DSTni-EX User Guide

Section Two

Part Number 900-335
Revision A 3/04

Copyright & Trademark

Lantronix
15353 Barranca Parkway
Irvine, CA 92618, USA
Phone: 949-453-3990
Fax: 949-453-3995

Technical Support
Phone: 630-245-1445
Fax: 630-245-1717

Master Distributor
Grid Connect
1841 Centre Point Circle, Suite 143
Naperville, IL 60563
Phone: 630-245-1445
www.gridconnect.com

Am186 is a trademark of Advanced Micro Devices, Inc.
Ethernet is a registered trademark of Xerox Corporation.
SPI is a trademark of Motorola, Inc.

REV	Changes	Released Date
A	Reformat. Add changes from Design Spec. 1.1	3-24-04

Contents

List of Figures

List of Tables

About This User Guide

This User Guide describes the technical features and programming interfaces of the Lantronix DSTni-EX chip (hereafter referred to as “DSTni”).

DSTni is an Application Specific Integrated Circuit (ASIC)-based single-chip solution (SCS) that integrates the leading-edge functionalities needed to develop low-cost, high-performance device server products. On a single chip, the DSTni integrates an x186 microprocessor, 16K-byte ROM, 256K-byte SRAM, programmable input/output (I/O), and serial, Ethernet, and Universal Serial Bus (USB) connectivity — key ingredients for device- server solutions. Although DSTni embeds multiple functions onto a single chip, it can be easily customized, based on the comprehensive feature set designed into the chip.

Providing a complete device server solution on a single chip enables system designers to build affordable, full-function solutions that provide the highest level of performance in both processing power and peripheral systems, while reducing the number of total system components. The advantages gained from this synergy include:

- Simplifying system design and increased reliability.
- Minimizing marketing and administration costs by eliminating the need to source products from multiple vendors.
- Eliminating the compatibility and reliability problems that occur when combining separate subsystems.
- Dramatically reducing implementation costs.
- Increasing performance and functionality, while maintaining quality and cost effectiveness.
- Streamlining development by reducing programming effort and debugging time.
- Enabling solution providers to bring their products to market faster.

These advantages make DSTni the ideal solution for designs requiring x86 compatibility; increased performance; serial, programmable I/O, Ethernet, and USB communications; and a glueless bus interface.

Intended Audience

This User Guide is intended for use by hardware and software engineers, programmers, and designers who understand the basic operating principles of microprocessors and their systems and are considering designing systems that utilize DSTni.

Conventions

This User Guide uses the following conventions to alert you to information of special interest.

The symbols # and n are used throughout this Guide to denote active LOW signals.

Notes: *Notes are information requiring attention.*

Navigating Online

The electronic Portable Document Format (PDF) version of this User Guide contains *hyperlinks*. Clicking one of these hyper links moves you to that location in this User Guide. The PDF file was created with Bookmarks and active links for the Table of Contents, Tables, Figures and cross-references.

Organization

This User Guide contains information essential for system architects and design engineers. The information in this User Guide is organized into the following chapters and appendixes.

- *Section 1: Introduction*
  Describes the DSTni architecture, design benefits, theory of operations, ball assignments, packaging, and electrical specifications. This chapter includes a DSTni block diagram.
- *Section 2: Microprocessor*
  Describes the DSTni microprocessor and its control registers.
- *Section 2: SDRAM*
  Describes the DSTni SDRAM and the registers associated with it.
- *Section 3: Serial Ports*
  Describes the DSTni serial ports and the registers associated with them.
- *Section 3: Programmable Input/Output*
  Describes DSTni's Programmable Input/ Output (PIO) functions and the registers associated with them.
- *Section 3: Timers*
  Describes the DSTni timers.
- *Section 4: Ethernet Controllers*
  Describes the DSTni Ethernet controllers.
- *Section 4: Ethernet PHY*
  Describes the DSTni Ethernet physical layer core.
- *Section 5: SPI Controller*
  Describes the DSTni Serial Peripheral Interface (SPI) controller.
- *Section 5: I2C Controller*
  Describes the DSTni I^2C controller.
- *Section 5: USB Controller*
  Describes the DSTni USB controller.
- *Section 5: CAN Controllers*
  Describes the DSTni Controller Area Network (CAN) bus controllers.
- *Section 6: Interrupt Controller*
  Describes the DSTni interrupt controller.
- *Section 6: Miscellaneous Registers*
  Describes DSTni registers not covered in other chapters of this Guide.
- *Section 6: Debugging In-circuit Emulator (Delce)*
- *Section 6: Packaging and Electrical*
  Describes DSTni's packaging and electrical characteristics.
- *Section 6: Applications*
  Describes DSTni's packaging and electrical characteristics.
- *Section 6: Instruction Clocks*
  Describes the DSTni instruction clocks.
- *Section 6: DSTni Sample Code*
- *Section 6: Baud Rate Calculations*
  Provides baud rate calculation tables.

Microprocessor

Overview

The DSTni chip is based on the x86 architecture. This chapter describes the DSTni microprocessor. Topics include:

- Block Diagram on page 5
- Theory of Operation on page 6
- Register Summary on page 17
- Register Definitions on page 18

Block Diagram

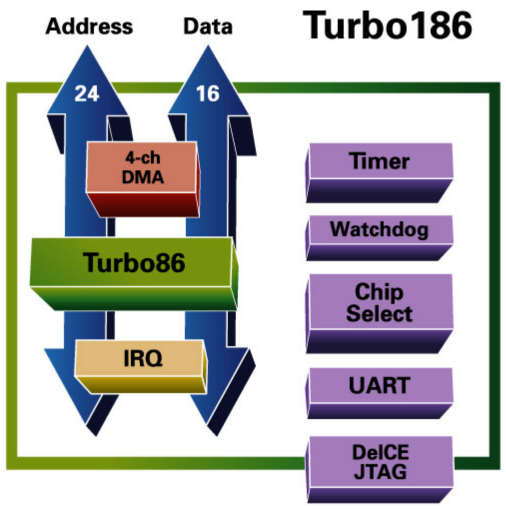


Figure 0-1. CPU Block Diagram

Theory of Operation

Overview

The embedded CPU is a performance-enhanced implementation of the industry-standard 80186 microprocessor. The CPU has been rearchitected from the ground up using the latest design techniques to produce an efficient, high clock-rate CPU core that is greatly improved over standard component implementations. The CPU greatly leverages many years of experience in providing 8086 software compatibility in dozens of SoC designs. All of the 8086 compatibility tests accumulated to establish the compatibility of the CPU core have been used to establish the compatibility of CPU.

The CPU provides approximately 2.5 times the performance of the AMD or Intel 80186 at the same clock frequency. Low-power applications that do not need this increased performance can run at half the clock frequency and still obtain greater throughput. The performance advantage of the CPU is magnified to over five times with its ability to operate at two to three times the clock frequency of standard components. The result is a processor core that exceeds the performance of the 386SX in real mode applications.

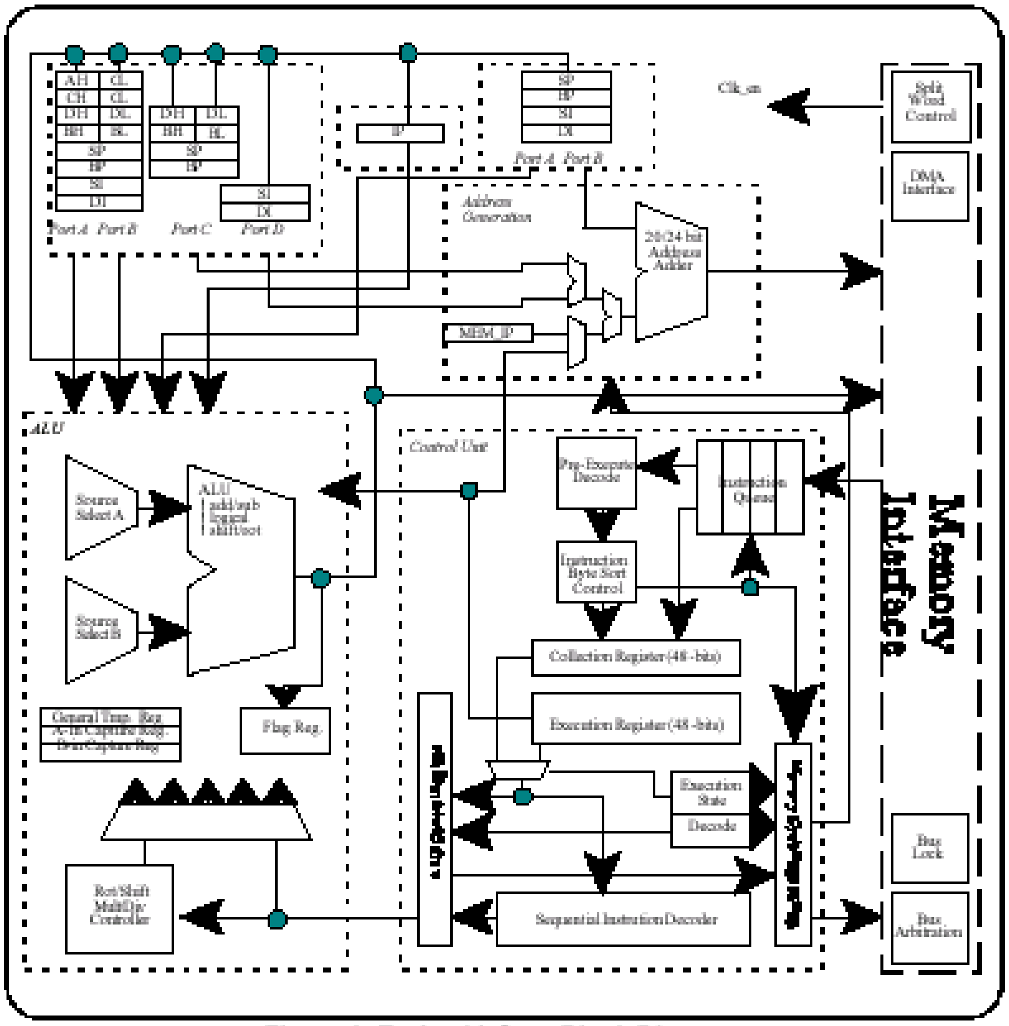
The CPU integrates the 186 set of versatile peripherals with compatibility with the Intel 80C186 or AMD Am186ES. The CPU has been enhanced with all of the 80186 instruction-set enhancements to provide 100% software compatibility with the industry-standard 80186.

On-chip peripherals are always accessed as 16-bit bus cycles.

