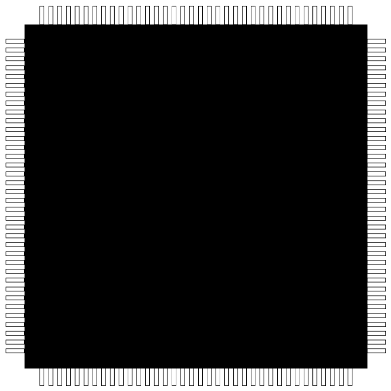
DSTni-LX Data Book



Revision F

Part Number GC-900-252

Copyright and Trademark

Grid Connect

1630 W. Diehl Rd.
Naperville, IL 60563
Phone: 630.245.1445

Technical Support

Phone: 630.245.1445
Fax: 630.245.1717
On-line: www.gridconnect.com

Revision	Date	Author	Comments
A	08-28-01	GR	Initial Release
B	09-30-01	AC	Cosmetic changes and copyright page.
C	10-15-01	AC	Removed Glossary and Index, edited block diagram, Baud Rate table and made cosmetic changes throughout.
D	11-27-01	GR	Replaced Index. Revised format.
E	2-18-02	GR	Ready is synchronous.
	12/14/2005 1:58 PM		Last edited and saved. New Format
F	12-14-05	GR	Revised tables. Added notes

Table of Contents

List of Figures

List of Tables

List of Registers

1 Introduction

1.1 Purpose of This Manual

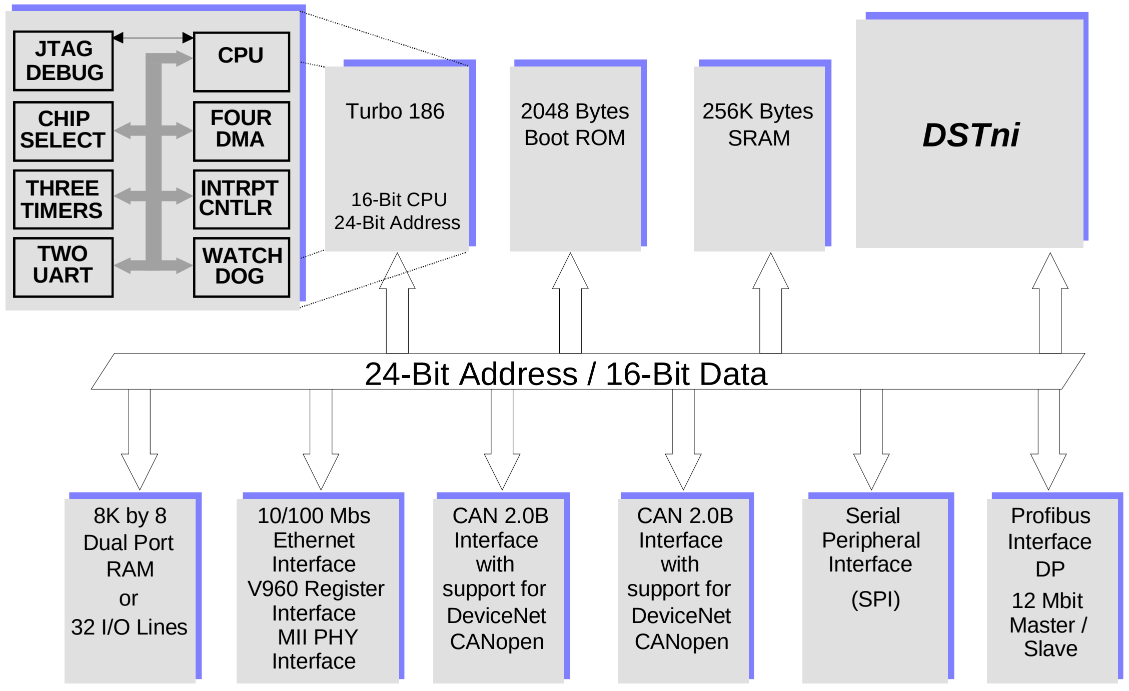
This manual describes the technical features and programming interface of the DSTni-LX-001/-002 Communications Single Chip Solution (SCS) device. This manual is intended for hardware engineers, software engineers, and system integrators who are considering using the DSTni-LX in their designs.

1.2 Manual Overview

The information in this manual is organized into the following chapters:

1.3 General Description

The DSTni-LX contains all of the necessary functions to support the most requested embedded communication technologies. This device provides Intel 80186 compatibility to existing software, while providing 12.5 MIPS of processor performance as a platform that is capable of handling the most demanding embedded communication tasks. Designed as a synchronous device providing multiple networks, the DSTni-LX has on board support for multiple CAN channels, Profibus (DSTni-LX-002), Ethernet, RS232, RS422 & RS485. To provide the highest performance possible, the DSTni-LX has on-board 256K Bytes of high-speed (10ns) SRAM, 8Kx8 dual port memory for co-processor applications and an SPI interface for communications to serial flash or other SPI networking applications.


JTAG DEBUG
CPU
CHIP SELECT
FOUR DMA
THREE TIMERS
INTRPT CNTLR
TWO UART
WATCH DOG
Turbo 186
16-Bit CPU
24-Bit Address
2048 Bytes Boot ROM
256K Bytes SRAM
DSTni
24-Bit Address / 16-Bit Data
8K by 8 Dual Port RAM or 32 I/O Lines
10/100 Mbs Ethernet Interface V960 Register Interface MII PHY Interface
CAN 2.0B Interface with support for DeviceNet CANopen
CAN 2.0B Interface with support for DeviceNet CANopen
Serial Peripheral Interface (SPI)
Profibus Interface DP 12 Mbit Master / Slave

**** Profibus interface is available in DSTni-LX-002***

Figure 1 - DSTni-LX Block Diagram

1.4 Key Features & Benefits

- Intel compatible 80186 16-bit microprocessor with on-chip peripherals operating at 48Mhz and providing 12+ MIPS of high-speed performance
- AMD 960 compatible 10/100 Mbps Ethernet controller with MII PHY interface
- Two CAN V2.0B 1 Mbps fully compatible with the Bosch specification.
- Profibus DP master/slave controller (100% Siemens ASPC2 compatible)(DSTni-LX-002 only)
- Two asynchronous RS232/422/485 serial channels, RTS/CTS and DMA support
- 256K bytes of SRAM memory, 2K bytes of boot ROM
- 8K bytes of dual port memory with interrupt and semaphore support
- Integrated DMA, interrupt controller, timers, Watchdog timer, and memory select logic
- External 16M byte addressable memory space (24-bit mode)
- 32 I/O pins or dual port memory interface
- 160-Pin LQFP package, 180BGA Package
- JTAG interface with in-circuit emulator support for breakpoints and trace buffer
- Industrial temperature specifications

1.5 Hardware Description

DSTni-LX features a high performance (12.5 MIPS) Intel 80186 compatible device operating at 48Mhz.

This chip combines a powerful microprocessor with on-chip memory and all required communication peripherals. For simple stand-alone applications, the DSTni-LX can be implemented as the primary microprocessor in the system and simply interfaced to required external components (see

Figure 2 - Typical DSTni-LX System Block Diagram). In more complex multi-processor applications, the DSTni-LX is applied as a co-processor and uses an on board 8k x 8 dual-port memory for two-way communications between the host processor and the DSTni-LX microprocessor.

Required peripherals of the DSTni-LX include an external serial flash device or parallel non-volatile memory (flash or EPROM) to load the 256K of internal RAM memory, a crystal/oscillator and any required transceiver components for the supported communication controllers. If serial or parallel Flash are not used, the serial port (SP0) can be used to download programs to the internal RAM.

The DSTni-LX is designed to use standard x86 development tools and operating systems for ease of development and to leverage existing software source code. Having been designed to be an effective solution for various communications technologies that are used in industrial communications, the DSTni-LX is ideal for medical electronics (see warning below), vehicle electronics, building control, embedded communications or web servers, protocol converters, and other intelligent commercial and industrial applications.

WARNING:

The DSTni-LX should NOT be used as a critical component in life-support devices or systems.

1.5.1 High Performance 16-Bit Microprocessor

The DSTni-LX has an on-chip 80186 compatible, 16-bit microprocessor with its related peripherals (interrupt controller, timer, chip select logic). This microprocessor has additional features to support a

24-bit address space, which allows up to 16M bytes of memory to be used. Internally, the DSTni-LX has 16-bit wide, 256K bytes of SRAM memory (20-bit mode), for maximum performance.

The turbo 186 microprocessor is the heart of the DSTni-LX controller. The microprocessor executes the software or firmware loaded in internal or external memory. The microprocessor instruction set is software compatible with the Intel/AMD 80186 and 80188 microprocessors and uses standard compilers and development tools for Intel x86 microprocessors. This allows the DSTni-LX to be fully software compatible with 8086, 8088, 80186, and 80188 family of processors. Normally, an 80186-48Mhz microprocessor executes instructions between 4-12 clock cycles. The DSTni-LX microprocessor has been enhanced to execute instructions in 1-4 clock cycles, thus giving 12.5 MIPS of performance.

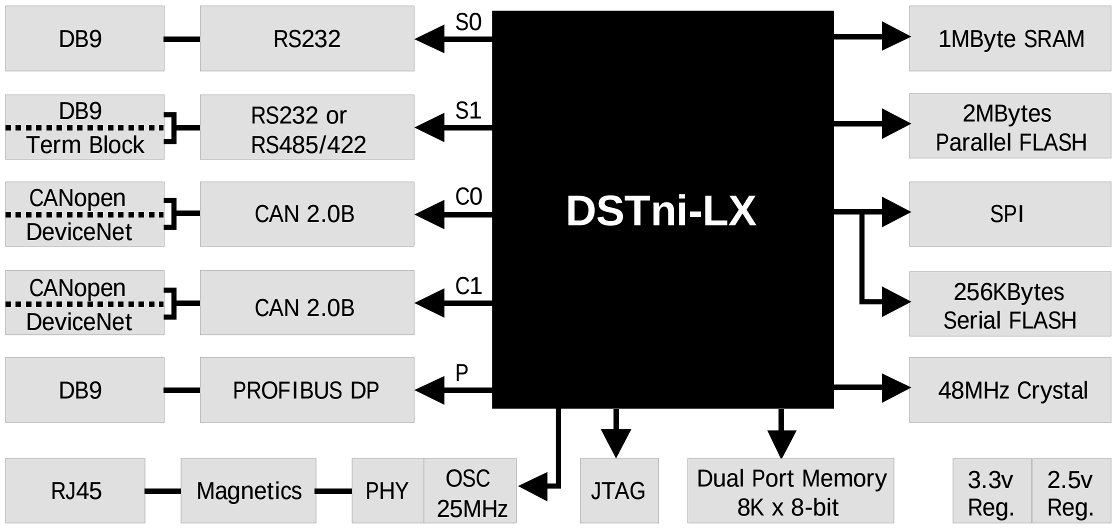


Figure 2 - Typical DSTni-LX System Block Diagram

1.5.2 Programmable Timers

The integrated timers provide a software vehicle to count or time external events. Each timer is equipped with a max count register, which defines the maximum count the timer will reach. Timer 0 and 1 each have two max counts registers. The programmer has the option of either normal timing functionality or when enabled, these timers can be defined to alternate. Timer 0 has an output that can be used to generate waveform signals with programmable duty cycles.

1.5.3 Asynchronous Serial Ports

The DSTni-LX provides two asynchronous serial ports. These ports provide a read/write port for full duplex operation with programmable baud rate up to 230400. Any combination of the following are supported:

- 7 or 8-bit data transfers
- 1 or 2 stop bits
- Even, odd or no parity

The receive portion of the serial ports provides break character recognition and error detection for frame, parity and overrun errors. In addition, they are programmable to generate interrupts whenever one of these error conditions is detected. Interrupts can be generated by the serial port to signal that the transmitter is available to accept a new character or that valid data has been received. Both of the serial channels support RTS/CTS control signals and DMA control. An additional register has been added to support high-speed protocols and signaling for RS-422/485.

1.5.4 Interrupt Controller

The DSTni-LX integrated interrupt controller manages and prioritizes the internal and external interrupt sources. Internal interrupt sources include the Timers and DMA channels. These sources can be disabled by their own control register or by the mask bits within the interrupt controller.

1.5.5 SPI Controller

The Serial Peripheral Interface (SPI Controller) provides communication with external devices in master mode. The SPI also supports communication between processors if an external processor is connected to the system. The DSTni-LX uses the SPI port to load from serial flash all boot strap data to initiate operation.

1.5.6 Hardware Watchdog Timer

The DSTni-LX has been developed to provide the developer with a configurable Watch Dog Timer. This externally accessible pin (wdog_n) can be used to monitor the on board DSTni-LX CPU. The Watch Dog Timer (WDT) is a count-down counter that is reset via software at a programmable periodic rate. Should the WDT count down to zero it will reset the DSTni-LX CPU.

1.5.7 10/100 Mbps Ethernet Controller

The DSTni-LX has an Ethernet 10/100 Mbps controller with Media Independent Interface (MII), and is register compatible with the Advanced Micro Devices AM79C960. Supported protocols are: IEEE 802.3 and ANSI88023. Additionally, the DSTni-LX has a buffer for message receive and transmit.

1.5.8 Static RAM

The DSTni-LX additionally provides 256K bytes of on-chip zero wait state 10NS static RAM (20-bit mode). The bootstrap will zero this SRAM after reset. The static RAM is 16 bits wide internally, which allows for maximum microprocessor performance. The 160-pin package has been designed to bring out all data lines to access the full externally supplied 16M Bytes of memory (24-bit mode).

1.5.9 Profibus DP Master/Slave Controller

The DSTni-LX-002 has a fully licensed Siemens Profibus core providing master or slave capability. Profibus DP Master/ Slave is a European standard for industrial communications. It supports up to 12 Mbps baud rate, with up to 128 devices, and was designed to use Siemens configuration tools.

1.5.10 Boot ROM Memory

The DSTni-LX has on-chip Boot ROM of 2048 bytes, which supports loading of code from any of the following:

- Serial flash device
- Parallel non-volatile memory (flash or EPROM)
- Serial port (SP0)

1.5.11 Dual Port Memory Shares 32 I/O Pins

Each of the 32 user-programmable I/O pins can be configured for normal operation, as inputs or outputs. Several different registers are provided for I/O control. Specifically, the mode and direction registers are used to establish I/O characteristics of each pin. All 32 I/O pins contain either an internal pull-down or pull-up circuit regardless of the function selected. The DSTni-LX has 8K bytes of 8-bit wide dual port memory, which is functionally compatible with the Integrated Device (IDT) dual port memory IDT7005 chip. When enabled, the dual port memory is made available as shared pins on 27 of the 32 I/O pins. If the dual port memory is not required, the 27 I/O pins are available as user defined I/O.

1.5.12 Dual 1 Mbps CAN Controllers

The DSTni-LX provides two individual CAN-2.0B channels. These CAN channels have been designed to function with high levels of data integrity in extremely noisy electrical environments. Both channels are capable of high-speed (1 Mbits/s) data transmission over short distances (40 m) and low-speed (5 kbits/s) transmissions at lengths of up to 10,000 meters. The multi-master CAN bus is highly fault tolerant, with powerful error detection and handling. The DSTni-LX adds filtering, input and output queues, and complete diagnostics bits; which provides significant feature enhancements over competitive products.

1.6 Communication Protocols

1.6.1 CAN

Included among the onboard communications peripherals are two, totally independent CAN controllers. Both channels are 100% Version 2.0B compliant. The maximum Baud rate is 1Mbps. Each CAN channel is integrated with a simple 16-bit internal I/O interface. There are 62, 16 bit Read/Write registers for each channel that allow full access and control of all CAN functions. The DSTni-LX with an internal CAN interface results in a major performance increase over typical 8-bit external CAN interfaces. This performance increase is directly related to the 48Mhz system clock speed, which is a true 16-bit synchronous data access interface, with no external timing constraints that typically add wait states.

Another distinguishing feature of the DSTni-LX CAN interfaces are the transmit holding registers. Each channel contains 3 separate transmit holding registers, which integrate a “Priority Arbiter.” This provides a two-fold advantage.

1. This allows “queuing” of any outgoing CAN messages, which reduces processor loading of the application.
2. The “Priority Arbiter” allows you to override a pending lower priority message that is already in one of the transmit buffers.

This is extremely useful for CAN protocols that require “SYNC” messages to be sent at defined intervals. Included in the design is a “listen-only” mode for CAN diagnostic and/or bus analyzing. Additionally, inclusion of 2 CAN channels makes this device a perfect candidate for simultaneous control and diagnostics of a CAN bus. A typical example of this flexibility is the use of CAN channel 1 as an active master or slave while using CAN channel 2 in the listen only mode for analyzing all CAN bus activity.

These features allow efficient, modular application software to be written since each channel operates independently. Having separate interrupts also enhances modularity.

Listed below are some of the notable features for each DSTni-LX CAN channel:

Processor Interface

- 14, fully programmable interrupt sources
- Fully synchronous, 0-wait states, efficient 16-bit data interface
- Simple 16-bit I/O access
- Internal “Loop-Back” mode for diagnostics
- Control and Status registers

3 fully programmable ACR (Acceptance Code Reg.) and AMR (Acceptance Mask Reg.) filters

- Receive Message filter includes: Full 29- bit ID, IDE, RTR, and 16 data bits
- Separate enable bit for all 3 filters

Transmitter Features

- 3 separate Transmitter holding registers
- Transmitter priority arbiter
- Transmission Abort function (pending messages still in registers)

Receive Features

- 4 deep receive message (14 bytes per message) FIFO
- FIFO Level status indicator with programmable threshold

Special CAN Features

- Listen only mode
- Supports both 11-bit ID and 29-bit ID
- Arbitration lost capture Error register
- Error event capture that includes an actual frame reference pointer to pinpoint errors

1.6.2 Profibus

Profibus is a one of several vendor-independent, open field-bus communication networks supported by the DSTni-LX-002. Profibus allows communication between devices of several different manufacturers on one network that comply with the international standards of EN 50170 and EN 50254. Profibus can be used for both high-speed time critical applications and complex communication tasks. The networking technology of Profibus is based on serial field bus systems for networking of distributed Non-or Intelligent programmable devices. Operation to 12Mbits/second is fully supported.

The DSTni-LX-002 Profibus feature is a multi-master system that allows joint operation of several automation, engineering or visualization systems and peripherals on one bus. Commonly called Profibus DP, the node definitions are as follows:

Master

These devices control the data communication on the bus. A master can send messages without an external request when it holds the bus access rights, which is called a token. Masters are also called active stations.

Slave

These are peripherals such as I/O devices, valves, drives and measuring transducers. Slave devices do not have bus access rights and they can only acknowledge received messages or send messages to the master when requested.

The DSTni-LX-002 supports both I/O and complex data transfer using Profibus. The Profibus DP implementation on the DSTni-LX-002 has been highly optimized for speed, ease of development and low connection costs. The most typical applications are for communication between automation systems and distributed peripherals. DSTni-LX-002 Profibus DP is suitable as a replacement for conventional, parallel signal transmission operating at 24 VDC in manufacturing automation as well as for analog signal transmissions implementing 4 to 20 mA technologies found in process automation.

The DSTni-LX-002 also includes all of the latest feature sets of Profibus DP communications, which include both DPV-1 & 2 extensions.

1.6.3 Ethernet

Ethernet is a mature networking solution with hundreds of vendors providing hardware and software products. Presently, Ethernet is widely used as the network backbone for businesses, government and educational institutions. Most recently, the embedded application world has taken a new look at the deployment of Ethernet in other applications that have only been proprietary in the past. One such application is for Industrial Automation.

Since its original inception in the early 1980s, the Ethernet standard has been updated and expanded to support higher data rates and more predictable throughput. Ethernet is now available at speeds of 10 Mbps, 100 Mbps and 1000 Mbps (Gigabit), full duplex or half duplex operation; and provides a variety of bus technologies, such as twisted pair wire, fiber optic cable and wireless.

The DSTni-LX on-chip Ethernet controller provides a choice 10 Mbps or 100 Mbps data rates and either full or half duplex operation. The controller interface is software compatible with the Am79C960 Ethernet Controller from AMD.

The DSTni-LX Ethernet controller uses its Media Independent Interface (MII) to interface with an Ethernet transceiver (PHY), such as the Level One LTX971. The MII provides the means to set parameters in the PHY such as data rate, full or half duplex and loopback. This provides a raw Ethernet packet interface that allows developers to implement their own network protocols over Ethernet.

1.7 Development Tools

The development environment for the DSTni-LX is the standard x86 tools. One of the development kits contains an in-circuit emulator (ICE) that communicates with the DSTni-LX through the JTAG port. The ICE tools allows the developer to load software, set breakpoints, view trace buffer, and many other standard ICE functions. The Paradigm C compiler with a 24-Bit supported tool set is available.

1.7.1 Paradigm

Paradigm C++ features a powerful integrated development environment (IDE), which permits all the tools needed for software development to be completely integrated. Now all the commonly used tools such as editors, browsers, compilers, debuggers, and source version control are available in one place, taking x86 software development to a new level of productivity. Paradigm C++ operates on Windows 95/98, Windows NT 4.0, Windows 2000 or Windows ME.

1.7.2 FS^2 ICE

The VSA-8X/18X System Analyzer from FS^2 was designed to support the special features and integrated peripherals of the DSTni-LX product. Special "silicon hooks" were integrated into the DSTni-LX that support “On-Chip” extensions, which provides a powerful debug tool with advanced features.

The VSA-8X/18X debugger is contained in a compact chassis that connects to the target system using a standard 20-pin AMP debug connector. It requires access to only 4 pins in the core processor. The system runs on a Windows® 95/98/NT/2000 PC over an IEEE-1284 EPP/ECP high speed parallel port. An optional graphical, source level debugger program provides an intuitive, easy to use interface.

1.7.3 Developers Evaluation Kits

Three evaluation kits are available. They contains software and/or hardware components suitable for initial setup and development of an operating communication system.

Kits may include:

- FS2 In-Circuit Emulator (ICE) /Debugger
- Paradigm C++ Integrated Development Environment (IDE), including 20 and 24-bit data addressing, editors, browsers, compilers, assemblers, linkers, locaters, and debuggers.
- Real-time Operating System (RTOS)
- DSTni-LX Hardware/software documentation

Optional components: Please refer to the software stacks and other modules in the price list.

1.8 System Diagram

1.8.1 DSTni-LX Block Diagram

The major circuits of the DSTni-LX are shown in the following diagram. Profibus is available only on the DSTni-LX-002.

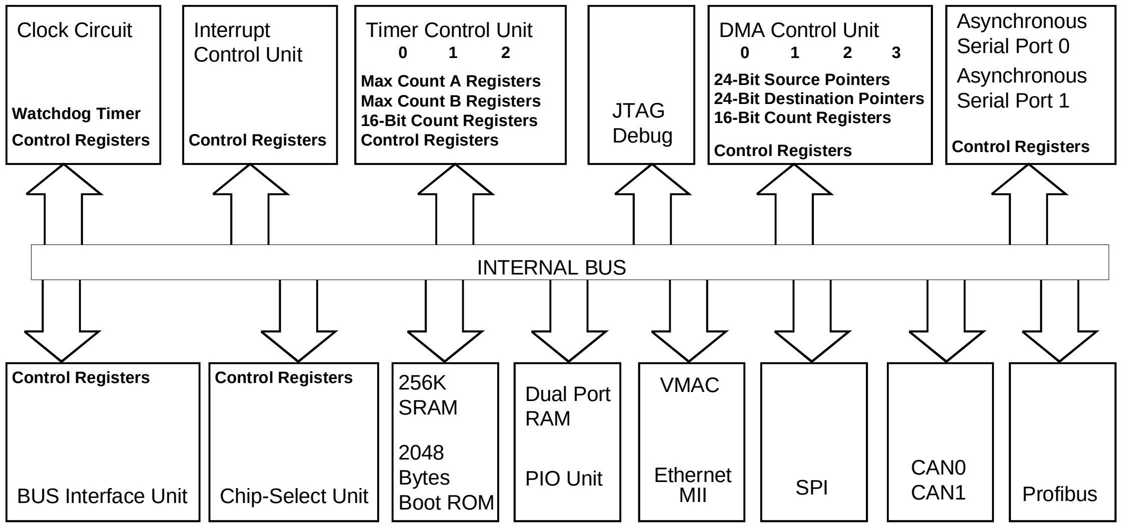


Figure 3 - Block Diagram: Major Circuits

The following diagrams show the major blocks with associated pins.

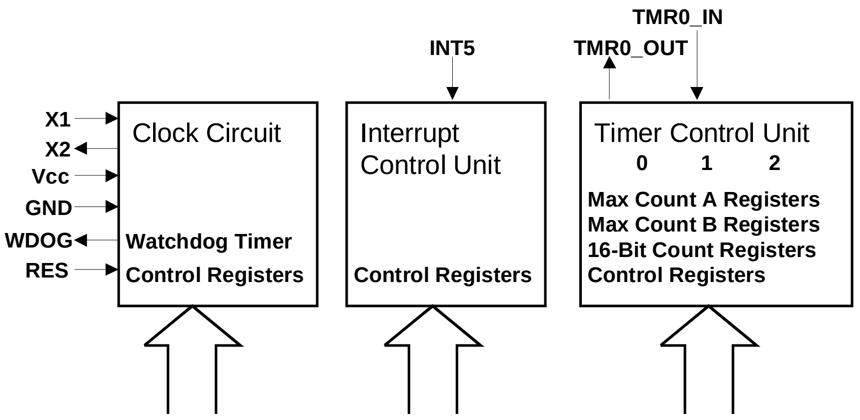


Figure 4 - Block Diagram: Clock, Power, Timers

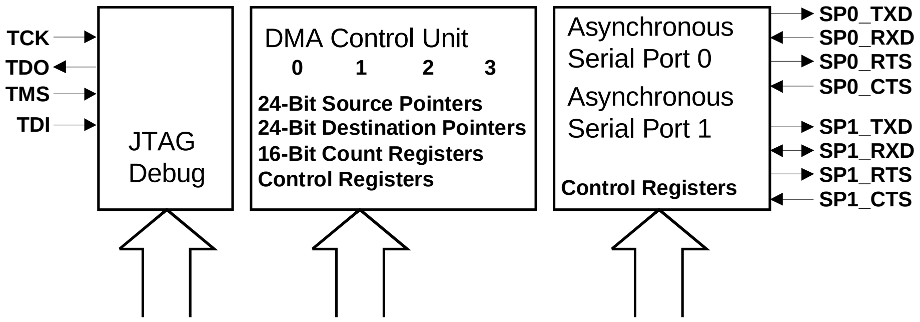


Figure 5 - Block Diagram: JTAG, DMA, Serial Ports

The four pins for Serial Port 1 are multiplexed to provide for support for the Profibus DPV2 Isochron Mode of Operation.

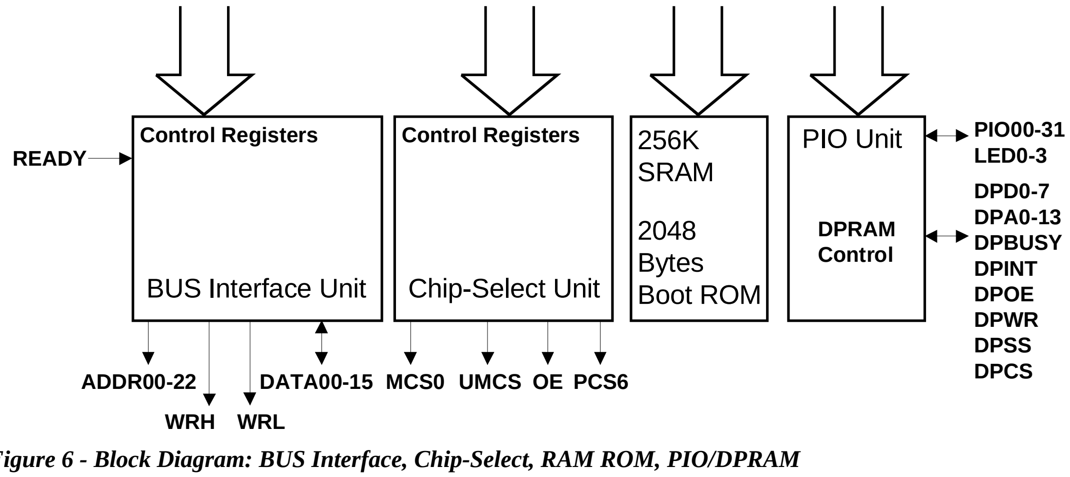


Figure 6 - Block Diagram: BUS Interface, Chip-Select, RAM ROM, PIO/DPRAM

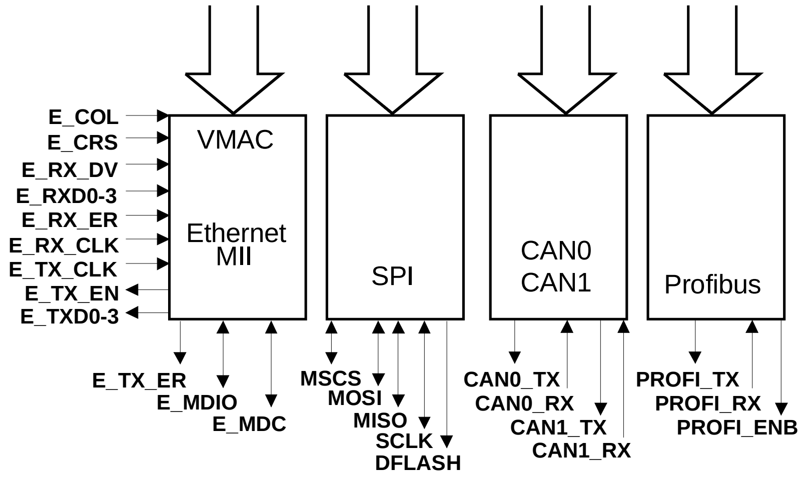


Figure 7 - Block Diagram: Ethernet, SPI, CAN, ProfiBus

Note: MSCS is used for SPI Master Collision Input. DFLASH is used for SPI Slave 0 Select or Serial Flash.

1.9 Programming

The DSTni-LX contains the same basic set of registers, instructions, and addressing modes as the industry-standard 186 microcontroller. For more information about the register set, memory organization, I/O space, instruction set, segments, data types and addressing modes, see the following document.

The Am186 and Am188 Family Instruction Set Manual, order# 21267

1.10 System Overview

1.10.1 DSTni-LX Package Options

The DSTni-LX is an enhanced 80186 16-bit microprocessor with integrated interrupt controller, 3 timers, 4-channel DMA controller, external chip selects, watchdog timer and JTAG port. All DSTni-LX chips contain 256KB SRAM, 2 KB Boot ROM, 8 KB dual-port RAM, 2 Serial Ports, 2 CAN 2.0B ports, and an SPI port.

