SGS-THOMSON MICROELECTRONICS

ST6x86

P90+, P120+, P133+, P150+, P166+ 3.52 Volt ST6x86 CPU

PRELIMINARY DATA

Sixth-Generation Superscalar Superpipelined Architecture
- Dual 7-stage integer pipelines
- High performance on-chip FPU with 64-bit interface
- Operating at P90+ speeds and above
- 16-KByte write-back cache

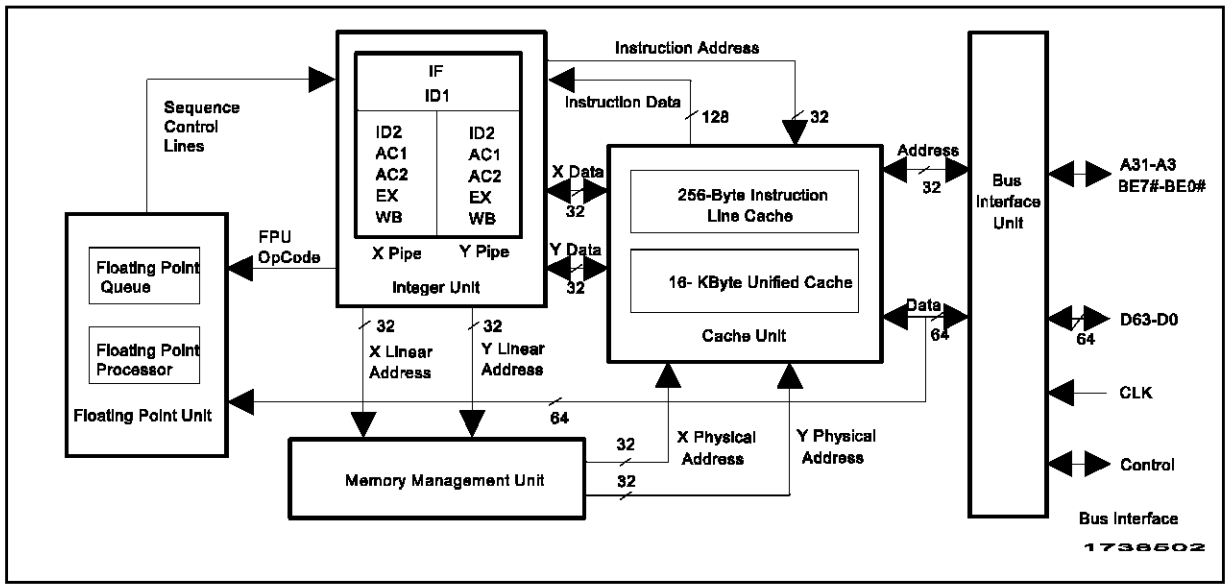
X86 Instruction Set Compatible
- Runs Windows 95, Windows NT, DOS, UNIX, Novell, OS/2, Solaris, and others

Optimum Performance for Windows®95
- Intelligent instruction dispatch
- Out-of-order execution
- Register renaming
- Data forwarding
- Branch prediction
- Speculative execution

64-Bit Data Bus
- P54C socket compatible for quick time to market

The ST6x86™ microprocessor is a superscalar, superpipelined CPU that provides sixth-generation performance for x86 software. Since the ST6x86 CPU is fully compatible with the x86 instruction set, it is capable of executing a wide range of existing operating systems and applications, including Windows 95, DOS, Unix, Windows NT, Novell, OS/2, and Solaris. The ST6x86 CPU achieves top performance levels through the use of two optimized superpipelined integer units and an on-chip floating point unit. The superpipelined architecture reduces timing constraints and allows the ST6x86 CPU to achieve P90+ performance levels and above. In addition, the ST6x86 CPU's integer and floating point units are optimized for maximum instruction throughput by using advanced architectural techniques, including register renaming, out-of-order execution, data forwarding, branch prediction, and speculative execution. These design innovations eliminate many data dependencies and resource conflicts that provide optimum performance for Windows 95 software.

BLOCK DIAGRAM



1.0 ARCHITECTURE OVERVIEW

The SGS-THOMSON ST6x86 CPU is a leader in the sixth generation of high performance, x86-compatible microprocessors. Increased performance is accomplished by the use of superscalar and superpipelined design techniques.

The ST6x86 CPU is superscalar in that it contains two separate pipelines that allow multiple instructions to be processed at the same time. The use of advanced processing technology and the increased number of pipeline stages (superpipelining) allow the ST6x86 CPU to achieve clocks rates of 80, 100, 110, 120, 133 MHz and above.

Through the use of unique architectural features, the ST6x86 processor eliminates many data dependencies and resource conflicts, resulting in optimal performance for both 16-bit and 32-bit x86 software.

The ST6x86 CPU contains two caches: a 16-KByte dual-ported unified cache and a 256-byte instruction line cache. Since the unified cache can store instructions and data in any ratio, the unified cache offers a higher hit rate than separate data and instruction caches of equal size. An increase in overall cache-to-integer unit bandwidth is achieved by supplementing the unified cache with a small, high-speed, fully associative instruction line cache. The inclusion of the instruction line cache avoids excessive conflicts between code and data accesses in the unified cache.

The on-chip FPU allows floating point instructions to execute in parallel with integer instructions and features a 64-bit data interface. The FPU incorporates a four-deep instruction queue and a four-deep store queue to facilitate parallel execution.

The ST6x86 CPU operates from a 3.3 volt power supply resulting in reasonable power consumption at all frequencies. In addition, the ST6x86 CPU incorporates a low power suspend mode, stop clock capability, and system management mode (SMM) for power sensitive applications.

1.1 Major Functional Blocks

The ST6x86 processor consists of five major functional blocks, as shown in the overall block diagram on the first page of this manual:

- Integer Unit
- Cache Unit
- Memory Management Unit
- Floating Point Unit
- Bus Interface Unit

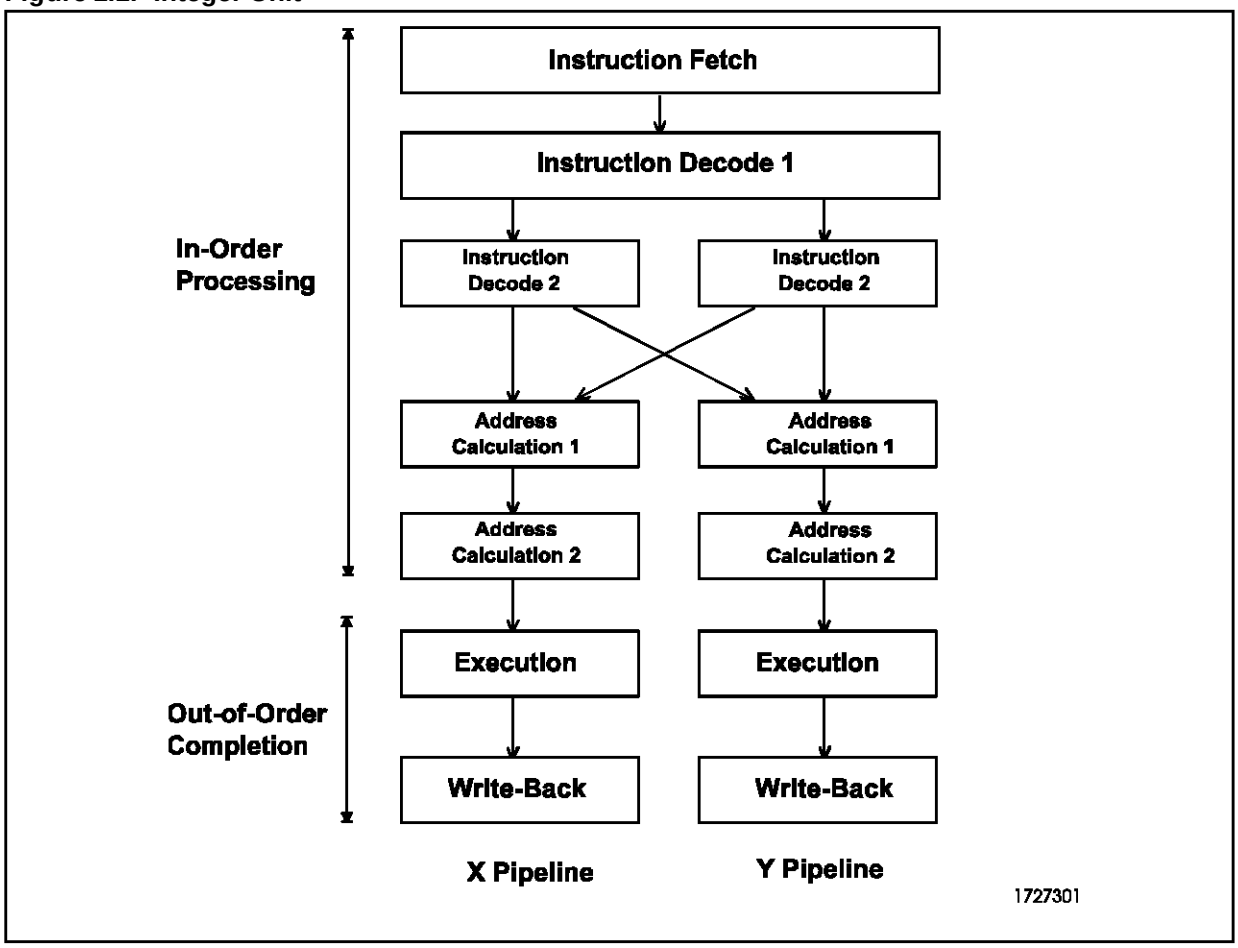
Instructions are executed in the X and Y pipelines within the Integer Unit and also in the Floating Point Unit (FPU). The Cache Unit stores the most recently used data and instructions to allow fast access to the information by the Integer Unit and FPU.

Physical addresses are calculated by the Memory Management Unit and passed to the Cache Unit and the Bus Interface Unit (BIU). The BIU provides the interface between the external system board and the processor's internal execution units.

1.2 Integer Unit

The Integer Unit (Figure 1.1) provides parallel instruction execution using two seven-stage integer pipelines. Each of the two pipelines, X and Y, can process several instructions simultaneously.

Figure 1.1. Integer Unit



The Integer Unit consists of the following pipeline stages:

- Instruction Fetch (IF)
- Instruction Decode 1 (ID1)
- Instruction Decode 2 (ID2)
- Address Calculation 1 (AC1)
- Address Calculation 2 (AC2)
- Execute (EX)
- Write-Back (WB)

The instruction decode and address calculation functions are both divided into superpipelined stages.

1.2.1 Pipeline Stages

The **Instruction Fetch** (IF) stage, shared by both the X and Y pipelines, fetches 16 bytes of code from the cache unit in a single clock cycle. Within this section, the code stream is checked for any branch instructions that could affect normal program sequencing.

If an unconditional or conditional branch is detected, branch prediction logic within the IF stage generates a predicted target address for the instruction. The IF stage then begins fetching instructions at the predicted address.

The superpipelined **Instruction Decode** function contains the ID1 and ID2 stages. ID1, shared by both pipelines, evaluates the code stream provided by the IF stage and determines the number of bytes in each instruction. Up to two instructions per clock are delivered to the ID2 stages, one in each pipeline.

The ID2 stages decode instructions and send the decoded instructions to either the X or Y pipeline for execution. The particular pipeline is chosen, based on which instructions are already in each pipeline and how fast they are expected to flow through the remaining pipeline stages.

The **Address Calculation** function contains two stages, AC1 and AC2. If the instruction refers to a memory operand, the AC1 calculates a linear memory address for the instruction.

The AC2 stage performs any required memory management functions, cache accesses, and register file accesses. If a floating point instruction is detected by AC2, the instruction is sent to the FPU for processing.

The **Execute** (EX) stage executes instructions using the operands provided by the address calculation stage.

The **Write-Back** (WB) stage is the last IU stage. The WB stage stores execution results either to a register file within the IU or to a write buffer in the cache control unit.

1.2.2 Out-of-Order Processing

If an instruction executes faster than the previous instruction in the other pipeline, the instructions may complete out of order. All instructions are processed in order, up to the EX stage. While in the EX and WB stages, instructions may be completed out of order.

If there is a data dependency between two instructions, the necessary hardware interlocks are enforced to ensure correct program execution. Even though instructions may complete out of order, exceptions and writes resulting from the instructions are always issued in program order.

1.2.3 Pipeline Selection

In most cases, instructions are processed in either pipeline and without pairing constraints on the instructions. However, certain instructions are processed only in the X pipeline:

- Branch instructions
- Floating point instructions
- Exclusive instructions

Branch and floating point instructions may be paired with a second instruction in the Y pipeline.

Exclusive Instructions cannot be paired with instructions in the Y pipeline. These instructions typically require multiple memory accesses. Although exclusive instructions may not be paired, hardware from both pipelines is used to accelerate instruction completion. Listed below are the ST6x86 CPU exclusive instruction types:

- Protected mode segment loads
- Special register accesses
  (Control, Debug, and Test Registers)
- String instructions
- Multiply and divide
- I/O port accesses
- Push all (PUSHA) and pop all (POPA)
- Intersegment jumps, calls, and returns

1.2.4 Data Dependency Solutions

When two instructions that are executing in parallel require access to the same data or register, one of the following types of data dependencies may occur:

- Read-After-Write (RAW)
- Write-After-Read (WAR)
- Write-After-Write (WAW)

Data dependencies typically force serialized execution of instructions. However, the ST6x86 CPU implements three mechanisms that allow parallel execution of instructions containing data dependencies:

- Register Renaming
- Data Forwarding
- Data Bypassing

The following sections provide detailed examples of these mechanisms.