CPU Core

The CPU core implements an 8086 microprocessor, and includes the 10 additional 80186 instructions. This allows the CPU to be fully software compatible with 8086 and 80186 family of processors. Figure 0-2 shows a block diagram of the CPU core.

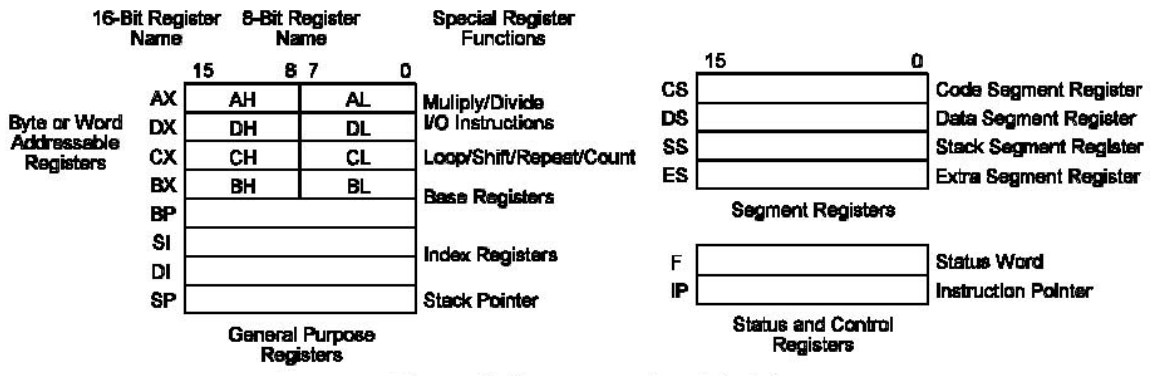
Figure 0-2. CPU Core



Programming Model

Figure 0-3 shows the CPU programming model.

Figure 0-3. CPU Programming Model



To maintain full program compatibility.with the 8086 and the 80186,all registers have been fully implemented.

General Purpose Register File

The general purpose register file consists of eight 16-bit registers, as shown in Figure 0-3. Any of these registers can be used as general purpose registers. Four of these 16-bit registers AX, BX, CX, and DX can also be accessed as a pair of 8-bit registers. When accessing these registers as byte registers they are referred to as AL, AH, BL, BH, CL, CH, DL, and DH. Some of the general purpose registers also have dedicated functions with certain instructions as noted in Figure 0-3. The AX and DX registers are used during multiply, divide, and I/O instructions. The CX register is used during instructions that have a repeat count for example Loop, shift, and rep. BX and BP can be used as base pointers used during effective address calculations. SI and DI can be used as index registers during effective address calculations.

Segment Register File

The segment register file consists of four 16-bit registers, as shown in Figure 0-3. The segment register serve a special function, they select the active segment that is used for code, stack, and data. The CPU core has implicit rules coded into each instruction regarding segment selection, as shown in **Table 2**.

Status and Instruction Pointer Registers

The status word and instruction pointer registers maintain CPU processor state information. The instruction pointer (IP) contains an offset into the current code segment where the next instruction to be executed resides. The status word register contains status and control flags. The format of the status register is shown in Figure 0-4. Arithmetic and logical instructions will affect the status flags (bits 0, 2, 4, 6, 7, and 11) based on the result of an instruction. While the control flags (bits [8], [9], and [10]) determine certain processing control information. The status word bit functions are shown in Figure 0-1 and defined in Table 0-1.

Figure 0-4. Status Word Bit Format

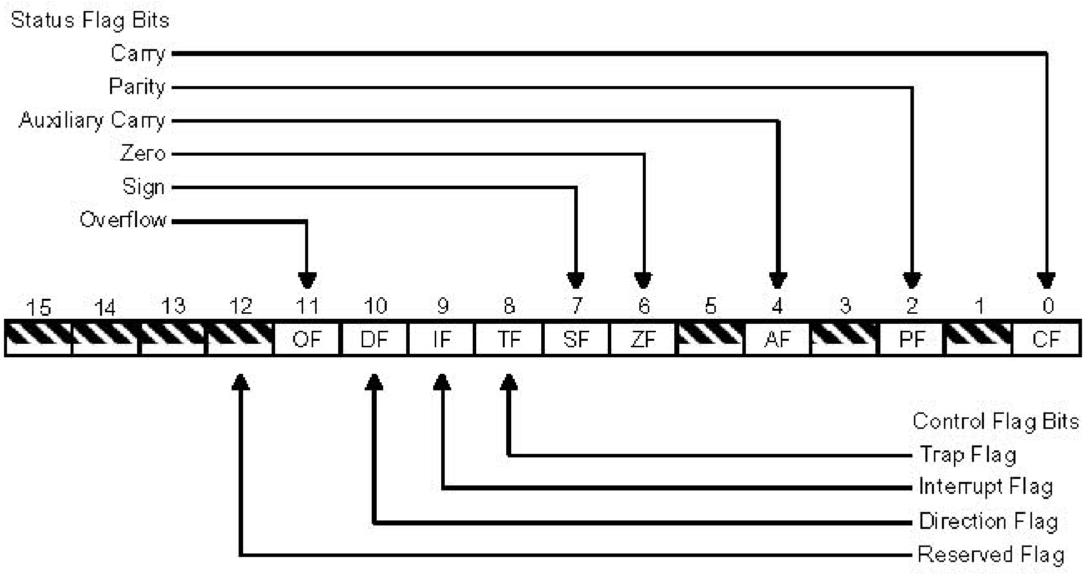


Table 0-1. Status Word Bit Functions

Name	Function
CF	**Carry Flag** Set if a high-order bit carry or borrow occurred. Otherwise, it is cleared.
PF	**Parity Flag** Set of the lower 8 bits of a result contain an even number of ‘1’s. Otherwise, it is cleared.
AF	**Auxiliary Carry** Set if a carry from or borrow to the low-order four bits of the General Purpose register AL. Otherwise, it is cleared.
ZF	**Zero Flag** Set if the result is ‘0’. Otherwise, it is cleared.
SF	**Sign Flag** Set equal to the high-order bit of a result (‘0’ is positive, ‘1’ if negative).
TF	**Single Step Flag** If set, a single-step interrupt occurs after the next instruction executes. TF is cleared by the single-step instruction.
IF	**Interrupt Enable Flag** Setting this bit enables maskable interrupt.
DF	**Direction Flag** Used during single instructions to determine auto-decrement (DF=1) or auto increment (DF=0) of index value.
OF	**Overflow Flag** Set if the signed result cannot be expressed by the number of bits in the resultant operand. Otherwise, it is cleared.

Instruction Set and Execution Times

This section lists all CPU instructions and gives execution times in clock cycles. The

8086/80186 instruction set is fully documented in a variety of publicly available documents and is not described in detail here.

Note: The cycle times represent the number of clock periods an instruction takes to execute after reaching the CPU execution unit. Additional delays may be present in the instruction stream due to the singular bus architecture used by x86. These delays will be deterministic for any given instruction stream but are difficult to quantify in simple terms since they result from the interaction of multiple instructions. The CPU minimizes these delays (often zero) through the use of pipelining and an instruction buffer queue.

Memory Organization

The CPU can address 1MB of physical memory in compatible mode, or 16Mb of physical memory in 24-bit enhanced mode. In either of these modes memory organization is similar.

Memory is organized in sets of segments. Each segment is a linear contiguous sequence of up to 64K 8-bit bytes. To address memory, two 16-bit pointers must be added together. In compatible mode, the 20-bit address is generated by shifting a segment value left by 4-bits and adding it to a 16-bit offset or effective address (see Figure 0-5**)**. In enhanced mode, the 24-bit address is generated by shifting a segment value left by 8-bits and adding it to a 16-bit offset or effective address (see Figure 0-6).

In either compatible or enhanced mode all instructions that address operands in memory must specify a segment register and the 16-bit offset value. Segment registers used for physical address generation are implied by the addressing mode used (see Table 0-2**)**. These rules follow the way programs are written as independent modules that require areas for code, data, stack, and external data areas.

Special segment override instruction prefixes allow the implicit segment register selection rules to be overridden for special cases.

Table 0-2. Segment Register Selection Rules

Memory Reference	Segment Register Used	Implicit Segment Selection Rule
Instructions	Code (CS)	Instruction prefetch and immediate data
Local Data	Data (DS)	All other data references
Stack	Stack (SS)	All stack pushes and pops; any memory references that use BP as a base register.
External Data (Global)	Extra (ES)	All string instruction references that use the DI register as an index.

Figure 0-5. 20-Bit Address Generation

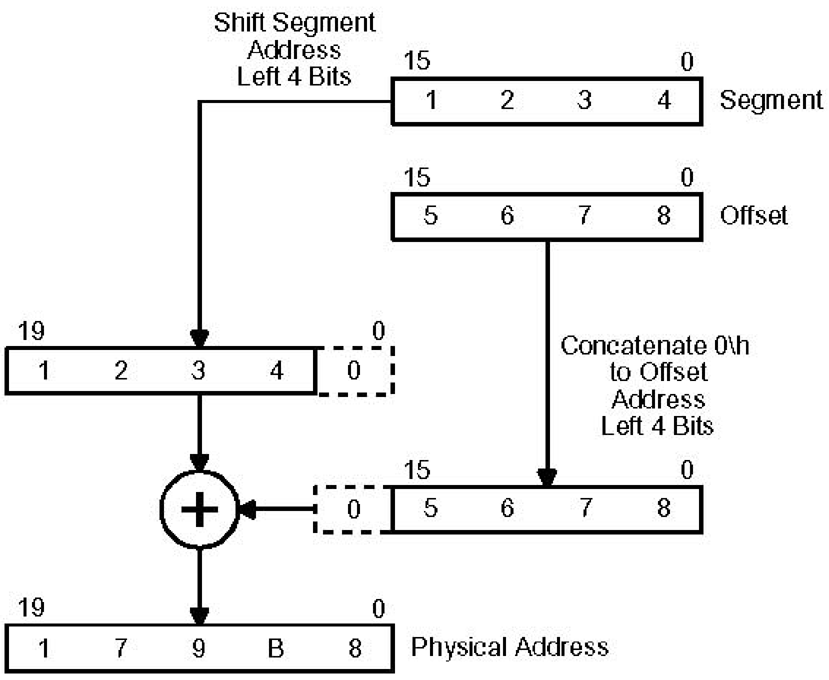
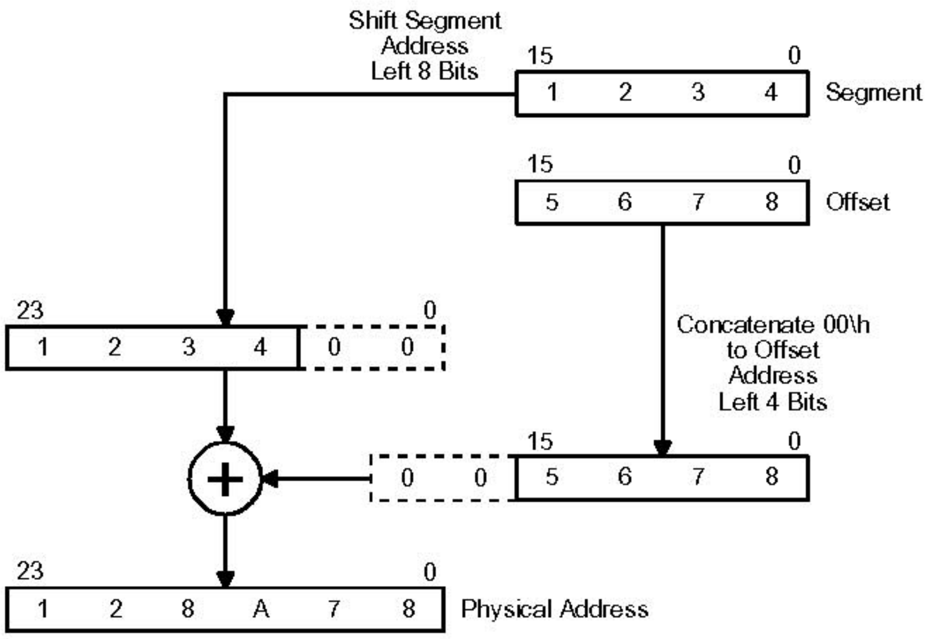