Part Number	160 Pin LQFP Device	180 Pin BGA Device	16 Mb External Memory	Two (2) CAN Channels	Profibus Master/Slave	10/100 Ethernet MAC
DSTni-LX-001	♦		♦	♦		♦
DSTni-LX-002	♦		♦	♦	♦	♦
DSTni-LX-180BP		♦	♦	♦	♦	♦

1.10.2 DSTni-LX 160-Pin Package

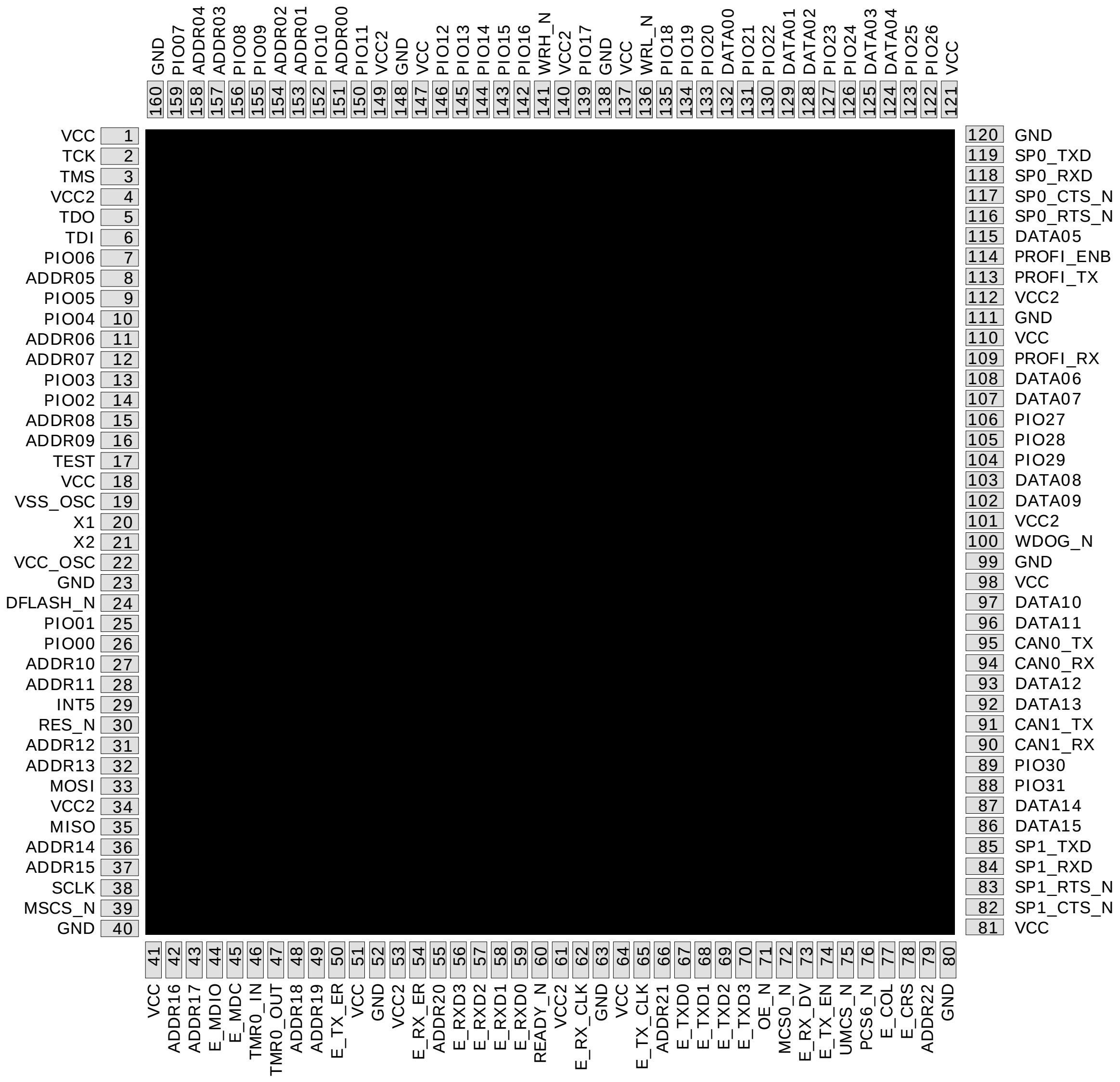


Figure 8 - DSTni-LX 160-Pin Package

1.10.3 Pin Descriptions

ADDR22 – ADDR00	Address Bus (Outputs)	These 23 signals are output pins. They supply non-multiplexed memory or I/O addresses to the connected system. ADDR22 is the MSB and ADDR00 is the LSB. **NOTE:** Although not available as an external pin, ADDR23 is internally used and is the MSB of the Address Bus. All lines are always enabled which means they do not enter a high-impedance state. Following a system reset, ADR22-20 remain in the logic low state for "Real" or 20-bit mode. In "extended" or 24-bit mode, ADR22-20 function as normal address lines.
DATA15 – DATA00	Data Bus (Bi-Directional)	These 16 signals make up the external Data Bus pins. DATA15 is the MSB and DATA00 is the LSB. This Data Bus is bi-directional to allow system data to be passed in and out through READ and WRITE accesses respectively. This bus also goes into the high-impedance state when the system is not accessing external I/O or memory.
X1	Crystal Input	This pin and the X2 pin provide connections for the internal oscillator circuit. If an external crystal is used, the X2 pin is required as well. For external clock or oscillator circuits with digital output drivers, this pin can be connected directly and the X2 pin is left unconnected. See 48MHz Clock, page 28, for information about selecting crystals and using a crystal oscillator.
X2	Crystal Output	This pin and the X1 pin provide connections for the internal oscillator circuit (see X1 description above). See 48MHz Clock, page 28, for information about selecting crystals and using a crystal oscillator.
RES_N	Reset (Input)	This input pin forces the microcontroller to perform a reset. When RES_N is asserted, the microcontroller immediately terminates its present activity, clears its internal logic, and CPU control is transferred to the reset address, FFFF0h (20-bit mode) or FFFFE0H (24-bit mode). Note this processor has an internal 2048 bytes of "BOOT-ROM" located at the top of the address space, so execution will always start here when reset occurs. Any externally connected devices in the reset vector address will be overridden by the internal logic. See the "BOOTSTRAP ROM" section for more details.
TCK	JTAG Clock (Input)	The JTAG interface is used by the In-Circuit Emulator to access the hardware registers that support the breakpoint and trace functions. The four interface pins are TCK, TMS, TDI, and TDO. See the "In-Circuit Emulator" section for more details. The ESD tolerance for the TDO output, pin 5 is only about 500 Volts. This output is used by the JTAG interface to the FS-2 In-System Emulator. ESD precautions such as a grounded work area, using wrist straps, and making the connection, or removing the connection when the power is off, need to be used. When the JTAG interface is not being used, there is no issue.
TMS	JTAG Mode (Input)	
TDI	JTAG Data In (Input)	
TDO	JTAG Data Out (Output)	
WDOG_N	Watchdog (Output)	This output pin is used to indicate that a Watchdog Reset or timeout has occurred. The signal is active low and will revert back to high when the processor has gone through the reset vector. Typically this output signal is used to reset or signal external circuitry that the Watchdog timer has timed out.

SP0_RXD	Serial Port 0 Receive Data (Input)	This pin is used to receive asynchronous serial data from an externally connected device through serial Port 0. See the “Asynchronous Serial Ports” section for more details including enhanced RS-485 and DMA features.
SP0_TXD	Serial Port 0 Transmit Data (Output)	This pin is used to send asynchronous serial data to an externally connected device through serial Port 0. See the “Asynchronous Serial Ports” section for more details including enhanced RS-485 and DMA features.
SP0_RTS_N	Serial Port 0 Request to Send (Output)	If enabled, this “flow control” pin is used to signal an external device that the processor has serial data and is requesting to send it through serial Port 0. See the “Asynchronous Serial Ports” section for more details including enhanced RS-485 and DMA features.
SP0_CTS_N	Serial Port 0 Clear to Send (Input)	If enabled, this “flow control” input pin is used to signal the processor that an externally connected device is ready to receive serial data through serial Port 0. See the “Asynchronous Serial Ports” section for more details including enhanced RS-485 and DMA features.
SP1_RXD/ DIAG2	Serial Port 1 Receive Data (Bi-directional)	This is a multiplexed pin. In the default Serial Port mode, this input pin is used to receive asynchronous serial data from an externally connected device through serial Port 1. See the “Asynchronous Serial Ports” section for more details including enhanced RS-485 and DMA features. The alternate mode of operation for this pin provides support for the Profibus DPV2 Isochron Mode of Operation. In this mode, the pin is an output utilizing the Profibus specific function of “DIAG2”.
SP1_TXD/ DIAG3	Serial Port 1 Transmit Data (Output)	This is a multiplexed pin. In the default Serial Port mode, this input pin is used to send asynchronous serial data to an externally connected device through serial Port 1. See the “Asynchronous Serial Ports” section for more details including enhanced RS-485 and DMA features. The alternate mode of operation for this pin provides support for the Profibus DPV2 Isochron Mode of Operation. In this mode, the pin is an output utilizing the Profibus specific function of “DIAG3”.
SP1_RTS_N/ DIAG4	Serial Port 1 Request to Send (Output)	This is a multiplexed pin. The default mode for this pin is the RTS Serial Port function. If this “flow control” function is enabled, this output pin is used to signal an external device that the processor has serial data and is requesting to send it through serial Port 1. See the “Asynchronous Serial Ports” section for more details including enhanced RS-485 and DMA features. The alternate mode of operation for this pin provides support for the Profibus DPV2 Isochron Mode of Operation. In this mode, the pin is an output utilizing the Profibus specific function of “DIAG4”.
SP1_CTS_N/ ASPC2_CTS_N	Serial Port 1 Clear to Send (Input)	This is a multiplexed pin. The default mode for this pin is the CTS Serial Port function. If this “flow control” function is enabled, this input pin is used to signal the processor that an externally connected device is ready to receive serial data through serial Port 1. See the “Asynchronous Serial Ports” section for more details including enhanced RS-485 and DMA features. The alternate mode of operation for this pin provides support for the Profibus DPV2 Isochron Mode of Operation. In this mode, the pin is an input utilizing the Profibus specific function of “ASPC2_CTS_N”.

OE_N	Output Enable (Output)	This pin (Output Enable) is an active low output signal used to access external memory along with the chip select signals. This signal goes active for all external memory access made by the processor. It is typically connected to the output enable pin of static RAM, or Flash device. It provides the tri-state bus disable function to the device when in the inactive or high state.
WRL_N	Write Low Byte (Output)	This pin (Write Low Byte) is an active low output signal used to indicate an external memory write access is being performed on the low byte of the system data bus, D07-D00. It is typically connected to the Write enable pin of the external memory device.
WRH_N	Write High Byte (Output)	This pin (Write High Byte) is an active low output signal used to indicate an external memory write access is being performed on the upper byte of the system data bus, D15-D08. It is typically connected to the Write enable pin of the external memory device.
UMCS_N	Upper Memory Chip Select (Output)	This pin is an active low output signal and indicates to the system that an external memory access is in progress to the upper memory block. The starting address and size of the upper memory block are programmable up to 512 Kbytes in 20-bit mode and up to 8 Mbytes in 24-bit mode. When the system comes out of reset, UMCS_N is active for the top 64 Kbyte memory block. This signal is typically connected to the chip enable of an external non-volatile storage device. Note that the on-chip BOOTROM overrides the very top 2048 bytes of the UMCS_N memory area.
MCS0_N	Middle Chip Select 0 (Output)	This pin is an active low output signal and indicates to the system that an external memory access is in progress to the middle memory block range. The internal logic is hard wired so that the MCS0_N pin is always active in the range of 40000h to 7FFFFh in 20-bit mode and 040000h to 7FFFFFh in 24-bit mode. MCS0_N is automatically enabled after a reset or power up.
READY_N	External Ready (Input)	This pin is an active low input signal. It is a synchronous ready that indicates to the processor that the addressed memory space or I/O device will complete a data transfer. This input contains an internal pull-down resistor, so to always assert the ready condition to the processor, this pin can be left open. **NOTE**: Do not use this signal to synchonize slow peripherals or slow memory. This will cause the Ethernet to fail and the processor to crash.
E_COL	Ethernet MII Bus Collision (Input)	The PHY asserts this output when a collision is detected. This output remains High for the duration of the collision. This signal is asynchronous and is inactive during full duplex operation.
E_CRS	Ethernet MII Carrier Sense (Input)	During half-duplex operation (bit 0.8 = 0), the PHY asserts this output when either transmitting or receiving data packets. During full-duplex operation (bit 0.8 = 1), E_CRS is asserted only during receive. E_CRS assertion is asynchronous with respect to E_RX_CLK. E_CRS is de-asserted on loss of carrier, synchronous to E_RX_CLK.
E_RX_DV	Ethernet MII Receive Data Valid (Input)	The PHY asserts this signal when it drives valid data on E_RXD. This output is synchronous to E_RX_CLK.

E_RXD3 – E_RXD0	Ethernet MII Receive Data 3-0 (Inputs)	E_RXD3-0 are parallel signals that transition synchronously with respect to the E_RX_CLK. E_RXD0 is the least significant bit.
E_RX_ER	Ethernet MII Receive Error (Input)	Signals a receive error condition has occurred. This input is synchronous to E_RX_CLK.
E_RX_CLK	Ethernet MII Receive Clock (Input)	Clock input for the Ethernet device. 25 MHz for 100 Mbps operation, 2.5 MHz for 10 Mbps operation.
E_TX_EN	Ethernet MII Transmit Enable (Output)	The controller asserts this signal when it drives valid data on E_TXD. This signal is synchronized to E_TX_CLK.
E_TXD3 – E_TXD0	Ethernet MII Transmit Data 3-0 (Outputs)	TXD3-0 are parallel data signals that are driven by the Ethernet Controller. TXD<3:0> will transition synchronously with respect to the E_TX_CLK. E_TXD0 is the least significant bit.
E_TX_ER	Ethernet MII Transmit Error (Output)	Signals a transmit error condition. This signal is synchronized to E_TX_CLK.
E_TX_CLK	Ethernet MII Transmit Clock (Input)	E_TX_CLK is sourced by the PHY in both 10 and 100 Mbps operations. 2.5 MHz for 10 Mbps operation, 25 MHz for 100 Mbps operation.
E_MDIO	Ethernet Management Data (Bi-directional)	Bi-directional serial data channel for PHY/STA communication.
E_MDC	Ethernet Management Clock (Output)	Clock output for the E_MDIO serial data channel.
PIO31/ LED3_N/ Address Mode Sel	LED Control 3 (Bi-directional)	This is a programmable multifunction pin. It can be used with the normal function or can be programmed for alternate general-purpose I/O use. This pin is also takes on a special function only during reset, Address Mode Select. It is sampled internally by the processor when going through the reset state to select the processor mode. **Address Mode Sel:** When left open or pulled up to a logic high during hardware reset, it places the processor into 20-bit address mode. When connected to Ground during hardware reset, it places the DSTni-LX into 24-bit address mode. **LED3_N:** This is the Normal function for this pin. It is typically used to drive a status LED but can be used freely. **PIO31:** Can be programmed as a general-purpose digital I/O pin through the PIO Mode and Direction registers. There are three options to choose from, Normal, Input, or Output. Note that regardless of the mode selected, this pin always contains a weak internal pull-up circuit.

PIO30/LED2_N/ Watchdog Disable_N	LED Control 2 (Bi-directional)	This is a programmable multifunction pin. It can be used with the normal function or can be programmed for alternate general-purpose I/O use. This pin is also takes on a special function only during reset, Watchdog Disable. It is sampled internally by the processor when going through the reset state. **Watchdog Disable_N:** When this pin is held low during power up or reset, it causes the watchdog timer to be disabled. **LED2_N:** This is the Normal function for this pin. It is typically used to drive a status LED but can be used freely. **PIO30:** Can be programmed as a general-purpose digital I/O pin through the PIO Mode and Direction registers. There are three options to choose from, Normal, Input, or Output. Note that regardless of the mode selected, this pin always contains a weak internal pull-up circuit.
PIO29/LED1_N	LED Control 1 (Bi-directional)	This is a programmable multifunction pin. It can be used with the normal function or can be programmed for alternate general-purpose I/O use. **LED1_N:** This is the Normal function for this pin. It is typically used to drive a status LED but can be used freely. During execution of the internal Boot ROM code, this pin and LED0 are used to indicate system status. **PIO29:** Can be programmed as a general-purpose digital I/O pin through the PIO Mode and Direction registers. There are three options to choose from, Normal, Input, or Output. Note that regardless of the mode selected, this pin always contains a weak internal pull-up circuit.
PIO28/LED0_N	LED Control 0 (Bi-directional)	This is a programmable multifunction pin. It can be used with the normal function or can be programmed for alternate general-purpose I/O use. **LED0_N:** This is the Normal function for this pin. It is typically used to drive a status LED but can be used freely. During execution of the internal Boot ROM code, this pin and LED1 are used to indicate system status. **PIO28:** Can be programmed as a general-purpose digital I/O pin through the PIO Mode and Direction registers. There are three options to choose from, Normal, Input, or Output. Note that regardless of the mode selected, this pin always contains a weak internal pull-up circuit.
PIO27	General Purpose I/O Pin (Bi-directional)	This is a general purpose Programmable I/O pin. **PIO27:** Can be programmed as a general-purpose digital I/O pin through the PIO Mode and Direction registers. There are three options to choose from, Normal, Input, or Output. Note that regardless of the mode selected, this pin always contains a weak internal pull-down circuit.

PIO26 – PIO00 / DPM	PIO / Dual Port Memory pins	The following 27 pins are multifunction pins, meaning they can be used with the “normal” function or can be programmed for general-purpose I/O use. If used in the normal mode, these pins make up a logical group consisting of the “Dual Port Memory” interface. They allow the user to gain external access to the “External-side” port of the internal 8k bytes of dual-port memory. The “Internal-side” port is internally connected to processor and is accessed through normal memory (for data) and I/O (for semaphore) accesses. If Dual Port Memory is not required, the user is free to use all 27 pins as general-purpose I/O pins. They can be programmed through the PIO Mode and Direction registers. Note that regardless of the mode selected, all pins always contain a weak internal pull-up circuit.
PIO26/DPCS_N	I/O – DPRAM Chip Select (Bi-directional)	**DPCS_N:** Dual Port RAM Chip Select. When used in this normal mode, this active low input pin is used during an external Read or Write Access to enable data transfers to and from the “External-Side” of the internal 8k dual port RAM. Note this signal must be used in conjunction with the DPRAM Data Bus, Address Bus, and one of the following control signals, DPWR_N, or DPOE_N to perform a valid data transfer.
PIO25/DPSS_N	I/O – DPRAM Semaphore Select (Bi-directional)	**DPSS_N:** Dual Port RAM Semaphore Select. When used in this normal mode, this active low input pin is used during an external Read or Write Access to obtain or release one of the eight semaphore flags contained in the dual port RAM interface. Note this signal must be used in conjunction with the DPRAM Data Bus, Address Bus, and one of the following control signals, DPWR_N, or DPOE_N to perform a valid semaphore transfer.
PIO24/DPWR_N	I/O – DPRAM Memory Write (Bi-directional)	**DPWR_N:** Dual Port RAM Memory Write. When used in this normal mode, this active low input pin is used during an external Write Access to enable data transfers from the DPRAM Data bus to the “External-side” port of the internal 8k bytes of dual-port memory. Note this signal must be used in conjunction with the DPRAM Data Bus, Address Bus and the DPCS_N to perform a valid data transfer.
PIO23/DPOE_N	I/O – DPRAM Output Enable (Bi-directional)	**DPOE_N:** Dual Port RAM Output Enable. When used in this normal mode, this active low input pin is used during an external Read Access to enable data transfers from the “External-side” port of the internal 8k bytes of dual-port memory to the DPRAM Data bus. Note this signal must be used in conjunction with the DPRAM Data Bus, Address Bus and the DPCS_N to perform a valid data transfer.
PIO22/DPINT_N	I/O – DPRAM **External** Port Interrupt (Bi-directional)	**DPINT_N:** Dual Port RAM External Port Interrupt. When used in this normal mode, this active low output pin is generally used to signal the external “External-side” user of the DPRAM that the internal-side port has information to be read or service is required. Note this interrupt is fully controlled by the user software and can be used to indicate any user-defined action. This signal will revert to the inactive state when the “External-side” makes a READ access from DPRAM address 1FFEh. The “External-side” user is also given the ability to generate an interrupt that will be sent to the internally connected “Internal-side” interrupt logic. To generate this interrupt, the “External-side” user simply writes to location 1FFFh. The “Internal-side” user resets and acknowledges this interrupt by making a READ access from DPRAM address 1FFFh.

PIO21/DPBUSY_N	I/O – DPRAM Busy (Bi-directional)	**DPBUSY_N: Dual Port RAM Busy.** When used in this normal mode, this active low output pin is used to signal the external “External-side” user of the DPRAM that a simultaneous access of a memory location has occurred and the Internal side currently has valid access control. Under this Busy condition, the “External-side” user will not be guaranteed valid data, so interface logic must take this into account. Typically this is handled by simply extending the current access until the busy condition is complete at which time the access is able to complete with valid data. Note that if the built in semaphores are used and managed correctly by the software on both sides, the Busy condition can be completely eliminated thus eliminating the need for external hardware logic.
PIO20/DPA12	I/O – DPRAM ADR_12	DPA12-DPA00: Dual Port RAM External Port Address Lines. When used in the normal mode, these 13 input signals make up the Address Bus for accessing the external “External-side” locations of the DPRAM. DPA12 is the MSB while DPA00 is the LSB of the Address Bus. These address lines are used in conjunction with the DPRAM Data Bus and control signals enabling full access to the “External-side” memory and semaphore locations.
PIO19/DPA11	I/O – DPRAM ADR_11	
PIO18/DPA10	I/O – DPRAM ADR_10	
PIO17/DPA09	I/O – DPRAM ADR_09	
PIO16/DPA08	I/O – DPRAM ADR_08	
PIO15/DPA07	I/O – DPRAM ADR_07	
PIO14/DPA06	I/O – DPRAM ADR_06	
PIO13/DPA05	I/O – DPRAM ADR_05	
PIO12/DPA04	I/O – DPRAM ADR_04	
PIO11/DPA03	I/O – DPRAM ADR_03	
PIO10/DPA02	I/O – DPRAM ADR_02	
PIO09/DPA01	I/O – DPRAM ADR_01	
PIO08/DPA00	I/O – DPRAM ADR_00	

PIO07/DPD7	I/O – DPRAM DATA7	**DPD7-DPD0:** Dual Port RAM Data Bus. When used in the normal mode, these 8 Bi-directional signals make up the Data Bus for accessing the external "External-side" locations of the DPRAM. DPD7 is the MSB while DPD0 is the LSB of the Data Bus. These data lines are used in conjunction with the DPRAM Address Bus and control signals enabling full access to the data on "External-side" memory and semaphore locations.
PIO06/DPD6	I/O – DPRAM DATA6	
PIO05/DPD5	I/O – DPRAM DATA5	
PIO04/DPD4	I/O – DPRAM DATA4	
PIO03/DPD3	I/O – DPRAM DATA3	
PIO02/DPD2	I/O – DPRAM DATA2	
PIO01/DPD1	I/O – DPRAM DATA1	
PIO00/DPD0	I/O – DPRAM DATA0	
PCS6_N	External Peripheral Chip Select (Output)	This pin is a dedicated active low output signal and indicates to the system that an external I/O access is in progress to this I/O memory block. The starting address is determined by the Peripheral Base Address (PBA) + 600h, which is set in the PACS register. The size of the I/O block is fixed at 256 bytes. This signal is typically connected to the chip enable of any user-selected device that contains less than or equal to 256 bytes of data.
CAN0_TX	CAN Bus 0 Transmit (Output)	This pin is a dedicated output signal used for transmitting CAN type protocol messages on channel CAN-0. +3.3 volt output. (See Note following this table.) For more details, see the CAN Controller section.
CAN0_RX	CAN Bus 0 Receive (Input)	This pin is a dedicated input signal used for receiving CAN type protocol messages on channel CAN-0. This pin must be current limited. (See Note following this table.) For more details, see the CAN Controller section.
CAN1_TX	CAN Bus 1 Transmit (Output)	This pin is a dedicated output signal used for transmitting CAN type protocol messages on channel CAN-1. +3.3 volt output. (See Note following this table.) For more details, see the CAN Controller section.
CAN1_RX	CAN Bus 1 Receive (Input)	This pin is a dedicated input signal used for receiving CAN type protocol messages on channel CAN-1. This pin must be current limited. (See Note following this table.) For more details, see the CAN Controller section.
PROFI_TX	Profibus Transmit Data (Output)	This pin is a dedicated output signal used for transmitting Profibus type protocol messages. +3.3 volt output. (See Note following this table.) For more details, see the Profibus Controller section. (DSTni-LX-002 only)

PROFI_RX	Profibus Receive Data (Input)	This pin is a dedicated input signal used for receiving Profibus type protocol messages. This pin must be current limited. (See Note following this table.) For more details, see the Profibus Controller section.(DSTni-LX-002 only)
PROFI_ENB	Transmit Enable (Output)	This pin is a dedicated output signal used for enabling the external Profibus transceivers when the controller is in the “transmit” mode. This signal is typically tied directly to the enable pin of the external transceiver. +3.3 volt output. (See Note following this table.) For more details, see the Profibus Controller section.(DSTni-LX-002 only)
TMR0_OUT	Timer 0 Output	This pin is a dedicated output signal that is logically connected to TIMER0, an internal 16-bit programmable timer. The duty cycle and rate of this signal is controlled through the Timer Registers. See the Timer section for more details.
TMR0_IN	Timer 0 Input	This pin is a dedicated input signal that can be programmed as the clock input to TIMER0, an internal 16-bit programmable timer. This pin is typically connected to an external clock when the user wants to use a different time base other than the internally supplied clock. Note that if this pin is not used, it must be pulled to logic high externally. See the Timer section for more details.
INT5	External Interrupt (Input)	This is a dedicated input pin that can be used to indicate to the processor that an external device requires servicing through an interrupt vector. The interrupt specific parameters for this pin are settable through the Interrupt control registers. See the Interrupt Controller section for more details.
DFLASH_N	SPI Slave 0 Select: Serial Flash (Output)	This is a dedicated active low signal used with the SPI Port. It is normally tied directly to the chip select of an externally supplied Serial Data Flash. This signal is controlled through the SPI Slave Select Register. See the SPI section for more details. Note this pin requires an external pull-up resistor.
MOSI	SPI:Master Out / Slave In (Bi-directional)	This is a dedicated bi-directional signal used with the SPI Port. It is normally tied directly to the Serial Input pins of externally supplied SPI slave devices. This signal is controlled through the SPI Data and Control Registers. See the SPI section for more details. Note this pin requires an external pull-up resistor.
MISO	SPI:Master In / Slave Out (Bi-directional)	This is a dedicated bi-directional signal used with the SPI Port. It is normally tied directly to the Serial output pins of externally supplied SPI slave devices. This signal is controlled through the SPI Data and Control Registers. See the SPI section for more details. Note this pin requires an external pull-up resistor.

SCLK	SPI:Master Clock Out / Slave Clock Input (Bi-directional)	This is a dedicated bi-directional signal used with the SPI Port. It currently is only supported in “master” mode. In master mode, this pin is an output that supplies the programmed SPI clock signal. It is normally tied directly to the Serial Clock input pins of externally supplied SPI slave devices. This signal is controlled through the SPI Data and Control Registers. See the SPI section for more details. Note this pin requires an external pull-up resistor.
MSCS_N	SPI:Master Collision Input	This is a dedicated active low input signal used with the SPI Port. It functions as the SPI Port, Master Collision Input for use with multiple masters. When driven low, an SPI collision will be detected. See the SPI section for more details. Note this pin requires an external pull-up resistor.
TEST	Test	Ground
GND	Ground	Logic Ground.
VCC	+3.3V	Supply voltage, +3.3VDC I/O Power.
VCC2	+2.5V	Supply voltage, +2.5VDC Core Power.
VSS_OSC	Ground	Ground for Internal Oscillator. (Must be connected to Ground)
VCC_OSC	+3.3V	+3.3V Supply voltage for internal oscillator. (Must be connected to +3.3V.)

NOTE

The DSTni-LX uses two power supplies. One is for the core logic, at +2.5 Volts. The second is for the I/O, at +3.3 Volts. If the +3.3 Volt supply is powered and the +2.5 Volt core logic supply is not powered, current in excess of 350 mA will flow into the chip. This will not be a problem for the chip for short periods (< 5 minutes). However, for very long periods (> 24 hours), the chip dissipation may be exceeded, and this may cause long term reliability issues.

We recommend that the +2.5 Volt core logic be powered first, or at least simultaneously with the +3.3 Volt I/O supply. Also, the device should never be subjected to long intervals where it is only powered by the +3.3 Volt I/O supply.

There is not a problem if the +2.5 Volt core logic supply is powered, and the +3.3 Volt I/O supply is not powered.

Further, it is not a problem if the +3.3 Volt I/O supply is not powered, but the connected +3.3 Volt I/O devices are powered. In this case, the +3.3 Volt I/O will actually power the +3.3 Volt I/O supply.

NOTE

The CAN0_RX, CAN1_RX, and PROFI_RX inputs operate normally as long as the current into them is limited. If the input is driven by a TTL or CMOS output, a series resistor of not less than 500 Ohms is required. If the input is driven by an open collector output, the pullup resistor cannot be less than 500 Ohms.

NOTE

The External Memory interfaces must use at least one wait state. This is required because of internal chip delays, which prevent the address signals from appearing early enough in the memory cycle to allow for the memory to be accessed and the data returned and stored at the appropriate place within a single 21nS memory cycle. The MMCS and UMCS registers can be programmed to easily provide this single wait state.

