**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | CPU32 |
| Core Size | 32-Bit Single-Core |
| Speed | 20.97MHz |
| Connectivity | EBI/EMI, SCI, SPI, UART/USART |
| Peripherals | WDT |
| Number of I/O | 48 |
| Program Memory Size | - |
| Program Memory Type | ROMless |
| EEPROM Size | - |
| RAM Size | 2K x 8 |
| Voltage - Supply (Vcc/Vdd) | 4.5V ~ 5.5V |
| Data Converters | - |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Through Hole |
| Package / Case | 132-BPGA |
| Supplier Device Package | 132-PGS (34.55x34.55) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/ts68332vr1-16a |

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.

**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 | |
| 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.

**Figure 7.** Read Cycle Timing Diagram

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

**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.

**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.

**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.

**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).

- 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*

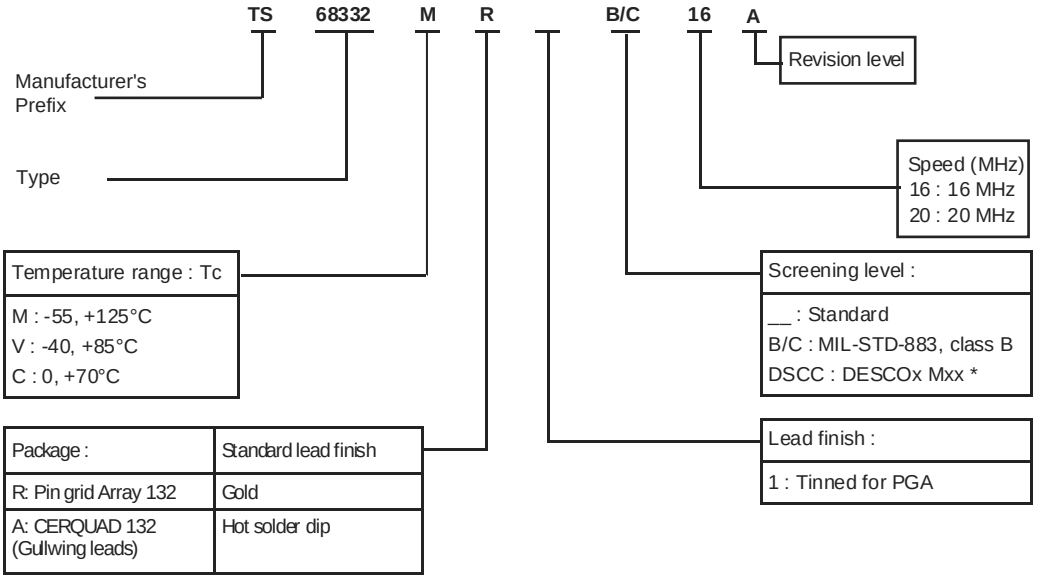
Latency is the interval of time from an even to the start of event servicing. The ability of the TPU to service its own interrupts or events reduces latency and the CPU is not required to service each input transition capture that occurs on a pin, or to determine each match time required for waveform 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).




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