Figure 0-6. 24-Bit Address Generation



Addressing Modes

The CPU supports all the original 8086 addressing modes. There are eight categories of addressing mode to specify operands. Two addressing modes are provided that operate on register or immediate data:

- Register Operand Mode: The operand is located in one of the 8 or 16-bit registers.
- Immediate Operand Mode: The operand is located in the instruction.

Six modes are provided to specify the location of an operand in a memory segment. A memory operand address consists of two 16-bit components: a segment and an offset. The segment base is supplied by a 16-bit segment register either implicitly chosen by the addressing mode or explicitly chosen by a segment override prefix (see Table 0-2 on page 10). The offset, which is also called the effective address, is calculated by summing any combination of the following three address elements:

- The displacement, which is an 8 or 16-bit immediate value contained in the instruction.
- The base, which is either the BX or BP registers.
- The index, which is either the SI or DI registers.

All effective address calculations are done as 16-bit additions any carry out is ignored. All 8-bit displacements are sign extended to 16-bits. Combinations of these three address elements define the six memory addressing modes for the CPU. These addressing modes are as follows:

- Direct Mode: The operand's offset is contained in the instruction as an 8- or 16-bit displacement element.
- Register Indirect Mode: The operand's offset is in one of the SI, DI, BX, or BP registers.
- Based Mode: The operand's offset is the sum of an 8 or 16-bit displacement and the contents of a base register (BX or BP).
- Indexed Mode: The operand's offset is the sum of an 8 or 16-bit displacement and the contents of an index register (SI or DI).
- Based Indexed Mode: The operand's offset is the sum of the contents of a base register (BX or BP) and an index register (SI or DI).
- Based Index Mode with Displacement: The operand's offset is the sum of a base register's contents, an index register's contents, and an 8 or 16-bit displacement.

Data Types

The CPU supports the following data types:

Integer:	A signed binary numeric value contained in an 8-bit byte or 16-bit word. All operations assume a two's complement representation.
Ordinal:	An unsigned binary numeric value contained in a 8-bit byte or a 16-bit word.
Pointer:	These are 16 or 32-bit quantities. Pointers can be composed of a single 16-bit offset or a 16-bit segment base and 16-bit offset.
String:	A contiguous sequence of bytes of word. A string maybe 1 to 64 Kbytes in length.
ASCII:	A byte representation of the alphanumeric and control characters using the ASCII standard.
BCD:	A byte representation of the decimal digits 0-9. Also referred to as an unpacked BCD representation.
Packed BCD:	A byte representation of two decimal digits (0-9). One digit is stored in the lower nibble (bits 3-0), and the second in the upper nibble (bits [7] through [4]).

Input/Output Space

The CPU can address a maximum of 64K bytes or 32K words of I/O. The IN and OUT instructions are used to transfer data between the AL register for byte I/O, and the AX register for word I/O. Accesses to and from I/O space are not segmented. To access a port, the BIU places the port address (0-64K) specified by the I/O instruction on address bits [15] through [0]. The address bits above bit [15] are always zero for I/O cycles.

Note: *I/O port addresses 00F8\h through 00FF\h are reserved.*

The CPU can be initialized to a defined state by asserting the RSTIN# input LOW. This causes all processor activity to terminate, the processor remains in an idle state while RSTIN# input low. When the RSTIN# input finally returns HIGH, certain internal registers are initialize,d as shown in Table 0-3. Program execution will begin at one of two addresses depending if operating in compatible or enhanced mode. In compatible mode program execution begins at 0FFFF0\h [CS:FFFF IP:0000], enhanced 24-bit mode program execution begins at FFFFE0\h [CS:FFFF;IP:00E0].

Table 0-3. CPU Registers After Reset

Register Name	Value After Reset
Status Word	F000\h
Instruction Pointer	0000\h (00E0\h in 24-bit mode)
Code Segment Register	FFFF\h
Data Segment Register	0000\h
Extra Segment Register	0000\h
Stack Segment Register	0000\h

DSTni Peripherals

DSTni is software compatible with popular 80186 designs. It is similar to an AMD AM186ES. A few minor functions, such as the refresh controller and certain control bits related to the design of the original 80C186XL bus controller, have been omitted from these versions. Where discrepancies exist between the original specifications from the manufacturers and this specification, the data in this specification takes precedent.

The following peripherals are included in all of the CPU peripheral collections.

- Chip select-and-ready control logic
- 4-channel DMA controller
- Three programmable 16-bit timers
- Interrupt controller (Master Mode Only)

Each peripheral has been designed from the ground up to provide a fully synchronous operation with the CPU core. All the peripherals support zero-wait state accesses and work with the single cycle CPU through an internal bridge.

Note: All internal peripherals should only be written with 16-bit writes. Byte writes are not supported to the CPU peripherals and may cause unpredictable behavior. No power-save features are available in the CPU. CPU clock speed may be reduced to save power.

Peripheral Control Block

All integrated DSTni peripherals are controlled by sets of 16-bit registers contained within a 256-byte integrated Peripheral Control Block (PCB). The peripheral control registers are physically located in the peripheral devices they control, but are addressed as a single 256-byte block. The PCB encompasses 256 contiguous bytes and can reside on any 256-byte boundary of memory or input/output (I/O) space. The PCB Relocation register, which is also located within the PCB, controls the PCB location.

Peripheral Control Registers

Each of the integrated peripheral's registers is located at a fixed offset above the programmed base location of the PCB. These registers are described in the chapters that cover the associated peripheral.

PCB Relocation Register

In addition to control registers for the integrated peripherals, the PCB contains a PCB Relocation register. The Relocation register is located at a fixed offset within the PCB. If the PCB moves, the Relocation register also moves.

The PCB Relocation register allows the PCB to relocate to any 256-byte boundary within memory or I/O space. In 24-bit mode, the PCB can only reside in the lower 1 MB. The Memory I/O bit (bit [12] of the Relocation register) selects either memory or I/O space, and bits [19:8] specify the starting (base) address of the PCB.

For information about setting the base location and restrictions on the PCB location, see Setting the PCB Base Location on page 15.

Reserved Locations

Some locations within the PCB are not assigned to any peripheral. Unused locations are reserved and marked with "///" in the register tables under Register Definitions on page 18. Reading from these locations yields an undefined result. If reserved register locations are written (for example, during a block MOV instruction), they should be set to 0h.

Accessing the PCB

Like all integrated peripherals, the PCB is always accessed 16 bits at a time.

Bus Cycles

The microprocessor runs an external bus cycle for any memory or I/O cycle accessing a location within the PCB. Address, data and control information are driven on the external pins as with an ordinary bus cycle. Information returned by an external device is ignored, even if the access does not correspond to the location of an integrated peripheral control register.

Writing the PCB Relocation Register

When mapping the PCB to another location, program the Relocation register with a word write (i.e., OUT DX, AX).

Setting the PCB Base Location

Upon reset, the PCB Relocation register contains the value 00FFh. Writing the PCB Relocation register lets you change this location.

For example, to relocate the PCB to the memory range 10000-100FFh, program the PCB Relocation register with the value 1100h. Since the Relocation register is part of the PCB, it relocates to word 10000h plus its fixed offset.

Direct Memory Access

In many applications, large blocks of data must transfer between memory and I/O space. A disk drive, for example, reads and writes data in blocks that can be thousands of bytes long. If the microprocessor had to handle each byte of the transfer, the main tasks would suffer a severe performance penalty. Even if the data transfers was interrupt-driven, the overhead for transferring control to the interrupt handler would still decrease system throughput.

DMA enables data to transfer between memory and peripherals without requiring microprocessor intervention. Data transfers can occur between memory and I/ O spaces or within the same (see DMA Transfer Process on page 16). DSTni provides four high-speed DMA channels. All four DMA channels are functionally identical and can directly connect to the asynchronous serial ports.

Each channel accepts a DMA request from one of four sources:

- The channel request pin (DRQ3–DRQ0)
- Timer 2
- A serial port
- System software

Furthermore, the channels can be programmed with different priorities in the event of a simultaneous DMA request or if there is a need to interrupt transfers on one of the other channels.

Either bytes or words can be transferred to or from even or odd addresses on DSTni. However, DSTni does not support word DMA transfers to or from memory configured for 8-bit accesses. Only two bus cycles are needed for each data transfer.

DMA Transfer Directions

The source and destination addresses for a DMA transfer are programmable and can be in either memory or I/O space. DMA transfers can be programmed for any of the following four directions:

- From memory space to I/O space
- From I/O space to memory space
- From memory space to memory space
- From I/O space to I/O space

DMA transfers can access the PCB.

DMA Transfer Process

The following sequence describes the DMA process:

1. A peripheral sends a request for a transfer to the DMA controller.
2. The DMA controller signals the microprocessor that it needs control of the system bus.
3. The microprocessor releases control of the bus and the DMA controller performs the transfer. In many cases, the microprocessor releases the bus and continues to execute instructions from the prefetch queue.

Every DMA transfer consists of two distinct bus cycles: a fetch and a deposit.