1.10.4 Pin Out Table

Pin Name	160 PIN	180 BGA	Cell
x1	20	F1	I, OSC
x2	21	G2	O, OSC
res_n	30	J2	I
test	17	E1	I
tck	2	B2	I, PU
tms	3	C3	I, PU
tdi	6	C2	I, PU
tdo	5	B1	O2
wdog_n	100	G11	O8
sp0_rxd	118	C13	I
sp0_txd	119	C14	O8
sp0_rts_n	116	D13	O8
sp0_cts_n	117	D14	I
sp1_rxd/diag2	84	L11	I,O8
sp1_txd/diag3	85	N14	O8
sp1_rts_n/diag4	83	M12	O8
sp1_cts_n/aspc2_cts_n	82	N13	I
addr22	79	P12	O8
addr21	66	N8	O8
addr20	55	M6	O8
addr19	49	P3	O8
addr18	48	L5	O8
addr17	43	M3	O8
addr16	42	N2	O8
addr15	37	L1	O8
addr14	36	L2	O8
addr13	32	J1	O8
addr12	31	K3	O8
addr11	28	K4	O8
addr10	27	J3	O8
addr9	16	G4	O8
addr8	15	F3	O8
addr7	12	F4	O8
addr6	11	E3	O8
addr5	8	E4	O8
addr4	158	B3	O8
addr3	157	A4	O8
addr2	154	C4	O8
addr1	153	B5	O8
addr0	151	C5	O8
data15	86	M13	I,O8
data14	87	L12	I,O8
data13	92	J11	I,O8
data12	93	L14	I,O8
data11	96	H11	I,O8
data10	97	K14	I,O8
data9	102	H13	I,O8
data8	103	G12	I,O8
data7	107	F12	I,O8
data6	108	E11	I,O8
data5	115	E14	I,O8
data4	124	D11	I,O8
data3	125	A13	I,O8
data2	128	D10	I,O8
data1	129	A12	I,O8
data0	132	D9	I,O8

Pin Name	160 PIN	180 BGA	Cell
oe_n	71	M10	O8
wrl_n	136	D8	O8
wrh_n	141	A9	O8
umcs_n	75	P10	O8
mcs0_n	72	P9	O8
ready_n	60	L8	I,PD
e_col	77	P11	I
e_crs	78	N12	I
e_rx_dv	73	N10	I
e_rxd3	56	L7	I
e_rxd2	57	P5	I
e_rxd1	58	N6	I
e_rxd0	59	M7	I
e_rx_er	54	N5	I
e_rx_clk	62	N7	I
e_tx_en	74	M11	O8
e_txd3	70	N9	O8
e_txd2	69	P8	O8
e_txd1	68	L10	O8
e_txd0	67	M9	O8
e_tx_er	50	N4	O8
e_tx_clk	65	P7	I
e_mdio	44	L4	I,O8
e_mdc	45	P2	O8
pio31/led3_n/Addr Mode Sel	88	K11	(I),O8,PU
pio30/led2_n/Watchdog Disbl_n	89	M14	(I),O8,PU
pio29/led1_n	104	F11	(I),O8,PU
pio28/led0_n	105	H14	(I),O8,PU
pio27	106	G13	I,(O)8,PD
pio26/dpcs_n	122	B13	(I),O8,PU
pio25/dpss_n	123	C12	(I),O8,PU
pio24/dpwr_n	126	B12	(I),O8,PU
pio23/dpoe_n	127	C11	(I),O8,PU
pio22/dpint_n	130	B11	I,(O)8,PU
pio21/dpbusy_n	131	C10	I,(O)8,PU
pio20/dpa12	133	A11	(I),O8,PU
pio19/dpa11	134	B10	(I),O8,PU
pio18/dpa10	135	C9	(I),O8,PU
pio17/dpa09	139	C8	(I),O8,PU
pio16/dpa08	142	B8	(I),O8,PU
pio15/dpa07	143	C7	(I),O8,PU
pio14/dpa06	144	D6	(I),O8,PU
pio13/dpa05	145	A8	(I),O8,PU
pio12/dpa04	146	B7	(I),O8,PU
pio11/dpa03	150	B6	(I),O8,PU
pio10/dpa02	152	A6	(I),O8,PU
pio09/dpa01	155	A5	(I),O8,PU
pio08/dpa00	156	B4	(I),O8,PU
pio07/dpd7	159	A3	I,O8,PU
pio06/dpd6	7	D3	I,O8,PU
pio05/dpd5	9	C1	I,O8,PU
pio04/dpd4	10	D2	I,O8,PU
pio03/dpd3	13	D1	I,O8,PU
pio02/dpd2	14	E2	I,O8,PU
pio01/dpd1	25	G1	I,O8,PU
pio00/dpd0	26	H2	I,O8,PU

Pin Name	160 PIN	180 BGA	Cell
pcs6_n	76	N11	O8
dflash_n	24	J4	O8
can0_tx	95	J12	O85T
can0_rx	94	K13	I (L)
can1_tx	91	K12	O85T
can1_rx	90	L13	I (L)
profi_tx	113	E13	O85T
profi_rx	109	G14	I (L)
profi_enb	114	D12	O85T
tmr0_out	47	M4	O8
tmr0_in	46	N3	I
int5	29	H1	I
mosi	33	K2	I,O8
miso	35	K1	I,O8
sclk	38	M2	I,O8
mscs_n	39	M1	I,O8
VSS_OSC	19	G3	
VCC_OSC	22	F2	
VCC 3.3V	1,18,41,51,64,81, 98,110,121,137, 147	A1, A10, A14, C6, F13, H3, J13, L9, M5, P1, P14	IO POWER
VCC2 2.5V	4,34,53,61,101, 112,140,149,	A7, D4, D7, F14, J14, L3, P4, P6	CORE POWER
GND (VSS)	23,40,52,63,80, 99,111,120,138, 148, 160	A2, B9, B14, D5, E12, H4, H12, L6, M8, N1, P13	GROUND

NOTE

The DSTni-LX uses two power supplies. One is for the core logic, at +2.5 Volts. The second is for the I/O, at +3.3 Volts. If the +3.3 Volt supply is powered and the +2.5 Volt core logic supply is not powered, current in excess of 350 mA will flow into the chip. This will not be a problem for the chip for short periods (< 5 minutes). However, for very long periods (> 24 hours), the chip dissipation may be exceeded, and this may cause long term reliability issues.

We recommend that the +2.5 Volt core logic be powered first, or at least simultaneously with the +3.3 Volt I/O supply. Also, the device should never be subjected to long intervals where it is only powered by the +3.3 Volt I/O supply.

There is not a problem if the +2.5 Volt core logic supply is powered, and the +3.3 Volt I/O supply is not powered.

Further, it is not a problem if the +3.3 Volt I/O supply is not powered, but the connected +3.3 Volt I/O devices are powered. In this case, the +3.3 Volt I/O will actually power the +3.3 Volt I/O supply.

Symbol Table

Symbol	Meaning
I	Input Pin
PU	Equivalent internal 10K-Ohm pull-up to Vcc_3.3V
O2	2 ma drive
O8	8 ma drive
I5T	Input, +5VDC tolerant
O85T	8 ma drive slow slew rate, +5VDC tolerant output
PD	Equivalent internal 10K-Ohm pull-down to GND
(I)	Normal direction is Input (PIO00-PIO31)
(O)	Normal direction is Output (PIO00-PIO31)
I, OSC	Oscillator Input
O, OSC	Oscillator Output
I (L)	Input Pin, Current limited (See Note below)

NOTE

The I(L) inputs operate normally as long as the current into them is limited. If the input is driven by a TTL or CMOS output, a series resistor of not less than 500 Ohms is required. If the input is driven by an open collector output, the pullup resistor cannot be less than 500 Ohms.

The power-on or Reset values for the PIO00-PIO31 pins are inputs with pullup/ pulldown.

1.10.5 48 MHz Clock

The DSTni-LX contains a clock generation circuit which allows the use of a times-one crystal up to 48MHz. You can also use a 48MHz Crystal Oscillator to drive pin X1. The following diagrams illustrate the clock circuits.

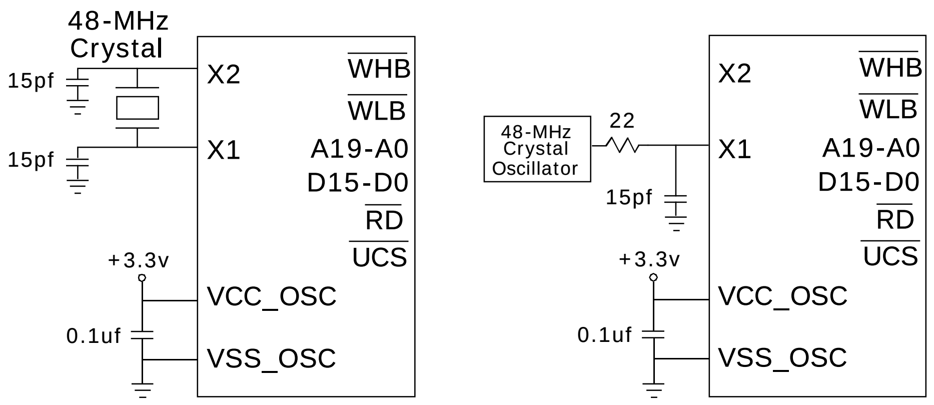


Figure 9 - 48 MHz Clock

1.10.6 20/24-Bit Address Support

The DSTni-LX has the ability to operate in either the standard 80x86 address mode (20-bit) having a 1M byte address space or in the extended address mode (24-bit) having a 16M byte address space. The mode of operation is determined by the state of the PIO31/LED3_N/24B_N pin at the time the DSTni-LX goes through hardware reset. If the PIO31/LED3_N/24B_N pin is at Vcc during reset, the DSTni-LX will operate in 20-bit mode. If the PIO31/LED3_N/24B_N pin is at Ground during reset, the DSTni-LX will operate in 24-bit mode.

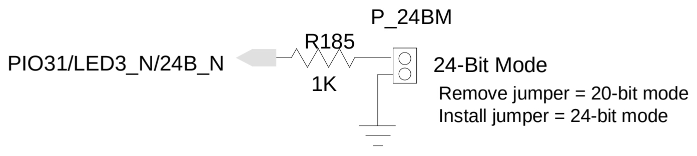


Figure 10 - 24-Bit Address Option

When the DSTni-LX is operating in 24-bit mode, the paragraph size for a segment increases from 16 bytes to 256 bytes. The following table of examples illustrate the differences.

Linear Address	**Equivalent segment:offset address**	
	20-bit	**24-bit**
040000h	4000:0000	0400:0000
0FF800h	FF80:0000	0FF8:0000
023456h	2345:0006	0234:0056
800000h	N/A	8000:0000
FFF800h	N/A	FFF8:0000

As the last few examples show, 24-bit mode provides access to a much larger address space than is possible with the standard 20-bit architecture. Segment size is limited to 64 Kbytes in 20-bit and 24-bit mode.

When in the 20-bit mode, memory accesses using the UMCS chip select will set the address bits A23:A20 to all ones. For accesses using the LMCS or the MMCS chip selects, the address bits A23:A20 will be all zeroes.

When in the 24-bit address mode, the address bits A23:A20 will be controlled by the 24-bit address adder.

The Reset Vector for the processor is FFFF0h for 20-bit mode and FFFFE0h for 24-bit mode.

See the memory map in the Bootstrap section page 89.

2 Peripheral Control Block

2.1 Memory Layout

The following memory map illustrates the basic layout of the internal 256K SRAM and 64K I/O space. The control registers and chip select functions are discussed in this section.

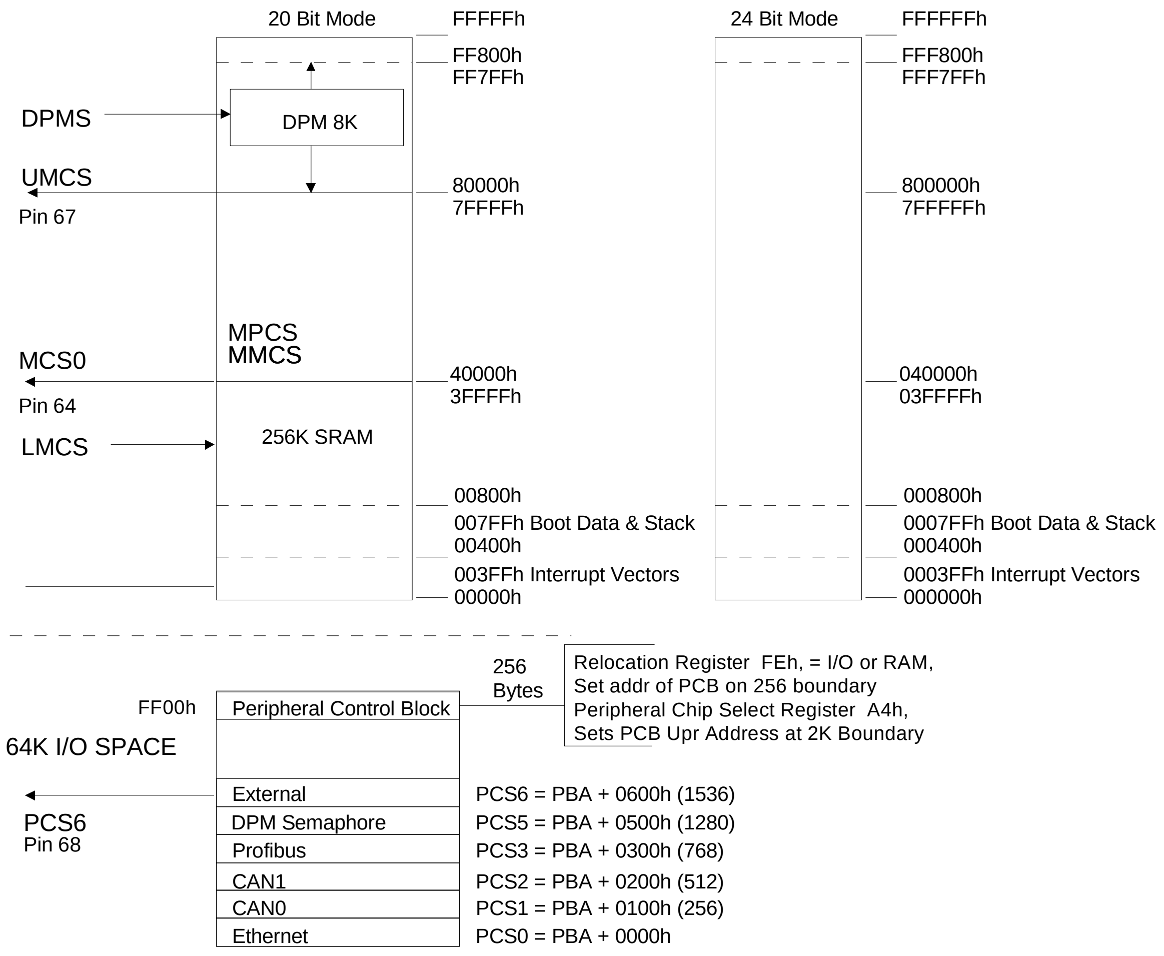


Figure 11 - Memory Layout

2.2 Peripheral Registers

The DSTni-LX integrated peripherals are controlled by 16-bit read/write registers. The peripheral registers are contained within an internal 256-byte control block—the peripheral control block. Registers are physically located in the peripheral devices they control, but they are addressed as a single 256-byte block. The following table shows a map of the peripheral control block registers.

Code that is intended to execute on the DSTni-LX should perform all writes to the PCB registers as word writes. There are no 8-bit wide registers in the Peripheral Control Block.

The peripheral control block can be mapped into either memory or I/O space. The base address of the control block must be on an even 256-byte boundary (i.e., the lower eight bits of the base address are 00h). Internal logic recognizes control block addresses and responds to bus cycles. During bus cycles to internal registers, the bus controller signals the operation externally (i.e., the RD, WR, status, address, and data lines are driven as in a normal bus cycle), but the data bus, SRDY, and ARDY are ignored.

At reset, the Peripheral Control Block Relocation register is set to 40FFh, which maps the control block to start at FF00h in I/O space.

NOTE: Programmers should set this register to 00FFh, clearing the Slave Interrupt mode bit. This is not a run-time issue, but will help correct operation when using the FS2 debugger to examine PCB registers.

An offset map of the 256-byte peripheral control register block is shown in the following table.

Table 1 - Peripheral Register Offset Map

Register Name	Offset
Relocation Register	FEH – FFH
Processor Release Level	F4H – F5H
Auxiliary Configuration Register	F2H – F3H
System Configuration Register	F0H – F1H
Watchdog Timer Control	E6H – E7H
DMA 1	D0H – DFH See Table 14, page 53
DMA 0	C0H – CFH See Table 14, page 53
DMA 3	90H – 9FH See Table 14, page 53
Chip Select Registers	A0H – AFH See Table 2, page 35
DMA 2	B0H – BFH See Table 14, page 53
Profibus Extended Control	8EH – 8FH (DSTni-LX-002 only)
Reserved	8CH – 8DH
Async Serial Port 0	80H – 8BH See Table 15, page 58
Reserved	7CH – 7FH
Programmable I/O	70H – 7BH See Table 21, page 72
Serial Peripheral Interconnect (SPI)	68H – 6FH See Table 19, page 67
Timer 2	60H – 67H See Table 8, page 41
Timer 1	58H – 5FH See Table 8, page 41
Timer 0	50H – 57H See Table 8, page 41
Interrupt Control	20H – 4FH See Table 13, page 48
Reserved	1CH – 1FH
Async Serial Port 1	10H – 1BH See Table 15, page 58
Reserved	00H – 0FH

2.2.1 PCB Relocation Register (RELREG, Offset = FEh)

This register controls whether the peripheral control block (PCB) is located in IO space or memory space. The Relocation Address bits set the starting address of the PCB to any 256 byte boundary.

Register 1 - PCB Relocation Register, RELREG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S/M	0	M/IO	R19	R18	R17	R16	R15	R14	R13	R12	R11	R10	R9	R8

S/M	Slave/Master. Configures the interrupt controller for slave mode when set to 1 and for master mode when set to 0. The DSTni-LX does not support slave mode so this bit must be set to 0.
M/IO	Memory/IO Space. When set to 1, the peripheral control block (PCB) is located in memory space. When set to 0, the PCB is located in IO space.
R19:R8	Relocation Address Bits. Allows the PCB to start at any 256 byte boundary. When the PCB is set for IO space, bits R19:R16 are ignored.

2.2.2 Processor Release Level Register (PRL, Offset = F4h)

This is a read only register to identify the version of the DSTni-LX.

Register 2 - Processor Release Level, PRL

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

REV	Revision Number
MONTH	Month (1-12) revision was generated.
DAY	Day (1-31) of month revision was generated.

2.2.3 System Configuration Register (SYSCON, Offset = F0h)

This register provides control over CPU features. For the DSTni-LX there are no writable bits.

Register 3 - System Configuration Register, SYSCON

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

MCSBIT	This bit is hardwired to 1 within the DSTni-LX, meaning that the Middle Chip Select pin, MCS0, is active over the entire Middle Chip Select range rather than ¼ of it. This is hardwired because MCS0 is the only Middle Chip Select pin available on the DSTni-LX.

2.2.4 Auxiliary Configuration Register(AUXCON, Offset = F2h)

This register is referenced by the Serial Port Control Registers Flow Control Bit.

Register 4 - Auxiliary Configuration Register, AUXCON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	ENRX1	RTS1	ENRX0	RTS0	0	0	0

ENRX1	Serial Port 1 Enable Receiver Request ENRX1. When this bit is 1, the CTS1/ENRX1 pin is configured as ENRX1. When this bit is 0, the CTS1/ENRX1 pin is configured as CTS1. This bit is 0 after processor reset.
RTS1	Serial Port 1 Request to Send RTS1. When this bit is 1, the RTR1/RTS1 pin is configured as RTS1. When this bit is 0, the RTR1/RTS1 pin is configured as RTR1. This bit is 0 after processor reset.
ENRX0	Serial Port 0 Enable Receiver Request ENRX0. When this bit is 1, the CTS0/ENRX0 pin is configured as ENRX0. When this bit is 0, the CTS0/ENRX0 pin is configured as CTS0. This bit is 0 after processor reset.
RTS0	Serial Port 0 Request to Send RTS0. When this bit is 1, the RTR0/RTS0 pin is configured as RTS0. When this bit is 0, the RTR0/RTS0 pin is configured as RTR0. This bit is 0 after processor reset.

2.2.5 Chip Select

The DSTni-LX contains logic that provides programmable chip select generation for both memories and peripherals. In addition, the logic can be programmed to provide ready or wait-state. The chip select lines are active for all memory and I/O cycles in their programmed areas, whether they are generated by the CPU or by the integrated DMA unit.

The DSTni-LX provides four chip select outputs for use with memory devices in the memory space and six more for use with peripherals in the I/O space. The memory chip selects can be used to address four memory ranges. Each peripheral chip select addresses a 256-byte block offset from a programmable base address.

The chip selects are programmed through the use of six 16-bit peripheral registers. The UMCS register, offset A0h, is used to program the Upper Memory Chip Select (UCS). The LMCS register, offset A2h, is used to program the Lower Memory Chip Select (LCS). The Midrange Memory Chip Select (MCS0) is programmed through the use of two registers—the MMCS register, offset A6h and the MPCS register, offset A8h. In addition to its use in configuring the MCS chip select, the MPCS register and the PACS register are used to program the Peripheral Chip Selects (PCS6–PCS5 and PCS3–PCS0). The DPMS register is used to program the 8k-byte block of available dual port internal memory.

Note: The SYSCON register contains the MCSBIT field which determines behavior of the MCS0 chip select. The PCS4 chip select is not implemented on the DSTni-LX.

Chip select registers are used to control access to memory and peripheral components such as FLASH, SRAM, and on-chip static RAM. The control registers are:

Table 2 - Chip Select Registers

Register Name	Mnemonic	Offset	Comments
Upper Memory Chip Select	UMCS	A0h	UMCS, top of address space
Lower Memory Chip Select	LMCS	A2h	LMCS, 256K SRAM
Peripheral Address Chip Select	PACS	A4h	PCS6-PCS5, PCS3-PCS0
Mid-Memory Chip Select	MMCS	A6h	MCS0
Memory / Peripheral Control Select	MPCS	A8h	MCS0 + PCS6-PCS5, PCS3-PCS0
Dual Port Memory Select	DPMS	AAh	Selects Internal 8k-bytes of DPM

2.2.6 Ready and Wait-State Programming

The DSTni-LX can be programmed to sense a ready signal for each of the peripheral or memory chip select lines. Each chip select control register (UMCS, LMCS, MMCS, PACS, and MPCS) contains a single-bit field, R2, that determines whether the external ready signal is required or ignored. When R2 is set to 1, external ready is ignored. When R2 is set to 0, external ready is required.

The number of wait states to be inserted for each access to a peripheral or memory region is programmable. Zero wait states to 3 wait states can be inserted for the PCS6–PCS0 peripheral chip selects. Zero wait states to 5 wait states can be inserted for UMCS.

Each of the chip select control registers other than the PACS register (DPMS, LMCS, MMCS, and MPCS) contains a two-bit field, R1–R0, whose value determines the number of wait states from zero to three to be inserted. A value of 00b in this field specifies no inserted wait states. A value of 11b specifies 3 inserted wait states.

The PCS6–PCS0 peripheral chip selects can be programmed for up to 3 wait states. When external ready is required (R2 is set to 0), internally programmed wait states will always complete before external ready can terminate or extend a bus cycle. For example, if the internal wait states are set to insert two wait states (R1–R0 = 10b), the processor samples the external ready pin during the first wait cycle. If external ready is asserted at that time, the access completes after 3 cycles (one cycle plus two wait states). If external ready is not asserted during the first wait cycle, the access is extended until ready is asserted, which is followed by one more wait state.

NOTE

The External Memory interfaces must use at least one wait state. This is required because of internal chip delays, which prevent the address signals from appearing early enough in the memory cycle to allow for the memory to be accessed and the data returned and stored at the appropriate place within a single 21nS memory cycle. The MMCS and UMCS registers can be programmed to easily provide this single wait state.

2.2.7 Upper Memory Chip Select Register (UMCS, Offset = A0h)

Upper Memory Chip Select (UMCS) controls access to non-volatile (FLASH) memory components. Upon power-up or reset, the UMCS signal will be set to select the top 64K bytes of the address space. The on-chip BOOTROM overrides the very top 2048 bytes of the UMCS area. The BOOTROM program will test for the presence of this memory by checking for a signature string at location FF7E0h. If the signature is present and correct, the bootstrap will jump to location FF7F0h, which will be the entry point for the application code. The bootstrap will program the UMCS pin to be active over the range F0000h – FF7FFh so any initialization code within the parallel FLASH device must be inside this range. Note: UMCS can also control RAM but is usually used to control non-volatile RAM.

The upper boundary is FFFFFh for 20-bit mode and FFFFFFh for 24-bit mode. The lower boundary, wait states, and READY signal are selected in the Upper Memory Chip Select Register.

See BOOTROM on page 87 for an explanation of the BOOTROM sequence.

Register 5 - Upper Memory Chip Select, UMCS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	LB2	LB1	LB0	0	0	0	0	0	0	1	1	1	R2	R1	R0

LB2:LB0	Selects the lower boundary for the chip select per Table 3. The upper boundary is always FFFFFh for 20-bit mode and FFFFFFh for 24-bit mode.
R2:0	Controls the wait state and READY generation logic, per Table 4. The R2 bit is used to configure the ready mode for the UMCS chip select. If R2 is set to 0, external ready is required. If R2 is set to 1, external ready is ignored. The processor uses the value of the R1-R0 bits to determine the number of wait states to insert into an access to UMCS memory area. R2 defaults to 0 at reset and R1-R0 default to 11 at reset.

Table 3 - UMCS Base Address and Size

LB2:LB0	**Starting Base Address**		**Memory Block Size**	
	20-bit mode	**24-bit mode**	**20-bit mode**	**24-bit mode**
000	80000h	800000h	512K	8M
100	C0000h	C00000h	256K	4M
110	E0000h	E00000h	128K	2M
111	F0000h	F00000h	64K	1M

Table 4 - UMCS Wait State and Ready Select

R2	**R1**	**R0**	**Wait States**	**Cycle (ns)**	**Ready Signal Used**
0	0	0	0	20	Yes
0	0	1	2	60	Yes
0	1	0	3	80	Yes
0	1	1	5	120	Yes
1	0	0	0	20	No
1	0	1	2	60	No
1	1	0	3	80	No
1	1	1	5	120	No

2.2.8 Lower Memory Chip Select (LMCS, Offset = A2h))

Lower Memory Chip Select (LMCS) controls access to the 256K bytes of on-chip SRAM in the lower portion of address space. The SRAM is organized as 128K x 16. The memory cycle time is less than 15nSec and runs with no wait states on the internal bus. The LMCS read-only register is hardwired to utilize the full range of on-chip SRAM at zero wait states. The range is from 00000h to 3FFFFh. Lower memory is automatically enabled after a reset or power-up.

Register 6 - Lower Memory Chip Select, LMCS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A19	A18	A17	A16	-	-	-	-	-	-	-	-	-	R2	R1	R0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0

2.2.9 Middle Memory Chip Select (MMCS, Offset = A6h)

The Middle Memory Chip Select (MMCS) is hard wired so that the MCS0 pin is always active in the range of 40000h to 7FFFFh in 20-bit mode and 040000h to 7FFFFFh in 24-bit mode. The wait state bits can be set to accommodate slower memories. MMCS is automatically enabled after a reset or power-up.

Register 7 - Middle Memory Chip Select, MMCS

MMCS 20-bit Mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A19	A18	A17	A16	A15	A14	A13	-	-	-	-	-	-	R2	R1	R0
0	1	0	0	0	0	0	1	1	1	1	1	1	0	0	0

MMCS 24-bit Mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A23	A22	A21	A20	A19	A18	A17	-	-	-	-	-	-	R2	R1	R0
0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0

R2:0	Controls the wait state and Ready generation. See Table 5 - MMCS Wait State and Ready Select

Table 5 - MMCS Wait State and Ready Select

R2	R1	R0	Wait States	Cycle (ns)	Ready Signal Used
0	0	0	0	20*	Yes
0	0	1	1	40*	Yes
0	1	0	2	60*	Yes
0	1	1	3	80*	Yes
1	0	0	0	20*	No
1	0	1	1	40*	No
1	1	0	2	60*	No
1	1	1	3	80*	No

*Note: Cycle time (ns) is based on a 48MHz clock.

2.2.10 Memory / Peripheral Control Select (MPCS, Offset = A8h)

This register controls the size of memory accessed via the Middle Memory Chip Select lines and the mode of operation for the Peripheral Chip Selects. The MPCS register fields provide program information for MCS3-MCS0 as well as PCS6-PCS5 and PCS3-PCS0. For the DSTni-LX the MCS0 pin is hardwired to always be active across the range 40000h to 7FFFFh in 20-bit mode and 040000h to 7FFFFFh in 24-bit mode.

Register 8 - Memory / Peripheral Control Select Register, MPCS

MPCS 20-bit Mode

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
-	M6	M5	M4	M3	M2	M1	M0	EX	MS	-	-	-	R2	R1	R0
1	0	1	0	0	0	0	0	1	0	1	1	1	0	0	0

MPCS 24-bit Mode

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
-	M6	M5	M4	M3	M2	M1	M0	EX	MS	-	-	-	R2	R1	R0
1	1	0	0	0	0	0	0	1	0	1	1	1	0	0	0

Bits 15-3	Hardwired. Cannot be changed.
M6:M0	This field determines the total block size for the MCS0 chip select. These bits are hardwired and cannot be changed.
EX	Set to 1. PCS6-PCS5 is always peripheral chip select.
MS	Set to 0. PCS outputs always active for I/O bus cycles.
R2:0	Ready generation for PCS6 and PCS5. See Table 5 - MMCS Wait State and Ready Select.

2.2.11 Peripheral Chip Selects (PACS, Offset = A4h)

The Peripheral Chip Selects are programmed through two registers—the Peripheral Chip Select (PACS) register and the PCS and MCS Auxiliary (MPCS) register. The Peripheral Chip Select (PACS) register determines the base address, the ready condition, and the wait states for the PCS3–PCS0 outputs.

The MPCS register bit 6 is set to 0, which forces PCS chip selects to be active during I/O bus cycles. The MPCS register also specifies the ready and wait states for the PCS6–PCS5 outputs.

The PCS pins are not active on reset. The PCS pins are activated as chip selects by writing to the PACS and MPCS registers.

The Peripheral Chip Selects do not need to support the 24-bit addressing. These have little value in the memory space because of their small, fixed range of 256 bytes. These will be used exclusively in I/O space.

Register 9 - Peripheral Address Chip Select Register, PACS

PACS 20-bit Mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A19	A18	A17	A16	A15	A14	A13	A12	A11	1	1	1	1	R2	R1	R0
0	0	0	0												

A19:11	Upper address bits of the starting address of the Peripheral Base Address (PBA). Allows programmer to set the PBA at any 2K byte address boundary. Since we're operating only in I/O space which is limited to 64 Kbytes, A19-A16 must always be zero.
R2:0	Ready generation select. See Table 5 - MMCS Wait State and Ready Select.

PACS 24-bit Mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	R2	R1	R0
0	0	0	0	0	0	0	0								

A23:11	Upper address bits of the starting address of the Peripheral Base Address (PBA). Allows programmer to set the PBA at any 2K byte address boundary. Since we're operating only in I/O space which is limited to 64 Kbytes, A23-A16 will always be zero.
R2:0	Ready generation select for PCS3-PCS0. See Table 5 - MMCS Wait State and Ready Select.

2.2.12 Dual Port Memory Select (DPMS, Offset = AAh)

This register is used to set the base address for the 8 Kbyte DPM block. If there is an overlap between this select and another memory select, DPMS will take precedence. As in the Middle Memory Chip Selects, DPMS is disabled after hardware reset until this register is written to.

Register 10 - Dual Port Memory Select, DPMS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A23	A22	A21	A20	A19	A18	A17	A16	0	0	0	0	0	R2	R1	R0

A23:16	Upper 8 address bits of the Dual Port Memory base. Allows programmer to set the DPM base at any 64 Kbyte address boundary. When the CPU is operating in normal 20-bit address mode, the contents of A23-A20 are ignored and are treated as if they were zero.
R2:0	Ready logic select for PCS3-PCS0. See Table 5 - MMCS Wait State and Ready Select.

2.2.13 Chip Select Assignments

The chip select lines are assigned per the following table.

Table 6 - Chip Select Assignments

Mnemonic	Name	Assigned To
UMCS	Upper Memory Chip Select	Boot ROM / UMCS_n pin
LMCS	Lower Memory Chip Select	Internal Static RAM
MMCS0_n	Middle Memory Chip Select 0	External pin MCS0_n
MMCS1_n	Middle Memory Chip Select 1	(Not Available)
MMCS2_n	Middle Memory Chip Select 2	(Not Available)
MMCS3_n	Middle Memory Chip Select 3	(Not Available)
DPMS	Dual Port Memory Select	Dual Port Memory
PCS0_n	Peripheral Chip Select 0	Ethernet
PCS1_n	Peripheral Chip Select 1	CAN0
PCS2_n	Peripheral Chip Select 2	CAN1
PCS3_n	Peripheral Chip Select 3	Profibus Controller
PCS4_n	Peripheral Chip Select 4	(Not Available)
PCS5_n	Peripheral Chip Select 5	Dual Port Memory Semaphores
PCS6_n	Peripheral Chip Select 6	External Pin PCS6_n

_n indicates active low signal

Each peripheral has 256 bytes of I/O address space allocated to it. The starting I/O address for each peripheral depends on the PCS line assigned to it. The following table summarizes the peripherals and the base I/O address for each.

Table 7 - Peripheral I/O Address Assignments

Select Line	Assigned To	Base I/O Address
PCS0_n	Ethernet	PBA + 0000h
PCS1_n	CAN0	PBA + 0100h
PCS2_n	CAN1	PBA + 0200h
PCS3_n	Profibus	PBA + 0300h
PCS4_n	(Not Available)	PBA + 0400h
PCS5_n	Dual Port Memory Semaphores	PBA + 0500h
PCS6_n	External Pin PCS6_n	PBA + 0600h

_n indicates active low signal

PBA is determined via the PACS register.

2.3 Timers

The DSTni-LX includes three internal 16-bit programmable timers and a Watchdog Timer.

Timer 0 connects to two external pins, one input and one output. Timers 1 and 2 are not connected to any external pins. Timers 1 and 2 can be used for real-time coding and time-delay applications. Timer2 can also be used as a prescaler to timer 0 and timer 1 or as a DMA request source. It can also be used as the clock source for the SPI port, when SPI is in master mode.