1.2.4.1 Register Renaming

The ST6x86 CPU contains 32 physical general purpose registers. Each of the 32 registers in the register file can be temporarily assigned as one of the general purpose registers defined by the x86 architecture (EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP). For each register write operation a new physical register is selected to allow previous data to be retained temporarily. Register renaming effectively removes all WAW and WAR dependencies. The programmer does not have to consider register renaming; it is completely transparent to both the operating system and application software.

Example #1 - Register Renaming Eliminates Write-After-Read (WAR) Dependency

A WAR dependency exists when the first in a pair of instructions reads a logical register, and the second instruction writes to the same logical register. This type of dependency is illustrated by the pair of instructions shown below:

X PIPE	Y PIPE
(1) MOV BX, AX	(2) ADD AX, CX
BX ← AX	AX ← AX + CX

Note: In this and the following examples the original instruction order is shown in parentheses.

In the absence of register renaming, the ADD instruction in the Y pipe would have to be stalled to allow the MOV instruction in the X pipe to read the AX register.

The ST6x86 CPU, however, avoids the Y pipe stall (Table 1.1). As each instruction executes, the results are placed in new physical registers to avoid the possibility of overwriting a logical register value and to allow the two instructions to complete in parallel (or out of order) rather than in sequence.

Table 1.1. Register Renaming with WAR Dependency

Instruction	Physical Register Contents					Action	
	Reg0	Reg1	Reg2	Reg3	Reg4	Pipe	
(Initial)	AX	BX	CX				
MOV BX, AX	AX		CX	BX		X	Reg3 ← Reg0
ADD AX, CX			CX	BX	AX	Y	Reg4 ← Reg0 + Reg2

Note: The representation of the MOV and ADD instructions in the final column of Table 1.1 are completely independent.

Example #2 - Register Renaming Eliminates Write-After-Write (WAW) Dependency

A WAW dependency occurs when two consecutive instructions perform writes to the same logical register. This type of dependency is illustrated by the pair of instructions shown below:

X PIPE	Y PIPE
(1) ADD AX, BX	(2) MOV AX, [mem]
AX ←AX + BX	AX ← [mem]

Without register renaming, the MOV instruction in the Y pipe would have to be stalled to guarantee that the ADD instruction in the X pipe would write its results to the AX register first.

The ST6x86 CPU uses register renaming and avoids the Y pipe stall. The contents of the AX and BX registers are placed in physical registers (Table 1.2). As each instruction executes, the results are placed in new physical registers to avoid the possibility of overwriting a logical register value and to allow the two instructions to complete in parallel (or out of order) rather than in sequence.

Table 1.2. Register Renaming with WAW Dependency

Instruction	Physical Register Contents				Action	
	Reg0	Reg1	Reg2	Reg3	Pipe	
(Initial)	AX	BX				
ADD AX, BX		BX	AX		X	Reg2 ← Reg0 + Reg1
MOV AX, [mem]		BX		AX	Y	Reg3 ← [mem]

Note: All subsequent reads of the logical register AX will refer to Reg 3, the result of the MOV instruction.

1.2.4.2 Data Forwarding

Register renaming alone cannot remove RAW dependencies. The ST6x86 CPU uses two types of data forwarding in conjunction with register renaming to eliminate RAW dependencies:

- Operand Forwarding
- Result Forwarding

Operand forwarding takes place when the first in a pair of instructions performs a move from register or memory, and the data that is read by the first instruction is required by the second instruction. The ST6x86 CPU performs the read operation and makes the data read available to both instructions simultaneously.

Result forwarding takes place when the first in a pair of instructions performs an operation (such as an ADD) and the result is required by the second instruction to perform a move to a register or memory. The ST6x86 CPU performs the required operation and stores the results of the operation to the destination of both instructions simultaneously.

Example #3 - Operand Forwarding Eliminates Read-After-Write (RAW) Dependency

A RAW dependency occurs when the first in a pair of instructions performs a write, and the second instruction reads the same register. This type of dependency is illustrated by the pair of instructions shown below in the X and Y pipelines:

X PIPE	Y PIPE
(1) MOV AX, [mem]	(2) ADD BX, AX
AX ← [mem]	BX ← AX + BX

The ST6x86 CPU uses operand forwarding and avoids a Y pipe stall (Table 1.3). Operand forwarding allows simultaneous execution of both instructions by first reading memory and then making the results available to both pipelines in parallel.

Table 1.3. Example of Operand Forwarding

Instruction	Physical Register Contents				Action	
	Reg0	Reg1	Reg2	Reg3	Pipe	
(Initial)	AX	BX				
MOV AX, [mem]		BX	AX		X	Reg2 ← [mem]
ADD BX, AX			AX	BX	Y	Reg3 ← [mem] + Reg1

Operand forwarding can only occur if the first instruction does not modify its source data. In other words, the instruction is a move type instruction (for example, MOV, POP, LEA). Operand forwarding occurs for both register and memory operands. The size of the first instruction destination and the second instruction source must match.

Example #4 - Result Forwarding Eliminates Read-After-Write (RAW) Dependency

In this example, a RAW dependency occurs when the first in a pair of instructions performs a write, and the second instruction reads the same register. This dependency is illustrated by the pair of instructions in the X and Y pipelines, as shown below:

X PIPE	Y PIPE
(1) ADD AX, BX	(2) MOV [mem], AX
AX ←AX + BX	[mem] ← AX

The ST6x86 CPU uses result forwarding and avoids a Y pipe stall (Table 1.4). Instead of transferring the contents of the AX register to memory, the result of the previous ADD instruction (Reg0 + Reg1) is written directly to memory, thereby saving a clock cycle.

Table 1.4. Result Forwarding Example

Instruction	Physical Register Contents			Action	
	Reg0	Reg1	Reg2	Pipe	
(Initial)	AX	BX			
ADD AX, BX		BX	AX	X	Reg2 ←Reg0 + Reg1
MOV [mem], AX		BX	AX	Y	[mem] ← Reg0 +Reg1

The second instruction must be a move instruction and the destination of the second instruction may be either a register or memory.

1.2.4.3 Data Bypassing

In addition to register renaming and data forwarding, the ST6x86 CPU implements a third data dependency-resolution technique called data bypassing. Data bypassing reduces the performance penalty of those memory data RAW dependencies that cannot be eliminated by data forwarding.

Data bypassing is implemented when the first in a pair of instructions writes to memory and the second instruction reads the same data from memory. The ST6x86 CPU retains the data from the first instruction and passes it to the second instruction, thereby eliminating a memory read cycle. Data bypassing only occurs for cacheable memory locations.

Example #1- Data Bypassing with Read-After-Write (RAW) Dependency

In this example, a RAW dependency occurs when the first in a pair of instructions performs a write to memory and the second instruction reads the same memory location. This dependency is illustrated by the pair of instructions in the X and Y pipelines as shown below:

X PIPE	**Y PIPE**
(1) ADD [mem], AX	(2) SUB BX, [mem]
[mem] ¨[mem] + AX	BX ¨ BX - [mem]

The ST6x86 CPU uses data bypassing and stalls the Y pipe for only one clock by eliminating the Y pipe's memory read cycle (Table 1.5). Instead of reading memory in the Y pipe, the result of the previous instruction ([mem] + Reg0) is used to subtract from Reg1, thereby saving a memory access cycle.

Table 1.5. Example of Data Bypassing

Instruction	Physical Register Contents			Action	
	Reg0	Reg1	Reg2	Pipe	
(Initial)	AX	BX			
ADD [mem], AX	AX	BX		X	[mem] ← [mem] + Reg0
SUB BX, [mem]	AX		BX	Y	Reg2 ← Reg1 - {[mem] + Reg0}

1.2.5 Branch Control

Branch instructions occur on average every four to six instructions in x86-compatible programs. When the normal sequential flow of a program changes due to a branch instruction, the pipeline stages may stall while waiting for the CPU to calculate, retrieve, and decode the new instruction stream. The ST6x86 CPU minimizes the performance degradation and latency of branch instructions through the use of branch prediction and speculative execution.

1.2.5.1 Branch Prediction

The ST6x86 CPU uses a 256-entry, 4-way set associative Branch Target Buffer (BTB) to store branch target addresses and branch prediction information. During the fetch stage, the instruction stream is checked for the presence of branch instructions. If an unconditional branch instruction is encountered, the ST6x86 CPU accesses the BTB to check for the branch instruction's target address. If the branch instruction's target address is found in the BTB, the ST6x86 CPU begins fetching at the target address specified by the BTB.

In case of conditional branches, the BTB also provides history information to indicate whether the branch is more likely to be taken or not taken. If the conditional branch instruction is found in the BTB, the ST6x86 CPU begins fetching instructions at the predicted target address. If the conditional branch misses in the BTB, the ST6x86 CPU predicts that the branch will not be taken, and instruction fetching continues with the next sequential instruction. The decision to fetch the taken or not taken target address is based on a four-state branch prediction algorithm.

Once fetched, a conditional branch instruction is first decoded and then dispatched to the X pipeline only. The conditional branch instruction proceeds through the X pipeline and is then resolved in either the EX stage or the WB stage. The conditional branch is resolved in the EX stage, if the instruction responsible for setting the condition codes is completed prior to the execution of the branch. If the instruction that sets the condition codes is executed in parallel with the branch, the conditional branch instruction is resolved in the WB stage.

Correctly predicted branch instructions execute in a single core clock. If resolution of a branch indicates that a misprediction has occurred, the ST6x86 CPU flushes the pipeline and starts fetching from the correct target address. The ST6x86 CPU prefetches both the predicted and the non-predicted path for each conditional branch, thereby eliminating the cache access cycle on a misprediction. If the branch is resolved in the EX stage, the resulting misprediction latency is four cycles. If the branch is resolved in the WB stage, the latency is five cycles.

Since the target address of return (RET) instructions is dynamic rather than static, the ST6x86 CPU caches target addresses for RET instructions in an eight-entry return stack rather than in the BTB. The return address is pushed on the return stack during a CALL instruction and popped during the corresponding RET instruction.

1.2.5.2 Speculative Execution

The ST6x86 CPU is capable of speculative execution following a floating point instruction or predicted branch. Speculative execution allows the pipelines to continuously execute instructions following a branch without stalling the pipelines waiting for branch resolution. The same mechanism is used to execute floating point instructions (see Section 1.5) in parallel with integer instructions.

The ST6x86 CPU is capable of up to four levels of speculation (i.e., combinations of four conditional branches and floating point operations). After generating the fetch address using branch prediction, the CPU checkpoints the machine state (registers, flags, and processor environment), increments the speculation level counter, and begins operating on the predicted instruction stream.

Once the branch instruction is resolved, the CPU decreases the speculation level. For a correctly predicted branch, the status of the checkpointed resources is cleared. For a branch misprediction, the ST6x86 processor generates the correct fetch address and uses the checkpointed values to restore the machine state in a single clock.

In order to maintain compatibility, writes that result from speculatively executed instructions are not permitted to update the cache or external memory until the appropriate branch is resolved. Speculative execution continues until one of the following conditions occurs:

1) A branch or floating point operation is decoded and the speculation level is already at four.

2) An exception or a fault occurs.

3) The write buffers are full.

4) An attempt is made to modify a non-checkpointed resource (i.e., segment registers, system flags).

1.3 Cache Units

The ST6x86 CPU employs two caches, the Unified Cache and the Instruction Line Cache (Figure 1.2).

1.3.1 Unified Cache

The 16-KByte unified write-back cache functions as the primary data cache and as the secondary instruction cache. Configured as a four-way set-associative cache, the cache stores up to 16 KBytes of code and data in 512 lines. The cache is dual-ported and allows any two of the following operations to occur in parallel:

- Code fetch
- Data read (X pipe, Y pipeline or FPU)
- Data write (X pipe, Y pipeline or FPU)

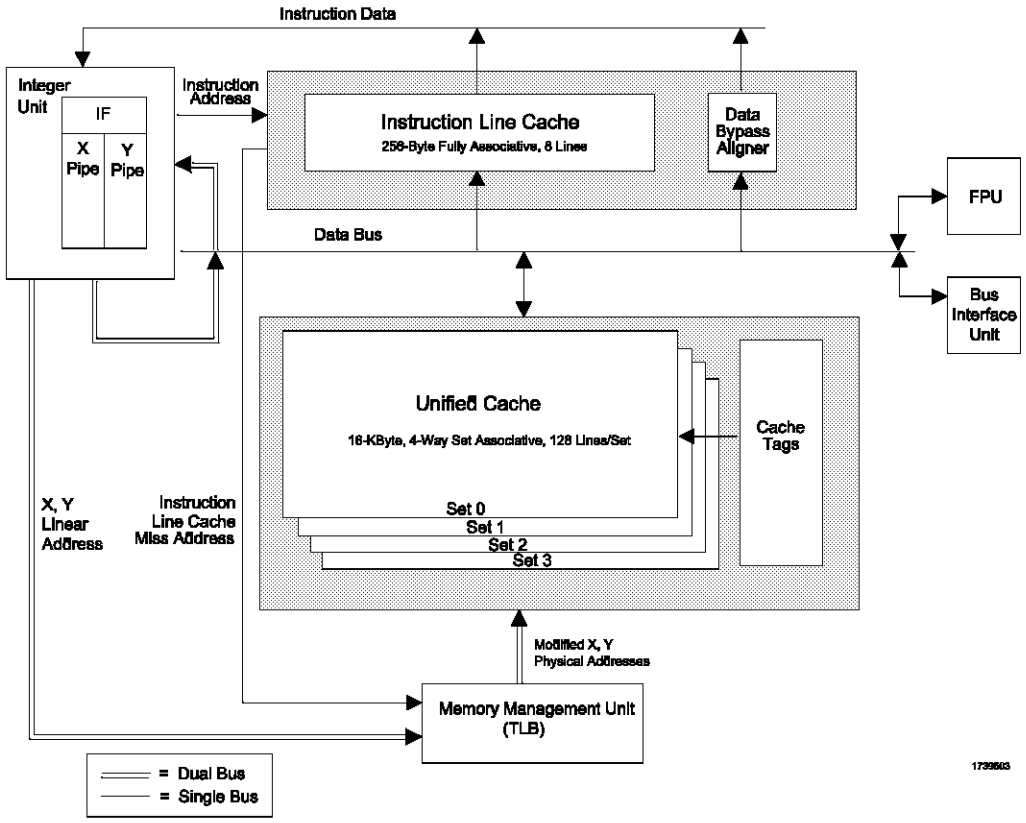
The unified cache uses a pseudo-LRU replacement algorithm and can be configured to allocate new lines on read misses only or on read and write misses.