- During the fetch cycle, the byte or word is read from the data source and placed in an internal temporary storage register.
- During the deposit cycle, the data in the temporary storage register writes to the destination.

The fetch and deposit bus cycles are indivisible and cannot be separated by a bus hold request, a refresh request, or another DMA request. If the DMA transfers are relatively infrequent, the DMA transfer is transparent to the microprocessor and there is no degradation of software performance.

DMA Operation

Each channel has the following 16-bit registers in the PCB that define specific channel operations:

- Two DMA Address High registers — one for source and one for destination.
- Two DMA Address Low registers — one for source and one for destination.
- A DMA Transfer Count register — which specifies the number of DMA transfers to be performed. Up to 64K of byte or word transfers can be performed with automatic termination.
- A DMA Channel Control register — which determines the operating mode for the particular DMA channel.

All registers can be modified during any DMA activity. Any changes made to the DMA registers are reflected immediately in DMA operation. For more information, see Register Summary on page 17.

DMA Priority

You can program the DMA channels so that one channel always has priority over the others, or so they alternate cycles when all four have DMA requests pending. DMA cycles always have priority over internal microprocessor cycles, except between locked memory accesses or word accesses to odd memory locations. However, an external bus hold takes priority over an internal DMA cycle.

Because an interrupt request cannot suspend a DMA operation, and because the microprocessor cannot access memory during a DMA cycle, interrupt latency time suffers during sequences of continuous DMA cycles. An NMI request, however, causes all internal DMA activity to halt. This allows the microprocessor to respond quickly to the NMI request.

Register Summary

Table 0-4. System Register Summary

Hex Offset	Mnemonic	Register Description	Page
FE	RELREG	Peripheral Control Block Relocation register	18
FC	///	///	///
FA	///	///	///
F8	DCR	DSTni Configuration register	19
F6	RCR	Reset Configuration register	22
F4	PRL	Processor Release Level register	22
F2	AUXCON	Auxiliary Configuration register	23
F0	SYSCON	System Configuration register	24

Table 0-5. DMA Controller Register Summary

Hex Offset	Mnemonic	Register Description	Page

DMA0 = CA
DMA1 = DA
DMA2 = 9A
DMA3 = BA
	D0CON
	D1CON
	D2CON
	D3CON
		DMA Control registers	25

DMA0 = C8
DMA1 = D8
DMA2 = 98
DMA3 = B8
	D0TC
	D1TC
	D2TC
	D3TC
		DMA Transfer Count registers	27

DMA0 = C6
DMA1 = D6
DMA2 = 96
DMA3 = B6
	D0DTSH
	D1DTSH
	D2DTSH
	D3DTSH
		DMA Destination Address High registers	28

DMA0 = C4
DMA1 = D4
DMA2 = 94
DMA3 = B4
	D0DSTL
	D1DSTL
	D2DSTL
	D3DSTL
		DMA Destination Address Low registers	29

DMA0 = C2
DMA1 = D2
DMA2 = 92
DMA3 = B2
	D0SRCH
	D1SRCH
	D2SRCH
	D3SRCH
		DMA Source Address High registers	29

DMA0 = C0
DMA1 = D0
DMA2 = 90
DMA3 = B0
	D0SRCL
	D1SRCL
	D2SRCL
	D3SRCL
		DMA Source Address Low registers	30

Register Definitions

The following sections provide the microprocessor register definitions. In these sections, the initialization value shown is the register's initialization value at reset.

PCB Relocation Register

The PCB Relocation register is a 16-bit register that maps the PCB into either memory or I/O space. This register provides the upper 12 bits of the control block's base address. The control block is effectively an internal chip select range.

Other chip selects can overlap the control block if they are programmed to zero wait states and ignore external ready. If the control register block maps into I/O space, the upper four bits of the base address must be programmed as 0000b, since I/O addresses are only 16 bits wide. At reset, this register is set to 00FFh, which maps the control block to start at FF00h in I/O space.

Table 0-6. PCB Relocation Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	FEh															
FIELD	///	///	///	M/IO	Relocation Address RA [19:8]											
RESET	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 0-7. PCB Relocation Register Definitions

Bits	Field Name	Description
15:13	///	**Reserved**
12	M/IO	**PCB Location** 0 = PCB is in I/O space. 1 = PCB is in memory space.
11:0	Relocation Address RA [19:8]	Defines the 256-byte boundary where the PCB resides.

DSTni Configuration Register

The DSTni Configuration register is a Read/Write register that is reset using the values on address lines 22 down to 8 as DSTni exits reset.

This register may be written to by the CPU. However exercise caution because these values can cause the CPU to enter invalid modes.

***Note:** The reset value of these bits is determined by internal pull-up/down resistors. These values may be overridden by the presence or absence of external pull-up/down resistors. If an external resistor is not applied, each bit has a default value as described by Table 0-10.*

The DCR allows hardware design-time configuration choices to be made when designing a new product around DSTni-EX. It is possible but not common for software to modify the contents of this register.

SPIEN selects between SPI or Serial Port 3 on 4 of the 16 PIO pins. When SPI is enabled, Serial Port 3 cannot be used.

The bit values in the DSTni Configuration register correspond to the address bus as the ASIC is reset. The values at the completion of the reset can be modified by applying the proper pull up or pull-down resistors externally to override the default internal resistors (see *Table 0-10*). DCR pins that are not assigned to hardware functions are used to configure the behavior of the on-chip boot ROM when it is enable (BROMEN = 1). Refer to the DSTni-EX Bootstrap User Guide for information about other uses of DCR.

Table 0-8. DSTni Configuration Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	F8h															
FIELD	///	BROMEN	ADDR24	WDOGEN	SPIEN	///	///	///	///	///	///	CSBE	///	///	///	///
RESET	0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 0-9. DSTni Configuration Register Definitions

Bits	Field Name	Description
15	///	**This bit value cannot be changed.**
14:0	Data	
14	BROMEN	**Boot ROM Enable** 1 = enable the internal Boot ROM. 0 = disable the internal Boot ROM. The external upper chip select is accessed when the Boot ROM is disabled. The internal Boot ROM always has zero wait states. This bit defaults to 1 with an internal pull-up resistor. Placing an external pull-down resistor on A22 resets this bit to 0.
13	ADDR24	**Address 24 Mode** 1= enable the CPU to execute in a 24 bit address mode with 16 megabytes of address space. 0 = the CPU executes in real mode x86 1 Megabyte of address space. This bit defaults to 1 with an internal pull-up resistor. Placing an external pull-down resistor on A21 resets this bit to 0.
12	WDOGEN	**Watch Dog Enable** 1 = enable the internal watchdog on reset. 0 = disable the watchdog on reset. This can be useful for testing and debugging. This bit defaults to a 1 with an internal pull-up resistor. Placing an external pull-down resistor on A20 resets this bit to 0.
10	SPIEN	**SPI Enable** 1 = enable the SPI controller on serial port 3. 0 = SPI controller is not connected to the I/O. This bit defaults to 0 with an internal pull-down resistor. Placing an external pull-up resistor on A18 resets this bit to reset to 1.
4	CSBE	**Chip Select Byte Enable** 1 = enable WRH_n and WRL_n to connect to byte write enable control signals. 0 = enable byte control signals. This is typical for x16 external static memory devices. This bit defaults to 1 with an internal pull-up resistor. Placing an external pull-down resistor on A12 resets this bit to 0.

Table 0-10. Default Reset Values

Data	ADR Pin	Default	Value	External Connection	0 =
Data 15		0	Reserved	Always zero	
Data 14	22	1	BROMEN	Address Bit 22 (pulled up)	disabled
Data 13	21	1	ADDR24	Address Bit 21 (pulled up)	20-bit
Data 12	20	1	WDOGEN	Address Bit 20 (pulled up)	disabled
Data 11	19	0	SPIBOOT	Address Bit 19 (pulled down)	disabled
Data 10	18	0	SPIEN	Address Bit 18 (pulled down)	Ser3 uses pins
Data 9	17	0	ETHBOOT	Address Bit 17 (pulled down)	disabled
Data 8	16	0	ETHCHAN	Address Bit 16 (pulled down)	Use chan 0
Data 7	15	1	LEDENC	Address Bit 15 (pulled up)	4 mono LEDs
Data 6	14	1	PARBOOT	Address Bit 14 (pulled up)	Disabled
Data 5	13	1	PHYBYPASS	Address Bit 13 (pulled up)	Set to 10Mbps
Data 4	12	1	CSBE CSMODE	Address Bit 12 (pulled up)	Use UCE/LCE
Data 3	11	0	SERBOOT	Address Bit 11 (pulled down)	Disabled
Data 2	10	0	SERCHAN	Address Bit 10 (pulled down)	SP0
Data 1	9	0	SERSPEED	Address Bit 9 (pulled down)	57600 baud
Data 0	8	0	DEBUG	Address Bit 8 (pulled down)	disabled

Reset Configuration Register

The Reset Configuration register is a Read Only register that latches system-configuration information presented to the microprocessor on the data bus when exiting reset. The interpretation of this information is system-specific. The microprocessor does not impose any predetermined interpretation; it simply acts as a conduit for communicating the information to software.

***Note:** This register resets to a default of all '0's unless pull-up resistors are placed on the external data bus.*

Table 0-11. Reset Configuration Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	F6h															
FIELD	Data															
RESET	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 0-12. Reset Configuration Register Definitions

Bits	Field Name	Description
15:0	Data	A value that corresponds to the data bus as it exits reset. The software uses this value as dictated by the application. The bus is pulled-down by internal resistors. Pull-up resistors placed externally on the data bus can present a 1 value in the corresponding data bit for this register.

Processor Release Level Register

The Processor Release Level register is a Read Only register that specifies the microprocessor version and type.

Table 0-13. Processor Release Level Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	F4h															
FIELD	PRL								OPTIONID				CPU TYPE			
RESET	—	—	—	—	—	—	—	—	0	0	0	0	—	—	—	—
RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 0-14. Processor Release Level Register Definitions

Bits	Field Name	Description
15:8	PRL	**Processor Release Level** A = 01h, B=02h, etc.
7:4	OPTIONID	**OPTIONID** Indicates the options included in DSTni. These bits also correspond to the DEVICE_ID in the boundary scan. DSTni = 0000.
3:0	CPU TYPE	**Type of CPU** 0h = AMD, 1h = DSTni-EX, 2h = DSTni-LX

Auxiliary Configuration Register

The Auxiliary Configuration register configures the asynchronous serial port flow-control signals. It also configures the data bus width for memory and I/O accesses. For more information, refer to the Serial Port theory of operation.

