 | 16.78 | Atmel-Grenoble datasheet |
| TS68332MA16 | Atmel-Grenoble Standard | CERQUAD 132 | -55/+125 | 16.78 | Atmel-Grenoble datasheet |
| TS68332VR20 | Atmel-Grenoble Standard | PGS 132 | -40/+85 | 20.97 | Atmel-Grenoble datasheet |
| TS68332MR20 | Atmel-Grenoble Standard | PGS 132 | -55/+125 | 20.97 | Atmel-Grenoble datasheet |
| TS68332VA20 | Atmel-Grenoble Standard | CERQUAD 132 | -40/+85 | 20.97 | Atmel-Grenoble datasheet |
| TS68332MA20 | Atmel-Grenoble Standard | CERQUAD 132 | -55/+125 | 20.97 | Atmel-Grenoble datasheet |




Note: For availability of different versions, contact your Atmel sales office.

## Atmel Headquarters


*Corporate Headquarters*
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

*Europe*
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

*Asia*
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

*Japan*
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581


## Atmel Operations


*Memory*
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

*Microcontrollers*
Atmel Corporate
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 436-4270
FAX 1(408) 436-4314

Atmel Nantes
La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

*ASIC/ASSP/Smart Cards*
Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Atmel Smart Card ICs
Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

*RF/Automotive*
Atmel Heilbronn
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

Atmel Colorado Springs
1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

*Biometrics/Imaging/Hi-Rel MPU/*
*High Speed Converters/RF Datacom*
Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80


---


*e-mail*
literature@atmel.com

*Web Site*
http://www.atmel.com





2118A–HIREL–03/02 0M