1.3.2 Instruction Line Cache

The fully associative 256-byte instruction line cache serves as the primary instruction cache. The instruction line cache is filled from the unified cache through the data bus. Fetches from the integer unit that hit in the instruction line cache do not access the unified cache. If an instruction line cache miss occurs, the instruction line data from the unified cache is transferred to the instruction line cache and the integer unit, simultaneously.

The instruction line cache uses a pseudo-LRU replacement algorithm. To ensure proper operation in the case of self-modifying code, any writes to the unified cache are checked against the contents of the instruction line cache. If a hit occurs in the instruction line cache, the appropriate line is invalidated.

Figure 1.2. Cache Unit Operations



1.4 Memory Management Unit

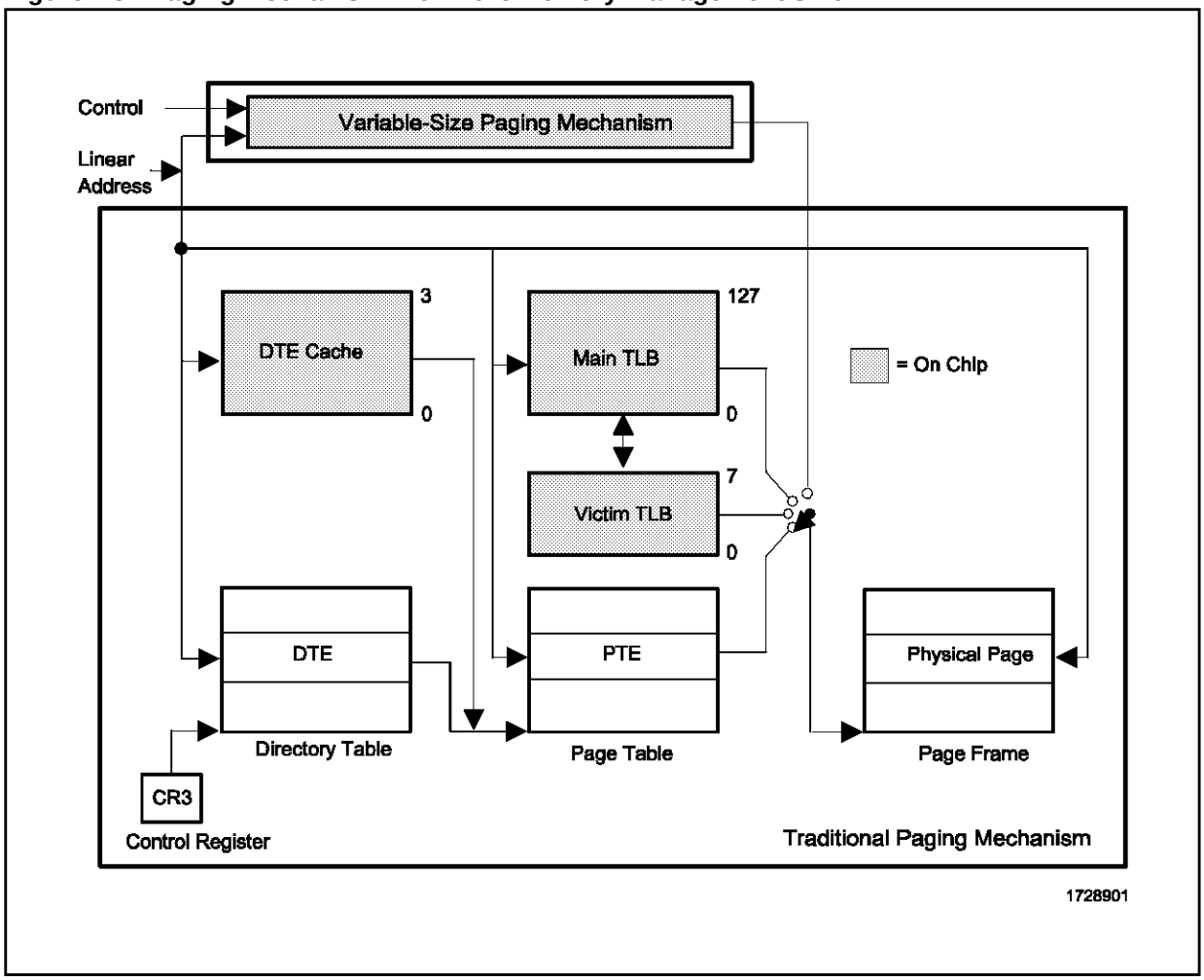
The Memory Management Unit (MMU), shown in Figure 1.3, translates the linear address supplied by the IU into a physical address to be used by the unified cache and the bus interface. Memory management procedures are x86 compatible, adhering to standard paging mechanisms.

The ST6x86 MMU includes two paging mechanisms (Figure 1.3), a traditional paging mechanism, and a ST6x86 variable-size paging mechanism.

1.4.1 Variable-Size Paging Mechanism

The SGS-THOMSON variable-size paging mechanism allows software to map pages between 4 KBytes and 4 GBytes in size. The large contiguous memories provided by this mechanism help avoid TLB (Translation Lookaside Buffer) thrashing associated with some operating systems and applications. For example, use of a single large page instead of a series of small 4-KByte pages can greatly improve performance in an application using a large video memory buffer.

Figure 1.3. Paging Mechanism within the Memory Management Unit



1.4.2 Traditional Paging Mechanism

The traditional paging mechanism has been enhanced on the ST6x86 CPU with the addition of the Directory Table Entry (DTE) cache and the Victim TLB. The main TLB (Translation Lookaside Buffer) is a direct-mapped 128-entry cache for page table entries.

The four-entry fully associative DTE cache stores the most recent DTE accesses. If a Page Table Entry (PTE) miss occurs followed by a DTE hit, only a single memory access to the PTE table is required.

The Victim TLB stores PTEs which have been displaced from the main TLB due to a TLB miss. If a PTE access occurs while the PTE is stored in the victim TLB, the PTE in the victim TLB is swapped with a PTE in the main TLB. This has the effect of selectively increasing TLB associativity. The ST6x86 CPU updates the eight-entry fully associative victim TLB on an oldest entry replacement basis.

1.5 Floating Point Unit

The ST6x86 Floating Point Unit (FPU) interfaces to the integer unit and the cache unit through a 64-bit bus. The ST6x86 FPU is x87 instruction set compatible and adheres to the IEEE-754 standard. Since most applications contain FPU instructions mixed with integer instructions, the ST6x86 FPU achieves high performance by completing integer and FPU operations in parallel.

FPU Parallel Execution

The ST6x86 CPU executes integer instructions in parallel with FPU instructions. Integer instructions may complete out of order with respect to the FPU instructions. The ST6x86 CPU maintains x86 compatibility by signaling exceptions and issuing write cycles in program order.

As previously discussed, FPU instructions are always dispatched to the integer unit's X pipeline. The address calculation stage of the X pipeline checks for memory management exceptions and accesses memory operands used by the FPU. If no exceptions are detected, the ST6x86 CPU checkpoints the state of the CPU and, during AC2, dispatches the floating point instruction to the FPU instruction queue. The ST6x86 CPU can then complete any subsequent integer instructions speculatively and out of order relative to the FPU instruction and relative to any potential FPU exceptions which may occur.

As additional FPU instructions enter the pipeline, the ST6x86 CPU dispatches up to four FPU instructions to the FPU instruction queue. The ST6x86 CPU continues executing speculatively and out of order, relative to the FPU queue, until the ST6x86 CPU encounters one of the conditions that causes speculative execution to halt. As the FPU completes instructions, the speculation level decreases and the checkpointed resources are available for reuse in subsequent operations. The ST6x86 FPU also uses a set of four write buffers to prevent stalls due to speculative writes.

1.6 Bus Interface Unit

The Bus Interface Unit (BIU) provides the signals and timing required by external circuitry. The signal descriptions and bus interface timing information is provided in Chapters 3 and 4 of this manual.

2.0 Programming Interface

In this chapter, the internal operations of the ST6x86 CPU are described mainly from an application programmer's point of view. Included in this chapter are descriptions of processor initialization, the register set, memory addressing, various types of interrupts and the shutdown and halt process.

An overview of real, virtual 8086, and protected operating modes is also included in this chapter. The FPU operations are described separately at the end of the chapter. This manual does not - and is not intended to - describe the ST6X86 microprocessor or its operations at the circuit level.

Table 2.1. Initialized Register Controls

REGISTER	REGISTER NAME	INITIALIZED CONTENTS	COMMENTS
EAX	Accumulator	xxxx xxxxh	0000 0000h indicates self-test passed.
EBX	Base	xxxx xxxxh	
ECX	Count	xxxx xxxxh	
EDX	Data	05 + Device ID	Device ID = 31h or 33h (2X clock) Device ID = 35h or 37h (3X clock)
EBP	Base Pointer	xxxx xxxxh	
ESI	Source Index	xxxx xxxxh	
EDI	Destination Index	xxxx xxxxh	
ESP	Stack Pointer	xxxx xxxxh	
EFLAGS	Flag Word	0000 0002h	
EIP	Instruction Pointer	0000 FFF0h	
ES	Extra Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh.
CS	Code Segment	F000h	Base address set to FFFF 0000h. Limit set to FFFFh.
SS	Stack Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh.
DS	Data Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh.
FS	Extra Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh.
GS	Extra Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh.
IDTR	Interrupt Descriptor Table Register	Base = 0, Limit = 3FFh	
GDTR	Global Descriptor Table Register	xxxx xxxxh, xxxxh	
LDTR	Local Descriptor Table Register	xxxx xxxxh, xxxxh	
TR	Task Register	xxxxh	
CR0	Machine Status Word	6000 0010h	
CR2	Control Register 2	xxxx xxxxh	
CR3	Control Register 3	xxxx xxxxh	
CCR (0-5)	Configuration Control (0-5)	00h	
ARR (0-7)	Address Region Registers (0-7)	00h	
RCR (0-7)	Region Control Registers (0-7)	00h	
DIR0	Device Identification 0	31h or 33h (2X clock) 35h or 37h (3X clock)	
DIR1	Device Identification 1	Step ID + Revision ID	
DR7	Debug Register 7	0000 0400h	

Note: x = Undefined value

2.1 ST6x86 Configuration Registers

A set of 24 on-chip ST6x86 configuration registers are used to enable features in the ST6x86 CPU. These registers assign non-cached memory areas, set up SMM, provide CPU identification information and control various features such as cache write policy, and bus locking control. There are four groups of registers within the ST6x86 configuration register set:

- 6 Configuration Control Registers (CCRx)
- 8 Address Region Registers (ARRx)
- 8 Region Control Registers (RCRx)
- 2 Device Identification Registers (DIRx)

Access to the configuration registers is achieved by writing the register index number for the configuration register to I/O port 22h. I/O port 23h is then used for data transfer.

Each I/O port 23h data transfer must be preceded by a valid I/O port 22h register index selection. Otherwise, the current 22h, and the second and later I/O port 23h operations communicate through the I/O port to produce external I/O cycles. All reads from I/O port 22h produce external I/O cycles. Accesses that hit within the on-chip configuration registers do not generate external I/O cycles.

After reset, configuration registers with indexes CO-CFh and FE-FFh are accessible. To prevent potential conflicts with other devices which may use ports 22 and 23h to access their registers, the remaining registers (indexes D0-FDh) are accessible only if the MAPEN(3-0) bits in CCR3 are set to 1h. See Figure 2.4 (Page 24) for more information on the MAPEN(3-0) bit locations.

If MAPEN[3-0] = 1h, any access to indexes in the range 00-FFh will not create external I/O bus cycles. Registers with indexes C0-CFh, FE, FFh are accessible regardless of the state of MAPEN[3-0]. If the register index number is outside the C0-CFh or FE-FFh ranges, and MAPEN[3-0] are set to 0h, external I/O bus cycles occur. Table 2.2 (Page 20) lists the MAPEN[3-0] values required to access each ST6x86 configuration register. All bits in the configuration registers are initialized to zero following reset unless specified otherwise.

Valid register index numbers include C0h to E3h, E8h, E9h, FEh and FFh (if MAPEN[3-0] = 1).

2.1.1 Configuration Control Registers

(CCR0 - CCR5) control several functions, including non-cacheable memory, write-back regions, and SMM features. A list of the configuration registers is listed in Table 2.2 (Page 20). The configuration registers are described in greater detail in the following pages.