Table 0-15. Auxiliary Configuration Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	F2h															
FIELD	///					ENRX3	DTE3	ENRX2	DTE2	ENRX1	DTE1	ENRX0	DTE0	LSIZ	MSIZ	IOSIZ
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	R	R

Table 0-16. Auxiliary Configuration Register Definitions

Bits	Field Name	Description
15:11	///	**Reserved**
10	ENRX3	**Enable Receiver Request** 1 = corresponding CTS/ENRX pin is configured as ENRX. 0 = corresponding CTS/ENRX pin is configured as CTS (*default*).
9	DTE3	**DTE/DCE Channel Configuration** 1 = corresponding serial channel is configured as DTE. 0 = corresponding serial channel is configured as DCE (*default*).
8	ENRX2	**Enable Receiver Request** 1 = corresponding CTS/ENRX pin is configured as ENRX. 0 = corresponding CTS/ENRX pin is configured as CTS (*default*).
7	DTE2	**DTE/DCE Channel Configuration** 1 = corresponding serial channel is configured as DTE. 0 = corresponding serial channel is configured as DCE (*default*).
6	ENRX1	**Enable Receiver Request** 1 = corresponding CTS/ENRX pin is configured as ENRX. 0 = corresponding CTS/ENRX pin is configured as CTS (*default*).
5	DTE1	**DTE/DCE Channel Configuration** 1 = corresponding serial channel is configured as DTE. 0 = corresponding serial channel is configured as DCE (*default*).
4	ENRX0	**Enable Receiver Request** 1 = corresponding CTS/ENRX pin is configured as ENRX. 0 = corresponding CTS/ENRX pin is configured as CTS (*default*).
3	DTE0	**DTE/DCE Channel Configuration** 1 = corresponding serial channel is configured as DTE. 0 = corresponding serial channel is configured as DCE (*default*).
2	LSIZ	**Data Bus Width for Accesses to LCS Space** 1 = perform 8-bit accesses. 0 = perform 16-bit accesses. This bit is always 0.
1	MSIZ	**Data Bus Width for Accesses Outside the UCS or LCS Address Space, including the MCS and PCS Address Space** 1 = perform 8-bit accesses. 0 = perform 16-bit accesses. This bit is always 0.
0	IOSIZ	**Data Bus Width for All I/O Space Accesses** 1 = perform 8-bit accesses. 0 = perform 16-bit accesses. This bit is always 0.

System Configuration Register

The System Configuration register enables MCS0-only mode. It also disables clock output.

Table 0-17. System Configuration Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	F0h															
FIELD	///	MCSBIT	///					CD	///							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 0-18. System Configuration Register Definitions

Bits	Field Name	Description
15	///	**Reserved**
14	MCSBIT	**Enable MCS0-Only Mode** 1 = the MCS0 pin is active over the entire MCS range. 0 = the MCS0 pin is active for only ¼ of the MCS range.
13:9	///	**Reserved**
8	CD	**Clock Disable** 1 = disable (tri-state) clock output (CPUCLK, pin E14). 0 = enable (tri-state) clock output. PLL clock is driven out on CPUCLK, pin E14.
7:0	///	**Reserved**

DMA Control Registers

The DMA Control registers determine the operating mode for the DMA channels. These registers specify whether:

- The destination pointer is memory or I/O space.
- The destination pointer increments, decrements, or keeps constant after each transfer.
- The source pointer is in memory or I/O space.
- The source pointer increments, decrements, or keeps constant after each transfer.
- DMA activity stops after a programmed number of DMA cycles.
- An interrupt generates with the last transfer

These registers also specify:

- The mode of synchronization.
- A DMA channel's relative priority with respect to the other DMA channel.
- Whether Timer 2 DMA requests are enabled or disabled.
- Whether a channel's DMA operations start or stop.
- The data size (bytes or words) of DMA transfers.

Table 0-19. DMA Control Registers

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	DMA 0 = CAh DMA 1 = DAh DMA 2 = 9Ah DMA3 = BAh															
FIELD	M/IO	DEC	INC	M/IO	DEC	INC	TC	INT	SYN		P	TDRQ	///	CHG/NCHG	ST/STOP	B/W
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW

Table 0-20. DMA Control Register Definitions

Bits	Field Name	Description
15	M/IO	**Destination Pointer** 0 = destination pointer is in I/O space. 1 = destination pointer is in memory space.
14	DEC	**Decrement Destination Pointer** 1 = decrement destination pointer after each access. Decrement by 1 or 2, depending on whether configured for byte or word operations.
13	INC	**Increment Destination Pointer** 1 = increment destination pointer after each access. Increment by 1 or 2, depending on whether configured for byte or word operations.
12	M/IO	**Source Pointer** 0 = source pointer is in I/O space. 1 = source pointer is in memory space.
11	DEC	**Source Decrement Pointer** 1 = decrement source pointer after each access. Decrement by 1 or 2, depending on whether configured for byte or word operations.

Bits	Field Name	Description
10	INC	**Source Increment Pointer** 1 = increment source pointer after each access. Increment by 1 or 2, depending on whether configured for byte or word operations.
9	TC	**Transfer Count** 1 = DMA operation will end when the Transfer Count register value reaches zero, and the ST/STOP bit will also be reset. 0 = Transfer Count register decrements on each transfer, but DMA operations do not end when the Transfer Count register reaches zero.
8	INT	**Interrupt Request** 1 = generate interrupt request when transfer count ends.
7:6	SYN	**Synchronization** Two bits that encode the type of synchronization used. 00 = no external synchronization necessary. DMA transfers can occur at any time. Back-to-back DMA bus cycles are performed. 01 = use source synchronization. DMA transfers occur when the source signals a DMA request via the DRQ3-0 signals. Back-to-back DMA bus cycles are performed. 10 = use destination synchronization. DMA transfers occur when the destination signals a DMA request via the DRQ3-0 signals. After each DMA bus cycle, the controller waits 2 clocks before starting another DMA bus cycle. This delay lets the destination release its DMA request if it cannot handle another DMA operation and gives the microprocessor time to perform a bus cycle between DMA operations. 11 = not used.
5	P	**Priority** Determines a channel’s priority level relative to other channels. 0 = low priority. 1 = high priority. If two or more channels have simultaneous requests with the same priority level, they alternate cycles. Default priority among DMA channels is DMA0 (highest), DMA1, DMA2, and DMA3 (lowest).
4	TDRQ	**DMA Request** Enables a channel’s DMA request from Timer 2 instead of the external DMA request signal. 1 = enable DMA request from Timer 2. 0 = disable this feature.
3	///	**Reserved** This bit is always set to 0.
2	CHG/NCHG	**Enable/Disable Updating of ST/STP Bit** Writes to the Control Word to enable or disable updating the ST/STP bit. 1 = the ST/STP bit in the control word can be changed. 0 = the ST/STP bit in the control word cannot be changed. This bit must be 1 during a write operation to enable writing the ST/STP bit.
1	ST/STOP	**DMA Operations** Starts and stops a channel’s DMA operations. 1 = start DMA operations. 0 = stop DMA operations. Bit [2], CHG/NCHG must be 1 during a write operation to write the ST/STP bit.
0	B/W	**Data Size of DMA Transfer** Determines the data size of the DMA transfer. 1 = enable word transfers. 0 = enable byte transfers. This bit also determines whether to increment or decrement source and destination pointers by 1 (byte) or 2 (word).

DMA Transfer Count Registers

Each DMA channel has a 16-bit DMA Transfer Count register. This register decrements after each DMA cycle. If the TC bit (bit [9]) in the DMA control word is set, DMA activity ends when the Transfer Count register reaches zero.

Table 0-21. DMA Transfer Count Registers

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	DMA 0 = C8h DMA 1 = D8h DMA 2 = 98h DMA3 = B8h															
FIELD	CNT [15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 0-22. DMA Transfer Count Register Definitions

Bits	Field Name	Description
15:0	CNT [15:0]	**16-bit Transfer Count Value**

DMA Destination Address High Registers

Each DMA channel has a 24-bit Destination register and a 24-bit Source register. Each 24-bit address occupies two 16-bit registers (the High register and the Low register) in the PCB. For each DMA channel to be used, all four Address registers for that channel must be initialized. These addresses can individually increment or decrement after each transfer.

- If word transfers are performed, the address increments or decrements by 2 after each transfer.
- If byte transfers are performed, the address increments or decrements by 1.

Each register can point into either memory or I/O space. You must program the upper eight bits to all '0's to address the normal 64K I/O space. Since the DMA channels can perform transfers to or from odd addresses, there is no restriction on values for the Destination and Source Address registers. However, DSTni can achieve higher transfer rates by performing all word transfers to or from even addresses, so that accesses occur in single 16-bit bus cycles.

Table 0-23. DMA Destination Address High Registers

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	DMA 0 = C6h DMA 1 = D6h DMA 2 = 96h DMA3 = B6h															
FIELD	///	///	///	///	///	///	///	///	A [23:16]							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W

Table 0-24. DMA Destination Address High Register Definitions

Bits	Field Name	Description
15:8	///	**Reserved**
7:0	A [23:16]	**Upper Address of the Destination Pointer**

DMA Destination Address Low Registers

This 16-bit register works with the eight bits of the DMA Destination Address High register to produce a 24-bit destination address.

Table 0-25. DMA Destination Address Low Registers

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	DMA 0 = C4h DMA 1 = D4h DMA 2 = 94h DMA3 = B4h															
FIELD	A [15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W

Table 0-26. DMA Destination Address Low Register Definitions

Bits	Field Name	Description
15:0	A [15:0]	**Lower Address of the Destination Pointer**

DMA Source Address High Registers

Each DMA channel has a 24-bit Destination register and a 24-bit Source register. Each 24-bit address occupies two 16-bit registers (the High register and the Low register) in the PCB. For each DMA channel to be used, all four Address registers for that channel must be initialized. These addresses can individually increment or decrement after each transfer.

- If word transfers are performed, the address increments or decrements by 2 after each transfer.
- If byte transfers are performed, the address increments or decrements by 1.

Each register can point to memory or I/O space. You must program the upper four bits to 0000b to address the normal 64K I/O space. Since the DMA channels can perform transfers to or from odd addresses, there is no restriction on Destination and Source Address register values. However, DSTni can achieve higher transfer rates by performing all word transfers to or from even addresses, so that accesses occur in single 16-bit bus cycles.

Table 0-27. DMA Source Address High Registers

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	DMA 0 = C2h DMA 1 = D2h DMA 2 = 92h DMA3 = B2h															
FIELD	///	///	///	///	///	///	///	///	A [23:16]							
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	R W	RW	R W	R W	RW	RW	R W	R W	R W	R W	R W	R W	R W	R W	R W	R W

Table 0-28. DMA Source Address High Register Definitions

Bits	Field Name	Description
15:8	///	**Reserved**
7:0	A [23:16]	**Upper Address of the Source Pointer**

DMA Source Address Low Registers

This 16-bit register works with the eight bits of the DMA Source Address High register to produce a 24-bit source address.