The Watchdog Timer function is wired to an external connection pin. This is an active low output pin (WDOG_n) that activates ('0') when the Watchdog Timer trips. It will be reset by software to the inactive state of '1' after receiving the "keyed" write operation to reset the WDT count (more details below).

2.3.1 Timer Registers

There are three timers controlled by eleven 16-bit registers located in the peripheral control block. See Timer Control Register Table below:

Table 8 - Timer Registers

Register Name	Mnemonic	Offset
Timer 0 Mode/Control	T0CON	56h
Timer 0 Count	T0CNT	50h
Timer 0 Maxcount Compare A	T0CMPA	52h
Timer 0 Maxcount Compare B	T0CMPB	54h
Timer 1 Mode/Control	T1CON	5Eh
Timer 1 Count	T1CNT	58h
Timer 1 Maxcount Compare A	T1CMPA	5Ah
Timer 1 Maxcount Compare B	T1CMPB	5Ch
Timer 2 Mode/Control	T2CON	66h
Timer 2 Count	T2CNT	60h
Timer 2 Maxcount Compare A	T2CMPA	62h
Watchdog Timer Control	WDTCON	E6h

The timer-count registers contain the current value of a timer. The timer-count registers can be read or written at any time, regardless of whether the corresponding timer is running. The microcontroller increments the value of a timer-count register each time a timer event occurs.

Each timer also has a maximum-count register that defines the maximum value for the timer. When the timer reaches the maximum value, it resets to 0 during the same clock cycle. (The value in the timer-count register never equals the maximum-count register.) In addition, timers 0 and 1 have a secondary maximum-count register. Using both the primary and secondary maximum-count registers lets the timer alternate between two maximum values. If the timer is programmed to use only the primary maximum-count register, the timer output pin switches Low for one clock cycle, the clock cycle after the maximum value is reached. If the timer is programmed to use both of its maximum-count registers, the output pin creates a waveform by indicating which maximum-count register is currently in control. The duty cycle and frequency of the waveform depend on the values in the alternating maximum-count registers.

2.3.2 Basic Timer Operation

Each timer is serviced on every fourth clock cycle. Therefore, a timer can operate at a maximum speed of one-quarter of the internal clock frequency. A timer can be clocked externally at the same maximum frequency of one-fourth of the internal clock frequency. However, because of internal synchronization and pipelining of the timer circuitry, the timer output takes up to six clock cycles to respond to the clock or gate input. The timers are run by the processor's internal clock.

Note that for the DSTni-LX Timer 0 is the only timer providing external pins TMRIN0 and TMROUT0. Timer 1 and Timer 2 provide no external connections.

2.3.3 Timer 0 and Timer 1 Control (T0CON, offset 56h; T1CON, offset 5Eh)

Registers T0CON and T1CON control the functionality of Timer0 and Timer1 respectively. The value of T0CON and T1CON at reset is 0000h.

Register 11 - Timers 0-1 Mode / Control Registers, T0CON, T1CON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	INH*	INT	RIU	0	0	0	0	0	0	MC	RTG	P	EXT	ALT	CONT

Note: * suffix means active low.

EN	Enable. When set to 1, the timer is enabled. When set to 0, the timer is inhibited from counting. This bit can only be written with the INH bit set at the same time.
INH	Inhibit. Allows selective updating of enable (EN) bit. When set to 1 during a write, EN can also be modified. When set to 0 during a write, the write to EN is ignored. This bit is not stored and is always read as 0.
INT	Interrupt. When set to 1, an interrupt request is generated when the count register equals a maximum count. If the timer is configured in dual maxcount mode, an interrupt is generated each time the count reaches maxcount A or maxcount B. When INT is set to 0, the timer will not issue interrupt requests. If the enable bit is cleared after an interrupt request has been generated but before the pending interrupt is serviced, the interrupt request will still be present.
RIU	Register In Use. When the maxcount compare A register is being used for comparison to the timer count value, this bit is set to 0. When the maxcount compare B register is being used, this bit is set to 1.
MC	Maximum Count. The MC bit is set to 1 when the timer reaches a maximum count. In dual maxcount mode, the bit is set each time either maxcount compare A or B register is reached. This bit is set regardless of the timer interrupt-enable bit. The MC bit can be used to monitor timer status through software polling instead of through interrupts.
RTG	Retrigger Mode. Determines the control function provided by the timer input pin. When set to 1, a 0 to 1 edge transition on TMRIN0 resets the count. When set to 0, a High input enables counting and a Low input holds the timer value. This bit is ignored when external clocking (EXT=1) is selected. Note that on the DSTni-LX only Timer 0 has an input pin, so this bit has no function for Timer 1.
P	Prescaler. When set to 1, the timer is prescaled by timer 2. When set to 0, the timer counts up every fourth CLKOUT period. This bit is ignored when external clocking is enabled (EXT=1).
EXT	External Clock. When set to 1, an external clock source connected to Timer 0 In pin is used. When set to 0, the internal clock is used. Note that on the DSTni-LX only Timer 0 has an input pin. This bit should always be 0 for Timer 1.

ALT	Alternate Compare. When set to 1, the timer counts to maxcount compare A, then resets the count register to 0. Then the timer counts to maxcount compare B, resets the count register to zero, and starts over with maxcount compare A. If ALT is clear, the timer counts to maxcount compare A and then resets the count register to zero and starts counting again against maxcount compare A. In this case, maxcount compare B is not used.
CONT	Continuous Mode. When set to 1, CONT causes the associated timer to run in the normal continuous mode. When CONT is set to 0, EN is cleared after each timer count sequence and the timer clears and then halts on reaching the maximum count. If CONT=0 and ALT=1, the timer counts to the maxcount compare A register value and resets, then it counts to the B register value and resets and halts.

2.3.4 Timer 2 Control (T2CON, offset 66h)

Register T2CON controls the functionality of Timer2. The value of T2CON at reset is 0000h.

Register 12 - Timer 2 Mode / Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	INH*	INT	0	0	0	0	0	0	0	MC	0	0	0	0	CONT

Note: * suffix means active low.

EN	Enable. When set to 1, the timer is enabled. When set to 0, the timer is inhibited from counting. This bit can only be written with the INH bit set at the same time.
INH	Inhibit. Allows selective updating of enable (EN) bit. When set to 1 during a write, EN can also be modified. When set to 0 during a write, the write to EN is ignored. This bit is not stored and is always read as 0.
INT	Interrupt. When set to 1, an interrupt request is generated when the count register equals a maximum count. If the timer is configured in dual maxcount mode, an interrupt is generated each time the count reaches maxcount A or maxcount B. When INT is set to 0, the timer will not issue interrupt requests. If the enable bit is cleared after an interrupt request has been generated but before the pending interrupt is serviced, the interrupt request will still be present.
MC	Maximum Count. The MC bit is set to 1 when the timer reaches a maximum count. In dual maxcount mode, the bit is set each time either maxcount compare A or B register is reached. This bit is set regardless of the timer interrupt-enable bit. The MC bit can be used to monitor timer status through software polling instead of through interrupts.
CONT	Continuous Mode. When this bit is set to 1, it causes the associated timer to run continuously. When set to 0, EN is cleared after each timer count sequence and the timer halts on reaching the maximum count.

2.3.5 Timer 0-2 Count (T0CNT, offset 50h; T1CNT, offset 58h; T2CNT, offset 60h)

These 16 bit registers can be incremented by one every four internal processor clocks. Timer 0 can also be configured to increment based on the TMRIN0 external signal. Timer 0 and Timer 1 can be prescaled by timer 2. The count registers are compared to maximum count registers and various actions are triggered based on reaching a maximum count. Note that the value of these registers at reset is undefined.

Register 13 - Timer Count Register, T0CNT, T1CNT, T2CNT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

T15:0	Current value of the T0, T1 or T2 timer.

2.3.6 Timer Maxcount Compare

These 16 bit registers serve as comparators for their associated count registers. Timer 0 and Timer 1 each have two maximum count compare registers. Timer 0 and Timer 1 can be configured to count and compare to register A and then count and compare to register B. Using this method, the TMROUT0 signal can be used to generate wave forms of various duty cycles.

Timer 2 has one compare register, T2CMPA. If a maximum count compare register is set to 0000h, the timer associated with that compare register will count from 0000h to FFFFh before requesting an interrupt. With a 48-MHz clock, a timer configured this way interrupts every 5.4613 ms. Note that the value of these registers at reset is undefined.

Register 14 - Timer Maxcount Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

T15:0	Programmed timer value.

2.3.7 Watchdog Timer Control (WDTCON, Offset = E6h)

The Watchdog Timer Control register is a combined status and control register through which all watchdog timer functionality is implemented.

The Watchdog Timer (WDT) is enabled out of reset and configured to system reset mode with a maximum timeout count. The WDTCON register can be opened for a single write following reset. To open the WDTCON register for writing, the keyed sequence of 3333h followed by CCCCh must be written to the WDTCON register. The register can then be written with the new configuration. Any number of processor cycles, including memory and I/O reads and writes, can be inserted between the two halves of the key or between the key and the writing of the new configuration as long as they do not access the WDTCON register. Note: The Watchdog Timer (WDT) is active after reset.

It is not possible to read the current count of the WDT, however it can be reset by writing the keyed sequence of AAAAh followed by 5555h to the WDTCON register. Any number of processor cycles, including memory and I/O reads and writes, can be inserted between the two halves of the key as long as they do not access the WDTCON register. The current count should be reset before modifying the WDT timeout period to ensure that an immediate WDT timeout does not occur. The value of the WDTCON register at reset is C080h.

In the DSTni-LX, the watchdog can be disabled via hardware which can be useful during software development. If the LED2 (PIO30) pin is tied to GND during a hardware reset or power up cycle, the watchdog timer will be disabled. After the reset cycle is complete, the LED2 pin can be used normally.

Register 15 - Watchdog Timer Control Register, WDTCON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENA	WRST	RSTFLAG	NMIFLAG	TEST	0	0	0	COUNT							

ENA	Watchdog Timer Enable. When this bit is 1, the watchdog timer is enabled. When this bit is 0, the watchdog timer is disabled. This bit is 1 after processor reset.
WRST	Watchdog Reset. When this bit is 1, the processor generates a WDT system reset when the WDT timeout count is reached. When this bit is 0, the processor generates an NMI interrupt when the WDT timeout count is reached if the NMIFLAG bit is 0. If the NMIFLAG bit is 1, a WDT system reset is generated upon WDT timeout. This bit is 1 after processor reset.

RSTFLAG	Reset Flag. When this bit is 1, a watchdog timer reset event has occurred. This bit is cleared by any keyed read or write to this register or by an externally generated system reset. This bit is 0 after an external system reset or 1 after a WDT system reset.
NMIFLAG	NMI Flag. When this bit is 1, a watchdog timer NMI event has occurred. This bit is cleared by any keyed write to this register. If this bit is set when a WDT timeout event occurs, a WDT system reset will be generated regardless of the setting of the WRST bit. This bit is 0 after processor reset.
TEST	Test Mode. This bit is reserved for an internal test mode. Setting this bit activates a special test mode that generates early WDT timeouts. This bit is 0 after processor reset.
COUNT	Watchdog Timer Timeout Count. This field determines the duration of the watchdog timer timeout interval. The duration is determined by using the value from the column titled "Exponent" in the "WDT COUNT Settings" Table in the following equation:

Duration = 2^{Exponent} / Frequency

Where Duration is the timeout period in seconds, Exponent is the value from "WDT Duration" Table, and Frequency is the processor frequency in Hz. For example, the following calculation determines the WDT timeout period for a 48-MHz processor with the COUNT field set to 20h.

Duration = (2^{24} cycles) / (48,000,000 Hz)
= (16,777,216 cycles) / (48,000,000 cycles/second)
= 0.3495 seconds

Setting more than one bit in the COUNT field results in the shorter timeout value. The COUNT field is 80h after reset.

Table 9 - WDT COUNT Settings

Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**	**Exponent**
0	0	0	0	0	0	0	0	N/A
X	X	X	X	X	X	X	1	10
X	X	X	X	X	X	1	0	20
X	X	X	X	X	1	0	0	21
X	X	X	X	1	0	0	0	22
X	X	X	1	0	0	0	0	23
X	X	1	0	0	0	0	0	24
X	1	0	0	0	0	0	0	25
1	0	0	0	0	0	0	0	26

Table 10 - WDT Duration

Exponent	**20MHZ**	**25MHZ**	**33MHZ**	**48MHZ**
10	51μs	40μs	30μs	21μs
20	52ms	41ms	31ms	22ms
21	104ms	83ms	62ms	43ms
22	209ms	167ms	125ms	87ms
23	419ms	335ms	251ms	174ms
24	838ms	671ms	503ms	349ms
25	1.67s	1.34s	1.00s	699ms
26	3.35s	2.68s	2.01s	1.398s

2.4 Interrupt Controller

The Interrupt Controller serves to manage and prioritizes both internal and external interrupt sources. Internal interrupt sources include the Timers and DMA channels. These sources can be disabled by their own control register or by the mask bits within the interrupt controller. External sources include the INT lines. The interrupt controller also has the ability to resolve priority among simultaneous interrupt requests. This interrupt controller is a functional subset of the actual 80C186 interrupt controller it ONLY SUPPORTS MASTER MODE, and FULLY NESTED MODE.

Fully Nested Mode

For Fully Nested Mode, the interrupt controller uses the interrupt lines (INT5-0) as direct inputs. Each interrupt source has an in-service bit. This bit is set to tell the interrupt controller which requesting interrupt is being serviced. The in-service bit also serves to mask all lower priority interrupts when set. It also will prevent additional interrupts from the current interrupt source. Only interrupts having a higher priority level than the current in-service interrupt will cause an interrupt. This allows software to run with interrupts enabled, and only be interrupted if a higher level interrupt occurs.

Polling Interrupts

The interrupt controller can be used for software polling of interrupts if desired. This is accomplished by disabling interrupts, and then reading the poll register whenever possible. When the poll register is read bit 15 indicates if an interrupt request of high enough priority is present. Bits 4-0 will indicate the vector type of the highest priority source requesting service. Bits 4-0 are only valid if bit 15 is one, indicating a valid interrupt source. When reading the poll register and there is an interrupt request the in-service bit for that source will be set.

If it is desirable to poll the interrupt controller without setting the in-service bits, then the poll status register can be read. This register has all the same information as the poll register except that when read it will not set the in-service bit.

Programmable Priority

Each interrupt source is equipped with a programmable priority level within it's control register. These bits can be programmed to one of eight different priorities (000\b - 111\b). Priority level 000\b has the highest priority and 111\b has the lowest priority. These are the bits that determine which interrupt source will have priority over another. If two or more simultaneous interrupts occur with the same programmed priority then priority is determined as shown in Table 11.

End-of-Interrupt Command

End-of-Interrupt (EOI) command is used by software to reset the in-service bit when an interrupt service routine is complete. The EOI command is generated by writing the proper bit pattern to the EOI register. Two types of EOI commands exist, specific and non-specific. A specific EOI command will specify which in-service bit will be reset, while a non-specific will not. To generate a specific EOI command, bit 15 should be loaded with a 0 and S4-0 should be loaded with the vector type corresponding to the in-service bit that will be reset. For non-specific mode the interrupt controller will reset the highest priority source whose in-service bit is set.

Trigger Mode

The external interrupts (INT5-0) can have their inputs configured for either edge-triggered or level-triggered mode. In either mode all interrupt inputs are active high, and should remain active (high) until acknowledged by the CPU. Within the control register for each external interrupt source the LTM bit determines which mode the input will be in. When the inputs are configured for level-triggered mode (LTM=1) an interrupt request will occur when ever the input is pulled high. The input must remain high long enough for the processor to recognize it, if not the interrupt will not be processed by the CPU. If the input continues to stay high it will cause another interrupt only after the first interrupt has been processed. When the inputs are configured for edge-triggered mode (LTM=0) an interrupt request will occur when the input makes a low to high transition. Unlike the level-triggered mode, if the input stays high no additional interrupts are generated until the input goes low for at least one clock cycle.

2.4.1 Interrupt Sources

The following table lists the available interrupt sources along with the interrupt type, vector and default priority for each source.

Table 11 - Interrupt Sources

Interrupt Name	Vector Type	Vector Address	Default Priority
Divide Error Exception	0	00H	1
Single Step Interrupt	1	04H	1A
NMI	2	08H	1B
Breakpoint Interrupt	3	0CH	1
INTO Detected Overflow Exception	4	10H	1
Array Bounds Exception	5	14H	1
Unused Opcode Exception	6	18H	1
ESC Opcode Exception	7	1CH	1
Timer 0	8	20H	2A
Timer 1	18	48H	2B
Timer 2	19	4CH	2C
DMA 0	10	28H	4
DMA 1	11	2CH	5
DMA 2	22	58H	12
DMA 3	23	5CH	13
INT 0	12	30H	6
INT 1	13	34H	7
INT 2	14	38H	8
INT 3	15	3CH	9
INT 4	16	40H	10
INT 5	21	54H	11
SP0	20	50H	15
SP1	17	44H	15
SPI	24	60H	15

At power-up or after hardware reset, all interrupt sources are masked (disabled).

2.4.2 Allocation of external interrupts (INT0 – INT5)

The external interrupts are connected to the following peripheral devices:

Table 12 - External Interrupt Assignments

Interrupt Signal	Assignment
INT0	Ethernet Controller
INT1	CAN0
INT2	CAN1
INT3	Profibus Controller
INT4	Dual Port Memory Flag
INT5	External Interrupt Pin (INT5)

2.4.3 Interrupt control registers

Control registers and associated offsets are listed below.

Table 13 - Interrupt Control Registers

Name	Mnemonic	Offset
End of Interrupt	EOI	22h
Poll	POLL	24h
Poll Status	POLLST	26h
Interrupt Mask	IMASK	28h
Priority Mask	PRIMSK	2Ah
In-Service	INSERV	2Ch
Interrupt Request	REQST	2Eh
Interrupt Status	INTSTS	30h
Timer Control	TCUCON	32h
DMA 0 Control	DMA0CON	34h
DMA 1 Control	DMA1CON	36h
DMA 2 Control	DMA2CON	48h
DMA 3 Control	DMA3CON	4Ah
INT 0 Control	I0CON	38h
INT 1 Control	I1CON	3Ah
INT 2 Control	I2CON	3Ch
INT 3 Control	I3CON	3Eh
INT 4 Control	I4CON	40h
INT 5 Control	I5CON	46h
SP0 Control	SP0CON	44h
SP1 Control	SP1CON	42h
SPI Control	SPICON	4Ch

2.4.4 In-Service Register (INSERV, offset 2Ch)

This register contains the in-service bits for each of the interrupt sources. An in-service bit is set to indicate that a source's service routine is being executed. When an in-service bit is set the interrupt controller will not generate interrupts from sources with a lower priority than the current interrupt being serviced. This allows interrupt service routines to run with interrupts enabled. This register is read/write.

Register 16 - In-Service Register, INSERV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	I2	I1	I0	D1	D0	0	TMR

SPI	In-service bit for the Serial Peripheral Interface
D3:0	In-service bits for the DMA channels (DMA3:0).
I5:0	In-service bits for the external interrupt sources (INT5:0).
SP1:0	In-service bits for the asynchronous serial ports (SP1:0).
TMR	In-service bit for all three timers (TMR2:0).

2.4.5 Interrupt Request Register (REQST, offset 2Eh)

This register contains the interrupt request bits for the Timers, DMA channels and external interrupt sources. This register is read/write except were noted.

Register 17 - Interrupt Request Register, REQST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	I2	I1	I0	D1	D0	0	TMR

SPI	Interrupt request bit for the SPI port.
D3:0	Interrupt request bits for the DMA channels (DMA3:0). Setting any of these bits will generate an interrupt request on the corresponding DMA channel interrupt request line, while resetting any of these bits will remove that interrupt request.
I5:0	Interrupt request bits for the external interrupt sources (INT5:0). These bits represent the actual state of the external interrupt pins, after they have been passed through a two level synchronizer, or a level detector depending on the LTM bit for that respective interrupt line. These bits will be set whenever an external interrupt line make a low to high transition if LTM=0. Otherwise, if LTM=1 these bits will follow the input pins after a two clock delay. Because these bits represent the state of the external interrupt pins they can not be written.
SP1:0	Interrupt request bits for the asynchronous serial ports, a bit is set when the corresponding serial port generates an interrupt request and is cleared when the interrupt acknowledge cycle occurs.
TMR	This bit is the logical OR of all three timer interrupt requests. The individual timer interrupt request bits are contained in the interrupt status register. This bit can not be written.

2.4.6 Interrupt Mask (IMASK, offset 28h)

This register contains the mask bits for each interrupt source. Setting any of these bits to a one will mask interrupts from that source and resetting any of these bits to zero will enable interrupts from that source. Note that these are the same physical mask bits contained in each source control register. Changing the mask bit in this register will also change the mask bit in the control register for that source, and vice versa. This register is read/write. Upon hardware reset or power-up, this register is set to 7FFDh (all interrupt sources masked).

Register 18 - Interrupt Mask Register, IMASK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	I2	I1	I0	D1	D0	0	TMR

SPI	Setting this bit will mask the SPI interrupt.
D3:0	Setting any of these bits will mask the respective DMA channel interrupt (DMA3:0).
I5:0	Setting any of these bits will mask the respective interrupt source (INT5:0).
SP1:0	Setting any of these bits will mask the respective asynchronous serial port interrupt.
TMR	Setting this bit is will mask all timer interrupt requests (TMR2:0).

2.4.7 Priority Mask (PRIMSK, offset 2Ah)

This register is used to mask all interrupts below a particular priority level. This register is read/write.

Register 19 - Interrupt Priority Mask Register, PRIMSK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	PR2	PR1	PR0

PR2:0	These bits represent the minimum priority level an interrupt request must have to be recognized. An interrupt request will be processed buy the interrupt controller if it's priority level is greater than or equal to the priority contained in this register. Bit combination 000\b has the highest priority while 111\b has the lowest priority. This register is set to 111\b upon reset, allowing no interrupts to be masked.

2.4.8 Interrupt Status (INTSTS, offset 30h)

This register contains DMA transfer and timer interrupt status information. This register is read/write.

Register 20 - Interrupt Status Register, INTSTS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DHLT	0	0	0	0	0	0	0	0	0	0	0	0	TMR2	TMR1	TMR0

DHLT	Setting this bit will halt all DMA operations. It is automatically set whenever a NMI occurs, and reset when an IRET instruction is executed. By suspending DMA operations during a NMI the processor can quickly service the NMI request. The programmer may also set this bit.
TMR2:0	These three bits are the individual timer interrupt request bits. These bits allows software to differentiate between the timer interrupts, because the TMR bit in the interrupt request register is the logical OR of all timer requests. Setting any of these bits will generate a timer interrupt request.

2.4.9 Timer, SP1-0, SPI and DMA3-0 Interrupt Control

These are the control registers for the timers, DMA channels and serial channels. A single control register is used for all three timers, while each DMA channel, each serial port and the SPI port each has its own control register. Each control register contains the mask bit and programmable priority level. The mask bits in these control registers is the same mask bits as in the mask register. Modifying them in the control register will also modify the corresponding mask bits in the mask register, and vice versa. This register is read/write.

Register 21 - Internal Peripheral Interrupt Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	MSK	PRI2	PRI1	PRI0

MSK	Setting this bit is will mask the respective interrupt requests. Resetting this bit will enable the respective interrupts.
PR2:0	These bits represent the programmable priority level for the respective interrupt source. Bit combination 000\b has the highest priority while 111\b has the lowest priority.

2.4.10 INT5-0 Interrupt Control

These are the control registers for the external interrupt sources. Each interrupt source has its own control register. Each control register contains the level-trigger mode bit, mask bit and programmable priority level. The mask bits in these control registers is the same mask bits as in the mask register. Modifying them in the control register will also modify the corresponding mask bits in the mask register, and vice versa. This register is read/write

Register 22 - External Interrupt Control Registers, IxCON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	LTM	MSK	PRI2	PRI1	PRI0

LTM	The level-trigger mode bit will set the respective interrupt source for either level triggered mode (LTM=1), or edge triggered mode (LTM=0). When set for level-triggered mode an interrupt is generated whenever the external interrupt signal is high. In edge triggered mode an interrupt request is generated when the external interrupt signal makes a low to high transition. In both cases, the level must remain high until the interrupt is acknowledged.
MSK	Setting this bit is will mask the respective interrupt requests. Resetting this bit will enable the respective interrupts.
PR2:0	These bits represent the programmable priority level for the respective interrupt source. Bit combination 000\b has the highest priority while 111\b has the lowest priority.

2.4.11 End of Interrupt (EOI, offset 22h)

When the end of interrupt register is written to it initiates an EOI command to the interrupt controller. This register is write only.

Register 23 - End of Interrupt Register, EOI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSPC	0	0	0	0	0	0	0	0	0	0	S4	S3	S2	S1	S0

NSPC	This bit determines the type of EOI command. A non-specific EOI command will be issued if NSPC=1, otherwise if NSPC=0 a specific EOI command will be issued.
S4:0	These bits represent the encoded vector type of an interrupt source as shown in Table 3. Writing the interrupt vector type to these bits will reset the corresponding in-service bit for that interrupt source. For example, to reset the in-service bit for INT0, these bit would be set to 01100\b, because the vector type for INT0 is 12. NOTE: To reset the single in-service bit for the timers, the vector type for Timer0 (8) should be written.

2.4.12 Poll and Poll Status (POLL, offset 24h; POLLST, offset 26h)

These registers contain interrupt polling information. Both registers have exactly the same information. The only difference is that reading the poll register generates a software poll. This will set the in-service bit for the highest priority pending interrupt. Reading the poll status register will not set the in-service bit. These registers are read only.

Register 24 - Poll and Poll Status Registers, POLL / POLLST

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQ	0	0	0	0	0	0	0	0	0	0	S4	S3	S2	S1	S0

IRQ	This bit determines if an interrupt request is pending. If set an interrupt request is present, otherwise it is reset.
S4:0	These bits contain the encoded vector type of the highest priority interrupt source. NOTE: These bits are only valid if IRQ=1.

Reset will initialize the interrupt controller register bits to the following values:

- All priority request bits (PR) are set to 111\b, which is the lowest priority level.
- All interrupt mask bits (MSK) are set, disabling interrupts.
- The level trigger mode bit (LTM), for external interrupt sources is reset to 0, enabling edge-sensitive mode.
- All interrupt service and interrupt request bits are reset to 0.

2.5 DMA Controller

Direct memory access (DMA) permits transfer of data between memory and peripherals without CPU involvement. The DMA unit in the DSTni-LX microcontroller provides four high-speed DMA channels (DMA3 - DMA0). Data transfers can occur between memory and I/O spaces (e.g., memory to I/O) or within the same space (e.g., memory-to-memory or I/O-to-I/O).

The DMA channels can be directly connected to the asynchronous serial ports. DMA/ asynchronous serial port transfer is accomplished by programming the DMA controller to perform transfers between a data source in memory or I/O space and an asynchronous serial port transmit or receive register.

DMA3 and DMA2 are allocated exclusively to Serial Port 1. DMA1 and DMA0 may be for Serial Port 0 or for other purposes.

The DMA controller supports 24-bit addresses. This transfer will be a synchronized transfer, and synched to the receiver or the transmitter, based on the direction of the transfer.

Six registers in the peripheral control block define the operation of each channel. The DMA registers consist of a 24-bit source address (2 registers), a 24-bit destination address (2 registers), a 16-bit transfer count register, and a 16-bit control register.

The DMA transfer count register (DTC) specifies the number of DMA transfers to be performed. Up to 64 Kbytes or 64 Kwords can be transferred with automatic termination. The DMA control registers define the channel operations. All registers can be modified or altered during any DMA activity. Any changes made to these registers are reflected immediately in DMA operation.

Table 14 - DMA Controller Register Map

Register Name	Mnemonic	Register Offset			
		Ch0	Ch1	Ch2	Ch3
Control Word	DxCON	CAH	DAH	BAH	9AH
Transfer Count	DxTC	C8H	D8H	B8H	98H
Destination Pointer (upper 8 bits)	DxDSTH	C6H	D6H	B6H	96H
Destination Pointer (lower 16 bits)	DxDSTL	C4H	D4H	B4H	94H
Source Pointer (upper 8)	DxSRCH	C2H	D2H	B2H	92H
Source Pointer (lower 16 bits)	DxSRCL	C0H	D0H	B0H	90H

2.5.1 DMA Channel Control Register (D0CON – D3CON)

Each DMA channel has a control word to control its operation. The control word is identical for all the channels. This is a read/write register.

Register 25 - DMA Control Registers, DxCON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Destination			Source			TC	INT	SYN		P	TDRQ	0	CHG/ NCHG	ST/ STP	B/W
MIO	DEC	INC	MIO	DEC	INC										

Destination Pointer

MIO	This bit determines if the destination is memory space (bit = 1) or I/O space (bit = 0).
DEC	If this bit is set to 1, the destination pointer will be decremented after each access. The decrement will be by one or two, depending if DMA is configured for byte or word operation.

INC If this bit is set to 1, the destination pointer will be incremented after each access. The increment will be by one or two, depending if DMA is configured for byte or word operation.

Note: If both INC and DEC bits are set, the pointer will remain unchanged after each cycle.

Source Pointer

MIO This bit determines if the source is memory space (bit = 1) or I/O space (bit = 0).

DEC If this bit is set to 1, the source pointer will be decremented after each access. The decrement will be by one or two, depending if DMA is configured for byte or word operation.

INC If this bit is set to 1, the source pointer will be incremented after each access. The increment will be by one or two, depending if DMA is configured for byte or word operation.

Note: If both INC and DEC bits are set, the pointer will remain unchanged after each cycle.

TC If this bit is set to 1, the DMA operation will terminate when the value of the transfer count register reaches zero. The ST/STOP bit will also be reset at this point. If this bit is set to 0, DMA operation will not terminate when the count register equals zero.

INT If this bit is set to 1, an interrupt will be generated upon transfer count termination.

SYN These two bits determine the type of synchronization used. SYN1 is bit 7 and SYN0 is bit 6.

SYN1	SYN0	
0	0	No synchronization. DMA transfer take place regardless of the DMA request line. The TC bit is ignored and ST/STOP is cleared upon transfer count reaching zero, stopping the channel. Back to back transfer cycles are possible.
0	1	Source Sync. DMA transfers can take place when the source signals a DMA request via the DRQx signal. Back to back transfer cycles are possible.
1	0	Destination Sync. DMA transfers can take place when the destination signals a DMA request via the DRQx signal. After each transfer cycle the controller waits two clocks before starting the next cycle.
1	1	Not used

P Channel priority, relative to the other DMA channels. Set to 1 for high priority or 0 for low priority. If two or more channels having the same priority have simultaneous requests, the channels will alternate cycles.

TDRQ Set this bit to 1 to enable a channel's DMA request to come from Timer 2 instead of the external DMA request signal. (External to DMA control block. See Interrupt Control)

CHG/NCHG If this bit is 0 when a write is performed to the control register, the contents of ST/STOP cannot be changed by the write. If this bit is 1 during a write to the control register, ST/STOP can be changed.

ST/STP Set this bit to 1 to start a DMA transfer. Clear this bit to stop a DMA transfer. The CHG/NCHG bit must be set to 1 during the write to allow ST/STP to be changed.

B/W Set this bit 1 to select word transfers and set it to 0 to select byte transfers. This also affects whether the source and/or destination pointer is incremented by one or two.