Table 2.2. ST6x86 CPU Configuration Registers

REGISTER NAME	ACRONYM	REGISTER INDEX	WIDTH (Bits)	MAPEN VALUE NEEDED FOR ACCESS
Configuration Control 0	CCR0	C0h	8	x
Configuration Control 1	CCR1	C1h	8	x
Configuration Control 2	CCR2	C2h	8	x
Configuration Control 3	CCR3	C3h	8	x
Configuration Control 4	CCR4	E8h	8	1
Configuration Control 5	CCR5	E9h	8	1
Address Region 0	ARR0	C4h - C6h	24	x
Address Region 1	ARR1	C7h - C9h	24	x
Address Region 2	ARR2	CAh - CCh	24	x
Address Region 3	ARR3	CDh - CFh	24	x
Address Region 4	ARR4	D0h - D2h	24	1
Address Region 5	ARR5	D3h - D5h	24	1
Address Region 6	ARR6	D6h - D8h	24	1
Address Region 7	ARR7	D9h - DBh	24	1
Region Control 0	RCR0	DCh	8	1
Region Control 1	RCR1	DDh	8	1
Region Control 2	RCR2	DEh	8	1
Region Control 3	RCR3	DFh	8	1
Region Control 4	RCR4	E0h	8	1
Region Control 5	RCR5	E1h	8	1
Region Control 6	RCR6	E2h	8	1
Region Control 7	RCR7	E3h	8	1
Device Identification 0	DIR0	FEh	8	x
Device Identification 1	DIR1	FFh	8	x

Note: x = Don't Care

Figure 2.1. ST6x86 Configuration Control Register 0 (CCR0)

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	NC1	Reserved

Table 2.3. CCR0 Bit Definitions

BIT POSITION	NAME	DESCRIPTION
1	NC1	No Cache 640 KByte - 1 MByte If = 1: Address region 640 KByte to 1 MByte is non-cacheable. If = 0: Address region 640 KByte to 1 MByte is cacheable.

Note: Bits 0, 2 through 7 are reserved.

Figure 2.2. ST6x86 Configuration Control Register 1 (CCR1)

7	6	5	4	3	2	1	0
SM3	Reserved	Reserved	NO_LOCK	Reserved	SMAC	USE_SMI	Reserved

Table 2.4. CCR1 Bit Definitions

BIT POSITION	NAME	DESCRIPTION
1	USE_SMI	Enable SMM and SMIACT# Pins If = 1: SMI# and SMIACT# pins are enabled. If = 0: SMI# pin ignored and SMIACT# pin is driven inactive.
2	SMAC	System Management Memory Access If = 1: Any access to addresses within the SMM address space, access system management memory instead of main memory. SMI# input is ignored. Used when initializing or testing SMM memory. If = 0: No effect on access.
4	NO_LOCK	Negate LOCK# If = 1: All bus cycles are issued with LOCK# pin negated except page table accesses and interrupt acknowledge cycles. Interrupt acknowledge cycles are executed as locked cycles even though LOCK# is negated. With NO_LOCK set, previously noncacheable locked cycles are executed as unlocked cycles and therefore, may be cached. This results in higher performance. Refer to Region Control Registers for information on eliminating locked CPU bus cycles only in specific address regions.
7	SM3	SMM Address Space Address Region 3 If = 1: Address Region 3 is designated as SMM address space.

Note: Bits 0, 3, 5 and 6 are reserved.

Figure 2.3. ST6x86 Configuration Control Register 2 (CCR2)

7	6	5	4	3	2	1	0
USE_SUSP	Reserved	Reserved	WPR1	SUSP_HLT	LOCK_NW	Reserved	Reserved

Table 2.5. CCR2 Bit Definitions

BIT POSITION	NAME	DESCRIPTION
2	LOCK_NW	Lock NW If = 1: NW bit in CR0 becomes read only and the CPU ignores any writes to the NW bit. If = 0: NW bit in CR0 can be modified.
3	SUSP_HLT	Suspend on Halt If = 1: Execution of the HLT instruction causes the CPU to enter low power suspend mode.
4	WPR1	Write-Protect Region 1 If = 1: Designates any cacheable accesses in 640 KByte to 1 MByte address region are write protected.
7	USE_SUSP	Use Suspend Mode (Enable Suspend Pins) If = 1: SUSP# and SUSPA# pins are enabled. If = 0: SUSP# pin is ignored and SUSPA# pin floats.

Note: Bits 0,1, 5 and 6 are reserved.

Figure 2.4. ST6x86 Configuration Control Register 3 (CCR3)

7	6	5	4	3	2	1	0
MAPEN				Reserved	LINBRST	NMI_EN	SMI_LOCK

Table 2.6. CCR3 Bit Definitions

BIT POSITION	NAME	DESCRIPTION
0	SMI_LOCK	SMI Lock If = 1: The following SMM configuration bits can only be modified while in an SMI service routine: CCR1: USE_SMI, SMAC, SM3 CCR3: NMI_EN ARR3: Starting address and block size. Once set, the features locked by SMI_LOCK cannot be unlocked until the RESET pin is asserted.
1	NMI_EN	NMI Enable If = 1: NMI interrupt is recognized while servicing an SMI interrupt. NMI_EN should be set only while in SMM, after the appropriate SMI interrupt service routine has been setup.
2	LINBRST	If = 1: Use linear address sequence during burst cycles. If = 0: Use “1 + 4” address sequence during burst cycles. The “1 + 4” address sequence is compatible with Pentium’s burst address sequence.
4 - 7	MAPEN	MAP Enable If = 1h: All configuration registers are accessible. If = 0h: Only configuration registers with indexes C0-CFh, FEh and FFh are accessible.

Note: Bit 3 is reserved.

Figure 2.5. ST6x86 Configuration Control Register 4 (CCR4)

7	6	5	4	3	2	1	0
CPUID	Reserved	Reserved	DTE_EN	Reserved	IORT		

Table 2.7. CCR4 Bit Definitions

BIT POSITION	NAME	DESCRIPTION
0 - 2	IORT	I/O Recovery Time Specifies the minimum number of bus clocks between I/O accesses: 0h = 1 clock delay 1h = 2 clock delay 2h = 4 clock delay 3h = 8 clock delay 4h = 16 clock delay 5h = 32 clock delay (default value after RESET) 6h = 64 clock delay 7h = no delay
4	DTE_EN	Enable Directory Table Entry Cache If = 1: the Directory Table Entry cache is enabled.
7	CPUID	Enable CPUID instruction. If = 1: the ID bit in the EFLAGS register can be modified and execution of the CPUID instruction occurs as documented in section 6.3. If = 0: the ID bit in the EFLAGS register can not be modified and execution of the CPUID instruction causes an invalid opcode exception.

Note: Bits 3 and bits 5 and 6 are reserved.

Figure 2.6. ST6x86 Configuration Control Register 5 (CCR5)

7	6	5	4	3	2	1	0
Reserved	Reserved	ARREN	LBR1	Reserved	Reserved	Reserved	WT_ALLOC

Table 2.8. CCR5 Bit Definitions

BIT POSITION	NAME	DESCRIPTION
0	WT_ALLOC	Write-Through Allocate If = 1: New cache lines are allocated for read and write misses. If = 0: New cache lines are allocated only for read misses.
4	LBR1	Local Bus Region 1 If = 1: LBA# pin is asserted for all accesses to the 640 KByte to 1 MByte address region.
5	ARREN	Enable ARR Registers If = 1: Enables all ARR registers. If = 0: Disables the ARR registers. If SM3 is set, ARR3 is enabled regardless of the setting of ARREN.

Note: Bits 1 through 3 and 6 though 7 are reserved.

2.1.2 Address Region Registers

The Address Region Registers (ARR0 - ARR7) (Figure 2.7) are used to specify the location and size for the eight address regions.

Attributes for each address region are specified in the Region Control Registers (RCR0-RCR7). ARR7 and RCR7 are used to define system main memory and differ from ARR0-6 and RCR0-6.

With non-cacheable regions defined on-chip, the ST6x86 CPU delivers optimum performance by using advanced techniques to eliminate data dependencies and resource conflicts in its execution pipelines. If KEN# is active for accesses to regions defined as non-cacheable by the RCRs, the region is not cached. The RCRs take precedence in this case.

A register index, shown in Table 2.9 (Page 28) is used to select one of three bytes in each ARR.

The starting address of the ARR address region, selected by the START ADDRESS field, must be on a block size boundary. For example, a 128 KByte block is allowed to have a starting address of 0 KBytes, 128 KBytes, 256 KBytes, and so on.

The SIZE field bit definition is listed in Table 2.10, on page 28. If the SIZE field is zero, the address region is of zero size and thus disabled.

Figure 2.7. Address Region Registers (ARR0 - ARR7)

START ADDRESS			SIZE
Memory Address Bits A31-A24	Memory Address Bits A23-A16	Memory Address Bits A15-A12	Size Bits 3-0
7 0	7 0	7 4	3 0

Table 2.9. ARR0 - ARR7 Register Index Assignment

ARR Register	Memory Address (A31 - A24)	Memory Address (A23 - A16)	Memory Address (A15 - A12)	Address Region Size (3 - 0)
ARR0	C4h	C5h	C6h	C6h
ARR1	C7h	C8h	C9h	C9h
ARR2	CAh	CBh	CCh	CCh
ARR3	CDh	CEh	CFh	CFh
ARR4	D0h	D1h	D2h	D2h
ARR5	D3h	D4h	D5h	D5h
ARR6	D6h	D7h	D8h	D8h
ARR7	D9h	DAh	DBh	DBh

Table 2.10. Bit Definitions for SIZE Field

SIZE (3-0)	BLOCK SIZE	
	ARR0-6	ARR7
0h	Disabled	Disabled
1h	4 KBytes	256 KBytes
2h	8 KBytes	512 KBytes
3h	16 KBytes	1 MBytes
4h	32 KBytes	2 MBytes
5h	64 KBytes	4 MBytes
6h	128 KBytes	8 MBytes
7h	256 KBytes	16 MBytes

SIZE (3-0)	BLOCK SIZE	
	ARR0-6	ARR7
8h	512 KBytes	32 MBytes
9h	1 MBytes	64 MBytes
Ah	2 MBytes	128 MBytes
Bh	4 MBytes	256 MBytes
Ch	8 MBytes	512 MBytes
Dh	16 MBytes	1 GBytes
Eh	32 MBytes	2 GBytes
Fh	4 GBytes	4 GBytes

.

2.1.3 Region Control Registers

The Region Control Registers (RCR0 - RCR7) specify the attributes associated with the ARRx address regions. The bit definitions for the region control registers are shown in Figure 2.8 (Page 30) and in Table 2.11 (Page 30). Cacheability, weak write ordering, weak locking, write gathering, cache write through policies and control of the LBA# pin can be activated or deactivated using the attribute bits.

If an address is accessed that is not in a memory region defined by the ARRx registers, the following conditions will apply:

- LBA# pin is asserted
- If the memory address is cached, write-back is enabled if WB/WT# is returned high.
- Writes are not gathered
- Strong locking takes place
- Strong write ordering takes place
- The memory access is cached, if KEN# is returned asserted.

Overlapping Conditions Defined. If two regions specified by ARRx registers overlap and conflicting attributes are specified, the following attributes take precedence:

- LBA# pin is asserted
- Write-back is disabled
- Writes are not gathered
- Strong locking takes place
- Strong write ordering takes place
- The overlapping regions are non-cacheable.

Figure 2.8. Region Control Registers (RCR0-RCR7)

7	6	5	4	3	2	1	0
Reserved	Reserved	NLB	WT	WG	WL	WWO	RCD / RCE*

*Note: RCD is defined for RCR0-RCR6. RCE is defined for RCR7.

Table 2.11. RCR0-RCR7 Bit Definitions

RCRx	BIT POSITION	NAME	DESCRIPTION
0 - 6	0	RCD	If = 1: Disables caching for address region specified by ARRx.
7	0	RCE	If = 1: Enables caching for address region ARR7.
0 - 7	1	WWO	If = 1: Weak write ordering for address region specified by ARRx.
0 - 7	2	WL	If = 1: Weak locking for address region specified by ARRx.
0 - 7	3	WG	If = 1: Write gathering for address region specified by ARRx.
0 - 7	4	WT	If = 1: Address region specified by ARRx is write-through.
0 - 7	5	NLB	If = 1:LBA# pin is not asserted for access to address region specified by ARRx

Note: Bits 6 and 7 are reserved.

Region Cache Disable (RCD). Setting RCD to a one defines the address region as non-cacheable. Whenever possible, the RCRs should be used to define non-cacheable regions rather than using external address decoding and driving the KEN# pin.

Region Cache Enable (RCE). Setting RCE to a one defines the address region as cacheable. RCE is used to define the system main memory as cacheable memory. It is implied that memory outside the region is non-cacheable.

Weak Write Ordering (WWO). Setting WWO=1 enables weak write ordering for that address region. Enabling WWO allows the ST6x86 CPU to issue writes in its internal cache in an order different than their order in the code stream. External writes always occur in order (strong ordering). Therefore, this should only be enabled for memory regions that are NOT sensitive to this condition. WWO should not be enabled for memory mapped I/O. WWO only applies to memory regions that have been cached and designated as write-back. It also applies to previously cached addresses even if the cache has been disabled (CD=1). Enabling WWO removes the write-ordering restriction and improves performance due to reduced pipeline stalls.

Weak Locking (WL). Setting WL=1 enables weak locking for that address region. With WWO enabled, all bus cycles are issued with the LOCK# pin negated except for page table accesses and interrupt acknowledge cycles. Interrupt acknowledge cycles are executed as locked cycles even though LOCK# is negated. With WL=1, previously non-cacheable locked cycles are executed as unlocked cycles and therefore, may be cached, resulting in higher performance. The NO_LOCK bit of CCR1 enables weak locking for the entire address space. The WL bit allows weak locking only for specific address regions. WL is independent of the cacheability of the address region.