Table 0-29. DMA Source Address Low Registers

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	DMA 0 = C0h DMA 1 = D0h DMA 2 = 90h DMA3 = B0h															
FIELD	A [15:0]															
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	R W	RW	R W	R W	RW	RW	RW	R W	R W	R W	R W	R W	R W	R W	R W	R W

Table 0-30. DMA Source Address Low Register Definitions

Bits	Field Name	Description
15:0	A [15:0]	**Lower Address of the Source Pointer**

READY Signal and Wait States

The microprocessor generates an internal READY signal:

- When an integrated peripheral is accessed. External READY is ignored.
- If an access is made to a location within the PCB that does not correspond to an integrated peripheral control register.

The microprocessor does not insert wait states for accesses to locations in the PCB.

Wait State Generation

The microprocessor supports 15 wait states to memory accesses external to DSTni. Bits [R3] through [R0] determine these wait state values, as shown in Table 0-31.

Table 0-31. Wait State Generation

R3	R2	R1	R0	Number of Wait States Generated
1	1	0	0	0 wait states
1	1	0	1	1 wait state
1	1	1	0	2 wait states
1	1	1	1	3 wait states
1	0	0	0	4 wait states
1	0	0	1	5 wait states
1	0	1	0	6 wait states
1	0	1	1	7 wait states
0	1	0	0	8 wait states
0	1	0	1	9 wait states
0	1	1	0	10 wait states
0	1	1	1	11 wait states
0	0	0	0	12 wait states
0	0	0	1	13 wait states
0	0	1	0	14 wait states
0	0	1	1	15 wait states

Page Chip Select Register

Burst and page mode flash devices can be used to decrease access times from 120-65 ns to 25-17 ns. Most page mode flash devices use a page size of 8 words, though some use 4 words. A page hit is determined by the page size. Any new accesses within this page size are considered a page hit, yielding faster access time. Any access outside this page size is detected as a page miss. The page size must match the type of memory device used, as well as the access time. Page mode increases performance by reducing access time by allowing the CPU to use less wait states for accesses in the same page as the previous access. This allows 14 of 16 bytes to be read from the same page with the lower access times.

Burst-mode flash devices can also be used to decrease access times. Most burst-mode flash use a page size of 32 words. This allows 62 of 64 bytes to be accessed using decreased access times.

Note: *Burst mode accesses must be sequential; otherwise, a miss is detected. Burst mode also uses page-size detection (typically 32 words).*

Since code typically executes sequentially, burst mode works well with DSTni. So well, in fact, that decreases in access times of 65% are common. Burst-mode flash requires 3 additional signals over Page-mode and regular flash:

- BCLK (Burst Clock)
- LBA# (Load Burst Address)
- BAA# (Burst Address Advance)

To activate these signals, set the Burst Enable bit (bit [13]) to 1.

Note: *This register works only with flash connected to upper chip select. Note, too, that burst-mode or page-mode devices may require activation in addition to programming the Page Chip Select register. For example, AMD burst-mode flash devices require a specific write sequence to be performed to the device to place it into burst mode once the Page Chip Select register has been set up.*

Table 0-32. Page Chip Select Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	AAh															
FIELD	PAGE ENABLE	ENHANCED PAGING	BURST ENABLE	ENHANCED BUS	///					PAGE SIZE		R3	R2	ER	R1	R0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	R W	RW	R W	R W	RW	RW	RW	R W	R W	R W	RW	R W	R W	R W	R W	R W

Table 0-33. Page Chip Select Register Definitions

Bits	Field Name	Description
15	PAGE ENABLE	**Page Mode** 1 = enable page mode flash on the UCS to allow page hits and increase performance. When a page hit occurs, the wait states in this register are used. A page miss uses the wait state value in the UMCS register. 0 = disable page mode.
14	ENHANCED PAGING	**Enhanced Paging** 1 = enable the UCS to stay active after the current memory cycle ends. By holding UCS active, any memory cycles separated by bus inactivity are page or burst hits instead of misses. Any bus cycles other than to UCS causes UCS to go inactive. This forces a page or burst miss on the next UCS access, even if it would otherwise be a hit. 0 = UCS goes inactive when a bus cycle is not to the memory addressed by UCS (this includes bus inactive).
13	BURST ENABLE	**Burst Enable** 1 = enable burst mode flash on the UCS to allow burst hits and increase performance. When a burst hit occurs, the wait states in this register are used. A burst miss uses the wait state value in the UMCS register. 0 = disable burst mode.
12	ENHANCED BUS	**Enhanced Bus** This bit should be set with the Enhanced Paging bit (bit [14]). 1 = modify the Enhanced Paging bit to ignore all bus cycles not to the external bus. With this setting, internal SRAM accesses will not cause UCS to go inactive. Therefore, the UCS stays active after the current memory cycle ends. By holding UCS active any memory cycles separated by bus inactivity or SRAM accesses are page or burst hits instead of misses. Any bus cycles other than to UCS or LCS cause UCS to go inactive. This forces a page or burst miss on the next UCS access, even if it would otherwise be a hit. See Note below table. 0 = disable enhanced bus mode.
11:7	///	**Reserved**
6:5	PAGE SIZE	**Page Size** Determine the page or burst size of the external flash connected to the UCS pin. Typical page size is 8 and burst size is 32. See Table 0-34.
4	R3	**Ready Generation** Used by ready generation logic to complete a bus cycle. See Wait State Generation on page 31.
3	R2	**Ready Generation** Used by ready generation logic to complete a bus cycle. See Wait State Generation on page 31.
2	ER	0 = wait state generation logic also uses external ready for wait state generation. 1 = ignore external ready signal. See Wait State Generation on page 31.
1	R1	**Ready Generation** Used by ready generation logic to complete a bus cycle. See Wait State Generation on page 31.
0	R0	**Ready Generation** Used by ready generation logic to complete a bus cycle. See Wait State Generation on page 31.

Note: *When setting Page or Burst mode, set only one enable bit at a time. One or both enhanced modes can be set. It is assumed that both enhanced bits can be set to increase flash hit rates. However, in configurations where MCS contains memory or where SRAM is external to DSTni, some bits may need to be OFF.*

Table 0-34. Page or Burst Size Values

Page Size		Page or Burst Size
Bit [6]	Bit [5]	
0	0	4
0	1	8
1	0	16
1	1	32

PCS and MCS Auxiliary Register

Table 0-35. PCS and MCS Auxiliary Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A8h															
FIELD	M[7:0]								EX	MS	SM	R3	R2	ER	R1	R0
RESET	1	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW

Table 0-36. PCS and MCS Auxiliary Register Definitions

Bits	Field Name	Description
15:8	M[7:0]	**MCS Block Size** Total block size for the MCS3-MCS0 chip selects. Each chip select is active for ¼ of the total block size. If the MCSBIT in the SYSCON register is set, MCS0 asserts over the entire programmed block size. MCS3-MCSa is still asserted over their programmed range. See Table 0-37.
7	EX	This Read Only bit is always 1.
6	MS	**Peripheral Block Mapping** 1 = peripheral block maps into memory. 0 = peripheral block maps into I/O space.
5	SM	1 = the device assigns Lower Chip Select (LCS) to the internal memory and MCS1 to an external pin. 0 = the device swaps the internal memory LCS and MCS1 pin functions with external memory on MCS1_n. The programming still is programmed by the corresponding pin. Therefore, LCS is always set to 256K if internal or external.
4	R3	**Ready Generation** Used by the ready-generation logic to complete a bus cycle. Applies to PCS4 to 7 only. Works with bits [3], [1], and [0].
3	R2	**Ready Generation** Used by the ready-generation logic to complete a bus cycle. Applies to PCS4 to 7 only. Works with bits [4], [1], and [0].
2	ER	**External Ready** 0 = wait-state generation logic uses external ready for wait-state generation. 1 = external ready signal is ignored. Applies to PCS4 to 7 only. See Wait State Generation on page 31.
1	R1	**Ready Generation** Used by the ready-generation logic to complete a bus cycle. Applies to PCS4 to 7 only. Works with bits [4], [3], and [0].
0	R0	**Ready Generation** Used by the ready-generation logic to complete a bus cycle. Applies to PCS4 to 7 only. Works with bits [4], [3], and [1].

Table 0-37. Block Sizes

Total Block Size		Individual Chip Select Size		Programmed MPCS[15:8] Bit Values							
20-Bit	24-Bit	20-Bit Mode	24-Bit Mode	Bit [15]	Bit [14]	Bit [13]	Bit [12]	Bit [11]	Bit [10]	Bit [9]	Bit [8]
None	None	None	None	1	0	0	0	0	0	0	0
8K	128K	2K	32K	1	0	0	0	0	0	0	1
16K	256K	4K	64K	1	0	0	0	0	0	1	x
32K	512K	8K	128K	1	0	0	0	0	1	x	x
64K	1M	16K	256K	1	0	0	0	1	x	x	x
128K	2M	32K	512K	1	0	0	1	x	x	x	x
256K	4M	64K	1M	1	0	1	x	x	x	x	x
512K	8M	128K	2M	1	1	x	x	x	x	x	x
1M	16M	256K	4M	0	x	x	x	x	x	x	x

Midrange Memory Chip Select Register

Table 0-38. Midrange Memory Chip Select Register (20-Bit Mode)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A6h															
FIELD	A[19:13]							///				R3	R2	ER	R1	R0
RESET	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	RW	RW	RW	RW	RW

Table 0-39. Midrange Memory Chip Select Register (24-Bit Mode)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A6h															
FIELD	A[23:17]							///	///	///	///	R3	R2	ER	R1	R0
RESET	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	R	R	R	R	RW	RW	RW	RW	RW

Table 0-40. Midrange Memory Chip Select Register Definitions

Bits	Field Name	Description
15:9	A[19:13]	In 20-bit mode, these bits define the starting address of the midrange memory chip select signals.
	A[23:17]	In 24-bit mode, these bits define the starting address of the midrange memory chip select signals.
8:5	///	**Reserved**
4	R3	**Wait State Value** Works with bits [3], [1], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
3	R2	**Wait State Value** Works with bits [4], [1], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
2	ER	**External Ready** 0 = wait-state generation logic uses external ready for wait-state generation. 1 = external ready signal is ignored. See Wait State Generation on page 31.
1	R1	**Wait State Value** Works with bits [4], [3], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
0	R0	**Wait State Value** Works with bits [4], [3], and [1] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).