2.5.2 Source and Destination Pointer Register

Each DMA channel is equipped with two 24-bit address pointers, A source pointer and a destination pointer. Each of these registers requires two 16-bit registers to implement one address pointer. The lower 16-bits of address are contained in the lower register, the remaining 8-bits are contained in the lower byte of the upper register. Either of the address pointers may be configured to increment, decrement, or not change during each DMA operation. The pointer registers will be incremented or decremented by two in word mode, or by one in byte mode. Either of the address pointers may be configured to point to memory or I/O. For I/O operations the upper address register, bits A23-16, should be programmed to 0 for DMA operations to or from I/O space. Like the CPU, the DMA controller can perform accesses to or from even and odd addresses. Word accesses to odd address will be broken into two bus cycles. To achieve the highest possible data transfer rates align word transfers to even word addresses.

Register 26 - DMA Destination Address High Register, DxDSTH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	A23	A22	A21	A20	A19	A18	A17	A16

A23-A16	High order address bits. Set to 0 for DMA operations to or from I/O space.

Register 27 - DMA Destination Address Low Register, DxDSTL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

A15-A0	Low order address bits. The sixteen bits of this register are combined with the eight bits of the Destination High Address register to form a 24-bit address.

Register 28 - DMA Source Address High Register, DxSRCH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	A23	A22	A21	A20	A19	A18	A17	A16

A23-A16	High order address bits. Set to 0 for DMA operations to or from I/O space.

Register 29 - DMA Source Address Low Register, DxSRCL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0

A15-A0	Low order address bits. The sixteen bits of this register are combined with the eight bits of the Source High Address register to form a 24-bit address.

2.5.3 Transfer Count Register (D0TC – D3TC)

Each DMA channel has a 16-bit transfer count register (TC). This register is decremented after every DMA cycle, regardless of the state of the TC bit in the DMA control register. If the TC bit in the DMA control register is set or if configured for unsynchronized transfers, DMA operations will be terminated when the value in the transfer count register reaches zero.

Register 30 - DMA Transfer Count Register, DxTC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0

C15-C0	DMA Transfer Count. Value decremented after every DMA cycle.

2.5.4 DMA Requests

DMA requests maybe be implicit, or explicit. For implicit requests, i.e. unsynchronized mode, no external DMA request is needed. The DMA request is "implied" by this mode of synchronization. DMA bus cycles will take place when the ST/STP bit is set to 1. In unsynchronized mode DMA cycles will take place, back to back, until the correct number of transfers have occurred. For explicit requests, i.e. source or destination synchronization, an external request must be generated for the transfer to begin. This request can come from the external DMA request signals (DRQ3:0) or from the Timer request signal (TMRQ). Source synchronization mode can yield the same data transfer rates as unsynchronized mode. For destination synchronization, the DMA controller releases the bus after every bus cycle. This allows the destination time to remove it's request if it can not handle another DMA cycle. After two clocks the DMA controller will perform another DMA bus cycle, if the request is still true. During these two clocks it is possible that the CPU will execute a bus cycle. If this occurs the second DMA cycle will be delayed by an additional clock, assuming no wait states.

2.5.5 DMA Acknowledge

There is no explicit DMA acknowledge signal or bus cycle provided from the DSTni-LX.

2.5.6 DMA Priority

Each DMA channel has a programmable priority bit within the control register. This allows certain DMA channels to have priority over others. If all DMA channels are programmed to the same priority level, and have pending DMA operations, the requesting channels will alternate DMA cycles. DMA bus cycles also have priority over all internal CPU cycles except between locked memory accesses or word accesses to odd addresses. Only external bus hold can take priority over an internal DMA operation. Because the DMA cycles have priority over CPU cycles interrupt latency times can increase during DMA operations.

2.5.7 DMA Programming

When programming the DMA channels the source pointer, destination pointer, and transfer count register (if used) must be programmed before any DMA operations commence. DMA operations occur whenever the ST/STP bit in the control register is set. If source or destination synchronized transfers are enabled, a corresponding DRQ signal must also be asserted. Any of the DMA channel registers may be modified during operations. If multiple registers are to be modified, it is recommended that a locked string transfer be used to prevent a DMA operation from occurring between updates to the channel registers.

2.5.8 DMA Controller Reset

A reset asserted to the DMA controller will perform the following:

- The ST/STP bit for each channel is reset to stop.
- Any DMA operations in progress are aborted.
- The values in the source pointer, destination pointer, and transfer count register are undefined.

2.6 Asynchronous Serial Ports

The DSTni-LX supports two serial ports.

Serial Port 0 register set is located at offset 80h from the Peripheral Control Block. Its DMA channels are DMA0 and DMA1.

Serial Port 1 register set is located at offset 10h from the Peripheral Control Block. The four pins used by the serial port interface, SP1_RXD, SP1_TXD, SP1_RTS, SP1_CTS can be reassigned for use by the Profibus interface by setting a bit in the Profibus Extended Control register. Its DMA channels are DMA3 and DMA2.

Both serial ports have a 4-word FIFO to enhance the support for high-speed protocols. Each port also has an additional register for control of RS-485 signals.

2.6.1 Receive FIFO

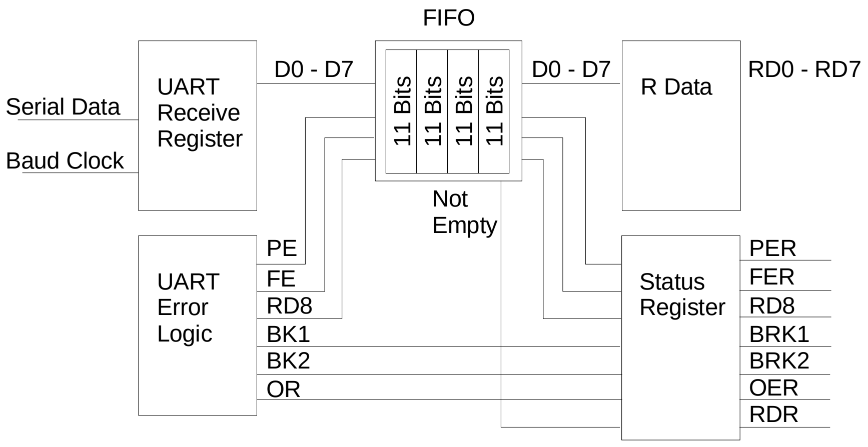
When receiving at a high baud rate, there is always a concern for having data overruns. This occurs when the software or DMA was not able to read the receive register before the next character was transferred into the receive register. In order to lessen the possibility of this condition, a 4 word FIFO is used as the receive register. The FIFO is 8 data bits wide and includes several flag bits as well.

The Receive Data Ready (RDR) status bit in the Serial Port Status Register is effectively the FIFO Not Empty flag. The bit is set when the FIFO holds one or more characters. This status bit is cleared when the FIFO is empty.

The Overrun status bit in the Serial Port Status Register is set when the FIFO is full and another character has arrived in the UART. Characters that generate an Overrun Error cannot be entered into the FIFO because the FIFO is already full.

The FIFO is 11 bits wide which includes 8 data bits, Framing Error, Parity Error, and Receive Bit 8. The FIFO will capture Receive Bit 8 that is available in Serial Modes 2 and 3. The FIFO will supply these bits to the Serial Port Status Register.

Once the number of bits have been received, all data bits and the corresponding error flags are clocked into the top (newest) location of the FIFO. The data bits and error bits move down thru the FIFO when unloaded. Once the error bit arrives at the bottom of the FIFO, it will set a register which can then only be cleared by software (which must first unload the FIFO by reading the data register).



10	9	8	7	6	5	4	3	2	1	0
PER	FER	RB8	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

PER	Parity Error
FER	Framing Error
RB8	Receive Bit 8. High bit available in Serial Modes 2 and 3.
RB7:0	Receive Bits 7-0

The FIFO will be advanced one position each time the receive register is read. This means that the status register can be read many times, and not advance the FIFO. However, it also means that the status register should be read first to determine the states of the error status bits or Receive Bit 8.

DMA operation should work in its normal fashion on receive. It will get its data from the receive register, which just happens to be the FIFO.

2.6.2 Asynchronous Serial Port Register Summary

The SP0RS485 and SP1RS485 registers are additions to the standard serial port registers.

Table 15 - Serial Port Registers

Register Name	Mnemonic	Offset	Section
Serial Port 0 Control	SP0CT	80h	2.6.3
Serial Port 0 Status	SP0STS	82h	2.6.4
Serial Port 0 Baud Rate Divisor	SP0BAUD	88h	2.6.7
Serial Port 0 Read Receive	SP0RD	86h	2.6.6
Serial Port 0 Transmit	SP0TD	84h	2.6.5
Serial Port 0 RS485/422 Control	SP0RS485	8Ah	2.6.8
Serial Port 1 Control	SP1CT	10h	2.6.3
Serial Port 1 Status	SP1STS	12h	2.6.4
Serial Port 1 Baud Rate Divisor	SP1BAUD	18h	2.6.7
Serial Port 1 Read Receive	SP1RD	16h	2.6.6
Serial Port 1 Transmit	SP1TD	14h	2.6.5
Serial Port 1 RS485/422 Control	SP1RS485	1Ah	2.6.8

2.6.3 Serial Port 0/1 Control (SP0CT, offset 80h; SP1CT, offset 10h)

The control register description is identical for both SP1 and SP0:

Register 31 - Serial Port 0 and 1 Control Registers, SP0CT, SP1CT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA			RSIE	BRK	TB8	FC	TXIE	RXIE	TMOD	RMOD	EVN	PE	MODE		

DMA	These three bits select the way SP0 and SP1 use DMA controller channels. For Serial Port 1, the mode bits are decoded as shown in Table 16. For Serial Port 0, the mode bits are decoded as shown in Table 17.

Table 16 - Serial Port 1 DMA modes

DMA bits	Receive	Transmit
000	No DMA	No DMA
001	DMA2	DMA3
010	DMA3	DMA2
011	Reserved	Reserved
100	DMA2	No DMA
101	DMA3	No DMA
110	No DMA	DMA2
111	No DMA	DMA3

Table 17 - Serial Port 0 DMA modes

DMA bits	Receive	Transmit
000	No DMA	No DMA
001	DMA0	DMA1
010	DMA1	DMA0
011	Reserved	Reserved
100	DMA0	No DMA
101	DMA1	No DMA
110	No DMA	DMA0
111	No DMA	DMA1

DMA transfers to a serial port are destination-synchronized DMA transfers. A new transfer is requested when the transmit holding register is empty. This corresponds with the assertion of the THRE bit in the serial port status register in non-DMA mode. When the serial port is configured for DMA transmits, the corresponding transmit interrupt is disabled regardless of the setting of the TXIE bit.

DMA transfers from the serial port are source-synchronized DMA transfers. A new transfer is requested when the serial port receive register contains valid data. This corresponds with the assertion of the RDR bit in the serial port status register in non-DMA mode. When the port is configured for DMA receivers, the corresponding receive interrupt is disabled regardless of the setting of the RXIE bit. Receive status interrupts may still be taken, as configured by the RSIE bit.

Hardware handshaking may be used in conjunction with serial port DMA transfers.

RSIE	Receive Status Interrupt Enable. The bit enables the serial port to generate in interrupt request when a exception occurs during data reception. When this bit is set, interrupt requests are generated from the error conditions reported in the serial port status register (BRK0, BRK1, OER, PER, FER).
BRK	Send Break. Setting this bit will cause the TxD pin to be driven Low.
TB8	Transmit Bit 8. This bit is transmitted as the ninth data bit in modes 2 and 3. This bit is not buffered and is cleared after every transmission. In order to transmit a character with the 8th data bit high, this bit must be set in the control register after the TEMT bit in the status register becomes set. The character to be transmitted is then written to the transmitter holding register.
FC	Flow Control Enable. When this bit is set, hardware flow control is enabled for the associated serial port. When this bit is cleared, hardware flow control is disabled. The nature of the flow control signals is determined by the setting of the ENRX0/ENRX1 and RTS0/RTS1 bits in the AUXCON register.
TXIE	Transmitter Ready Interrupt Enable. When this bit is set, the serial port generates an interrupt request whenever the transmit holding register is empty (THRE bit in the status register is set), indicating that the transmitter is available to accept a new character for transmission. When this bit is cleared, the serial port does not generate transmit interrupt requests. Interrupt requests continue to be generated as long as the TXIE bit is set and the transmitter is empty, i.e. the THRE bit in the status register remains set.
RXIE	Receive Data Ready Interrupt. When this bit is set, the serial port generates an interrupt request whenever the receive FIFO contains valid data (RDR bit in the status register is set). When this bit is cleared, the serial port does not generate receive interrupt requests. Interrupt requests continue to be generated as long as the RXIE bit is set and the FIFO contains unread data (the RDR bit in the status register is set).
TMOD	Transmit Mode. When this bit is set, the transmit section of the serial port is enabled. When this bit is cleared, the transmitter and the transmit interrupt requests are disabled.
RMOD	Receive Mode. When this bit is set, the receive section of the serial port is enabled. When this bit is cleared, the receiver is disabled.
EVN	Even Parity. This bit determines the parity sense. When EVN is set, even parity checking is enforced (even number of 1's in frame). When EVN is cleared, odd parity checking is enforced (odd number of 1's in frame). Note: This bit is valid only when the PE bit (parity enable) is set.
PE	Parity Enable. When this bit is set, parity checking is enabled. When this bit is cleared, parity checking is disabled.

MODE Serial Mode of Operation. This field determines the operating mode of the serial port. See the following table:

Mode Bits	Description	Data Bits	Parity Bits	Stop Bits
000	Reserved			
001	Mode 1	7 or 8	1 or 0	1
010	Mode 2	9	N/A	1
011	Mode 3	8 or 9	1 or 0	1
100	Mode 4	7	N/A	1
101	Reserved			
110	Reserved			
111	Reserved			

Mode 1 This mode supports 7 data bits when parity is enabled or 8 data bits when parity is disabled. When using parity, the eighth bit becomes the parity bit and is generated for transmits, or checked for receives automatically by the processor.

Mode 2 When configured in this mode, the serial port receiver will not complete a data reception unless the ninth data bit is set. Any character received with the ninth data bit cleared is ignored.

Mode 3 This mode supports 8 data bits when parity is enabled or 9 data bits when parity is disabled. When not using parity, the ninth bit (bit 8) for transmission is set by writing a 1 to the TB8 field in the serial port control register. The ninth data bit for receive can be read in the RB8 field in the serial port status register.

Mode 4 This mode supports 7 data bits, one start bit, and one stop bit. Parity is not available.

2.6.4 Serial Port 0/1 Status (SP0STS, offset 82h; SP1STS, offset 12h)

The Serial Port Status Register bit assignments are as follows:

Register 32 - Serial Ports 0 and 1 Status Registers, SP0STS, SP1STS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	BRK1	BRK0	RB8	RDR	THRE	FER	OER	PER	TEMT	HSO	0

BRK1 Long Break Detected. This bit is set when a long break is detected on the asynchronous serial interface. A long break is defined as a ‘0’ signal on the RXD pin for greater than 2M+3 bit times, where M = (start bit + # data bits + #parity bits + stop bit).

If a serial port is receiving a character when the break begins, the reception of the character will be completed (generating a framing error) before timing for the break begins. To guarantee detection with a specified 2M+3 bit time, the break must begin outside of a frame.

Note: This bit should be reset by software.

BRK0	Short Break Detected. This bit is set when a short break is detected on the asynchronous serial interface. A short break is defined as a ‘0’ signal on the RXD pin for greater than M bit times, where M = (start bit + #data bits + # parity bits + stop bit). If the serial port is receiving a character when the break begins, the reception of the character will be completed (generating a framing error) before timing for the break begins. To guarantee detection with the specified M bit times, the break must begin outside of a frame. **Note:** This bit should be reset by software.
RB8	Received Bit 8. This bit contains the ninth data bit received in modes 2 and 3. **Note:** This bit should be reset by software.
RDR	Receive Data Ready. The Receive Data Ready bit indicates whether a character is available in the FIFO. When set, the FIFO contains one or more characters to be read from the data register. When cleared, the FIFO is empty. This bit will be set whenever a character is entered into the FIFO. This bit will be cleared only when the FIFO is empty. If a serial port is using DMA to transfer characters from the data register to memory, this bit is used as the DMA request signal.
THRE	Transmit Holding Register. When this bit is set, the transmit holding register is ready to accept data for transmission. This bit is read-only. Writing data to the corresponding transmit holding register is the only way to clear this bit.
FER	Framing Error Detected. A Framing Error occurs when the stop bit of the character is detected as a logic 0 (spacing state). The Framing Error Flag is set when this condition occurs. It is cleared when the condition does not exist. The Framing Error Flag is loaded into the FIFO at the same time the received character is loaded into the FIFO. The Framing Error Bit is an output from the FIFO, and reflects the Framing Error status of the character at the top of the FIFO. **Note:** This bit should be reset by software.
OER	Overrun Error Detected. An Overrun Error occurs when the FIFO is full AND the receiver buffer register is full, AND the next character is transferred to the receiver buffer register. When this condition occurs, the character in the receive register is overwritten and the Overrun Error Flag is set. The FIFO contents remain unchanged. **Note:** This bit should be reset by software.
PER	Parity Error Detected. The Parity Error Flag is enabled only when using serial port modes 1 or 3. If Parity is not enabled, the Parity Error Flag will remain cleared. The Parity Error Flag is set when a Parity Error is detected. It is cleared when there is no Parity Error. The Parity Error Flag is loaded into the FIFO at the same time the received character is loaded into the FIFO. The Parity Error bit is an output from the FIFO, and reflects the parity status of the character at the top of the FIFO. **Note:** This bit should be reset by software.
TEMT	Transmitter Empty. This bit is set when the transmitter has no data to transmit and the transmit shift register is empty. This indicates to software that is safe to disable the transmit section. This bit is read-only.
HS0	Handshake Signal 0. This bit reflects the inverted value of the external CTS_n pin. If CTS_n is asserted, then HS0 is set to ‘1’. This bit is read-only.

2.6.5 Serial Port 0/1 Transmit (SP0TD/SP1TD, offset 84h/14h)

The value to be transmitted over the serial interface is written to the transmit register. The transmitter is double buffered. The data from the transmit register is transferred to the transmit shift register before transmitting. The state of the transmit register is reflected in the THRE status bit. This bit is a '1' when the transmit register is available to accept the next character. The state of the transmit shift register is reflected in the TEMT status bit. This bit is a '1' when the transmit buffer is empty.

When hardware handshaking is enabled, the transmitter will not transmit data while the RTS/RTR inputs are a '1'. Data is held in the transmit and transmit shift registers without affecting the transmit pin.

Register 33 - Serial Port 0 Transmit Registers, SP0TD, offset 84h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0

Register 34 - Serial Port 1 Transmit Registers, SP1TD, offset 14h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0

D7:0 Serial transmit data.

2.6.6 Serial Port 0/1 Receive (SP0RD/SP1RD, offset 86h/16h)

The value received from the serial interface is read from the receive register. The receiver is a 4-word FIFO, so another character can be received while the receive register is being read. The state of the FIFO is reflected in the RDR bit in the serial port status register. This bit is set when unread data is available in the FIFO. The bit is cleared when the FIFO becomes empty. The FIFO will contain 8 data bits, the Parity Error Bit, and the Framing Error bit for each character. All characters will be entered into the FIFO, even if there is a Parity Error or a Framing Error. Characters that generate an Overrun Error cannot be entered into the FIFO because it is already full.

When hardware handshaking is enabled, the CTS/ENRX signals are deasserted while the receive register contains valid unread data. Reading the receive register causes the CTS/ENRX signals to be asserted. The behavior prevents overrun errors, but may result in delays between character transmissions.

Register 35 - Serial Port 0 Receive Registers, SP0RD, offset 86h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0

Register 36 - Serial Port 1 Receive Registers, SP1RD, offset 16h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0

D7:0 Serial receive data.

2.6.7 Serial Port 0 and 1 Baud Rate Divisor (SP0BAUD/SP1BAUD, offset 88h/18h)

Each serial port has its own baud rate divisor register, so the serial ports can operate at different baud rates. The serial clock rate is 16 times the baud rate of transmission or reception of data. The register specifies the number of internal processor cycles in one phase of the 16x serial clock.

Register 37 - Serial Port 0 Baud Rate Divisor Register, SP0BAUD, offset 88h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Register 38 - Serial Port 1 Baud Rate Divisor Register, SP1BAUD, offset 18h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

D15:0 Baud rate divisor value.

The formula for the baud rate divisor is: BAUDDIV = CPU Clock / (16 * baud rate) The serial port receiver can tolerate a 3.0% overspeed and a 2.5% underspeed baud rate deviance. The maximum baud rate is 1/16 of the internal processor clock and is achieved by setting the BAUDDIV = 0001h. This results in 3000 Kb at 48 MHz.

The serial ports sample the data at the 70% point in the bit time, rather than the 50% point. This may change the range of acceptable off-limit data.

Table 18 - Common Baud Rates

Baud Rate	48 MHz CPU Clock Rate	Percent of Error
300	10000	0
600	5000	0
1200	2500	0
2400	1250	0
4800	625	0
9600	312	0.16
14400	208	0.16
19200	156	0.16
28800	104	0.16
38400	78	0.16
57600	52	0.16
76800	39	0.16
115200	26	0.16
187500	16	0
230400	13	0.16

2.6.8 Serial Port 0/1 RS485 / RS422 Control (SP0RS485/SP1RS485, offset 8Ah/1Ah)

This register controls the enhanced operation of Serial Ports 0 and 1. During a power up sequence, or hardware reset, this register is cleared. The bit assignments for this register are as follows:

Register 39 - Serial Port 0 RS485 / RS422 Control Register, SP0RS485, offset 8Ah

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	0	0	0	0	0	0	0	0	RTS	RXM	CTSM	RTSP

Register 40 - Serial Port 1 RS485 / RS422 Control Register, SP1RS485, offset 1Ah

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	0	0	0	0	0	0	0	0	RTS	RXM	CTSM	RTSP

RTS	This read/write bit provides control over the state of the RTS_n pin. If the FC (Flow Control) bit within the serial port control register is set, then this bit has no effect. When FC is 0, then setting the RTS bit will force the RTS_n output to its active state and clearing the RTS bit will force the RTS_n output to its inactive state.
RXM	Receiver Mode. When this bit is set, the serial port is placed in half-duplex mode. In this mode, the RTS signal determines the input to the serial port receiver. When the RTS signal is asserted, then the serial port receiver input is forced to a ‘1’. When the RTS signal is de-asserted, then the serial port receiver input is from the normal RXD input pin. When the RXM bit is cleared, the serial port is in full-duplex mode, and the serial port receiver input is from the normal RXD input pin.
CTSM	CTS Mode. When this bit is 1, the CTS signal is internally forced to its active state, regardless of the state of the CTS_n input pin. When this bit is 0, CTS will follow the state of the CTS_n pin. This permits the use of the RTS/CTS hardware flow control logic to be used to control the state of the TXEN pin of the SN75176 RS-485 driver without any external logic to simulate the RTS/CTS handshake protocol.
RTSP	RTS polarity. When this bit is set, the RTS signal is asserted as a ‘1’. When this bit is cleared, the RTS signal is asserted as a '0’. This permits the RTS pin to control the TXEN pin of RS-485 drivers such as the SN75176 without any external logic to invert the polarity of the RTS signal.

2.7 Serial Peripheral Interface (SPI) Port

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that is standard across many microprocessors and other peripheral chips. It provides support for a high bandwidth (1megabaud) network connection amongst CPUs and other devices supporting the SPI.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPI's. During a data transfer, the DSTni-LX acts as the “master” which controls the data flow, while the other system acts as the “slave” which has data shifted into and out of it by the master.

The SPI system consists of two data lines and two control lines:

Master Out Slave In

Abbreviated MOSI, this data line supplies the output data from the master which is shifted into the input(s) of the slave(s).

Master In Slave Out

Abbreviated MISO, this data line supplies the output data from a slave to the input of the master. There may be no more than one slave which is transmitting data during any particular transfer.

Serial Clock

Abbreviated SCLK, this control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SCLK line cycles once for each bit that is transmitted.

Slave Select

This control line allows slaves to be turned on and off with hardware control.

The SPI bus is a 3-wire bus that in effect links a serial shift register between the "master" and the "slave". This design will fully support the “master” SPI operation. Typically both the master and slave have an 8 bit shift register so the combined register is 16 bits. When an SPI transfer takes place, the master and slave shift their shift registers 8 bits and thus exchange their 8 bit register values.

The DSTni-LX implementation of the SPI interface includes a total of five I/O port pins. The four standard SPI pins are MSCS(Slave Select), SCLK, MISO, and MOSI. A fifth pin, DFLASH, is allocated to select the serial flash device. This pin is always available. See the Serial Flash circuit diagram on page 84.

The DSTni-LX SPI interface is software configurable. The clock polarity, clock phase, the clock frequency in master mode, and the number of bits to be transferred are all software programmable. Multiple masters are also fully supported and some support is provided for detecting collisions when multiple masters attempt to transfer at the same time.

A Wired-OR mode is provided which allows multiple masters to collide on the bus without risk of damage. In this mode, an external pull-up resistor is required on the MOSI (Master Out Slave In) and MISO (Master In Slave Out) pins. The Wired-OR mode also allows the SPI bus to operate as a 2 wire bus by connecting the MOSI and MISO pins together to form a single bi-directional data pin. Generally, pull-ups are recommended on all of the external SPI signals to ensure they are held in a valid state even when the DSTni-LX SPI interface is disabled.

The DSTni-LX SPI interface includes an internal interrupt connection, SPI interrupt. In SPI Master mode, an SPI interrupt will occur when the most recent transfer is completed. A familiar Interrupt Control register is provided for the SPI interrupt (more details below). An SPI bit is included in each of the Interrupt Mask, Interrupt Request, and In-Service Registers.

2.7.1 SPI Register Summary

The SPI uses 5 registers to configure and control its operation.

Table 19 - SPI Registers

Register Name	Mnemonic	Offset
SPI Data	SPIDATA	68h
SPI Control	SPICTRL	6Ah
SPI Status	SPISTAT	6Ch
SPI Slave Select	SPISSEL	6Eh
SPI Interrupt Control	SPICON	4Ch

2.7.2 SPI Data (SPIDATA, Offset = 68h)

The SPI interface contains a register for reading and writing the 8-bit input or output data. This is the actual data that is transferred over the SPI MISO and MOSI lines.

Register 41 - SPI Data Register, SPIDATA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	SPID 07	SPID 06	SPID 05	SPID 04	SPID 03	SPID 02	SPID 01	SPID 00

SPID07:00	Contains the input data from the receive FIFO when doing a read. Following a write operation, this register contains the data that will be loaded into the SPI transmit register. Note the value of this register at reset is undefined.

2.7.3 SPI Control (SPICTRL, Offset = 6Ah)

The SPI interface contains a register for configuring the port operation. The value of this register at reset is 0000h.

Register 42 - SPI Control Register, SPICTRL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	IRQEN	DVD1	DVD0	PHASE	CKPOL	WOR	MSTEN	0

IRQEN	Interrupt Request Enable. Set to 1 to enable the SPI to generate interrupts. This corresponds to the “Mask” bit in the SPI Interrupt control register.
DVD	Divisor Select. Chooses one of the following clock speeds when the SPI is operating in master mode.

DVD1	DVD0	Clock Divisor	Clock Rate @ 48MHz
0	0	CLK divided by 52	923,000
0	1	CLK divided by 12	4,000,000
1	0	CLK divided by 6	8,000,000

PHASE	Used to select one of two modes of operation for the SPI interface. The two modes select where the opposite edge D-Flip-Flop is placed. When PHASE=0, a negative edge flop is inserted into the shift_in path. When PHASE=1, the negative edge flop is inserted into the shift_out path to hold the data for an extra 1/2 clock.
CKPOL	Controls the polarity of the SCLK (SPI clock). If CKPOL is “1”, SCLK idles “high”. If CKPOL is “0”, SCLK idles “low”.
WOR	Wired-OR. If this bit is “high”, the WOR bit configures the SPI bus to operate in an Open-Drain fashion typically used to prevent conflicts on the SPI bus when there are multiple bus masters.
MSTEN	Master Enable. When set, this bit causes the SPI port to assert itself on the SPI bus as a master. When this bit is cleared, the SPI port is passive, allowing other SPI masters, if present, to drive the bus.

2.7.4 SPI Interrupt Control (SPICON, Offset = 4Ch)

This control register contains the mask bit and programmable priority level. The mask bit in this control register is the same mask bit as in the mask register. Modifying the bit in the control register will also modify the corresponding mask bit in the mask register, and vice versa. This register is read/write.

Register 43 - SPI Interrupt Control Register, SPICON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	MSK	PRI2	PRI1	PRI0

MSK	Setting this bit will mask the SPI interrupt request. Resetting this bit will enable the SPI interrupt.
PR2:0	These bits represent the programmable priority level for the SPI interrupt. Bit combination 000\b has the highest priority while 111\b has the lowest priority.

2.7.5 SPI Status (SPISTAT, Offset = 6Ch)

The SPI interface contains a register for viewing specific hardware or software states. Unless otherwise noted, clearing the individual status bits to a zero is achieved by writing a one to the respective bit.

Register 44 - SPI Status Register, SPISTAT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	IRQ	OVER-RUN	COL	0	0	0	TXRUN	SLVSEL

IRQ	Interrupt Request. Set at the end of a master mode transfer, or when SLVSEL goes high on a slave transfer.
OVERRUN	Set when the SPIDATA register is written to while an SPI transfer is already in progress.
COL	Set when there is a master mode collision between multiple SPI masters. It is set when SLVSEL goes low while MSTEN = 1.
TXRUN	Set when a Master mode operation is underway.
SLVSEL	Corresponds to the external MSCS_n pin on the DSTni-LX. Read only. When set, indicates an external master is attempting to drive the DSTni-LX as an SPI slave.

2.7.6 SPI Slave Select (SPISSEL, Offset = 6Eh)

This register controls which slave is selected and also the number of bits to shifted during an exchange.

Register 45 - SPI Slave Select Register, SPISSEL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	BCNT2	BCNT1	BCNT0	0	0	0	0	DFLASH

BCNT2:0 Bit Shift Count. These bits control the number of bits shifted between the master and slave device during a transfer where this device is the master.

BCNT2	BCNT1	BCNT0	# of Bits Shifted
0	0	0	8
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

DFLASH SPI Slave Select. In Master mode, this bit is used to activate the DFLASH_n select line to the serial flash device. The standard PIO bits can be used for additional SPI Slave Selects. Setting this bit to 1 causes the DFLASH_n signal to be driven low (active). Setting this bit to 0 causes DFLASH_n to be driven high. This bit is 0 upon hardware reset.

2.8 Programmable I/O Pins

There a total of 32 programmable I/O pins numbered from 0 to 31. Some of these pins may be multiplexed with other I/O or control functions.

Table 20 - Programmable I/O Pin Assignments

PIO No	Pin Name	Function / Alternate Function
0	PIO00 / DPD0	I/O - Dual Port RAM Data 0
1	PIO01 / DPD1	I/O - Dual Port RAM Data 1
2	PIO02 / DPD2	I/O - Dual Port RAM Data 2
3	PIO03 / DPD3	I/O - Dual Port RAM Data 3
4	PIO04 / DPD4	I/O - Dual Port RAM Data 4
5	PIO05 / DPD5	I/O - Dual Port RAM Data 5
6	PIO06 / DPD6	I/O - Dual Port RAM Data 6
7	PIO07 / DPD7	I/O - Dual Port RAM Data 7
8	PIO08 / DPA00	I/O - Dual Port RAM Address 0
9	PIO09 / DPA01	I/O - Dual Port RAM Address 1
10	PIO10 / DPA02	I/O - Dual Port RAM Address 2
11	PIO11 / DPA03	I/O - Dual Port RAM Address 3
12	PIO12 / DPA04	I/O - Dual Port RAM Address 4
13	PIO13 / DPA05	I/O - Dual Port RAM Address 5
14	PIO14 / DPA06	I/O - Dual Port RAM Address 6
15	PIO15 / DPA07	I/O - Dual Port RAM Address 7
16	PIO16 / DPA08	I/O - Dual Port RAM Address 8
17	PIO17 / DPA09	I/O - Dual Port RAM Address 9
18	PIO18 / DPA10	I/O - Dual Port RAM Address 10
19	PIO19 / DPA11	I/O - Dual Port RAM Address 11
20	PIO20 / DPA12	I/O - Dual Port RAM Address 12
21	PIO21 / DPBUSY_N	I/O - Dual Port RAM BUSY
22	PIO22 / DPINT_N	I/O - Dual Port RAM Right Port Interrupt
23	PIO23 / DPOE_N	I/O - Dual Port RAM Output Enable
24	PIO24 / DPWR_N	I/O - Dual Port RAM Write
25	PIO25 / DPSS_N	I/O - Dual Port RAM Semaphore Select
26	PIO26 / DPCS_N	I/O – Dual Port RAM Chip Select
27	PIO27	General purpose pin
28	PIO28 / LED0	LED0 Control
29	PIO29 / LED1	LED1 Control
30	PIO30 / LED2	LED2 Control / Watchdog disable
31	PIO31 / LED3	LED3 Control / Address mode select

Note: All pins are internally pulled up except for PIO27, which is pulled down.