Write Gathering (WG). Setting WG=1 enables write gathering for the associated address region. Write gathering allows multiple byte, word, or dword sequential address writes to accumulate in the on-chip write buffer. (As instructions are executed, the results are placed in a series of output buffers. These buffers are gathered into the finial output buffer).

When access is made to a non-sequential memory location or when the 8-byte buffer becomes full, the contents of the buffer are written on the external 64-bit data bus. Performance is enhanced by avoiding as many as seven memory write cycles. WG should not be used on memory regions that are sensitive to write cycle gathering. WG can be enabled for both cacheable and non-cacheable regions.

Write Through (WT). Setting WT=1 defines the address region as write-through instead of write-back, assuming the region is cacheable. Regions where system ROM are loaded (shadowed or not) should be defined as write through.

LBA # Not Asserted (NLB). Setting NLB = 1 prevents the microprocessor from asserting the Local Bus Access (LBA#) output pin for accesses to that address region. The RCR regions may be used to define non-local bus address regions. The LBA# pin could then be asserted for all regions, except those defined by the RCRs. The LBA# signal may be used by the external hardware (e.g., chipsets) as an indication that local bus accesses are occurring.

3.0 ELECTRICAL SPECIFICATIONS

3.1 Electrical Connections

This section provides information on electrical connections, absolute maximum ratings, recommended operating conditions, DC characteristics, and AC characteristics. All voltage values in Electrical Specifications are measured with respect to V_{SS} unless otherwise noted.

3.1.1 Power and Ground Connections and Decoupling

Testing and operating the ST6x86 CPU requires the use of standard high frequency techniques to reduce parasitic effects. The high clock frequencies used in the ST6x86 CPU and its output buffer circuits can cause transient power surges when several output buffers switch output levels simultaneously. These effects can be minimized by filtering the DC power leads with low-inductance decoupling capacitors, using low impedance wiring, and by utilizing all of the V_{CC} and GND pins. The ST6x86 CPU contains 296 pins with 53 pins connected to V_{CC} and 53 connected to V_{SS} (ground).

3.1.2 Pull-Up/Pull-Down Resistors

Table 3.1 lists the input pins that are internally connected to pull-up and pull-down resistors. The pull-up resistors are connected to V_{CC} and the pull-down resistors are connected to V_{SS}. When unused, these inputs do not require connection to external pull-up or pull-down resistors. The SUSP# pin is unique in that it is connected to a pull-up resistor only when SUSP# is not asserted.

Table 3.1. Pins Connected to Internal Pull-Up and Pull Down Resistors

SIGNAL	PIN NO.	RESISTOR
BRDYC#	Y3	20-kΩ pull-up
CLKMUL	Y33	20-kΩ pull-down
QDUMP#	AL7	20-kΩ pull-up
SMI#	AB34	
SUSP#	Y34	20-kΩ pull-up (see text)
TCK	M34	20-kΩ pull-up
TDI	N35	
TMS	P34	
TRST#	Q33	
Reserved	J33	
Reserved	W35	
Reserved	Y35	
Reserved	AN35	20-kΩ pull-down

3.1.3 Unused Input Pins

All inputs not used by the system designer and not listed in Table 3.1 should be connected either to ground or to V_{CC}. Connect active-high inputs to ground through a 20 kΩ (± 10%) pull-down resistor and active-low inputs to V_{CC} through a 20 kΩ (± 10%) pull-up resistor to prevent possible spurious operation.

3.1.4 NC and Reserved Pins

Pins designated NC have no internal connections. Pins designated RESV or RESERVED should be left disconnected. Connecting a reserved pin to a pull-up resistor, pull-down resistor, or an active signal could cause unexpected results and possible circuit malfunctions.

3.2 Absolute Maximum Ratings

The following table lists absolute maximum ratings for the ST6x86 CPU microprocessors. Stresses beyond those listed under Table 3.2 limits may cause permanent damage to the device. These are stress ratings only and do not imply that operation under any conditions other than those listed under "Recommended Operating Conditions" ST6x86 is possible. Exposure to conditions beyond Table 3.2 may (1) reduce device reliability and (2) result in premature failure even when there is no immediately apparent sign of failure. Prolonged exposure to conditions at or near the absolute maximum ratings may also result in reduced useful life and reliability.

Table 3.2. Absolute Maximum Ratings

PARAMETER	MIN	MAX	UNITS	NOTES
Operating Case Temperature	-65	110	°C	Power Applied
Storage Temperature	-65	150	°C	
Supply Voltage, V_{CC}	-0.5	4.0	V	
Voltage On Any Pin	-0.5	V_{CC} +0.5	V	
Input Clamp Current, I_{IK}		10	mA	Power Applied
Output Clamp Current, I_{OK}		25	mA	Power Applied

3.3 Recommended Operating Conditions

Table 3.3 presents the recommended operating conditions for the ST6x86 CPU device.

Table 3.3. Recommended Operating Conditions

PARAMETER	MIN	MAX	UNITS	NOTES
T_C Operating Case Temperature	0	70	°C	Power Applied
V_{CC} Supply Voltage	3.15	3.7	V	
V_{IH} High-Level Input Voltage	2.0	5.5	V	
V_{IL} Low-Level Input Voltage	-0.3	0.8	V	
I_{OH} High-Level Output Current All outputs except A20-A3 and W/R# A20-A3 and W/R#		 -1.0 -2.0	mA	$V_O=V_{OH(MIN)}$
I_{OL} Low-Level Output Current All outputs except A20-A3 and W/R# A20-A3 and W/R#		 5.0 10.0	mA	$V_O=V_{OL(MAX\}}$

3.4 DC Characteristics

Table 3.4. DC Characteristics (at Recommended Operating Conditions)

PARAMETER	MIN	TYP	MAX	UNITS	NOTES
V_{OL} Output Low Voltage I_{OL} = 5 mA			0.4	V	
V_{OH} Output High Voltage I_{OH} = -1 mA	2.4			V	
I_I Input Leakage Current For all pins except those listed in Table 4-1.			±15	µA	$0 < V_{IN} < V_{CC}$
I_{IH} Input Leakage Current For all pins with internal pull-downs.			200	µA	V_{IH} = 2.4 V See Table 3-1.
I_{IL} Input Leakage Current For all pins with internal pull-ups.			-400	µA	V_{IL} = 0.45 V See Table 3-1.
I_{CC} Active I_{CC} 80 MHz 100 MHz 110 MHz 120 MHz 133 MHz		 3.9 4.5 4.8 5.1 5.5	 4.7 5.4 5.8 6.1 6.6	A	Note 1, 5
I_{CCSM} Suspend Mode I_{CC} 80MHz 100 MHz 110 MHz 120 MHz 133 MHz		 43 48 50 51 54	 75 80 83 105 115	mA	Note 1, 3, 5
I_{CCSS} Standby I_{CC} 0 MHz (Suspended/CLK Stopped)		35	75	mA	Note 4,5
C_{IN} Input Capacitance			15	pF	f = 1 MHz, Note 2
C_{OUT} Output Capacitance			20	pF	f = 1 MHz, Note 2
C_{IO} I/O Capacitance			25	pF	f = 1 MHz, Note 2
C_{CLK} CLK Capacitance			15	pF	f = 1 MHz, Note 2

Notes:
1. Frequency (MHz) ratings refer to the internal clock frequency.
2. Not 100% tested.
3. All inputs at 0.4 or V_{CC} - 0.4 (CMOS levels). All inputs held static except clock and all outputs unloaded (static I_{OUT} = 0 mA).
4. All inputs at 0.4 or V_{CC} - 0.4 (CMOS levels). All inputs held static and all outputs unloaded (static I_{OUT} = 0 mA).
5. Typical, measured at V_{CC} = 3.3 V

3.5 AC Characteristics

Table 3.6 through 3.11 (Pages 38 through 43) list the AC characteristics including output delays, input setup requirements, input hold requirements and output float delays. These measurements are based on the measurement points identified in Figure 3.1 (Page 37) and Figure 3.2 (Page 38). The rising clock edge reference level V_{REF} and other reference levels are shown in Table 3.5. Input or output signals must cross these levels during testing.

Figure 3.1 shows output delay (A and B) and input setup and hold times (C and D). Input setup and hold times (C and D) are specified minimums, defining the smallest acceptable sampling window a synchronous input signal must be stable for correct operation.

Figure 3.1. Drive Level and Measurement Points for Switching Characteristics

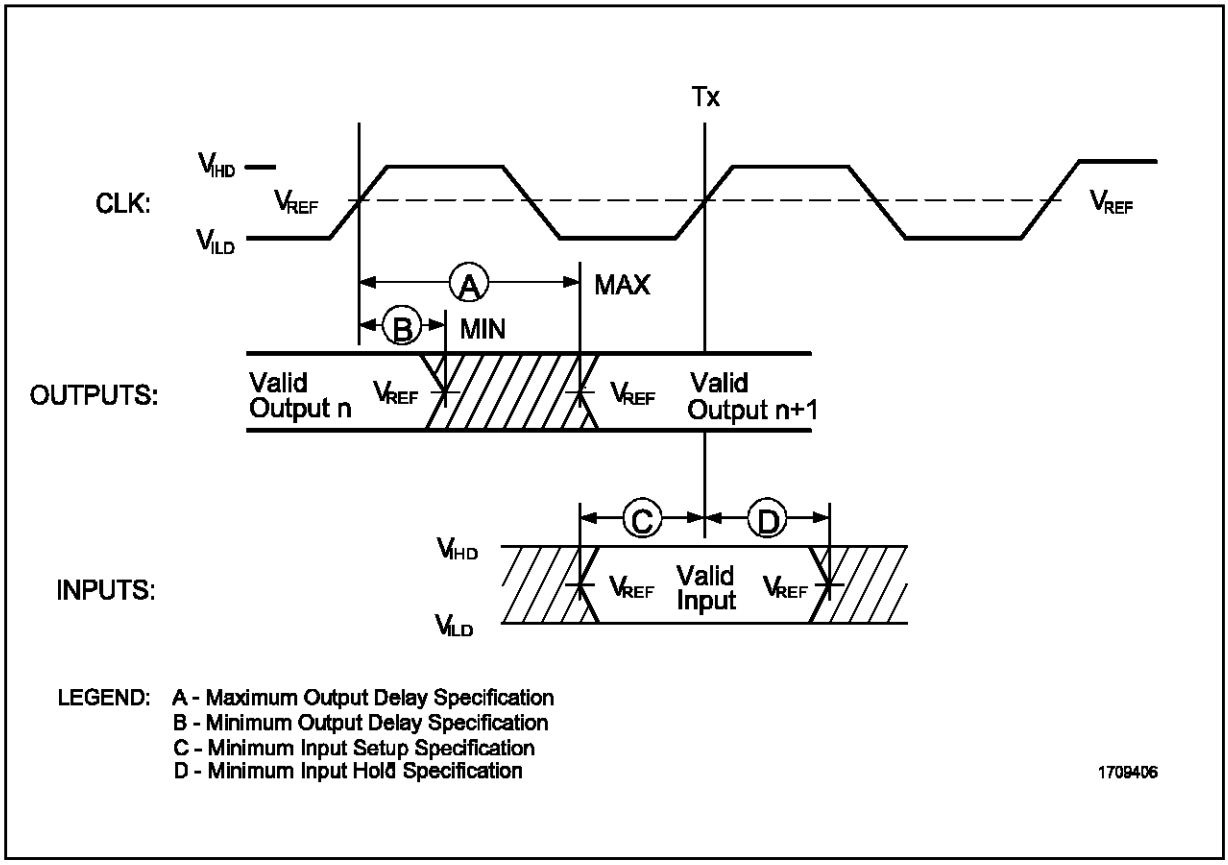


Table 3.5. Drive Level and Measurement Points for Switching Characteristics

SYMBOL	VOLTAGE (Volts)
V_{REF}	1.5
V_{IHD}	2.3
V_{ILD}	0

Note: Refer to Figure 3-1.