Peripheral Chip Select Register

Table 0-41. Peripheral Chip Select Register (20-Bit Mode)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A4h															
FIELD	A[19:11]									///			R2	ER	R1	R0
RESET	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 0-42. Peripheral Chip Select Register (24-Bit Mode)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A4h															
FIELD	A[23:12]												R2	ER	R1	R0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 0-43. Peripheral Chip Select Register Definitions

Bits	Field Name	Description
15:7	A[19:11]	In 20-bit mode, these bits define the starting address of the peripheral chip select signals. Available wait states are limited in this mode.
15:4	A[23:12]	In 24-bit mode, these bits define the starting address of the peripheral chip select signals. Available wait states are limited in this mode.
6:4	///	**Reserved (20-Bit Mode)**
3	R2	**Wait State Value** Works with bits [1] and [0] to specify the number of wait states used during a bus cycle (see Table 0-45).
2	ER	**External Ready** 0 = wait-state generation logic uses external ready for wait-state generation. 1 = external ready signal is ignored. Applies to PCS0 to 3 only. PCS4 to 7 are set in the PCS and MCS Auxiliary register (see page 34).
1	R1	**Wait State Value** Works with bits [3] and [0] to specify the number of wait states used during a bus cycle (see Table 0-45).
0	R0	**Wait State Value** Works with bits [3] and [1] to specify the number of wait states used during a bus cycle (see Table 0-45).

Table 0-44. PCS Address Ranges

PCS Signal	Active Addresses (20-Bit Mode)	Active Addresses (24-Bit Mode)
PCS0	PBA through PBA + 255	PBA through PBA + 511
PCS1	PBA +256 through PBA +511	PBA +512 through PBA +1023
PCS2	PBA +512 through PBA +767	PBA +1024 through PBA +1535
PCS3	PBA +768 through PBA +1023	PBA +1536 through PBA +2047
PCS4	PBA +1024 through PBA +1279	PBA +2048 through PBA +2559
PCS5	PBA +1280 through PBA +1535	PBA +2560 through PBA +3071
PCS6	PBA +1536 through PBA +1791	PBA +3072 through PBA +3583
PCS7	PBA +1792 through PBA +2047	PBA +3584 through PBA +4095

Table 0-45. Wait State Generation

R2	R1	R0	Number of Wait States Generated
0	0	0	0 wait states
0	0	1	1 wait state
0	1	0	2 wait states
0	1	1	3 wait states
1	0	0	5 wait states
1	0	1	7 wait states
1	1	0	9 wait states
1	1	1	15 wait states

Lower Memory Chip Select Register

The Lower Memory Chip Select register is decoded with the highest priority over all other memory accesses. This register is hard-coded for 256K bytes, starting at address 0 in both 20- and 24-bit modes. Bit R0 is Read/Write. All other bits are Read Only.

Table 0-46. Lower Memory Chip Select Register (20-Bit Mode)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A2h															
FIELD	///	A[18:16]			///							R3	R2	ER	R1	R0
RESET	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0
RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R W

Table 0-47. Lower Memory Chip Select Register (24-Bit Mode)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A2h															
FIELD	///	A[22:16]							///			R3	R2	ER	R1	R0
RESET	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
RW	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R W

Table 0-48. Lower Memory Chip Select Register Definitions

Bits	Field Name	Description
15	///	**Reserved**
14:12	A[18:16]	**Starting Address Lower Memory Chip Select Signal (20-bit Mode)** In 20-bit mode, these bits define the starting address of the lower memory chip select signal.
14:8	A[22:16]	**Starting Address Lower Memory Chip Select Signal (24-bit Mode)** In 24-bit mode, these bits define the starting address of the lower memory chip select signal.
11:5	///	**Reserved (20-Bit Mode)**
7:5	///	**Reserved (24-Bit Mode)**
4	R3	**Wait State Value** Works with bits [3], [1], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
3	R2	**Wait State Value** Works with bits [4], [1], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
2	ER	**External Ready** 0 = wait-state generation logic uses external ready for wait-state generation. 1 = external ready signal is ignored. See Wait State Generation on page 31.
1	R1	**Wait State Value** Works with bits [4], [3], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
0	R0	**Wait State Value** Works with bits [4], [3], and [1] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).

Upper Memory Chip Select Register

The upper memory chip select signal decodes a programmable region of upper memory downwards. The upper limit of this chip select is set to:

- FFFFFh (20-bit address mode)
- FFFFFFh (24-bit address mode)

The lower limit or size of the block is programmable and legal block sizes are shown in the tables below.

An internally generated address whose value is greater than the starting base address encoded in this register generates an upper memory chip-select. Upper memory chip select is the only chip select that is active after reset.

The internal Boot ROM is always in the upper 16K bytes of the memory space. The Boot ROM has priority and will be accessed instead of the upper memory chip select if enabled. Resetting the Boot ROM Enable bit (bit [14]) in the DSTni Configuration Register (see page 19) disables the internal Boot ROM. If the Boot ROM is disabled, the memory connected to the upper memory chip select is accessed in the Boot ROM address space.

If access for memory priority overlap, the priority is as follows:

Lower memory chip select and Boot Rom	(no overlap)	Highest Priority
Upper memory chip select		
Midrange memory chip select		Lowest Priority

Table 0-49. Upper Memory Chip Select Register (20-Bit Mode)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A0h															
FIELD	A[19:16]				///							R3	R2	ER	R1	R0
RESET	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1
RW	RW	RW	R W	RW	R	R	R	R	R	R	R	R W	R W	R W	R W	R W

Table 0-50. Upper Memory Chip Select Register (24-Bit Mode)

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	A0h															
FIELD	A[23:16]								///	///	///	R3	R2	ER	R1	R0
RESET	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
RW	RW	RW	R W	RW	RW	RW	R W	R W	R	R	R	R W	R W	R W	R W	R W

Table 0-51. Upper Memory Chip Select Register Definitions

Bits	Field Name	Description
15:12	A[19:16]	In 20-bit mode, these bits define the size of the upper memory chip select signal.
15:8	A[23:16]	In 24-bit mode, these bits define the size of the upper memory chip select signal.
11:5	///	**Reserved (20-Bit Mode)**
7:5	///	**Reserved (24-Bit Mode)**
4	R3	**Wait State Value** Works with bits [3], [1], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
3	R2	**Wait State Value** Works with bits [4], [1], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
2	ER	**External Ready** 0 = wait-state generation logic uses external ready for wait-state generation. 1 = external ready signal is ignored. See Wait State Generation on page 31.
1	R1	**Wait State Value** Works with bits [4], [3], and [0] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).
0	R0	**Wait State Value** Works with bits [4], [3], and [1] to specify the number of wait states used during a bus cycle (see Wait State Generation on page 31).

Table 0-52. Example of Upper Memory Chip Select Settings

Starting Base Address	Memory Size Block	Programmed Upper Memory Chip Select Register Value (R(3:0) = 1111b, ER = 1)
20-Bit Mode		
*F0000h	64K	1111,0000,0011,1111b
E0000h	128K	1110,0000,0011,1111b
C0000h	256K	1100,0000,0011,1111b
80000h	512K	1000,0000,0011,1111b
00000h	1M	0000,0000,0011,1111b
24-Bit Mode		
*FF0000h	64K	1111,1111,0011,1111b
FE0000h	128K	1111,1110,0011,1111b
FC0000h	256K	1111,1100,0011,1111b
F80000h	512K	1111,1000,0011,1111b
F00000h	1M	1111,0000,0011,1111b
E00000h	2M	1110,0000,0011,1111b
C00000h	4M	1100,0000,0011,1111b
800000h	8M	1000,0000,0011,1111b
000000h	16M	0000,0000,0011,1111b

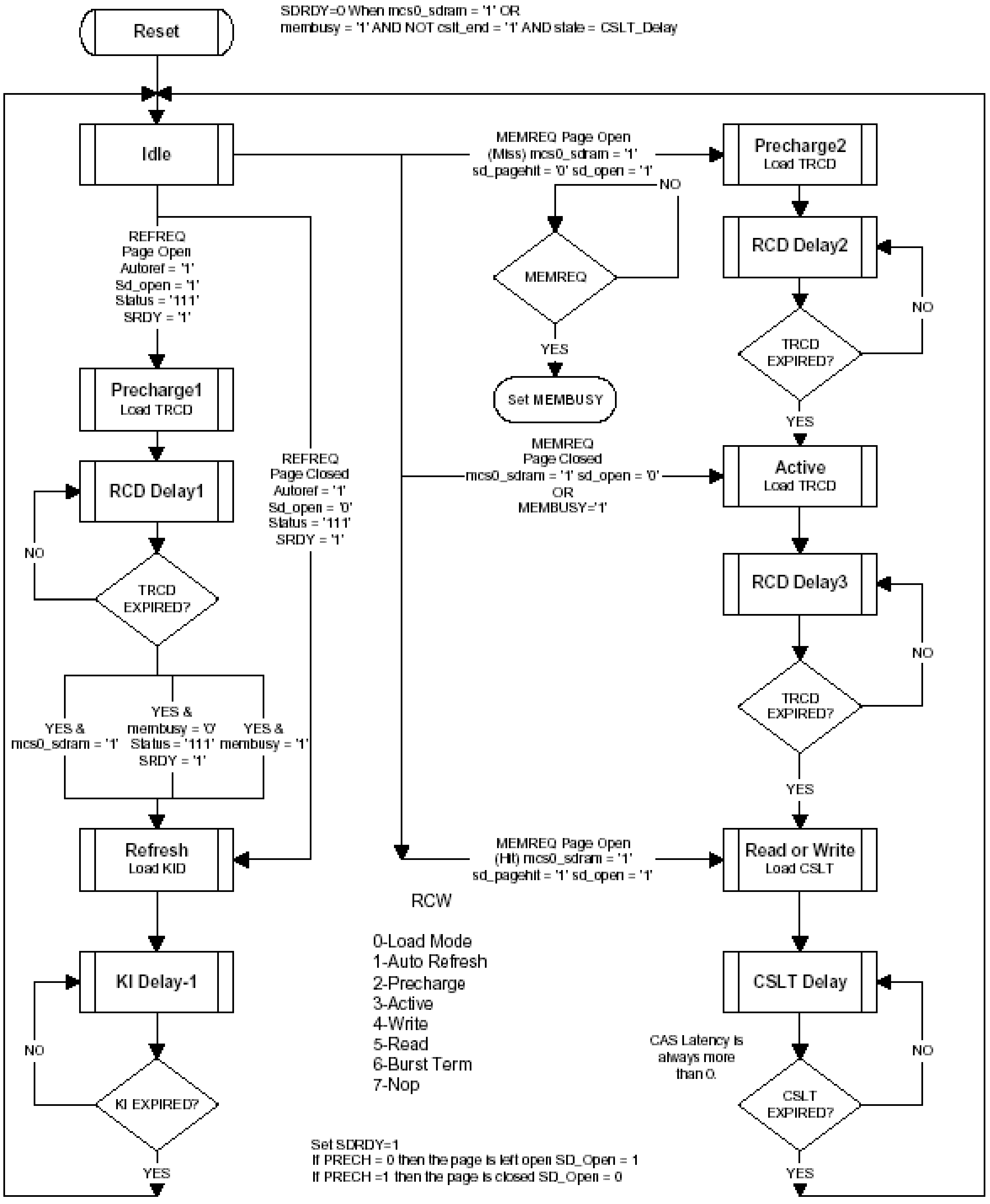
***Notes:** These are the only values legal for the block sizes.*
** Default values 3 waits, no external ready.*