The function of each pin is individually controlled by setting the appropriate bin within the PIO Mode, PIO Direction and PIO Data registers. The following table lists the PIO registers.

Table 21 - PIO Registers

Register Name	Mnemonic	Offset
PIO Mode 0	PIOMODE0	70h
PIO Direction 0	PDIR0	72h
PIO Data 0	PDATA0	74h
PIO Mode 1	PIOMODE1	76h
PIO Direction 1	PDIR1	78h
PIO Data 1	PDATA1	7Ah

2.8.1 PIO Mode (PIOMODE0, offset 70h; PIOMODE1, offset 76h)

Two registers, PIOMODE1 and PIOMODE0 control whether each PIO pin functions as its normal operation or as a general purpose Input / Output pin. PIOMODE0 controls the function of PIO pins 0 through 15 and PIOMODE1 controls the function of PIO pins 16 through 31. Set the bit corresponding to a pin to 0 to enable the normal function for that pin. Set the bit to 1 to make that pin an input or output, depending on the setting of the corresponding DIR bit.

Register 46 - PIO Mode 1 Register, PIOMODE1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO 31	PIO 30	PIO 29	PIO 28	PIO 27	PIO 26	PIO 25	PIO 24	PIO 23	PIO 22	PIO 21	PIO 20	PIO 19	PIO 18	PIO 17	PIO 16

Register 47 - PIO Mode 0 Register, PIOMODE0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO 15	PIO 14	PIO 13	PIO 12	PIO 11	PIO 10	PIO 09	PIO 08	PIO 06	PIO 06	PIO 05	PIO 04	PIO 03	PIO 02	PIO 01	PIO 00

2.8.2 PIO Direction (PIODIR0, offset 72h; PIODIR1, offset 78h)

Two registers, PDIR1 and PDIR0 control whether each PIO pin functions as an input or as an output. PDIR0 controls the function of PIO pins 0 through 15 and PDIR1 controls the function of PIO pins 16 through 31. Set the bit corresponding to a pin to 0 to configure it as an output. Set the bit to 1 configure it as an input.

Register 48 - PIO Direction 1 Register, PIODIR1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO 31	PIO 30	PIO 29	PIO 28	PIO 27	PIO 26	PIO 25	PIO 24	PIO 23	PIO 22	PIO 21	PIO 20	PIO 19	PIO 18	PIO 17	PIO 16

Register 49 - PIO Direction 0 Register, PIODIR0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO 15	PIO 14	PIO 13	PIO 12	PIO 11	PIO 10	PIO 09	PIO 08	PIO 06	PIO 06	PIO 05	PIO 04	PIO 03	PIO 02	PIO 01	PIO 00

2.8.3 PIO Data (PDATA0, offset 74h; PDATA1, offset 7Ah)

Two registers, PDATA0 and PDATA1 are used to read and write the state of the PIO pins. If a pin is programmed as an output, then setting the corresponding bit in the data register will drive that pin. If a pin is programmed to be an input, then its current state can be read by reading the appropriate PDATA register.

Register 50 - PIO Data 1 Register, PDATA1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO 31	PIO 30	PIO 29	PIO 28	PIO 27	PIO 26	PIO 25	PIO 24	PIO 23	PIO 22	PIO 21	PIO 20	PIO 19	PIO 18	PIO 17	PIO 16

Register 51 - PIO Data 0 Register, PDATA0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO 15	PIO 14	PIO 13	PIO 12	PIO 11	PIO 10	PIO 09	PIO 08	PIO 06	PIO 06	PIO 05	PIO 04	PIO 03	PIO 02	PIO 01	PIO 00

The combination of PIO Mode and Direction controls the pin functions.

Table 22 - PIO Mode and Direction Combinations

PIO Mode	PIO Direction	Pin Function
0	0	Normal operation with pullup/pulldown
0	1	PIO input with pullup/pulldown
1	0	PIO output with pullup/pulldown
1	1	INVALID

3 Dual Port RAM Memory

The Dual Port Ram Memory (DPM) provides two ports with separate control, address, and data pins to permit independent access for reads or writes to any location in the memory. For additional interface information, see page 167.

3.1 Control Signals

Each port of the DPM has its own set of control signals. When the CS (Chip Select) signal is inactive, the on-chip power down circuitry causes the respective port to go into a standby mode. When the CS signal is active, the DPM is considered "Selected", and can perform a read or a write operation, based on the states of the WR and OE control signals. The control signals for the left port are connected internally, and the control signals for the right port are available as pins of the DSTni-LX device.

PIO26/DPCS_N	I/O – DPRAM Chip Select (Active Low)
PIO24/DPWR_N	I/O – DPRAM Memory Write (Active Low)
PIO23/DPOE_N	I/O – DPRAM Output Enable (Active Low)

3.1.1 Dual Port Memory Select (DPMS, Offset = AAh)

This register is used to set the base address for the 8 Kbyte DPM block. If there is an overlap between this select and another memory select, DPMS will take precedence. As in the Middle Memory Chip Selects, DPMS is disabled after hardware reset until something is written to this register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A23	A22	A21	A20	A19	A18	A17	A16	0	0	0	0	0	R2	R1	R0

A23:16	Upper 8 address bits of the Dual Port Memory base. Allows programmer to set the DPM base at any 64 Kbyte address boundary. When the CPU is operating in normal 20-bit address mode, the contents of A23-A20 are ignored and are treated as if they were zero.
R2:0	Ready logic select. See Table 5 - MMCS Wait State and Ready Select.

3.1.2 Dual Port Memory Chip Select Assignments

The Dual Port Memory chip select lines are assigned per the following table.

Mnemonic	Name	Assigned To
DPMS	Dual Port Memory Select	Dual Port Memory (8k-bytes)
PCS5_n	Peripheral Chip Select 5	Dual Port Memory Semaphores (8)

Use of the 8k-byte Dual Port Memory requires initializing two chip selects. The DPMS is the internal chip select signal for accessing the 8k-bytes of memory. The DPMS starting address offset in memory is selected through the DPMS register. The PCS5_n signal is used to access the eight available semaphores built into the interface. The starting I/O address for each peripheral depends on the PCS line assigned to it. The following table shows the semaphore select line and the base I/O address for the Dual Port Memory.

Select Line	Assigned To	Base I/O Address
PCS5_n	Dual Port Memory Semaphores	PBA + 0500h

PBA is determined via the PACS register.

3.2 Address Lines

Each port has 13 address lines, A00L-A12L, or A00R-A12R. The address lines for the left port are connected internally to the internal memory address bus. The address lines for the right port are available as pins on the DSTni-LX device. These lines are inputs when programmed as the DPM interface.

PIO20/DPA12	I/O – DPRAM ADR_12
PIO19/DPA11	I/O – DPRAM ADR_11
PIO18/DPA10	I/O – DPRAM ADR_10
PIO17/DPA09	I/O – DPRAM ADR_09
PIO16/DPA08	I/O – DPRAM ADR_08
PIO15/DPA07	I/O – DPRAM ADR_07
PIO14/DPA06	I/O – DPRAM ADR_06
PIO13/DPA05	I/O – DPRAM ADR_05
PIO12/DPA04	I/O – DPRAM ADR_04
PIO11/DPA03	I/O – DPRAM ADR_03
PIO10/DPA02	I/O – DPRAM ADR_02
PIO09/DPA01	I/O – DPRAM ADR_01
PIO08/DPA00	I/O – DPRAM ADR_00

3.3 Data Lines

Each port has 8 data lines, D00L-D07L, or D00R-D07R. The data lines for the left port are connected internally to the internal memory data bus. The data lines for the right port are available as pins on the DSTni-LX device. These lines are bi-directional 8-bit data lines.

PIO07/DPD7	I/O – DPRAM DATA7
PIO06/DPD6	I/O – DPRAM DATA6
PIO05/DPD5	I/O – DPRAM DATA5
PIO04/DPD4	I/O – DPRAM DATA4
PIO03/DPD3	I/O – DPRAM DATA3
PIO02/DPD2	I/O – DPRAM DATA2
PIO01/DPD1	I/O – DPRAM DATA1
PIO00/DPD0	I/O – DPRAM DATA0

3.4 Interrupts

The DP Ram supports two interrupts, one for each port. The left port interrupt flag is asserted when the right port writes to memory location 1FFEh. The left port clears the interrupt by reading from the same memory location. The left port interrupt flag is internally connected to the interrupt controller. The right port interrupt flag is asserted when the left port writes to memory location 1FFFh. The right port clears the interrupt by reading from memory location 1FFFh. The right port interrupt flag is available as a pin on the DSTni-LX device.

PIO22/DPINT_N	I/O – DPRAM Right Port Interrupt (Output, Active Low)

3.4.1 Interrupt Sources

The following table lists the interrupt, the interrupt type, vector and default priority for the Dual Port Memory Flag.

Interrupt Name	Vector Type	Vector Address	Default Priority
INT 4	16	40H	10

At power-up or after hardware reset, all interrupt sources are masked (disabled).

The control register for the Dual Port RAM is I4CON.

Name	Mnemonic	Offset
INT 4 Control	I4CON	40h

The control register contains the level-trigger mode bit, mask bit and programmable priority level. The mask bits in this control register are the same mask bits as in the mask register. Modifying them in the control register will also modify the corresponding mask bits in the mask register, and vice versa. This register is read/write.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	LTM	MSK	PRI2	PRI1	PRI0

LTM	The level-trigger mode bit will set the respective interrupt source for either level triggered mode (LTM=1), or edge triggered mode (LTM=0). When set for level-triggered mode an interrupt is generated whenever the external interrupt signal is high. In edge triggered mode an interrupt request is generated when the external interrupt signal makes a low to high transition. In both cases, the level must remain high until the interrupt is acknowledged.
MSK	Setting this bit is will mask the respective interrupt requests. Resetting this bit will enable the respective interrupts.
PR2:0	These bits represent the programmable priority level for the respective interrupt source. Bit combination 000\b has the highest priority while 111\b has the lowest priority.

3.5 Busy Logic

Busy logic provides a hardware indication that both ports of the DP RAM have accessed the same location at the same time. It also allows one of the two accesses to proceed and signals the other side that the DP Ram is “busy”. The busy pin can then be used to stall the access until the operation on the other port has completed. If a write operation has been attempted from the port that receives a busy indication, the write signal is gated internally to prevent the write operation from proceeding. The left port busy signal is internally connected to the wait state generator for the DPM. The right port busy signal is provided as a pin on the DSTni-LX device.

PIO21/DPBUSY_N	I/O – DPRAM Busy (Output, Active low)

3.6 Semaphores

Eight semaphores are provided. The semaphores are completely independent of the DPM, being accessed in I/O address space. They do, however, share the Control lines for Write, Output Enable, the Address lines, and the Data lines with the DPM. They are selected via the "semaphore select" signals, SEML and SEMR signals. SEML is connected to the PCS5 signal and SEMR is connected to a pin on the DSTni-LX, PIO25/DPSS_N. An individual semaphore is selected using the A00L-A02L, and A00R-A02R address lines. The data for the semaphore appears on bit D00L and D00R. The remaining data lines are not used, and will return a value of '0'. The control signals, WL and WR, and OEL and OER control the read and write functions of the semaphore when the SEML and SEMR signals are active.

Note: The SEMR and DPCS signals should never be permitted to go active simultaneously.

Writing a '1' to a semaphore marks the semaphore as available. The semaphore is set to a '1' during hardware reset. To request a semaphore, a '0' is written to the semaphore, and then the semaphore is read. If the value read is a '0', then the request has been granted. If the value read is a '1', then the request has been denied because the semaphore is owned by the other port. At this point, the requesting port may cancel his request by writing a '1' to the semaphore, or may wait for the other side to release the semaphore. If the requesting port decides to wait, he will continue to read the semaphore until the value read is a '0', and thus the request has now been granted.

PIO25/DPSS_N	I/O – DPRAM Semaphore Select (Active low input)
PIO24/DPWR_N	I/O – DPRAM Memory Write (Active low input)
PIO23/DPOE_N	I/O – DPRAM Output Enable (Active low input)

3.7 Other Issues

The DPM appears as an 8 bit device to the right and left port. This is a compatibility requirement. All access to the DPM should be performed as 8-bit accesses.

When interfacing to any Dual Port Memory, the first concern is what happens when both ports attempt to do a write operation at the same time. If the two write addresses are different, there will be no problem. However, when the write addresses are identical, the dual port memory must have a means of permitting one port to proceed, and the other port has to wait. Most dual port memories provide busy signals that indicate that the specific port must hold off completing its write cycle until the busy signal becomes inactive. This is usually done using the ready input of the microprocessor.

The next most common concern is what happens when one port is writing, and the second port is reading. Will the second port get the new data? That usually depends upon which port started first. If the port starting first is doing a write, the write will complete, and the other port will get the new data for its read. If the port starting first is doing a read, the read will complete first. The starting port will get the 'old' data, and then the second port will complete its write. Again, this is only true when both ports are accessing the same address. Also, the busy signal into the ready input of the microprocessor makes all of this work nicely.

Using the busy signal works fine as long as the two processors are running at roughly the same clock speed. In the case of the DSTni-LX, the Turbo186 runs with a memory cycle of 21ns (48MHz). If the other processor is a PC with a PC/ISA bus interface, the cycle time is about 500ns. If the PC/ISA starts its cycle first, then the Turbo186 would have to wait for about 25 memory cycles. This is long enough to cause the Ethernet MAC running at 100Mbits to run out of data, and cause an error. If the Turbo186 starts its cycle first, then the PC/ISA bus would be held off by only 21ns. This has no effect on the PC/ISA bus interface. Using the ready approach does not work well here.

To solve this problem, the external accesses to the DSTni-LX Dual Port Memory are internally synchronized to the Turbo186 memory cycle. This technique actually removes the possibility of the Turbo186 and the external interface accessing the dual port memory at the same time. The internal memory bus arbitration logic insures that this cannot happen. The external interface may have to wait for several Turbo186 memory cycles, but this is a short 80-120ns compared to the 500ns of the PC/ISA cycle. When the PC/ISA interface starts its access to the dual port memory, the request is synchronized, and the memory cycle to the dual port memory is completed during a normal Turbo186 memory cycle of 21ns. The only additional requirement is that the write data has to be valid when the write strobe for the external memory access becomes active. Fortunately, this is the normal case.

The DSTni-LX does have a busy signal to synchronize the external accesses to the dual port memory. The busy signal is a normally low signal that goes high once the dual port memory access has completed. It will remain high until the external cycle completes.   If the external memory cycle is longer than about 120ns, then the busy signal can just be ignored.

4 External Non-Volatile Memory

The DSTni-LX can use an external non-volatile memory devices from which program and configuration data are accessed. The DSTni-LX provides support for two types of non-volatile memory – serial flash or parallel memory. The bootstrap will test for the presence of parallel flash before checking for the presence of serial flash.

4.1 Parallel Flash

An external parallel flash memory device may be connected to the UMCS_n pin. The bootstrap logic will test for the presence of this memory by checking for a signature string at location FF7E0h. If the signature is present and correct, the bootstrap will jump to location FF7F0h which will be the entry point for the application code. The bootstrap will program the UMCS_n pin to be active over the range F0000h – FF7FFh so any initialization code within the parallel flash device must be inside this range.

A functional block diagram of a basic memory system, consisting of external FLASH and SRAM, is shown in the following diagram. Note the use of UMCS to access the FLASH and MCS0 to access the SRAM.

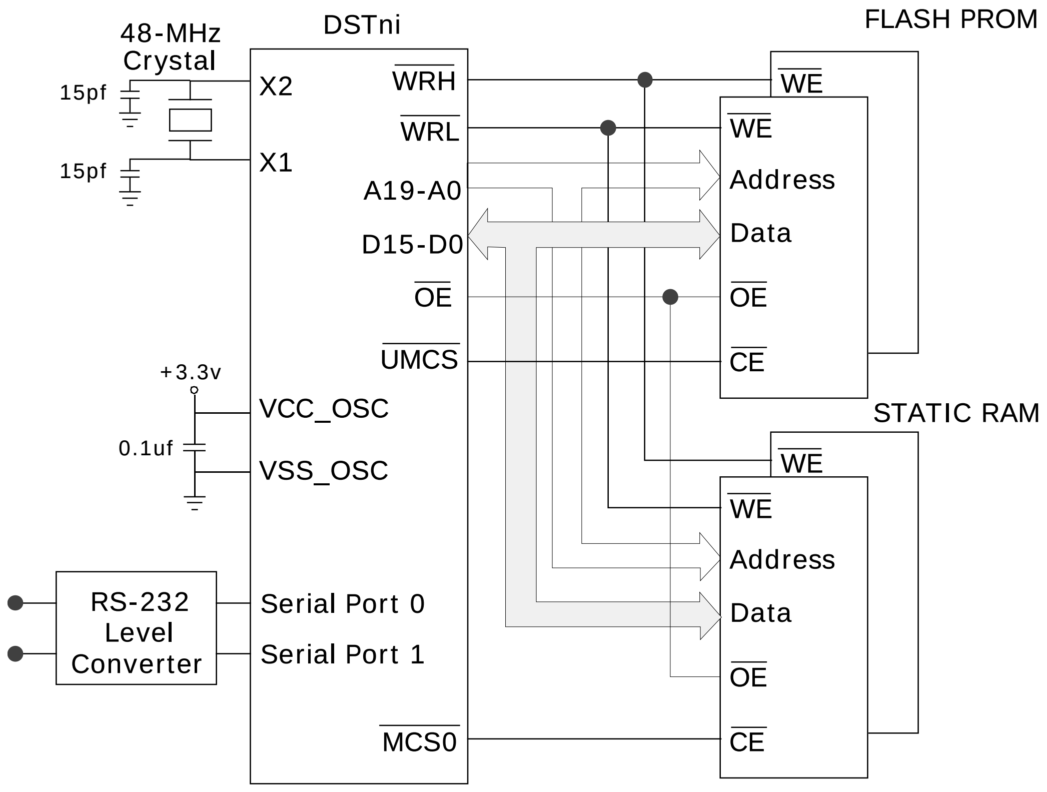


Figure 12 - External Parallel Flash Block Diagram

The following schematic shows the application of a 2MB FLASH.

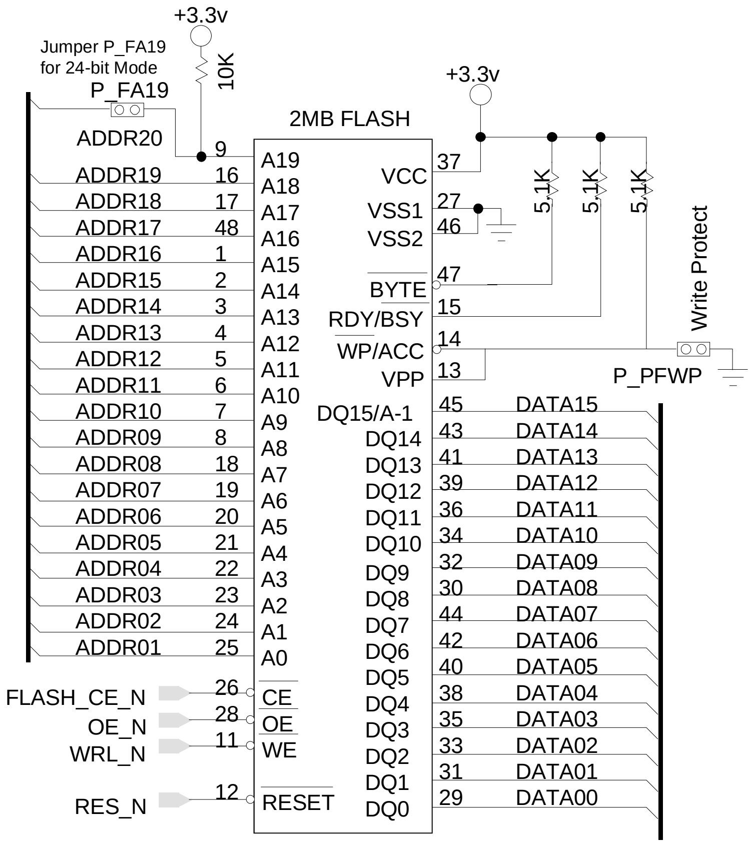


Figure 13 - 2MB Flash Circuit Diagram

The 2MB Flash Circuit Diagram uses the AM29DL163CT or equivalent flash memory part. This is a 16Mbit device arranged as 1,048,576 16 bit words, or 2,097,152 bytes. The available space on the device exceeds the total address space of the DSTni-LX CPU when it is operating in real (20-bit address) mode. As such, only a portion of the device is available to the CPU in this mode. When the CPU is in extended (24-bit address) mode, the entire range of the part is available to the CPU's 2M byte address space.

The following table illustrates how the flash memory physical offsets map to the two different CPU address modes.

Device Offset	Real Mode (20-bit)	Extended Mode (24-bit)
1FFFFF	N/A	(bootstrap)
1FF800	N/A	(bootstrap)
1FF7FF	N/A	FFF7FF
100000	N/A	F00000
FFFFF	(bootstrap)	EFFFFF
FF800	(bootstrap)	EFF800
FF7FF	FF7FF	EFF7FF
F0000	F0000	EF0000
80000	80000	E80000
7FFFF	N/A	E7FFFF
0	N/A	E00000

As the table illustrates, the CPU in real address mode can access at most only one quarter of the available space of the device. If you were to place the CPU into real mode, and write data to the flash device at address 80000h and then switch the CPU into extended mode, the data would be accessed at address E80000h.

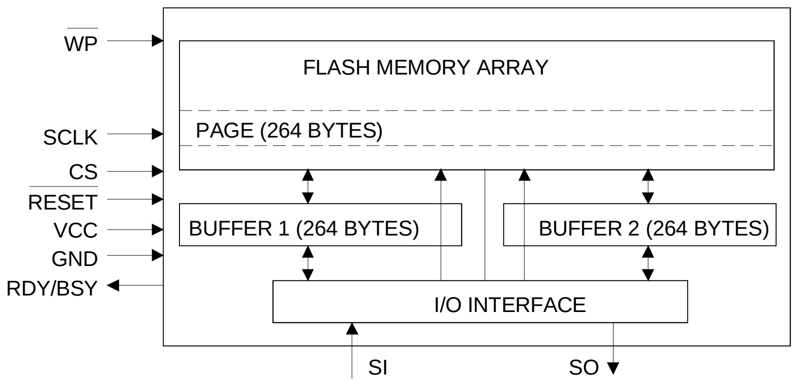
Note also that the upper 2K bytes of each mode's address space maps to the on-chip bootstrap code, making the corresponding regions of the flash device inaccessible.

By power-up default, the Upper Chip Select (UCS) of the DSTni-LX accesses only the top 64K bytes of address space for the address mode in effect. For real mode, this would be the address range F0000h – FFFFFh. For extended mode the default address range would be FF0000h – FFFFFFh. User software desiring to access flash memory below this range would first have to reprogram the UMCS register in the CPU.

4.2 Serial Flash

An external serial flash memory, with an SPI interface, stores the configuration parameters and also the firmware that will be boot-loaded immediately following a power up or hardware reset. This device can be programmed through the SPI port by both the firmware and by the JTAG interface.

The Serial Flash used in this example is an AT45DB021, a 2.7-volt only, serial interface Flash memory suitable for in-system programming. For specific device operation information, see the AT45DB021 data sheet from ATMEL, part number 1642B. Its 2,162,688 bits of memory are organized as 1024 pages of 264 bytes each. In addition to the main memory, the AT45DB021 also contains two SRAM data buffers of 264 bytes each. The buffers allow receiving of data while a page in the main memory is being reprogrammed.



The simple serial interface facilitates hardware layout, increases system reliability, minimizes switching noise, and reduces package size and active pin count. The chip is enabled through the chip select pin (CS) using the DFLASH_N signal. The chip is accessed via a three-wire SPI interface consisting of MOSI (Serial In), MISO (Serial Out), and the serial clock SCLK.

The device operation is controlled by instructions from the host processor. The list of instructions and their associated opcodes can be found in the device data sheet. A valid instruction starts with the falling edge of CS (DFLASH_N) followed by the appropriate 8-bit opcode and the desired buffer or main memory address location. While the CS (DFLASH_N) pin is low, toggling the SCLK pin controls the loading of the opcode and the desired buffer or main memory address location through the SI (serial input) pin. All instructions, addresses, and data are transferred with the most significant bit (MSB) first.

The Flash memory is selected when the CS pin is low. When the device is not selected, data will not be accepted on the SI pin, and the SO pin will remain in a high-impedance state. A high-to-low transition on the CS pin is required to start an operation, and a low-to-high transition on the CS pin is required to end an operation.

If the WP pin is held low, the first 256 pages of the main memory cannot be reprogrammed. The only way to reprogram the first 256 pages is to first drive the protect pin high and then use the program commands previously mentioned. The WP pin is internally pulled high; therefore, connection of the WP pin is not necessary if this pin and feature will not be utilized. However, it is recommended that the WP pin be driven high externally whenever possible. In the Serial Flash example circuit, a jumper is used to enable/disable the write protect feature.

A low state on the reset pin (RESET) will terminate the operation in progress and reset the internal state machine to an idle state. The device will remain in the reset condition as long as a low level is present on the RESET pin. Normal operation can resume once the RESET pin is brought back to a high level.

READY/BUSY is an open-drain output pin driven low when the device is busy in an internally self-timed operation. This pin will be pulled low during programming operations, compare operations, and during page-to-buffer transfers. The busy status indicates that the Flash memory array and one of the buffers cannot be accessed; read and write operations to the other buffer can still be performed. A jumper is used to enable/disable the serial flash write protect.

See the Serial Peripheral Interface (SPI) Port on page 66 for programming information.

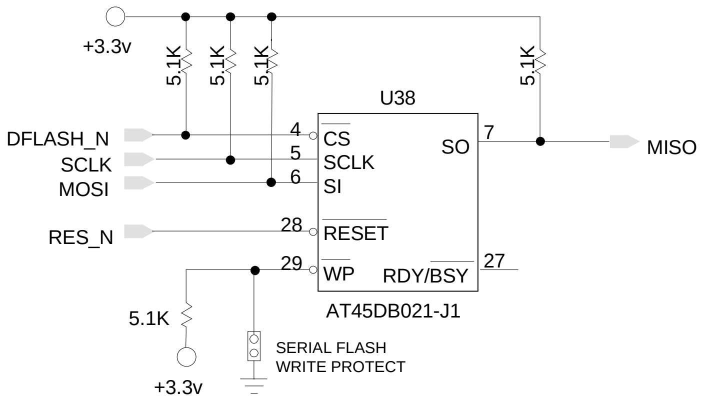


Figure 14 - Serial Flash Circuit Diagram

5 External RAM

The DSTni-LX has 256K bytes of on-chip zero wait state 10NS static RAM. It has all data and address lines to access 16M Bytes of external memory. A typical external RAM circuit is shown below.

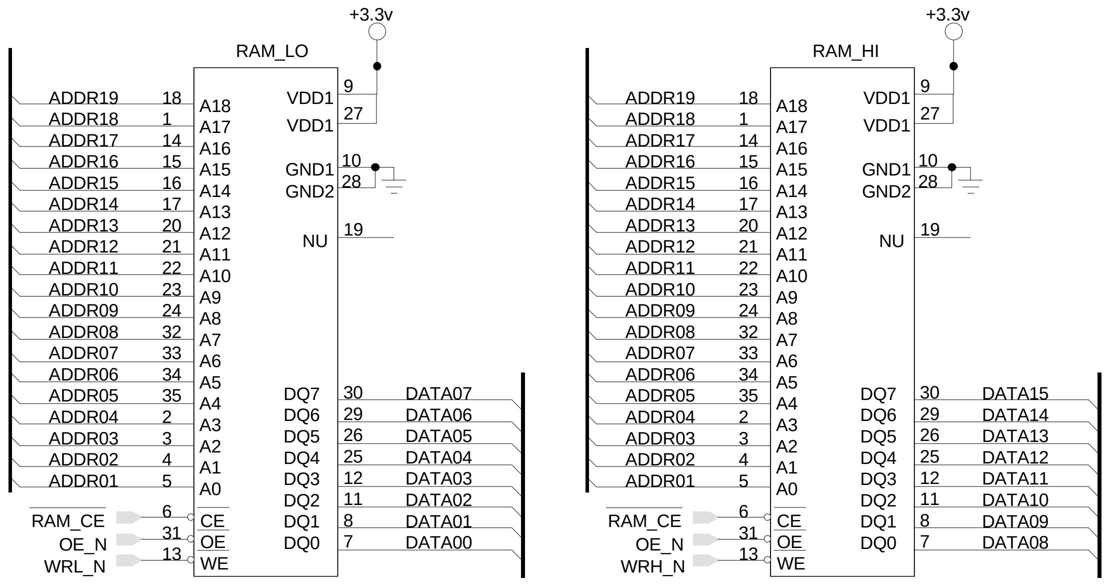


Figure 15 - External RAM

6 Bootstrap ROM

6.1 Bootstrap Description

The bootstrap is the first code to be executed when the DSTni-LX is powered up or reset. It is responsible for loading software from non-volatile storage and for providing the means to load new code via a serial port in the event that the non-volatile storage becomes corrupted. The DSTni-LX has two possible modes of operation – 20-bit addressing and 24-bit addressing. The bootstrap will operate in either mode of operation.

The bootstrap code is hardwired in the DSTni-LX chip where it cannot be erased or altered. This provides a fail-safe mechanism for restoring the DSTni-LX to operation should the non-volatile storage device become corrupted.

6.1.1 Memory

The DSTni-LX bootstrap uses three different memory areas – ROM, RAM and non-volatile code storage. The ROM and RAM are contained on the DSTni-LX chip and the non-volatile storage is an external component. Additional external RAM may be connected to the middle chip select or the upper chip select. The non-volatile storage may be either a parallel flash memory device connected to the DSTni-LX Upper Memory Chip Select (UMCS) or a serial flash memory device connected to the DSTni-LX SPI port. The serial flash memory is selected via the DSTni-LX DFLASH pin.

Memory Usage	20-bit Mode	24-bit Mode	Memory Type
Boot ROM	`FFFFFH` `FF800H`	`FFFFFFH` `FFF800H`	On-Chip
Upper Memory	`FF7FFH` `80000H`	`FFF7FFH` `800000H`	Off-Chip (UMCS_n)
Middle Memory	`7FFFFH` `40000H`	`7FFFFFH` `040000H`	Off-Chip (MCS0_n)
Available RAM	`3FFFFH` `00800H`	`03FFFFH` `000800H`	On-Chip
Boot Data & Stack	`007FFH` `00400H`	`0007FFH` `000400H`	
Interrupt Vectors	`003FFH` `00000H`	`0003FFH` `000000H`	

Figure 16 - Bootstrap Memory Map

6.1.2 Boot ROM

The boot ROM resides in the top 2048 bytes of the CPU address space. The CPU begins execution here at address FFFF0H (20-bit mode) or FFFFE0H (24-bit mode) immediately after a power-up or a reset. The boot ROM code is responsible for loading the DSTni-LX application code from the external serial flash memory. The boot ROM also has the ability to accept a download through via the CPU serial port, in the event the serial flash memory become corrupted. The boot ROM tests and zeros the on-chip SRAM.