Table 3.6. Clock Specifications
T_{case} = 0°C to 70°C, See Figure 3.2

SYMBOL	PARAMETER	40-MHz BUS		50-MHz BUS		55-MHz BUS		60-MHz BUS		66-MHz BUS		UNITS
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
	CLK Frequency		40		50		55		60		66.6	MHz
T1	CLK Period	25		20		18		16.67	33.33	15.0	30.0	ns
T2	CLK Period Stability		± 250		± 250		± 250		± 250		± 250	ps
T3	CLK High Time	9		7		4.0		4.0		4.0		ns
T4	CLK Low Time	9		7		4.0		4.0		4.0		ns
T5	CLK Fall Time	0.15	2	0.15	2	0.15	1.5	0.15	1.5	0.15	1.5	ns
T6	CLK Rise Time	0.15	2	0.15	2	0.15	1.5	0.15	1.5	0.15	1.5	ns

Figure 3.2. CLK Timing and Measurement Points

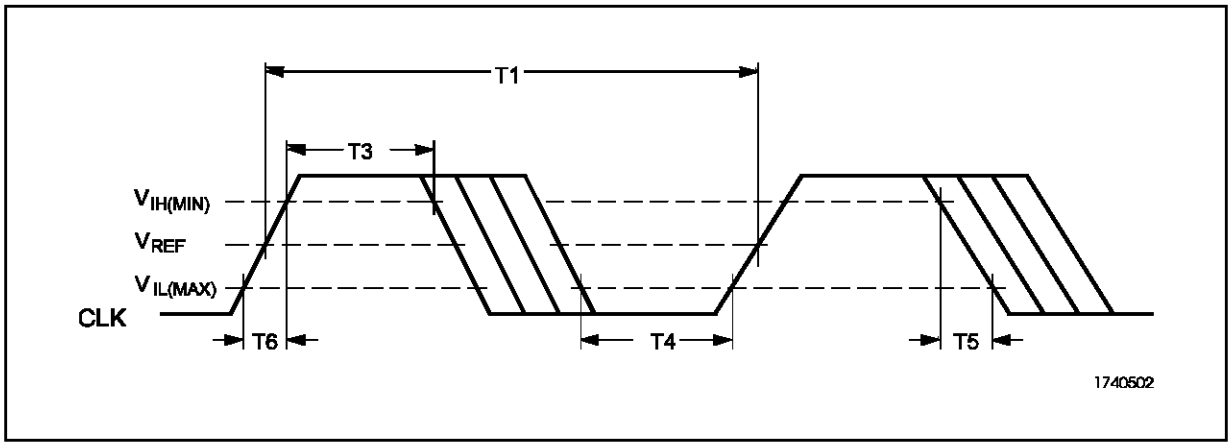


Table 3.7. Output Valid Delays
C_L=50 pF, T_{case} = 0°C to 70°C, See Figure 3.3

SYMBOL	PARAMETER	40-MHz BUS		50-MHz BUS		55-MHz BUS		60-MHz BUS		66-MHz BUS		UNITS
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
T7	A31-A3, BE7#-BEO# CACHE#, D/C#, LBA#, LOCK#, PCD, PWT, SCYC, SMIACT#, W/R#	3	14	1	12	1	7	1	8	1	7	ns
T7b	ADS#, M/IO#	3	14	1	12	1	7.5	1	7.5	1	6	ns
T8	ADSC#	3	14	1	12	1	7	1	8	1	7	ns
T9	AP	3	14	1	12	1	8.5	1	8.5	1	8.5	ns
T10	APCHK#, PCHK#, FERR#	3	16	1	14	1	8.3	1	7	1	7	ns
T11	D63-DO, DP7-DPO (Write)	3	14	1.3	12	1.3	9	1.3	9	1.3	7.5	ns
T12a	HIT#	3	14	1	12	1	8	1	8	1	8	ns
T12b	HITM#	3	14	1.1	12	1.1	7	1.1	7	1.1	6	ns
T13	BREQ, HLDA	3	14	1	12	1	8	1	8	1	8	ns
T14	SUSPA#	3	16	1	14	1	8	1	8	1	8	ns

Figure 3.3. Output Valid Delay Timing

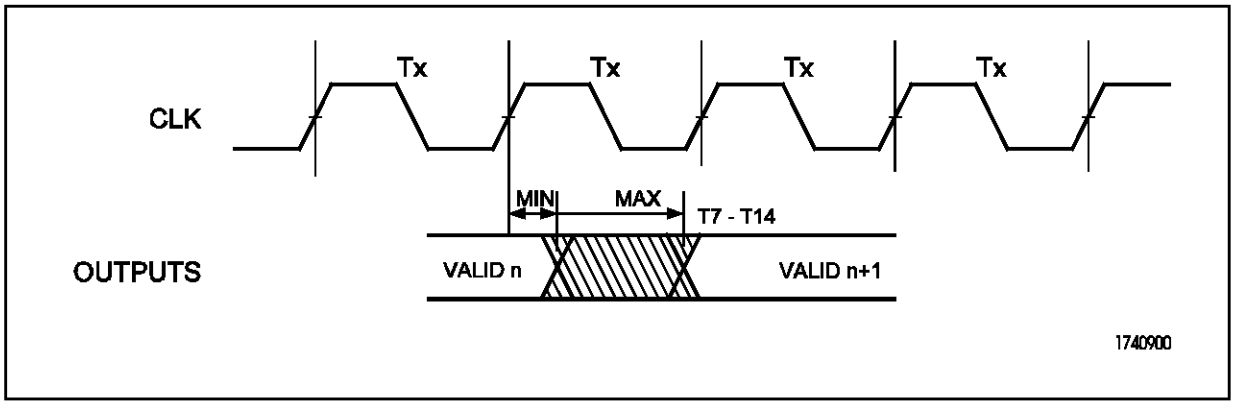


Table 3.8. Output Float Delays
C_L=50 pF_{case} = 0°C to 70°C, See Figure 3.4

SYMBOL	PARAMETER	40-MHz BUS		50-MHz BUS		55-MHz BUS		60-MHz BUS		66-MHz BUS		UNITS
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
T15	A31-A3, ADS#, BE7#-BE0#, BREQ, CACHE#, D/C#, LBA#, LOCK#, M/IO#, PCD, PWT, SCYC, SMIACT#, W/R#		19		16		10		10		10	ns
T16	AP		19		16		10		10		10	ns
T17	D63-D0, DP7-DP0 (Write)		19		16		10		10		10	ns

Figure 3.4. Output Float Delay Timing

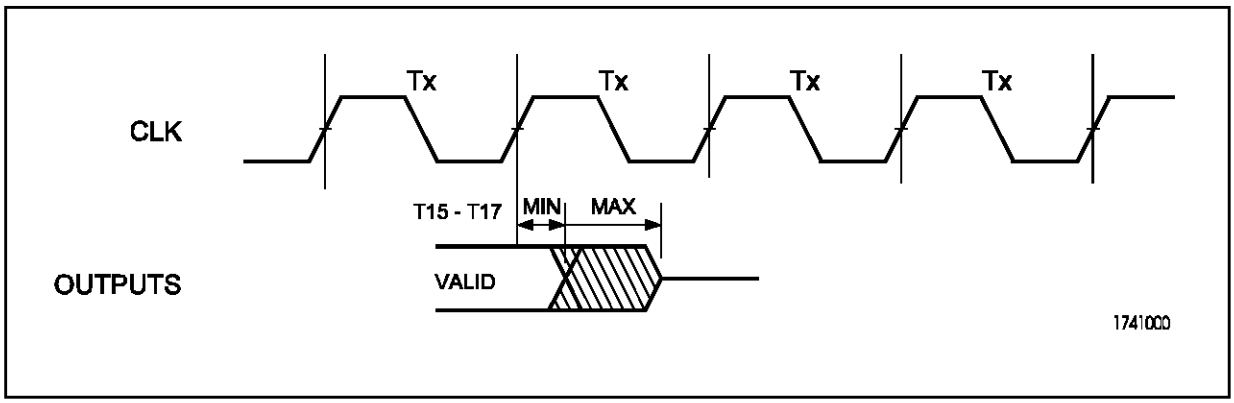


Table 3.9. Input Setup Times
T_{case} = 0 °C to 70 °C, See Figure 3.5

SYMBOL	PARAMETER	40-MHz BUS MIN	50-MHz BUS MIN	55-MHz BUS MIN	60-MHz BUS MIN	66-MHz BUS MIN	UNITS
T18	A20M#, FLUSH#, IGNNE#, SUSP#	5	5	5	5	5	ns
T19	AHOLD, BHOLD, BOFF#, DHOLD, HOLD	5	5	5	5	5	ns
T20	BRDY#	5	5	5	5	5	ns
T21	BRDYC#	5	5	5	5	5	ns
T22	A31-A3, AP, BE7#-BE0#,	5	5	5	5	5	ns
T22a	D63-D0 (Read), DP7-DP0 (Read)	3.8	3.8	3.8	3	3	ns
T23	EADS#, INV	5	5	5	5	5	ns
T24	INTR, NMI, RESET, SMI#, WM_RST	5	5	5	5	5	ns
T25	EWBE#, KEN#, NA#, WB/WT#	5	5	4.5	4.5	4.5	ns
T26	QDUMP#	5	5	5	5	5	ns

Table 3.10. Input Hold Times
T_{case} = 0 °Cto 70 °C, See Figure 3.5

SYMBOL	PARAMETER	40-MHz BUS MIN	50-MHz BUS MIN	55-MHz BUS MIN	60-MHz BUS MIN	66-MHz BUS MIN	UNITS
T27	A20M#, FLUSH#, IGNNE#, SUSP#	3	2	1	1	1	ns
T28	AHOLD, BHOLD, BOFF#, DHOLD, HOLD	3	2	1	1	1	ns
T29	BRDY#	3	2	1	1	1	ns
T30	BRDYC#	3	2	1	1	1	ns
T31a	A31-A3, AP, BE7#-BE0#	3	2	1	1	1	ns
T31b	D63-D0, DP7-DP0 (Read)	3	2	2	2	2	ns
T32	EADS#, INV	3	2	1	1	1	ns
T33	INTR, NMI, RESET, SMI#, WM_RST	3	2	1	1	1	ns
T34	EWBE#, KEN#, NA#, WB/WT#	3	2	1	1	1	ns
T35	QDUMP#	3	2	1	1	1	ns

Figure 3.5. Input Setup and Hold Timing

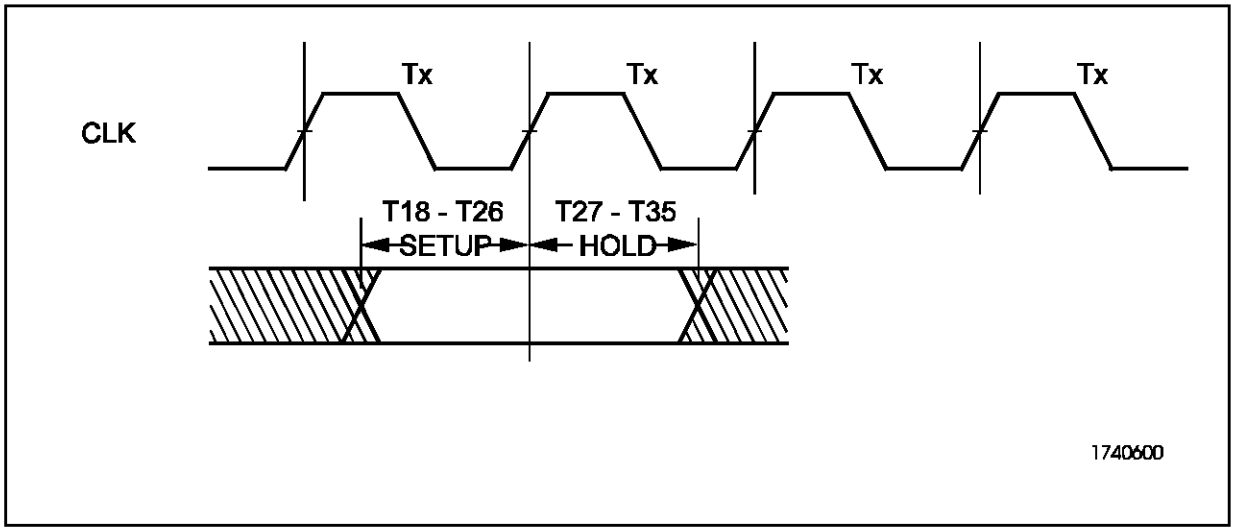


Table 3.11. JTAG AC Specifications

SYMBOL	PARAMETER	ALL BUS FREQUENCIES		UNITS	FIGURE
		MIN	MAX		
	TCK Frequency (MHz)		20	ns	
T36	TCK Period	50		ns	3-6
T37	TCK High Time	25		ns	3-6
T38	TCK Low Time	25		ns	3-6
T39	TCK Rise Time		5	ns	3-6
T40	TCK Fall Time		5	ns	3-6
T41	TDO Valid Delay	3	20	ns	3-7
T42	Non-test Outputs Valid Delay	3	20	ns	3-7
T43	TDO Float Delay		25	ns	3-7
T44	Non-test Outputs Float Delay		25	ns	3-7
T45	TRST# Pulse Width	40		ns	3-8
T46	TDI, TMS Setup Time	20		ns	3-7
T47	Non-test Inputs Setup Time	20		ns	3-7
T48	TDI, TMS Hold Time	13		ns	3-7
T49	Non-test Inputs Hold Time	13		ns	3-7