SDRAM

This chapter describes the DSTni SDRAM. Topics in this chapter include:

SDRAM Flow Diagram

Figure 0-1. SDRAM Flow Diagram



Reset
SDRDY=0 When mcs0_sdram = '1' OR membusy = '1' AND NOT cslt_end = '1' AND state = CSLT_Delay
Idle
MEMREQ Page Open (Miss) mcs0_sdram = '1' sd_pagehit = '0' sd_open = '1'
Precharge2 Load TRCD
NO
REFREQ Page Open Autoref = '1' Sd_open = '1' Status = '111' SRDY = '1'
MEMREQ
RCD Delay2
NO
TRCD EXPIRED?
YES
Set MEMBUSY
YES
Precharge1 Load TRCD
MEMREQ Page Closed mcs0_sdram = '1' sd_open = '0' OR MEMBUSY='1'
Active Load TRCD
RCD Delay1
REFREQ Page Closed Autoref = '1' Sd_open = '0' Status = '111' SRDY = '1'
NO
TRCD EXPIRED?
RCD Delay3
NO
TRCD EXPIRED?
YES & mcs0_sdram = '1'
YES & membusy = '0' Status = '111' SRDY = '1'
YES & membusy = '1'
YES
Refresh Load KID
MEMREQ Page Open (Hit) mcs0_sdram = '1' sd_pagehit = '1' sd_open = '1'
Read or Write Load CSLT
RCW
0-Load Mode
1-Auto Refresh
2-Precharge
3-Active
4-Write
5-Read
6-Burst Term
7-Nop
KI Delay-1
CSLT Delay
NO
KI EXPIRED?
CAS Latency is always more than 0.
NO
CSLT EXPIRED?
YES
Set SDRDY=1
If PRECH = 0 then the page is left open SD_Open = 1
If PRECH =1 then the page is closed SD_Open = 0
YES

SDRAM Operation

Table 0-1 summarizes SDRAM operation information.

Table 0-1. SDRAM Operation (16/64/128 MB)

SDRAM	NOP	Active	Read	Write	Precharge	Refresh	Load MR	Address Line ASIC
RAS	H	L	H	H	L	L	L	A4
CAS	H	H	L	L	H	L	L	A5
WE	H	H	H	L	L	H	L	A6
DQML	X	X	L	L/H	X	X	X	A7
DQMH	X	X	L	L/H	X	X	X	A8
A0	X	A9	A1	A1	X	X	L	A9
A1	X	A10	A2	A2	X	X	L	A10
A2	X	A11	A3	A3	X	X	L	A11
A3	X	A12	A4	A4	X	X	L	A12
A4	X	A13	A5	A5	X	X	L/H	A13
A5	X	A14	A6	A6	X	X	L/H	A14
A6	X	A15	A7	A7	X	X	L	A15
A7	X	A16	A8	A8	X	X	L	A16
A8	X	A17	A23	A8	X	X	L	A17
A9	X	A18	L	A23	X	X	L	A18
A10	X	A19	APRE	L	X	X	L	A19
BA/BA0	X	A20	A20	APRE	X	X	L	A20
NC/A11	X	A21	L	A20	X	X	L	A21
NC/BA1	X	A22	A22	L	X	X	X	A22
NC	0	0	0	A22	0	0	0	A23

Initializing SDRAM

The SDRAM must be initialized before it can be enabled and used. To initialize the SDRAM, perform the following steps with the SDRAM disabled. Memory accesses can either be a read or write. Data is ignored during this procedure.

4. Wait 100 us.
5. Set the refresh interval to 15.6 us (374 @24 MHz). See SDRAM Refresh Max Count Register on page 47.
6. Configure MCS0 for the appropriate memory size, starting address with wait states of 0, and no external ready. Valid memory sizes are:
   - 2 Mbyte (16 Mbit part)
   - 8 Mbyte (64 Mbit part)
   - 16 Mbyte (128 Mbit part)

See Midrange Memory Chip Select Register on page 35 and PCS and MCS Auxiliary Register on page 34.

7. Issue a NOP (access memory at x000070h).
8. Issue a precharge to all banks (access memory at x080020h).
9. Issue a refresh (access memory at x000040h).
10. Issue another refresh (access memory at x000040h).
11. Issue a Mode register load (access memory as indicated by Table 0-2).

Table 0-2. Addresses and Corresponding CAS Latencies

Address	CAS Latency
x000000h	0
x002000h	1
x004000h	2
x006000h	3

The SDRAM can now be enabled and refresh will begin.

SDRAM Register Summary

Table 0-3. SDRAM Registers

Hex Offset	Register Description	Page
64	Refresh Maxcount register	47
68	SDRAM Control register	49

SDRAM Register Definitions

SDRAM Refresh Max Count Register

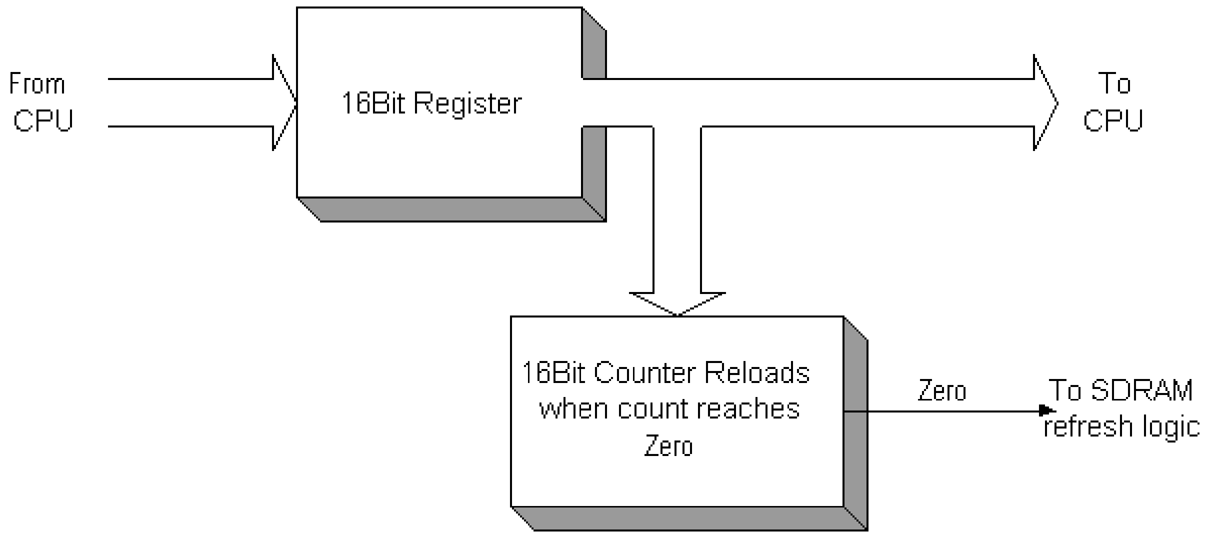
Table 0-4. SDRAM Refresh Max Count Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	64h															
FIELD	REFMAX [15:0]															
RESET	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 0-5. SDRAM Refresh Max Count Register Definitions

Bits	Field Name	Description
15:0	REFMAX [15:0]	**Refresh Reload Value** This register contains the value used to reload the 16-bit Refresh Count register. A refresh is attempted when the count decrements to zero and is reloaded with this value. The processor can read the value of this register; however, the decrementing value in this register cannot be read. The reset value of 0000,0001,0111,0110b corresponds to 15.6 us@24 MHz.

Figure 0-2. SDRAM Refresh Diagram



Note: The zero output from this counter is latched and is only cleared after a refresh is performed. This maintains the refresh interval even if the refresh is delayed by a large number of clocks.

SDRAM Control Register

Table 0-6. SDRAM Control Register

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET	68h															
FIELD	SDRAMEN	APRE	///						REFD				RCD		CSLT	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Table 0-7. SDRAM Control Register Definitions

Bits	Field Name	Description
15	SDRAMEN	**SDRAM Enable** 1 = enables the DSTni-EX SDRAM logic. When enabled the internal logic performs necessary functions using the address bus to access the memory and perform refresh cycles to the SDRAM using unused bus cycles. It assumes that one 16, 64, or 128 Mbit SDRAM is connected to MCS0_n pin. The SDRAM must be connected as 16 bits wide. 0 = the SDRAM logic is disabled and MCS0_n can be used for other functions; no refreshes will be performed.
14	APRE	**Auto-Precharge Mode** 1 = enables auto-precharge mode. This bit forces the internal state machine to assume the SDRAM pages are always closed after each page access. 0 = the SDRAM logic assumes the last page is open after each access and allow memory page hits. This reduces page-cycle time by many clocks. SDRAM refresh always closes any page that is open before refreshing and then performs an autorefresh. Some applications perform better with pages closed after access, while others perform better with pages left open. It is up to the user to make this design decision.
13:8	///	**Reserved**
7:4	REFD	**Refresh Recovery Period** These bits are the refresh recovery period. They hold off the internal state machine until this number of clocks has been counted after a refresh. Valid numbers are 0 to 15. Typical numbers are 70 ns. The correct value depends on CPU clock speed.
3:2	RCD	**RAS-to-CAS Delay** These bits hold off the internal state machine until this number of clocks has been counted. When a RAS cycle to is started, this many clocks must pass before CAS can be issued. Valid numbers are 0, 1, 2, or 3. Typical numbers are 20 ns. The correct value depends on the CPU clock speed.
1:0	CLST	**CAS Latency** These bits indicate how many clocks to count before the SDRAM is to provide data. This number must match the SDRAM minus 2. Valid number are 0, 1, 2, or 3. SDRAM CAS Latency: 2, 3 CLST: 0, 1