6.1.3 RAM

The DSTni-LX contains 256K bytes of static RAM, organized as 128K 16 bit wide words. The bottom 1K bytes (00000H – 003FFH) are reserved for interrupt vectors. The next 1K bytes (00400H – 007FFH) are reserved for data and stack space for the bootstrap. The remaining 254K bytes (00800H – 3FFFFH) are available for code download.

If external zero wait state external RAM is connected to the DSTni-LX middle memory chip select, the bootstrap will support loading beyond the end of the on-chip RAM. If there is external RAM, but it does not run at zero wait states, then an intermediate loader will be required to first set the wait state for the middle chip select before the middle RAM can have code loaded to it.

6.1.4 Serial Flash

The bootstrap is designed to work with either of two types of non-volatile memory – serial data flash or parallel flash.

The serial flash device is connected as a slave to the DSTni-LX Serial Peripheral Interface (SPI). The serial flash is expected to be the Atmel AT45DB011 (1 Mbit) or the Atmel AT45DB021 (2 Mbit) Data Flash® part or equivalent.

Physically, the memory is organized into pages of 264 bytes each. Depending on the part used, there will be either 512 pages or 1024 pages. Logically, the memory will be divided into regions: reserved, configuration storage, code storage and application data storage. Each region will start on a physical page boundary. The following table shows the use of pages.

Table 23 - Serial flash page usage

Page(s)	Bytes	Region
0	264	Reserved
1 - 4	1056	Configuration
5 - n		Program storage
(n+1) – m		Application data storage

In order to execute code from the serial flash device, the code must first be copied to RAM. The bootstrap will begin loading code from page 5, ignoring the contents of pages 0 through 4. Starting with page 5, the code is referred to as “File 0”.

6.1.5 Parallel Flash

The bootstrap is designed to work with either of two types of non-volatile memory – serial data flash or parallel flash.

The parallel Flash must be a 16 bit wide part, at least 64 Kbytes in size. It will be controlled by the DSTni-LX Upper Memory Chip Select (UMCS) line. Initial code will reside within the address range F0000H – FF7FFH. The entry point to the initial code is at location FF7F0H and the following 4 byte ASCII signature pattern must be present at location FF7E0H: “FC-1” (case sensitive). The string must be ordered from left to right starting in the least significant byte of the range FF7E0H – FF7E3H. The following figure illustrates the string.

0			3												
F	C	-	1												

Upon power-up, the bootstrap will test for the presence of parallel flash memory by looking for the signature pattern. If the pattern is found, the bootstrap will jump to the entry point at FF7F0H.

Using parallel flash provides two options for application execution: first, application code can be executed directly from flash or, application code can be copied to RAM and executed there.

6.2 Bootstrap Operation

6.2.1 Determine Operating Mode

Depending on the state of the LED3 pin at reset time, the CPU will be running in either 20-bit (normal) mode or 24-bit (extended address) mode. The bootstrap will determine the address mode and adjust its operation accordingly.

6.2.2 Boot Checksum

The bootstrap performs a checksum on itself. This is an 8 bit two’s complement checksum covering the range FF800H to FFFFEH. The 8 bit result of the checksum is stored at location FFFFFH. The bootstrap verifies the checksum by performing the 8 bit sum from FF800H through FFFFFH. The resulting sum must be 0.

The pass/fail result of the checksum is stored in the AH register – 0 for pass, 1 for fail. If the checksum fails, the bootstrap will drop into an endless loop. If the checksum passes, the bootstrap will proceed to hardware initialization.

6.2.3 Hardware Initialization

The bootstrap performs the following initializations:

- Program the Upper Memory Chip Select (UMCS) for the bootstrap region.
- Program the SPI controller so as to facilitate the loading of code from the serial flash device.
- Program the serial ports so as to support downloading of code via the serial port.
- Program one of the timers to be used as a time-out for polling and downloading operations.

After the hardware initialization is complete, the bootstrap operation continues with the RAM test.

6.2.4 RAM Test

The bootstrap performs a simple test of the on-chip static RAM to ensure that it is safe to use for code, data and stack. If the RAM test fails, the DSTni-LX signals a failure of the RAM test via LED and halts operation. Upon completion of the test, the bootstrap will zero the on-chip SRAM and proceed to the test for serial download.

6.2.5 Test for Serial Download

After the RAM test has passed, the bootstrap next provides the opportunity for an external host to initiate download of software via the serial port. The bootstrap polls the CPU serial port 0 for a period of time, checking for a particular ASCII character sequence.

If time period expires or the expected ASCII sequence is not received, the bootstrap proceeds to check for parallel flash.

If the expected ASCII sequence is received, the bootstrap will enter a mode of receiving a binary format file. Once the file is completely loaded, the bootstrap will jump to the entry point of the loaded code.

6.2.6 Test for Parallel Flash

A parallel non-volatile memory device, such as a flash memory part, may optionally be connected to the DSTni-LX. Access to the part will be controlled by the Upper Memory Chip Select (UMCS) pin.

The bootstrap looks for a signature pattern at address FF7E0H. If the pattern is found at this location, the bootstrap performs a far jump to location FF7F0H. At this location there should be a far jump to the entry point of the code stored within the parallel flash device.

The locations FF7E0H and FF7F0H were chosen because they are at the top of the address range that the UMCS pin will be active for. Addresses FF800H and above are reserved for the on-chip DSTni-LX boot code.

The code resident within the parallel flash device can either execute directly from the device or can copy an application to RAM and execute there.

It there is no recognizable pattern at location FF7E0H the bootstrap will proceed to check for the presence of a serial flash device.

6.2.7 Test for Serial Flash

The bootstrap communicates with the serial flash device via the DSTni-LX SPI port and copies the executable code image from the serial flash to RAM. Once the code is completely loaded, the bootstrap performs a checksum of the loaded code. If the checksum passes, then the bootstrap transfers execution to the RAM entry point of the loaded code. If a valid serial flash data file does not exist or the checksum fails, the bootstrap signals failure of the load via LED and reverts to a mode of waiting for an external host to initiate a software download via serial port 0.

6.2.8 Serial Port Software Download to Flash

The bootstrap supports the ability to download via serial port 0. After testing RAM and prior to checking for non-volatile code storage, the bootstrap polls the serial port for one second, waiting for a command to initiate a serial download operation. If the time out period elapses without the bootstrap receiving any command, it will automatically proceed with checking for non-volatile code storage. If there is no non-volatile storage or no valid code found within the storage, the bootstrap will revert to waiting for download via the serial port.

6.2.9 Bootstrap LEDs

The following table shows the usage of the LEDs during the operation of the bootstrap. The LEDs are controlled by the LED0 and LED1 signal pins.

PIO29/LED1	**LED Control 1**
PIO28/LED0	**LED Control 0**

Table 24 - Bootstrap LED Usage

LED1	LED0	Meaning
Off	Off	No power or bootstrap checksum failed
Off	On	RAM Test in progress, stays on if test failed
On	Off	Serial port being polled
Blink	On	Serial port load in progress
On	On	Serial port load failed
Off	Blink	Serial flash load in progress
Blink	Blink	Serial flash load failed
Blink	Off	No valid non-volatile storage found, serial port being polled indefinitely.

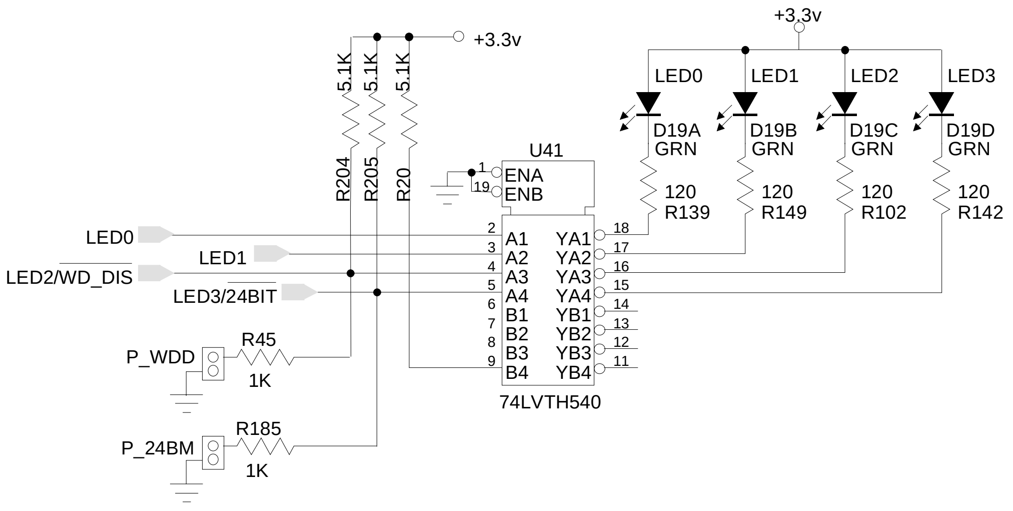


Figure 17 - Bootstrap LEDs

6.3 Application Execution

Depending on the type of non-volatile memory used for application storage, there are different procedures to be followed to start application software.

6.3.1 Execution from Serial Flash

Serial flash requires that code be loaded into RAM for execution. Code may be compiled as standard 20-bit addressing or it may be compiled as 24-bit extended addressing. The code to be loaded must match the mode the CPU will operate in. Also, the code may require more RAM than is available on the DSTni-LX chip. In this case, external RAM must be connected to the DSTni-LX and the loading procedure may have to be modified to take this into account. The following sections describe these options in more detail.

6.3.2 Single Stage Load

A single stage load is one that can be done without having to configure external memory. This would be the case if the application code fit completely within the on-chip 256K bytes of static RAM, or if zero wait state static RAM is connected to the middle chip select line, providing contiguous memory above 40000H. Given that adequate RAM is provided, the single stage load will support load sizes up to 510K bytes.

Here is the sequence of events that take place during a single stage load:

The CPU will undergo a power-up or hardware reset. Depending on the state of the LED3/Mode pin, the CPU will be in either 20-bit or 24-bit mode. Execution will begin at location FFFF0H for 20-bit mode and location FFFE0H for 24-bit mode. Depending on which entry point is used, the bootstrap will set an internal flag to keep track of the mode of operation and then jump to the initialization logic.

The bootstrap code will perform initialization and RAM test and then proceed to load and execute the code image stored in serial flash. This code image may be located at any address above 007FFH.

←-----------256K on-chip RAM --------------→ ←------- optional external RAM -------→

0000	00800H	40000H	7FFFFH max
Reserved	20-bit or 24-bit application code, data & stack		

6.3.3 Two Stage Load

A two stage load is required for any of the following conditions:

The application code to be loaded exceeds the range of the on-chip RAM and external RAM requires configuration before it can be accessed. This may be the case for either 20-bit applications or 24-bit applications. In this case, a 2nd stage loader would configure the external RAM and then proceed to load the application file.

The application to be loaded is stored in a compressed format and must be decompressed during the process of being loaded from non-volatile storage. In this case, the 2nd stage loader would contain decompression logic compatible with the compression method used when building the application image.

Depending on the application, there may be other conditions which require the use of an intermediate loader application. In any case requiring the use of an intermediate loader program, the final application must be located at a region which does not overlap the code and data areas of the 2nd stage loader, as shown in the sample memory map below.

←----------------256K on-chip RAM -------------------→ ← External RAM →

	00800H	01000H	40000H
Reserved	2nd stage	Application code, data & stack	

6.3.4 Execution from Parallel Flash

Using parallel flash offers two choices – executing directly from flash or copying code to RAM and executing from there. Finally, there is the possibility of having external RAM in addition to the on-chip RAM. Each possible combination of these factors requires different procedures. These will be discussed in the following sections.

The bootstrap, not knowing the speed of the flash device will have programmed UMCS for the maximum number of wait states when doing accesses to the device. One of the first things the flash resident startup code should do is to adjust UMCS to reflect the correct size and speed of the flash device in use.

6.3.5 Execute Directly from Flash

The application code executes directly from flash, leaving the on-chip RAM and any external RAM available for data and stack. Executing from flash may impose a performance penalty, but this may be an acceptable tradeoff, depending on the application. The code resident in the flash memory must be compiled for the address mode (20 or 24-bit) the CPU will operate in.

If the application needs to erase and reprogram parts of the flash, it may be necessary to have a small segment of code resident in RAM to do this as most current flash parts presently do not support execution from flash while being programmed.

6.3.6 Execute from RAM

The flash resident code executed by the bootstrap acts as a first stage. It will program the CPU chip select registers to reflect the true size and speed of the flash memory and to enable any external RAM that may be present. The first stage then copies the application code from flash to RAM and then jumps to the entry point of the application.

Note that all code must be built according to the address mode of the CPU.

7 Ethernet

7.1 10/100MBps Ethernet Controller

The Ethernet controller is an interface between the DSTni-LX and an IEEE 802.3 Ethernet Local Area Network (LAN). It is software compatible with the AM79C960 PCnet-ISA chip and the AM7990 LANCE chip from Advanced Micro Devices.

The Ethernet controller provides all of the logic necessary to connect the DSTni-LX to a 802.3 compatible Physical Layer Device (PHY: such as AM79C873 or LXT970A) through Media Independent Interface (MII).

The Ethernet controller interfaces directly to the DSTni-LX internal system bus, and operates with no wait states. The Ethernet controller has its own DMA controller. The DMA controller uses the hold/hold acknowledge signals from the DSTni-LX. Because the controller is tightly coupled to the DSTni-LX, and interrupt latencies will be low, the need for large TX and RX FIFOs is reduced. However, small FIFOs are implemented to guard against data over-runs and under-runs.

The Ethernet controller provides one interrupt signal to the DSTni-LX interrupt controller. The controller registers are accessed in I/O address space. The base address for the register set is controlled by the Peripheral Chip Select logic. See Table 6 for peripheral select assignments.

The Ethernet controller includes the following features:

- Supports IEEE 802.3-1998/ANSI 8802-3 and Ethernet standards
- Supports 10/100Mbs in half or full-duplex mode.
- Compatible with industry standard AM7990 LANCE & AM79C960 PCnet-ISA Ethernet Controllers
- Individual Transmit and Receive FIFOs to support full-duplex mode
- Automatic re-transmission
- Automatic Padding of small Transmit frames
- Automatic deletion of collision fragments
- Dynamic Transmit FCS generation on a frame-by-frame basis
- Complete DMA Buffer Management Unit for minimal CPU overhead
- Requires MII compatible external PHY

7.1.1 Differences between the DSTni-LX Ethernet Controller and the AM79C960

The differences between the DSTni-LX Ethernet controller and the AM79C960 center around the facts that the DSTni-LX Ethernet controller connects to an external Physical Layer Device (PHY) through a Media Independent Interface (MII), connects directly to a microprocessor bus and supports both 10Mbps and 100Mbps modes. The Am79C960 has an integrated Media Attachment Unit (MAU), connects to ISA bus and does not support 100Mbps mode. Regardless these differences, once an external PHY has been attached to the DSTni-LX Ethernet controller, the two circuits should function similarly. The DSTni-LX Ethernet controller was specifically designed to be register compatible with the AM79C960.

Below is a list of the differences between the DSTni-LX Ethernet controller and the AM79C960. Be sure to carefully review the register definitions section as there are some additional small differences in some bits of certain registers. The register definitions section highlights bits that are different between the DSTni-LX Ethernet controller and the AM79C960.

Table 25 - Ethernet Controller Differences

DSTni-LX Ethernet Controller	AM79C960
External MII compatible PHY.	Integrated MAU.
10Mbps and 100 Mbps rates for data transfers.	10Mbps data transfers only.
Connects directly to 186Turbo bus.	Supports ISA architecture.
Bus Master Mode only.	Bus Master/Shared memory modes.
No Boot PROM support.	Boot PROM support included.
No JTAG included.	JTAG Test controller included.
No External Address Detection Interface (EADI) support.	EADI interface included.
Does not require any software programmable registers associated with the system bus.	Numerous configuration registers associated with the system bus.
Gate level buffer management unit. Does not perform any buffer look-ahead.	Microcoded Engine for Buffer management including buffer look ahead.
CSR112 & CSR114 are 8 bits.	CSR112 & CSR114 are 16 bits.
No Built in Sleep Mode (can be implemented by simply stopping the clock).	Reduced Power Sleep Mode.
Automatically decodes MAC control frames with Pause Opcode.	No hardware support for Pause operation.
Debugging/factory testing registers are not implemented.	CSR6, 16, 17, 18, 19, 20, 21, 22, 23, 26, 27, 28, 29, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 48, 49, 50-74, 82, 84, 85, 92, 94, 96, 97, 98, 99, 104, 105, 124 are not "user" registers but are used primarily for factory testing.

NOTE: In the following pages, some register descriptions are shown shaded. This indicates a difference between the DSTni-LX controller and the AM79C960.

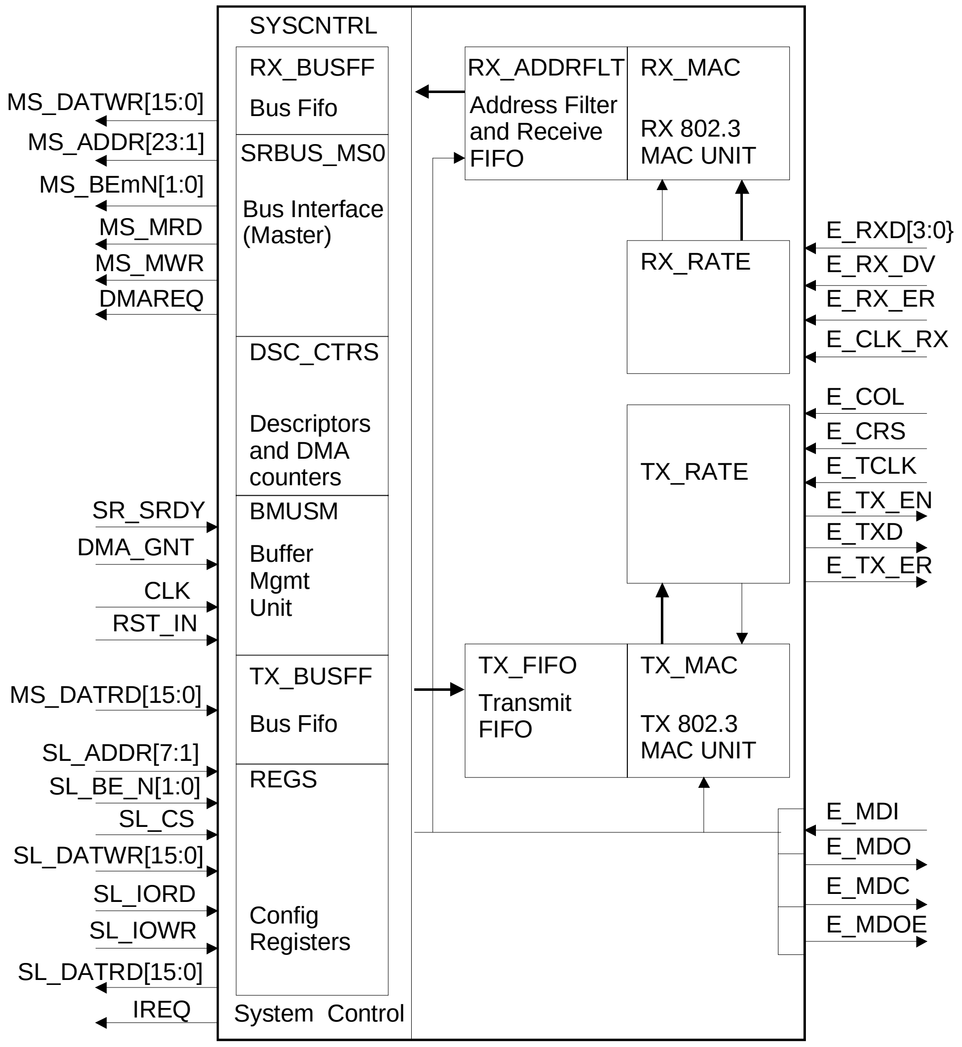


Figure 18 - Ethernet Controller Block Diagram

7.2 Software Interface

The controller relies on descriptor rings built in memory, and modified by both the host software and the controller. The controller uses DMA transfers to access memory in a fashion that minimizes the impact on overall system throughput. The controller uses a very similar ring structure as the AM79C960.

7.2.1 Initialization

At power up, or anytime the Ethernet controller is reset, the controller needs to be initialized. Numerous registers within the controller are set to default values. However, some registers are simply cleared and must be initialized with software. Initialization can be done in two ways; 1) using the initialization block to DMA data into the required registers, or 2) directly writing the registers with the proper values. Using the initialization block is somewhat easier and faster to perform, but directly writing to the registers enables the maximum flexibility in register settings.

To use the initialization block, the programmer should build the initialization block in memory (see diagram below), then write the starting address of this block to the IADR registers (CSR1&2), and then set the INIT bit in CSR0. The controller will then read the initialization block in 2 separate DMA accesses of 4 words each. Once this DMA sequence is complete, the controller will set the IDON bit in CSR0 and an interrupt will be generated if enabled. The programmer should not access the controller while the initialization DMA sequence is in operation. Note that the initialization block must start on an 8-byte boundary.

Table 26 - Initialization Block Format

ADDRESS	Bits 15-12	Bits 11-8	Bits 7-4	Bits 3-0
IADR+22	TLEN (CSR78)	0000	TDRA[23:16] (CSR31)	
IADR+20	TDRA[15:0] (CSR30)			
IADR+18	RLEN (CSR76)	0000	RDRA[23:16] (CSR25)	
IADR+16	RDRA[15:0] (CSR24)			
IADR+14	reserved			
IADR+12	reserved			
IADR+10	reserved			
IADR+8	reserved			
IADR+6	PADR[47:32] (CSR14)			
IADR+4	PADR[31:16] (CSR13)			
IADR+2	PADR[15:0] (CSR12)			
IADR+0	MODE (CSR15)			

RLEN and TLEN:

The RLEN and TLEN fields listed in the initialization block above provide the number of buffer descriptors that are in each ring buffer. These values are programmed into CSR76 and CSR78 as indicated. Note that values other than those listed here can be programmed by writing to CSR76 and CSR78 directly. The RLEN and TLEN fields must be programmed in bits [15:13]. Bit 12 should always be zero.

R/TLEN	Buffers
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

RDRA and TDRA:

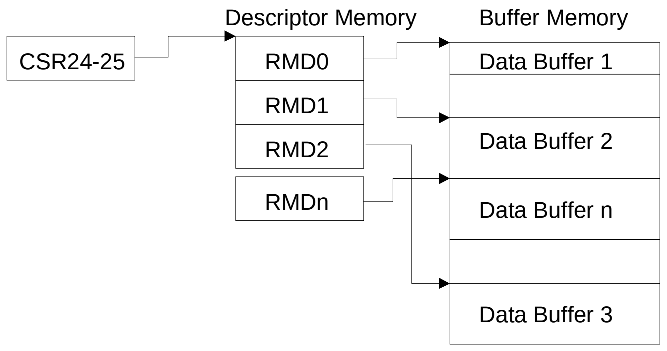
The RDRA and TDRA fields are the base address of the RX and TX buffer rings. See the description for CSR24, CSR25, CSR30 and CSR31 in the register definitions section for more detail.

7.2.2 Descriptor Rings

Data is sent and received via buffers that are built in memory. These buffers are pointed to by the buffer descriptor rings and registers within the controller. There are two descriptor rings: the TX and RX rings. Each descriptor entry is 4 words (8 bytes) and must be aligned on 8-byte boundaries.

Each descriptor ring must occupy a contiguous area of memory. Each descriptor is made up of 4 words with various fields in each word. The descriptor primarily contains the address of the start of the buffer, the number of bytes in the buffer and some status information on the buffer, such as whether it is full or empty.

The following diagram shows how the descriptor rings are built in memory and how they in turn point to the buffers of data in another part of memory. There are two of these structures, one for TX and one for RX. The diagram only shows the registers used for the RX descriptor ring. The TX ring is exactly the same except that it uses a different set of registers. CSR24 and CSR25 form a 24-bit address that points to the start of the RX descriptor ring. An internal register within the Ethernet controller points to the current buffer in use. CSR76 contains the number of buffers in the RX descriptor ring (in this example only 4 are shown). The descriptor entries in turn contain a 24-bit address that points to the start of the buffer where data is to be placed. Note that the buffers can be placed anywhere in memory in any order.



After initialization, the controller will load its internal current descriptor register with the descriptor base register. It will then DMA the descriptor to determine the state of the buffer. If the buffer is "owned" by the Ethernet controller, then the buffer will be used when needed.

Receive Descriptors

Register 52 - RMD0: ADR[15:0]

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

15:0	This field makes up the low 16 bits of the starting address of the receive data buffer. The buffer may be placed on any byte boundary.

Register 53 - RMD1: Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OWN	ERR	FRAM	OFLO	CRC	0	STP	ENP	AD23	AD22	AD21	AD20	AD19	AD18	AD17	AD16

OWN	OWN=1 Indicates that the buffer is "owned" or controlled by the controller. A 0 indicates that the buffer is owned by the host. The controller clears this bit after filling the buffer with data.
ERR	ERR is the logical OR of FRAM, OFLO, CRC. It is written by the controller.
FRAM	FRAM=1 indicates that there has been a framing error. A framing error occurs when there has been a non-integer multiple of bytes in frame. If the CRC and OFLO bits do not indicate other errors, then it is a driver (software) choice to invalidate such a frame (strict check), or to accept the packet (relaxed check). In the latter case, the last byte in the corresponding receive buffer must be discarded. FRAM is valid only when ENP is set and OFLO is not. It is written by the controller.
OFLO	OFLO=1 Indicates that the RX FIFO has overflowed and some data has been lost. OFLO is valid only when ENP is set. It is written by the controller.

CRC	CRC=1 Indicates that the FCS of the incoming frame did not match the computed FCS (CRC), or an external PHY generated a “receive error “ when sending data to the controller (RX_ER signal on MII bus). Data is therefore invalid. CRC is valid only when ENP is set and OFLO is not. It is written by the controller.
10	Always 0. The AM79C960 had a buffer error condition, the controller does not.
STP	STP=1 Indicates that this is the first buffer used to store this frame. It is used for chaining buffers. It is written by the controller.
ENP	ENP=1 Indicates that this is the last buffer used for this frame. If both STP and ENP are set in the same descriptor, then the frame fits in a single buffer, and no data chaining was needed. It is written by the controller.
7-0	High order 8 bits of the starting address of the buffer. Unchanged by the controller.

Register 54 - RMD2: Buffer Byte Count, BCNT

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

15:12	Must be all ones.
11-0	This field is the length of this buffer in bytes, expressed as the two's-complement of the length. Unchanged by the controller.

Register 55 - RMD3: Message Byte Count, MCNT

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

15:12	Must be all ones.
11-0	This field is the number of bytes of the received frame. It is expressed as an unsigned binary integer. MCNT is valid only when ERR is clear and ENP is set. Written by the controller.

Transmit Descriptors

Register 56 - TMD0: ADR[15:0]

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

15:0	This field makes up the low 16 bits of the starting address of the transmit data buffer. The buffer may be placed on any byte boundary.

Register 57 - TMD1: Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OWN	ERR	ADD _FCS	MORE	ONE	DEF	STP	ENP	AD23	AD22	AD21	AD20	AD19	AD18	AD17	AD16

OWN	OWN=1 Indicates that the buffer is "owned" or controlled by the controller. A 0 indicates that the buffer is owned by the host. The controller clears this bit after transmitting all data from the buffer.
ERR	ERR is the logical OR of UFLO, LCOL, LCAR, RTTY. It is written by the controller.
ADD_FCS	ADD_FCS dynamically controls the generation of FCS on a frame by frame basis. ADD_FCS=1 will cause the controller to append the FCS sequence to the end of a frame, regardless of the state of the DTXFCS bit. ADD_FCS is valid only when the STP bit is set. Unchanged by the controller.
MORE	MORE=1 Indicates that the more than one retry was needed to transmit this frame. It is valid only when ENP is set. Written by the controller.

ONE	ONE=1 Indicates that exactly one retry was needed to transmit this frame. ONE is valid only when ENP is set. Written by the controller.
DEF	DEF=1 Indicates that the controller had to defer transmitting this frame because the media was busy. Valid only when ENP or ERR is set. Written by the controller.
STP	STP=1 Indicates that this is the first buffer used for this frame. It is used for chaining buffers. STP must be set for the first buffer in a chain or the buffer will be skipped. Unchanged by the controller.
ENP	ENP=1 Indicates that this is the last buffer used for this frame. If both STP and ENP are set in the same descriptor, then the frame fits in a single buffer, and no data chaining is to be used. Unchanged by the controller.
7-0	High order 8 bits of the starting address of the buffer. Unchanged by the controller.

Register 58 - TMD2: Buffer Byte Count, BCNT

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

15:12	Must be all ones.
11-0	This field is the length of this buffer in bytes, expressed as the two's-complement of the length. There is no minimum buffer size restriction. Zero length buffers are allowed. Unchanged by the controller.

Register 59 - TMD3: Error Status

15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BUFF	UFLO	0	LCOL	LCAR	RTRY	0	0	0	0	0	0	0	0	0	0

BUFF	BUFF=1 when the controller finds OWN=1 and STP=0, and is not currently transmitting a chained frame. Written by the controller.
UFLO	UFLO=1 indicates that a FIFO underflow has occurred. The frame will have had a bad FCS appended to it and will be dropped. To enforce visibility of such error conditions on MII bus, the controller will generate TX_ER before deasserting TX_EN. Written by the controller.
13	Always 0.
LCOL	LCOL=1 indicates that a late collision occurred. The frame is dropped. Written by the controller.
LCAR	LCAR=1 Indicates that the carrier sense signal (CRS) went inactive during the transmission. Written by the controller.
RTRY	RTRY=1 Indicates that the transmitter tried to transmit the frame 16 times but was not able to. This indicates a heavily loaded network with excessive traffic and collisions. If DRTY=1 then RTRY will be set if the first attempt at transmission fails. Written by the controller.
9-0	This field is reserved. Note that this was the TDR bits on the LANCE and PCnet-ISA. The TDR function is not particularly useful so it was not implemented in the controller.

7.2.3 Descriptor Ring Polling

Once the host has initialized the descriptor rings and the START bit in CSR0 has been set, the controller will begin polling the descriptor rings. The controller will first poll the current RX and TX descriptor ring entries and evaluate their OWN bits. Three words are read from each descriptor, xMD0, xMD1 and xMD2. This results in two 3-word DMA transfers, after which the bus is released.

If there is no activity on the network the controller will continue to poll the descriptor rings periodically based on the value in the POLLINT register, until the OWN bit in the currently-polled ring is found to be 1. Once the OWN bit is found to be 1, then that ring will not be polled again until the operation for that buffer has been completed. Thus, once the RX OWN bit has polled and found to be 1, the RX descriptor ring will no longer be polled until a receive frame uses the buffer.

The POLL TIME COUNTER is a free-running counter with the max poll time of 1.3ms (clk=50 MHz). Note that a poll of the descriptor rings can be forced by setting the TDMD bit in CSR0. Setting TDMD after placing a frame in the TX ring can help reduce the transmit latency, since the controller will immediately poll and begin operating on the buffer.

Each time a descriptor's OWN bit is cleared by the controller, the controller will perform a poll of the descriptor rings to determine the state of the OWN bit of the next descriptor. Note that only the required rings will be polled. Thus, if we have just released an RX buffer, but have a TX frame waiting in the TX FIFO, only the RX ring will be polled. Only one descriptor is polled at a time. No look-ahead operations are performed.