Figure 3.6. TCK Timing and Measurement Points

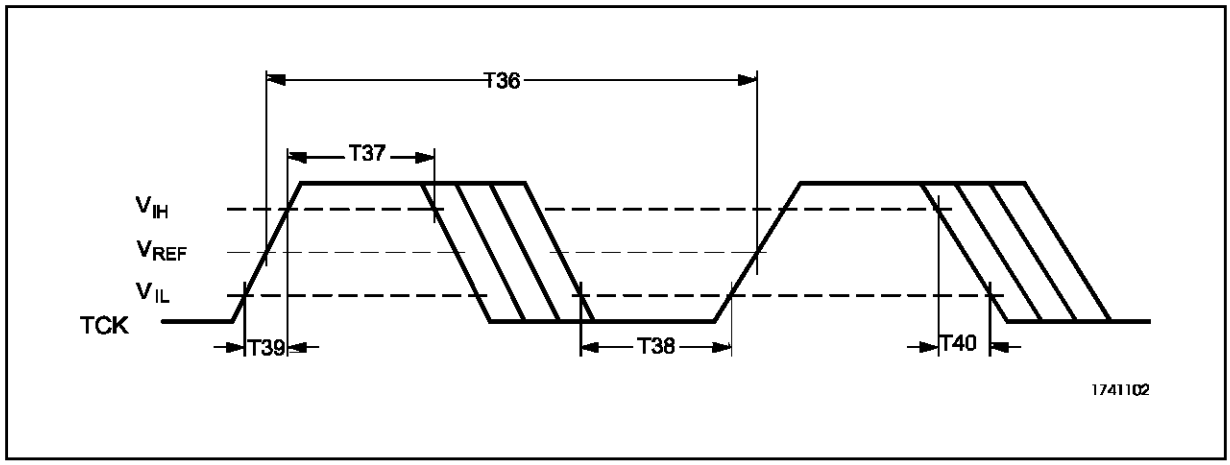


Figure 3.7. JTAG Test Timings

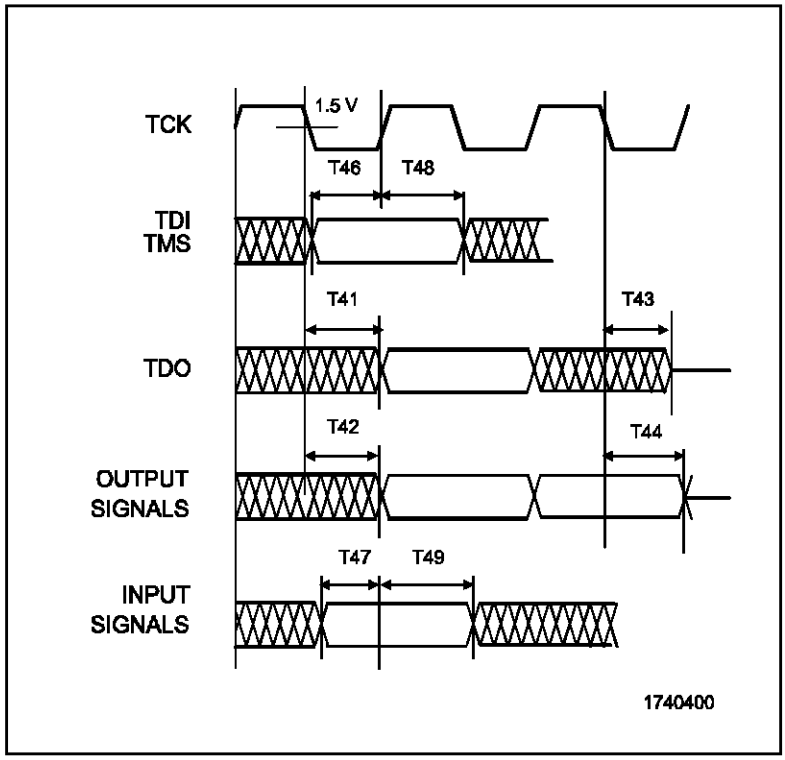
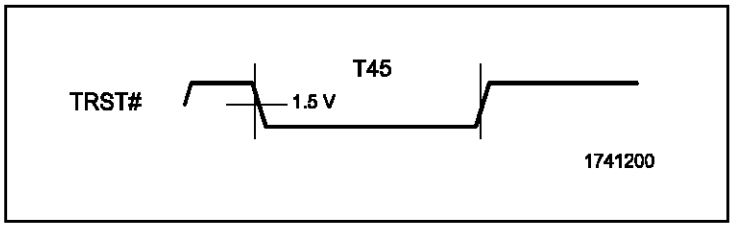


Figure 3.8. Test Reset Timing



4.0 MECHANICAL SPECIFICATIONS

4.1 296-Pin CPGA Package

The pin assignments for the ST6x86 CPUin a 296-pin CPGA package are shown in Figure 4.1. The pins are listed by signal name in Table 4.1 (Page 46) and by pin number in Table 4.2 (Page 47). Dimensions are shown in Table 4.2 (Page 47) and Table 4.3 (Page 49).

Figure 4.1. 296-Pin CPGA Package Pin Assignments

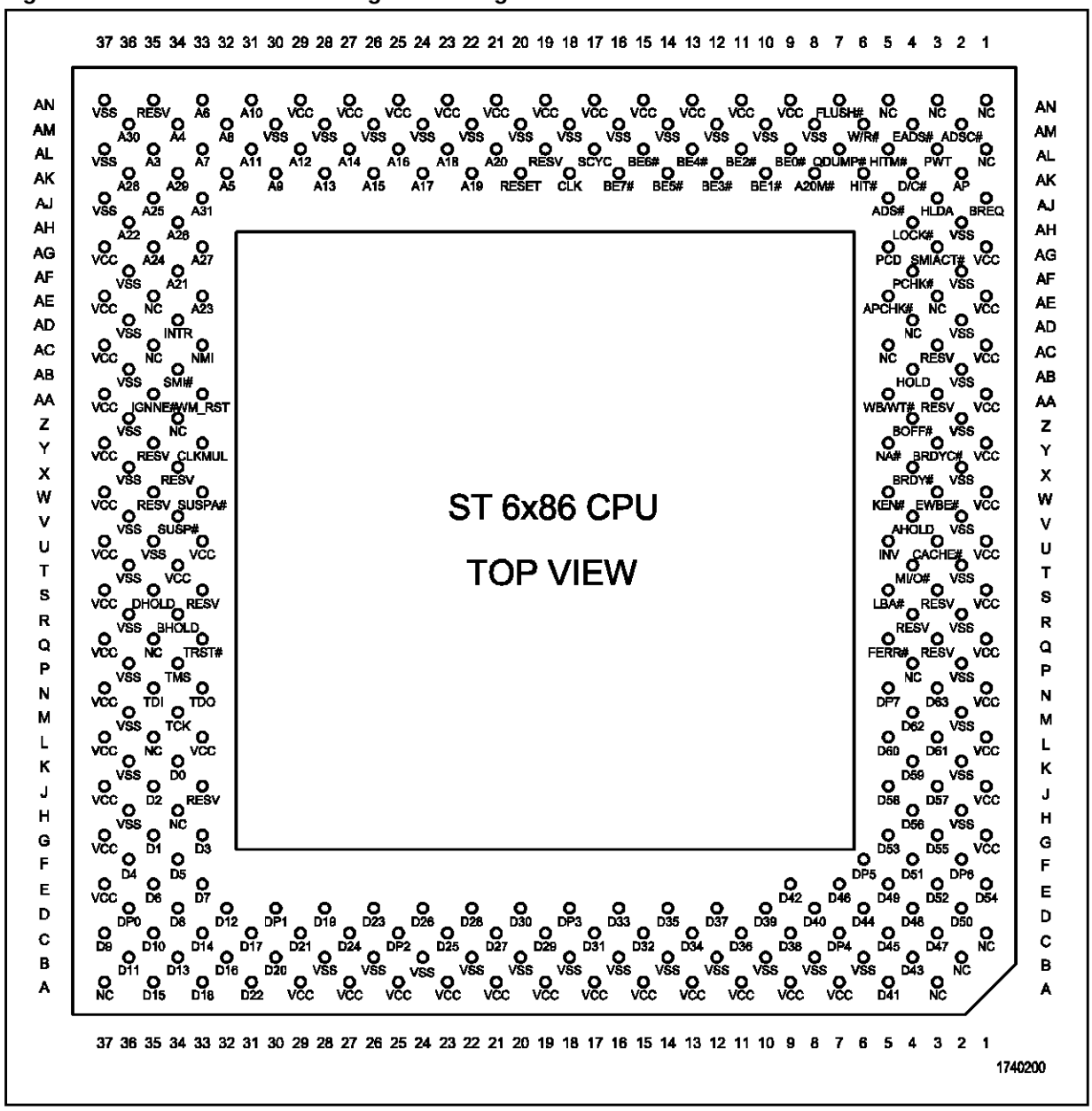


Table 4.1. 296-Pin CPGA Package Signal Names Sorted by Pin Number

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
A3	NC	C29	D21	J35	D2	U35	Vss	AE35	NC	AL21	A20
A5	D41	C31	D17	J37	Vcc	U37	Vcc	AE37	Vcc	AL23	A18
A7	Vcc	C33	D14	K2	Vss	V2	Vss	AF2	Vss	AL25	A16
A9	Vcc	C35	D10	K4	D59	V4	AHOLD	AF4	PCHK#	AL27	A14
A11	Vcc	C37	D9	K34	D0	V34	SUSP#	AF34	A21	AL29	A12
A13	Vcc	D2	D50	K36	Vss	V36	Vss	AF36	Vss	AL31	A11
A15	Vcc	D4	D48	L1	Vcc	W1	Vcc	AG1	Vcc	AL33	A7
A17	Vcc	D6	D44	L3	D61	W3	EWBE#	AG3	SMIACT#	AL35	A3
A19	Vcc	D8	D40	L5	D60	W5	KEN#	AG5	PCD	AL37	Vss
A21	Vcc	D10	D39	L33	Vcc	W33	SUSPA#	AG33	A27	AM2	ADSC#
A23	Vcc	D12	D37	L35	NC	W35	Reserved	AG35	A24	AM4	EADS#
A25	Vcc	D14	D35	L37	Vcc	W37	Vcc	AG37	Vcc	AM6	W/R#
A27	Vcc	D16	D33	M2	Vss	X2	Vss	AH2	Vss	AM8	Vss
A29	Vcc	D18	DP3	M4	D62	X4	BRDY#	AH4	LOCK#	AM10	Vss
A31	D22	D20	D30	M34	TCK	X34	Reserved	AH34	A26	AM12	Vss
A33	D18	D22	D28	M36	Vss	X36	Vss	AH36	A22	AM14	Vss
A35	D15	D24	D26	N1	Vcc	Y1	Vcc	AJ1	BREQ	AM16	Vss
A37	NC	D26	D23	N3	D63	Y3	BRDYC#	AJ3	HLDA	AM18	Vss
B2	NC	D28	D19	N5	DP7	Y5	NA#	AJ5	ADS#	AM20	Vss
B4	D43	D30	DP1	N33	TDO	Y33	CLKMUL	AJ33	A31	AM22	Vss
B6	Vss	D32	D12	N35	TDI	Y35	Reserved	AJ35	A25	AM24	Vss
B8	Vss	D34	D8	N37	Vcc	Y37	Vcc	AJ37	Vss	AM26	Vss
B10	Vss	D36	DP0	P2	Vss	Z2	Vss	AK2	AP	AM28	Vss
B12	Vss	E1	D54	P4	NC	Z4	BOFF#	AK4	D/C#	AM30	Vss
B14	Vss	E3	D52	P34	TMS	Z34	NC	AK6	HIT#	AM32	A8
B16	Vss	E5	D49	P36	Vss	Z36	Vss	AK8	A20M#	AM34	A4
B18	Vss	E7	D46	Q1	Vcc	AA1	Vcc	AK10	BE1#	AM36	A30
B20	Vss	E9	D42	Q3	Reserved	AA3	Reserved	AK12	BE3#	AN1	NC
B22	Vss	E33	D7	Q5	FERR#	AA5	WB/WT#	AK14	BE5#	AN3	NC
B24	Vss	E35	D6	Q33	TRST#	AA33	WM_RST	AK16	BE7#	AN5	NC
B26	Vss	E37	Vcc	Q35	NC	AA35	IGNNE#	AK18	CLK	AN7	FLUSH#
B28	Vss	F2	DP6	Q37	Vcc	AA37	Vcc	AK20	RESET	AN9	Vcc
B30	D20	F4	D51	R2	Vss	AB2	Vss	AK22	A19	AN11	Vcc
B32	D16	F6	DP5	R4	Reserved	AB4	HOLD	AK24	A17	AN13	Vcc
B34	D13	F34	D5	R34	BHOLD	AB34	SMI#	AK26	A15	AN15	Vcc
B36	D11	F36	D4	R36	Vss	AB36	Vss	AK28	A13	AN17	Vcc
C1	NC	G1	Vcc	S1	Vcc	AC1	Vcc	AK30	A9	AN19	Vcc
C3	D47	G3	D55	S3	Reserved	AC3	Reserved	AK32	A5	AN21	Vcc
C5	D45	G5	D53	S5	LBA#	AC5	NC	AK34	A29	AN23	Vcc
C7	DP4	G33	D3	S33	Reserved	AC33	NMI	AK36	A28	AN25	Vcc
C9	D38	G35	D1	S35	DHOLD	AC35	NC	AL1	NC	AN27	Vcc
C11	D36	G37	Vcc	S37	Vcc	AC37	Vcc	AL3	PWT	AN29	Vcc
C13	D34	H2	Vss	T2	Vss	AD2	Vss	AL5	HITM#	AN31	A10
C15	D32	H4	D56	T4	MI/O#	AD4	NC	AL7	QDUMP#	AN33	A6
C17	D31	H34	NC	T34	Vcc	AD34	INTR	AL9	BE0#	AN35	Reserved
C19	D29	H36	Vss	T36	Vss	AD36	Vss	AL11	BE2#	AN37	Vss
C21	D27	J1	Vcc	U1	Vcc	AE1	Vcc	AL13	BE4#		
C23	D25	J3	D57	U3	CACHE#	AE3	NC	AL15	BE6#		
C25	DP2	J5	D58	U5	INV	AE5	APCHK#	AL17	SCYC		
C27	D24	J33	Reserved	U33	Vcc	AE33	A23	AL19	Reserved		

Note: *Reserved* pins are reserved for future use by SGS-THOMSON only. Pins marked NC are not internally connected.

Table 4.2. 296-Pin CPGA Package Pin Numbers Sorted by Signal Name

Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin
A3	AL35	CLKMUL	Y33	D48	D4	NC	AN3	Vcc	AA37	Vss	AM12
A4	AM34	D/C#	AK4	D49	E5	NC	AN5	Vcc	AC1	Vss	AM14
A5	AK32	D0	K34	D50	D2	NC	B2	Vcc	AC37	Vss	AM16
A6	AN33	D1	G35	D51	F4	NC	C1	Vcc	AE1	Vss	AM18
A7	AL33	D2	J35	D52	E3	NC	H34	Vcc	AE37	Vss	AM20
A8	AM32	D3	G33	D53	G5	NC	L35	Vcc	AG1	Vss	AM22
A9	AK30	D4	F36	D54	E1	NC	P4	Vcc	AG37	Vss	AM24
A10	AN31	D5	F34	D55	G3	NC	Q35	Vcc	AN11	Vss	AM26
A11	AL31	D6	E35	D56	H4	NC	Z34	Vcc	AN13	Vss	AM28
A12	AL29	D7	E33	D57	J3	NMI	AC33	Vcc	AN15	Vss	AM30
A13	AK28	D8	D34	D58	J5	PCD	AG5	Vcc	AN17	Vss	AM8
A14	AL27	D9	C37	D59	K4	PCHK#	AF4	Vcc	AN19	Vss	AN37
A15	AK26	D10	C35	D60	L5	PWT	AL3	Vcc	AN21	Vss	B6
A16	AL25	D11	B36	D61	L3	QDUMP#	AL7	Vcc	AN23	Vss	B8
A17	AK24	D12	D32	D62	M4	RESET	AK20	Vcc	AN25	Vss	B10
A18	AL23	D13	B34	D63	N3	SCYC	AL17	Vcc	AN27	Vss	B12
A19	AK22	D14	C33	DHOLD	S35	Reserved	AA3	Vcc	AN29	Vss	B14
A20	AL21	D15	A35	DP0	D36	Reserved	AC3	Vcc	AN9	Vss	B16
A20M#	AK8	D16	B32	DP1	D30	Reserved	AL19	Vcc	E37	Vss	B18
A21	AF34	D17	C31	DP2	C25	Reserved	AN35	Vcc	G1	Vss	B20
A22	AH36	D18	A33	DP3	D18	Reserved	J33	Vcc	G37	Vss	B22
A23	AE33	D19	D28	DP4	C7	Reserved	Q3	Vcc	J1	Vss	B24
A24	AG35	D20	B30	DP5	F6	Reserved	R4	Vcc	J37	Vss	B26
A25	AJ35	D21	C29	DP6	F2	Reserved	S3	Vcc	L1	Vss	B28
A26	AH34	D22	A31	DP7	N5	Reserved	S33	Vcc	L33	Vss	H2
A27	AG33	D23	D26	EADS#	AM4	Reserved	W35	Vcc	L37	Vss	H36
A28	AK36	D24	C27	EWBE#	W3	Reserved	X34	Vcc	N1	Vss	K2
A29	AK34	D25	C23	FERR#	Q5	Reserved	Y35	Vcc	N37	Vss	K36
A30	AM36	D26	D24	FLUSH#	AN7	SMI#	AB34	Vcc	Q1	Vss	M2
A31	AJ33	D27	C21	HIT#	AK6	SMIACT#	AG3	Vcc	Q37	Vss	M36
ADS#	AJ5	D28	D22	HITM#	AL5	SUSP#	V34	Vcc	S1	Vss	P2
ADSC#	AM2	D29	C19	HLDA	AJ3	SUSPA#	W33	Vcc	S37	Vss	P36
AHOLD	V4	D30	D20	HOLD	AB4	TCK	M34	Vcc	T34	Vss	R2
AP	AK2	D31	C17	IGNNE#	AA35	TDI	N35	Vcc	U1	Vss	R36
APCHK#	AE5	D32	C15	INTR	AD34	TDO	N33	Vcc	U33	Vss	T2
BE0#	AL9	D33	D16	INV	U5	TMS	P34	Vcc	U37	Vss	T36
BE1#	AK10	D34	C13	KEN#	W5	TRST#	Q33	Vcc	W1	Vss	U35
BE2#	AL11	D35	D14	LBA#	S5	Vcc	A7	Vcc	W37	Vss	V2
BE3#	AK12	D36	C11	LOCK#	AH4	Vcc	A9	Vcc	Y1	Vss	V36
BE4#	AL13	D37	D12	MI/O#	T4	Vcc	A11	Vcc	Y37	Vss	X2
BE5#	AK14	D38	C9	NA#	Y5	Vcc	A13	Vss	AB2	Vss	X36
BE6#	AL15	D39	D10	NC	A3	Vcc	A15	Vss	AB36	Vss	Z2
BE7#	AK16	D40	D8	NC	A37	Vcc	A17	Vss	AD2	Vss	Z36
BHOLD	R34	D41	A5	NC	AC35	Vcc	A19	Vss	AD36	WB/WT#	AA5
BOFF#	Z4	D42	E9	NC	AC5	Vcc	A21	Vss	AF2	W/R#	AM6
BRDY#	X4	D43	B4	NC	AD4	Vcc	A23	Vss	AF36	WM_RST	AA33
BRDYC#	Y3	D44	D6	NC	AE3	Vcc	A25	Vss	AH2		
BREQ	AJ1	D45	C5	NC	AE35	Vcc	A27	Vss	AJ37		
CACHE#	U3	D46	E7	NC	AL1	Vcc	A29	Vss	AL37		
CLK	AK18	D47	C3	NC	AN1	Vcc	AA1	Vss	AM10		

Note: *Reserved* pins are reserved for future use by SGS-THOMSON only. Pins marked NC are not internally connected.

Figure 4.2. 296-Pin CPGA Package

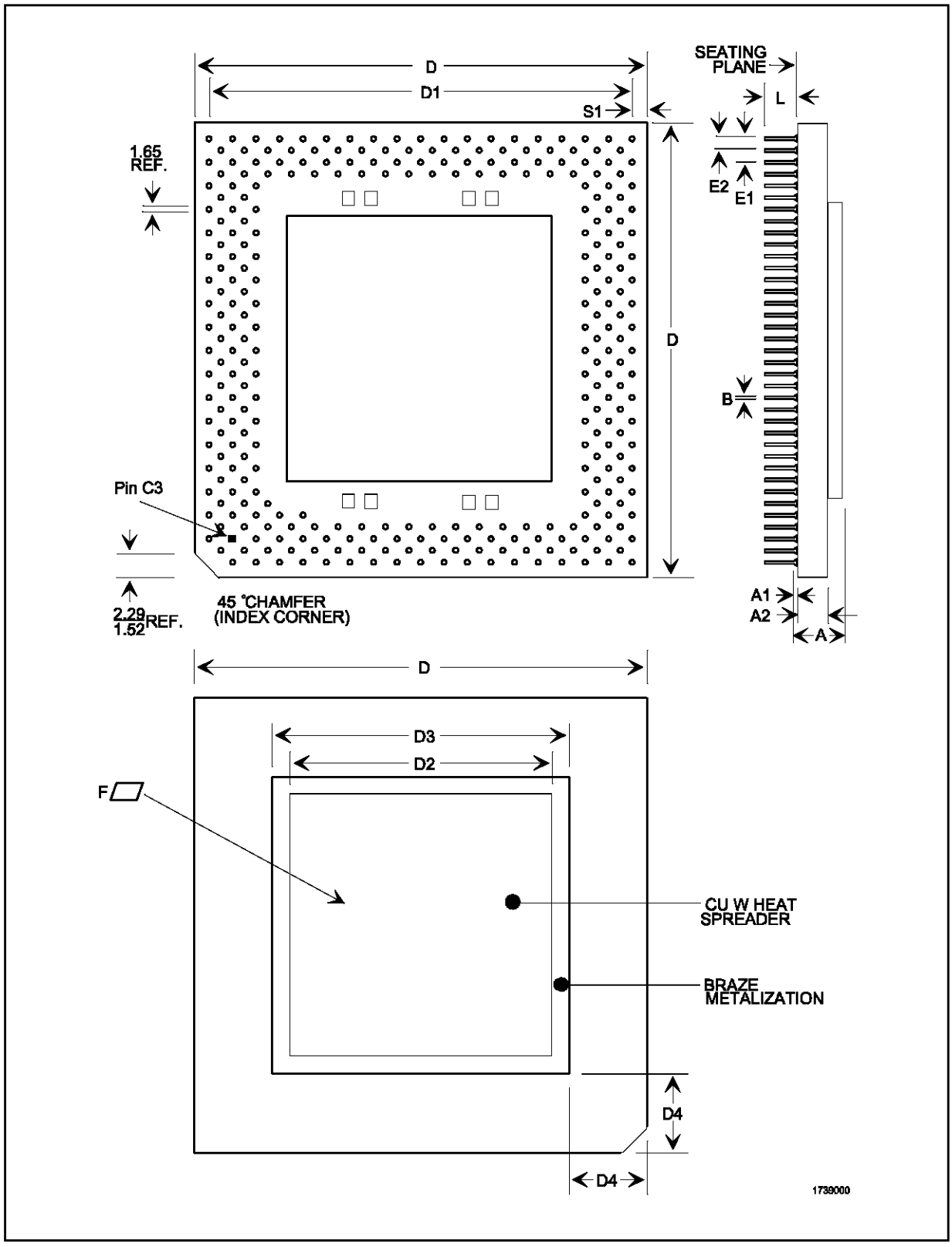


Table 4.3. 296-Pin CPGA Package Dimensions

SYMBOL	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	3.91	4.70	0.154	0.185
A1	0.33	0.43	0.013	0.017
A2	2.51	3.07	0.099	0.121
B	0.43	0.51	0.017	0.020
D	49.28	49.91	1.940	1.965
D1	45.47	45.97	1.790	1.810
D2	31.50 Sq.	32.00 Sq.	1.240 Sq.	1.260 Sq.
D3	33.99	34.59	1.338	1.362
D4	8.00	9.91	0.315	0.390
E1	2.41	2.67	0.095	0.105
E2	1.14	1.40	0.045	0.055
F	-	0.127 Diag.	-	0.005 Diag.
L	3.05	3.30	0.120	0.130
N	296 (Pin Count)			
S1	1.65	2.16	0.065	0.085

4.2 Thermal Characteristics

The ST6x86 processor is designed to operate when the case temperature at the top center of the package is between 0°C and 70°C. The maximum die (junction) temperature, $T_{J\,MAX}$, and the maximum ambient temperature, $T_{A\,MAX}$, can be calculated by substituting thermal resistance and maximum values for case or junction temperature and power dissipation in the following equations:

$$T_J = T_C + (P * \theta_{JC})$$
$$T_A = T_J - (P * \theta_{JA})$$

where:

T_A = Ambient temperature (°C)
T_J = Average junction temperature (°C)
T_C = Case temperature at top center of package (°C)
P = Power dissipation (W)
θ_{JC} = Junction-to-case thermal resistance (°C/W)
θ_{JA} = Junction-to-ambient thermal resistance (°C/W).

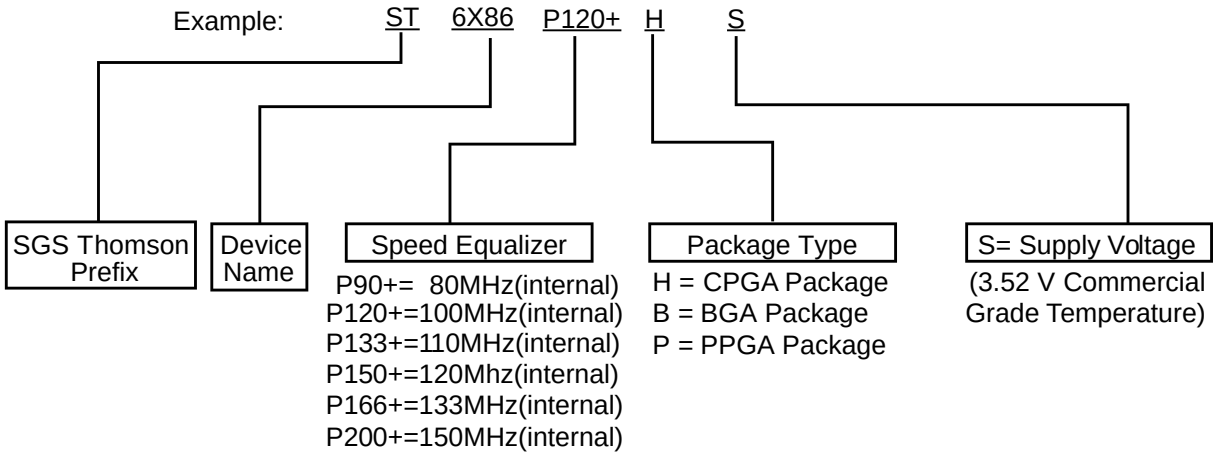
Table 4.4 lists the junction-to-case and case-to-ambient thermal resistances for the SPGA package.

Table 4.4. Thermal Resistances for CPGA Package With and Without Heatsinks

Thermal Resistance	θ_{JC} °C/W	θ_{CA} °C/W					
Laminar Air Flow (ft/min)	**0**	**0**	**100**	**200**	**400**	**600**	**800**
1.95 x 1.95 x 0.25 Heatsink	0.9	8.4	7.4	6.0	4.0	3.1	2.6
1.95 x 1.95 x 0.40 Heatsink	0.9	7.7	6.6	4.9	3.2	2.7	2.1
1.95 x 1.95 x 0.65 Heatsink	0.9	5.9	4.7	3.2	2.1	1.7	1.4
Without Heatsink	1.4	14.7	11.5	9.1	7.3	7.0	6.2

Notes:
For a ST6x86 processor with 1.25 x 1.25 x 0.40 inch CuW heat spreader.
Heatsinks are omni-directional pin aluminum alloy.
Features are based on standard extrusion practices for a given height.
Heatsink attachment was made with 0.006 inch of thermal grease applied between heatsink and case.
Maximum air temperature is assumed to be 40 °C

Ordering Information



Please contact your nearest SGS-THOMSON sales office to confirm availability of specific valid combinations and to check on newly released combinations.

The ST6x86 CPU part numbers are listed below.

ST6x86 Part Numbers

PART NUMBER	NOM Vcc (V)	FREQUENCY (MHz)	
		BUS	INTERNAL
ST6X86P90+HS	3.52	40	80
ST6X86P120+HS	3.52	50	100
ST6X86P133+HS	3.52	55	110
ST6X86P150+HS	3.52	60	120
ST6X86P166+HS	3.52	66	133