7.2.4 Transmit Descriptor Processing

If the controller finds the OWN bit of a TX descriptor to be a zero after a poll operation, then the controller will wait until the next poll operation to re-evaluate the OWN bit. At any time (except transmission of chained frames), when the controller finds the OWN bit =1 and STP=0, then the controller will request the bus, clear the OWN bit and set BUFF bit in TMD3. The controller always writes TMD1 and TMD3 whenever updating a TX descriptor ring entry.

If OWN=1 but the buffer length is zero, the controller will immediately request the bus to clear the OWN bit. After releasing the bus, another poll will be requested.

If OWN=1 and STP=1, then the controller will request the bus and begin DMA operations to move data to the TX fifo. DMA continues until all data in this buffer has been fetched by the controller. If ENP=0, then the descriptor will be released (OWN cleared to zero), a poll operation performed, and the data in the next buffer will be DMAed. Once the controller finds a descriptor with ENP=1, the data will be DMAed into the FIFO and then the controller will wait until the entire frame has been transmitted. The controller then updates TMD1 and TMD3 with the appropriate status bits. A poll operation is then performed and processing continues. Note that the current frame must complete transmission, the descriptor released, a poll operation performed and the next TX frame loaded into the FIFO, when there are back to back frames.

Whenever the last transmit descriptor of a frame is released (OWN cleared to 0 and ENP or ERR set to 1), the TINT bit of CSR0 is set to indicate to the host that the frame has been released.

7.2.5 Receive Descriptor Processing

If the controller finds the OWN bit of an RX descriptor to be zero after a poll operation, then the controller will wait until the next poll operation to re-evaluate the OWN bit. If the OWN bit is found to be a one, then no more polls of the RX descriptor ring will take place until the current buffer is filled with an incoming frame.

When a frame is received, the controller waits until destination address field of the frame has been placed in the FIFO. This provides time for the controller to verify the destination address with the currently active addressing schemes . If the address does not match, then the FIFO write pointer is reset and no data is sent from the address filter block. Any received frame shorter than 64 bytes is considered to be a fragment resulting from collision. The next frame that meets the minimum size will use same set of descriptors, since the descriptor pointer is not incremented by short frames. When RPA bit in CSR4(7) is set to 1, then frames shorter then 64 bytes are accepted by controller. This bit is meant for testing only.

Once the destination address has been verified, the BMU will check the status of the current receive buffer which is stored in the controller at this point. If the OWN bit is a one, DMA will begin immediately to store the data into the host RAM. If the OWN bit is a zero, and the FIFO does overflow, the data in the FIFO will remain, and will have to be DMAed eventually. However, the OFLO bit will be set in the descriptor to indicate that the data is bad. If the OWN bit remains zero, and consecutive frames cannot be received, a "missed frames counter" in CSR112 gets incremented.

If the frame length exceeds the length of the current receive buffer, the controller will update the RMD1 and RMD3 fields of the current descriptor. The controller will then immediately poll the next descriptor. If the OWN bit of the next descriptor is a one, then DMA will begin again until all data for the frame has been stored into the host's memory. In this case, data chaining is used to spread the frame across multiple buffers. Note that RMD3 will contain the total number of bytes of the frame when ENP is set.

7.2.6 MII Management functions

The MII management interface provides a simple, two-wire, serial interface to connect the controller and a managed PHY, for the purposes of controlling the PHY and gathering status from the PHY. The management interface consists of a pair of signals that physically transport the management information across the MII , a frame format and a protocol specification for exchanging management frames, and a register set that can be read and written using these frames. The register definition specifies a basic register set with an extension mechanism. The MII basic register set consists of two registers, referred to as the Control register (Register 0) and the Status register (Register 1). All PHYs that provide an MII shall incorporate the basic register set. Registers 2 through 10 are part of the extended register set. The full set of management registers (802.3) is listed in the following table.

Table 27 - MII Management Register Set

Register address	Register name	Basic/Extended
0	Control	B
1	Status	B
2,3	PHY Identifier	E
4	Auto-Negotiation Advertisement	E
5	Auto-Negotiation Link Partner Base Page Ability	E
6	Auto-Negotiation Expansion	E
7	Auto-Negotiation Next Page Transmit	E
8	Auto-Negotiation Link Partner Received Next Page	E
9	100BASE-T2 Control Register	E
10	100BASE-T2 Status Register	E
11 through 15	Reserved	Reserved
16 through 31	Vendor Specific	E

This document describes only basic registers 0 and 1, which have to be implemented in MII compatible PHY. All other PHY registers should be carefully reviewed in applicable documentation provided by the PHY manufacturer.

Table 28 - MII Management - Control Register Bit Definitions

Bit(s)	Name	Description	R/W SC(self clearing)
0.15	Reset	1 = PHY reset 0 = normal operation	R/W SC
0.14	Loopback	1 = enable loopback mode 0 = disable loopback mode	R/W
0.13	Speed Selection(LSB)	0.6 0.13 1 1 = Reserved 1 0 = 1000Mbps 0 1 = 100 Mbps 0 0 = 10 Mbps	R/W
0.12	Auto-Negotiation Enable	1 = Enable Auto-Negotiation Process 0 = Disable Auto-Negotiation Process	R/W
0.11	Power Down	1 = power down 0 = normal operation	R/W
0.10	Isolate	1 = electrically Isolate PHY from MII 0 = normal operation	R/W
0.9	Restart Auto-Negotiation	1 = Restart Auto-Negotiation Process 0 = normal operation	R/W SC
0.8	Duplex Mode	1 = Full Duplex 0 = Half Duplex	R/W
0.7	Collision Test	1 = enable COL signal test 0 = disable COL signal test	R/W
0.6	Speed Selection (MSB)	0.6 0.13 1 1 = Reserved 1 0 = 1000 Mbps 0 1 = 100 Mbps 0 0 = 10 Mbps	R/W
0.5:0	Reserved	Write as 0, ignore on Read	R/W

7.2.7 Reset

Resetting a PHY sets the status and control registers to their default states. As a consequence, this action may change the internal state of the PHY and the state of the physical link associated with the PHY. This bit is self-clearing, and a PHY shall return a value of one in bit 0.15 until the reset process is completed. A PHY is not required to accept a write transaction to the control register until the reset process is completed, and writes to bits of the control register other than 0.15 may have no effect until the reset process is completed. The reset process shall be completed within 0.5 s from the setting of bit 0.15.

7.2.8 Speed selection

Link speed can be selected via either the Auto-Negotiation process, or manual speed selection. Manual speed selection is allowed when Auto-Negotiation is disabled, by clearing bit 0.12 to zero. When Auto-Negotiation is disabled and bit 0.6 is cleared to a logic zero, setting bit 0.13 to a logic one configures the PHY for 100 Mbps operation, and clearing bit 0.13 to a logic zero configures the PHY for 10 Mbps operation. When Auto-Negotiation is disabled and bit 0.6 is set to a logic one, clearing bit 0.13 to a logic zero selects 1000 Mbps operation, which is a speed setting called out in the MII register standard, but not supported in the controller core. The combination of both bits 0.6 and 0.13 set to a logic one is reserved for future standardization. When Auto-Negotiation is enabled, bits 0.6 and 0.13 can be read or written, but the state of bits 0.6 and 0.13 have no effect on the link configuration, and it is not necessary for bits 0.6 and 0.13 to reflect the operating speed of the link when it is read. If a PHY reports via bits 1.15:9 and bits 15.15:12 that it is not able to operate at all speeds, the value of bits 0.6 and 0.13 shall correspond to a speed at which the PHY can operate, and any attempt to change the bits to an invalid setting shall be ignored. The default value of bits 0.6 and 0.13 are the encoding of the highest data rate at which the PHY can operate as indicated by bits 1.15:9.

7.2.9 Duplex mode.

The duplex mode can be selected via either the Auto-Negotiation process, or manual duplex selection. Manual duplex selection is allowed when Auto-Negotiation is disabled, by clearing bit 0.12 to zero. When Auto-Negotiation is disabled, setting bit 0.8 to a logic one configures the PHY for full-duplex operation, and clearing bit 0.8 to a logic zero configures the PHY for half-duplex operation. When Auto-Negotiation is enabled, bit 0.8 can be read or written, but the state of bit 0.8 has no effect on the link configuration. If a PHY reports via bits 1.15:9 and 15.15:12 that it is able to operate in only one duplex mode, the value of bit 0.8 shall correspond to the mode in which the PHY can operate, and any attempt to change the setting of bit 0.8 shall be ignored.

7.2.10 Auto-Negotiation.

The Auto-Negotiation process is enabled by setting bit 0.12 to a logic one. If bit 0.12 is set to a logic one, then bits 0.13, 0.8, and 0.6 shall have no effect on the link configuration and station, operation other than that specified by the Auto-Negotiation protocol. If bit 0.12 is cleared to a logic zero, then bits 0.13, 0.8, and 0.6 will determine the link configuration, regardless of the prior state of the link configuration and the Auto-Negotiation process. If a PHY reports via bit 1.3 that it lacks the ability to perform Auto-Negotiation, the PHY shall return a value of zero in bit 0.12. If a PHY reports via bit 1.3 that it lacks the ability to perform Auto-Negotiation, bit 0.12 should always be written as zero, and any attempt to write a one to bit 0.12 shall be ignored. The default value of bit 0.12 is one, unless the PHY reports via bit 1.3 that it lacks the ability to perform Auto-Negotiation, in which case the default value of bit 0.12 is zero.

7.2.11 Restarting Auto-Negotiation.

If a PHY reports via bit 1.3 that it lacks the ability to perform Auto-Negotiation, or if Auto-Negotiation is disabled, the PHY returns a value of zero in bit 0.9. In such a case bit 0.9 should always be written as zero, and any attempt to write a one to bit 0.9 shall be ignored. Otherwise, the Auto-Negotiation process shall be restarted by setting bit 0.9 to a logic one. This bit is self-clearing, and a PHY shall return a value of one in bit 0.9 until the Auto-Negotiation process has been initiated. The Auto-Negotiation process shall not be affected by writing a zero to bit 0.9. The default value of bit 0.9 is zero. All of the bits in status register are read only, a write to status register have no effect. Note that neither Control register nor Status register provide status about current operational mode regarding full duplex or 100Mbps mode. These registers show abilities of the PHY or link and do not necessarily match parameters of actual connections. To take advantage of full duplex mode the controller should be set properly (see: Register MIIP bit FDEN).

If auto-negotiation is disabled then PHY and the controller can easily be set to the same mode (full or half duplex). If auto-negotiation has been enabled and performed, the current mode can usually be read from one of the vendor specific MII registers (register 16 through 31). For example: LevelOne LXT970A PHY shows full duplex status in Reg 20-bit 12 and 100Mb mode in Reg 20-bit 11.

Table 29 - MII Status Register 1 Bit Definitions

Bit(s)	Name	Description	R/W
1.15	100BASE-T4	1 = PHY able to perform 100BASE-T4 0 = PHY not able to perform 100BASE-T4	R
1.14	100BASE-X Full Duplex	1 = PHY able to perform full duplex 100BASE-X 0 = PHY not able to perform full duplex 100BASE-X	R
1.13	100BASE-X Half Duplex	1 = PHY able to perform half duplex 100BASE-X 0 = PHY not able to perform half duplex 100BASE-X	R
1.12	10 Mbps Full Duplex	1 = PHY able to operate at 10 Mbps in full duplex mode 0 = PHY not able to operate at 10 Mbps in full duplex mode	R
1.11	10 Mbps Half Duplex	1 = PHY able to operate at 10 Mbps in half duplex mode 0 = PHY not able to operate at 10 Mbps in half duplex mode	R
1.10	100BASE-T2 Full Duplex	1 = PHY able to perform full duplex 100BASE-T2 0 = PHY not able to perform full duplex 100BASE-T2	R
1.9	100BASE-T2 Half Duplex	1 = PHY able to perform half duplex 100BASE-T2 0 = PHY not able to perform half duplex 100BASE-T2	R
1.8	Extended Status	1 = Extended status information in Register 15 0 = No extended status information in Register 15	R
1.7	Reserved	ignore when read	R
1.6	MF Preamble Suppression	1 = PHY will accept management frames with preamble suppressed. 0 = PHY will not accept management frames with preamble suppressed.	R
1.5	Auto-Negotiation Complete	1 = Auto-Negotiation process completed 0 = Auto-Negotiation process not completed	R
1.4	Remote Fault	1 = remote fault condition detected 0 = no remote fault condition detected	R
1.3	Auto-Negotiation Ability	1 = PHY is able to perform Auto-Negotiation 0 = PHY is not able to perform Auto-Negotiation	R
1.2	Link Status	1 = link is up 0 = link is down	R
1.1	Jabber Detect	1 = jabber condition detected 0 = no jabber condition detected	R
1.0	Extended Capability	1 = extended register capabilities 0 = basic register set capabilities only	R

7.2.12 Management frame

Format of Management frames transmitted on the MII Management Interface is shown in the following table. The order of bit transmission is from left to right. The first transmitted bits of Device Address, Register Address and DATA fields are MSb of the PHY address, register address, and data of the register being accessed..

The IDLE condition on MDIO is a high-impedance state. All three state drivers shall be disabled and the PHY's pull-up resistor will pull the MDIO line to a logic one.

At the beginning of each transaction, the controller should send a preamble sequence (PRE) of 32contiguous logic one bits on MDIO with 32 corresponding cycles on MDC to provide the PHY with a pattern that it can use to establish synchronization. A PHY shall observe a sequence of 32 contiguous one bits on MDIO with 32 corresponding cycles on MDC before it responds to any transaction.

Table 30 - MII Management Frame Format

Fields								
	PRE	Start	Opcode	Device Address	Register Address	TA	Data	Idle
READ	1...1	01	10	AAAAA	RRRRR	Z0	DDDDDDDDDDDDDDDD	Z
WRITE	1...1	01	01	AAAAA	RRRRR	10	DDDDDDDDDDDDDDDD	Z

7.2.13 Management Frame Format

If the controller determines that the PHY that is connected to the MDIO signal is able to accept management frames that are not preceded by the preamble pattern, then the controller may suppress the generation of the pre-amble pattern, and may initiate management frames with the Start of Frame pattern.

7.3 Ethernet Operation

The Ethernet controller appears to the microprocessor as a memory-mapped I/O device.

Each peripheral has 256 bytes of I/O address space allocated to it. The starting I/O address for each peripheral depends on the Peripheral Chip Select (PCS) line assigned to it. PCS0 is assigned to Ethernet. The Base I/O address is determined by the Peripheral Base Address (PBA) and the offset. The PBA is determined via the PACS register.

Table 31 - Ethernet I/O Address

Select Line	Assigned To	Base I/O Address
PCS0	Ethernet	PBA + 0000h

The Peripheral Chip Selects are programmed through two registers—the Peripheral Address Chip Select (PACS) register and the PCS and MCS Auxiliary (MPCS) register. The Peripheral Address Chip Select (PACS) register determines the base address, the ready condition, and the wait states for the PCS3–PCS0 outputs.

The MPCS register determines whether PCS chip selects are active during memory or I/O bus cycles and specifies the ready and wait states for the PCS6–PCS5 outputs. The Ethernet device must be mapped into the I/O space.

The PCS pins are not active on reset. The PCS pins are activated as chip selects by writing to the PACS and MPCS registers.

Peripheral Address Chip Select Register, PACS (20-bit mode)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A19	A18	A17	A16	A15	A14	A13	A12	A11	1	1	1	1	R2	R1	R0
0	0	0	0												

A19:11	Upper address bits of the starting address of the Peripheral Base Address (PBA). Allows programmer to set the PBA at any 2K byte address boundary. Since we’re operating only in I/O space which is limited to 64 Kbytes, A19-A16 must always be zero.
R2:0	Ready generation select. See Table 5 - MMCS Wait State and Ready Select.

Peripheral Address Chip Select Register, PACS (24-bit mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	R2	R1	R0
0	0	0	0	0	0	0	0								

A23:11	Upper address bits of the starting address of the Peripheral Base Address (PBA). Allows programmer to set the PBA at any 2K byte address boundary. Since we’re operating only in I/O space which is limited to 64 Kbytes, A23-A16 will always be zero.
R2:0	Ready generation select. See Table 5 - MMCS Wait State and Ready Select.

7.3.1 Ethernet Interrupt (INT0)

The Ethernet controller uses the following interrupt.

Register 60 - Ethernet Interrupts

Interrupt Signal	Assignment	Vector Type	Vector Address	Default Priority
INT0	Ethernet	12	30H	6

7.3.2 Ethernet Interrupt Control Register

This is the control register for the external interrupt sources. Each control register contains the level-trigger mode bit, mask bit and programmable priority level. The mask bits in these control registers are the same mask bits as in the mask register. Modifying them in the control register will also modify the corresponding mask bits in the mask register, and vice versa. This register is read/write

Register 61 - External Interrupt Control Register, I0CON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	LTM	MSK	PRI2	PRI1	PRI0

LTM	The level-trigger mode bit will set the respective interrupt source for either level triggered mode (LTM=1), or edge triggered mode (LTM=0). When set for level-triggered mode an interrupt is generated whenever the external interrupt signal is high. In edge triggered mode an interrupt request is generated when the external interrupt signal makes a low to high transition. In both cases, the level must remain high until the interrupt is acknowledged.
MSK	Setting this bit is will mask the respective interrupt requests. Resetting this bit will enable the respective interrupts.
PR2:0	These bits represent the programmable priority level for the respective interrupt source. Bit combination 000\b has the highest priority while 111\b has the lowest priority.

Name	Mnemonic	Offset
End of Interrupt	EOI	22h
Poll	POLL	24h
Poll Status	POLLST	26h
Interrupt Mask	IMASK	28h
Priority Mask	PRIMSK	2Ah
In-Service	INSERV	2Ch
Interrupt Request	REQST	2Eh
Interrupt Status	INTSTS	30h
INT 0 Control	**I0CON**	**38h**

7.3.3 In-Service Register (INSERV, offset 2Ch)

This register contains the in-service bits for each of the interrupt sources. An in-service bit is set to indicate that a source's service routine is being executed. When an in-service bit is set the interrupt controller will not generate interrupts from sources with a lower priority than the current interrupt being serviced. This allows interrupt service routines to run with interrupts enabled. This register is read/write.

Register 62 - In-Service Register for Ethernet

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	I2	I1	**I0**	D1	D0	0	TMR

SPI	In-service bit for the Serial Peripheral Interface
D3:0	In-service bits for the DMA channels (DMA3:0).
I5:0	**In-service bits for the external interrupt sources (INT5:0).**
SP1:0	In-service bits for the asynchronous serial ports (SP1:0).
TMR	In-service bit for all three timers (TMR2:0).

7.3.4 Interrupt Request Register (REQST, offset 2Eh)

This register contains the interrupt request bits for the Timers, DMA channels and external interrupt sources. This register is read/write except were noted.

Register 63 - Interrupt Request Register for Ethernet

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	I2	I1	**I0**	D1	D0	0	TMR

I5:0 Interrupt request bits for the external interrupt sources (INT5:0). These bits represent the actual state of the external interrupt pins, after they have been passed through a two level synchronizer, or a level detector depending on the LTM bit for that respective interrupt line. These bits will be set whenever an external interrupt line make a low to high transition if LTM=0. Otherwise, if LTM=1 these bits will follow the input pins after a two clock delay. Because these bits represent the state of the external interrupt pins they can not be written.

7.3.5 I/O Address Map

Most of the non-essential registers are not implemented in the Ethernet controller. Register bits that are different between the Ethernet controller and the AM79C960 are highlighted. All Ethernet registers are addressed in IO space.

Offset	# Bytes	Register
0x00	16	Reserved
0x10	2	RDP
0x12	2	RAP
0x14	2	RESET
0x18	2	MIIP

All CSR registers are accessed using a two-step process. First, the address of the register to be accessed is written into the Register Address Port (RAP) register. Then the contents of the register pointed to by the RAP is then accessed via the Register Data Port (RDP). Thus the RAP is used as an index register or pointer to the register you wish to access. All CSR registers are cleared on reset, unless shown differently.

Note: Register bits shown in shaded blocks are different than the AM79C960.

Register 64 - CSR0: Status Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR	0	CERR	MISS	0	RINT	TINT	IDON	INSTR	IENA	RXON	TXON	TDMD	STOP	STRT	INIT

ERR	ERR is the logical OR of CERR and MISS. ERR is READ-ONLY.
Bit 14	Reserved, R/W as zero. The controller supports only MII interface to communicate with PHY. This bit is unused and always reads back as 0. In AM79C960 it is the BABL bit, which indicates that the transmitter has been enabled for more than the maximum length frame.
CERR	CERR indicates that the COL pin did not go high within 20 network bit times after the termination of a transmission. This bit is used during SQE test, and is valid only at 10Mbps half-duplex mode. CERR does not assert IRQ but does set ERR. CERR is cleared by writing a 1. Writing a 0 has no effect. CERR is cleared by setting STOP.
MISS	MISS is set when an incoming frame has been lost. A frame will be lost when there is no room available in the RX_ADDRFLT FIFO for an incoming packet. This is the only indication that data has been lost since the packet was not placed in the fifo and/or no descriptors are available to record the information. MISS will assert IRQ if the mask bits are 0. MISS is cleared by writing a 1. Writing a 0 has no effect. MISS is cleared by setting STOP.
Bit 11	Reserved, R/W as zero. (It is the MERR bit in AM79C960.)
RINT	RINT is set when the OWN bit RMD1 is cleared and ENP=1. This happens when a complete frame has been placed in the RX descriptor ring. RINT will assert IRQ if the mask bits are 0. RINT is cleared by writing a 1. Writing a 0 has no effect. RINT is cleared by setting STOP.

TINT	TINT is set when the OWN bit TMD1 is cleared and ENP=1 and TX_EN goes low. This happens when a complete frame has been Transmitted. TINT will assert IRQ if the mask bits are 0. TINT is cleared by writing a 1. Writing a 0 has no effect. TNT is cleared by setting STOP.
IDON	IDON is set when the controller has read the entire Initialization Block from memory. IDON will assert IRQ if the mask bits are 0. IDON is cleared by writing a 1. Writing a 0 has no effect. IDON is cleared by setting STOP.
INSTR	INSTR is the logical OR of, MISS, MFCO, RCVCCO, RINT, TINT, IDON, TXSTRT and PAUSE the IRQ pin will be asserted if IENA is 1. INSTR is READ-ONLY.
IENA	IENA enables the INSTR bit to cause the IRQ pin to be asserted when IENA=1. If IENA=0, IRQ is disabled. IENA is set by writing a 1 and cleared by writing a 0. IENA is cleared by STOP.
RXON	RXON=1 indicates that the Receive function is enabled. RXON is set when DRX (CSR15.0)=0 and STRT=1. RXON is READ-ONLY.
TXON	TXON=1 indicates that the Transmit function is enabled. TXON is set when DTX (CSR15.1)=0 and STRT=1 TXON is READ-ONLY.
TDMD	TDMD when set, causes the Buffer Management State Machine to perform a poll of the RX and TX Descriptor Rings. Typically this bit is used to hasten the transmitting of a frame. If this bit is not set high after a TX frame is added to the ring, then the controller will not begin DMA until after the poll-time counter has expired. TDMD is set by writing a 1. Writing a 0 has no effect. TDMD is cleared by the Buffer Management State Machine after the Descriptor Ring Poll is complete. TDMD is cleared by setting the STOP bit.
STOP	STOP assertion disables the chip from all external activity. Setting STOP overrides STRT or INIT when the bits are set at the same time. Note that STOP is set when RESET is asserted. This causes numerous bits to be forced into their reset state. STOP is set by writing a 1 or by RESET. Writing a 0 has no effect. STOP is cleared by setting either STRT or INIT.
STRT	STRT assertion enables the controller. Buffer Management begins operation and frames can be transmitted or received. Setting STRT clears the STOP bit. If STRT and INIT are set at the same time, the INIT block is read first, then the controller begins operation. STRT is set by writing a 1. Writing a 0 has no effect. STRT is cleared by setting STOP.
INIT	INIT assertion causes the controller to DMA the INIT block from memory. Setting INIT clears the STOP bit. If STRT and INIT are set at the same time, the INIT block is read first, then the controller begins operation. INIT is set by writing a 1. Writing a 0 has no effect. INIT is cleared by setting STOP.

Register 65 - CSR1: Initialization Address Register, IADR[15:0]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

IADR 15:0	Lower 16 bits of the Initialization address register. Bits 2:0 must be 0. This register is the same as CSR16. READ/WRITE only when STOP=1.

Register 66 - CSR2: Initialization Address Register, IADR[23:16]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	AD23	AD22	AD21	AD20	AD19	AD18	AD17	AD16

15:8	Reserved. R/W as zero.
IADR 23:16	Upper 8 bits of the Initialization address register. This register is the same as CSR17. READ/WRITE only when STOP=1.

Register 67 - CSR3: Interrupt Mask Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	MISSM	0	RINTM	TINTM	IDONM	0	0	0	DTX2PD	0	0	0	0

15	R/W as zero.
14	R/W as zero. (It is BABLM bit in AM79C960. If BABLM is set, the BABL bit in CSR0 will be masked and will not set the INSTR bit.)
13	R/W as zero.
MISSM	If MISSM is set, the MISS bit in CSR0 will be masked and will not set the INSTR bit. MISSM is not affected by STOP.
11	Reserved, R/W as zero. (It is MERRM bit in AM79C960. The controller does not generate Memory Errors and thus does not need a mask bit)
RINTM	If RINTM is set, the RINT bit in CSR0 will be masked and will not set the INSTR bit. RINTM is not affected by STOP.
TINTM	If TINTM is set, the TINT bit in CSR0 will be masked and will not set the INSTR bit. TINTM is not affected by STOP.
IDONM	If IDONM is set, the IDON bit in CSR0 will be masked and will not set the INSTR bit. IDONM is not affected by STOP.
7-5	R/W as zero.
DTX2PD	DTX2PD=1 Disables transmit two part deferral. DTX2PD is not affected by STOP.
3-0	R/W as zero. (Bit 3 in the AM79C960 enabled the Modified Backoff algorithm. The controller does not implement this functionality).

Register 68 - CSR4: Features Control

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENTST	0	0	DPOLL	APAD _TX	0	MF C0	MF COM	RPA	BAK _FAST	RCV CCO	RCV COM	TX STRT	TX STRTM	PAUSE	PAUSEM

ENTST	ENTST=1 enables reading and writing to various registers, even when the STOP bit is 0. It also enables various test modes to ease chip testing. This bit enables a test mode in the AM79C960 but it functions differently. This bit is only meant to be used during chip testing.
14	Reserved , R/W as zero. (It is DMAPLUS bit in AM79C960. Bus cycles in the controller don't require programming)
13	Reserved, R/W as zero. (It is TIMER bit in AM79C960, used to enable the bus timer register. Bus cycles in the controller don't require programming).
DPOLL	DPOLL=1 disables transmit polling. The transmit ring will only be polled when the receive ring is polled or when the TDMD bit in CSR0 is set. DPOLL=0 enables automatic polling of the transmit ring buffers when the Poll-Time Counter (CSR46) reaches zero.
APAD_TX	APAD_TX=1 enables automatic padding of frames that are less than 64 bytes long. Zeroes will be added to the frame, to extend the length of the frame to be 64 bytes, including the FCS. The FCS includes the pad bytes. APAD_TX will override the DXMTFCS bit (CSR15.3).
10	Reserved, R/W as zero. (It is ASTRP_RCV in AM79C960 . The controller does not support automatic pad/FCS stripping of received frames)
MFCO	MFCO=1 indicates that the Missed Frame Counter in CSR112 has overflowed. MFCO will assert INSTR if the mask bits are 0. MFCO is cleared by writing a 1 or setting STOP. Writing a 0 has no effect.
MFCOM	MFCOM=1 will mask the MFCO from asserting INSTR. MFCOM=0 enables MFCO to assert INSTR. MFCOM is set by RESET.
RPA	RPA=1 enables runt packets to be accepted by the RX MAC. Runts are always accepted by the TX MAC. This bit is meant for testing purposes only. (This is a reserved bit in the AM79C960)
BAK_FAST	This bit is for test purposes only. BAK_FAST must be programmed with a 0 during normal operation. If ENTEST and BAK_FAST are both 1, then the backoff timer in the TX MAC state machine will limit the backoff time to maximum of 32 slot times even if number of retransmission attempts is grater than 5. This bit is meant for testing purposes only (This is a reserved bit in the AM79C960).
RCVCCO	RCVCCO=1 indicates that the Receive Collision Counter (CSR114) has overflowed. RCVCCO will assert INSTR if the mask bits are 0. RCVCCO is cleared by writing a 1 or setting STOP. Writing a 0 has no effect.
RCVCCOM	RCVCCOM=1 will mask the RCVCCO from asserting INSTR. RCVCCOM=0 enables RCVCCOto assert INSTR. RCVCCOM is set by RESET.
TXSTRT	TXSTRT=1 indicates that a Transmit frame has begun. The TX_MAC state machine sets this bit when TENA is asserted. TXSTRT is cleared by writing a 1 or by setting STOP. Writing a zero has no effect.
TXSTRTM	TXSTRTM=1 will mask the TXSTRT from asserting INSTR. TXSTRTM=0 enables TXSTRT to assert INSTR. TXSTRTM is set by RESET.

PAUSE PAUSE=1 indicates that the controller received a Pause Control Frame. PAUSE is cleared by writing a 1 or setting STOP. Writing a 0 has no effect. PAUSE will assert INSTR if the mask bit is 0 (In the AM79C960 with an integrated MAU it was JAB bit)

PAUSEM PAUSEM=1 will mask the PAUSE from asserting INSTR. PAUSEM=0 enables PAUSE to assert INSTR. PAUSEM is set by RESET. (In the AM79C960 with an integrated MAU it was JABMask bit)

Register 69 - CSR8: Reserved

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

15:0 Reserved R/W as zero. The controller doesn’t support Logical Address filter.

Register 70 - CSR9: Reserved

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

15:0 Reserved R/W as zero. The controller doesn’t support Logical Address filter.

Register 71 - CSR10: Reserved

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

15:0 Reserved R/W as zero. The controller doesn’t support Logical Address filter.

Register 72 - CSR11: Reserved

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

15:0 Reserved R/W as zero. The controller doesn’t support Logical Address filter.

Register 73 - CSR12: PADR[15:0]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

PADR 15:0 Physical Address bits 15 through 0. The PADR bits are received PADR[0] first and PADR[47] last. READ/WRITE only when STOP=1.

Register 74 - CSR13: PADR[31:16]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD31	AD30	AD29	AD28	AD27	AD26	AD25	AD24	AD23	AD22	AD21	AD20	AD19	AD18	AD17	AD16

PADR 31-16 Physical Address bits 31 through 16. READ/WRITE only when STOP=1.

Register 75 - CSR14: PADR[47:32]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD47	AD46	AD45	AD44	AD43	AD42	AD41	AD40	AD39	AD38	AD37	AD36	AD35	AD34	AD33	AD32

PADR 47-32 Physical Address bits 47 through 32. READ/WRITE only when STOP=1.

Register 76 - CSR15: Mode Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PROM	DRXBC	DRXPA	0	DPAUSE	0	0	0	0	0	DRTY	0	DTXFCS	0	DTX	DRX

PROM	PROM=1 enables "Promiscuous Mode". All receive frames are accepted regardless of the destination address. Read/Write only when STOP is set.
DRXBC	DRXBC=1 disables receive Broadcast messages. Read/Write only when STOP is set.
DRXPA	DRXPA=1 disables receive Physical Address messages. Frames matching the Physical address will be ignored. Read/Write only when STOP is set.
12	Reserved, R/W as zero. (In Am79C960 it is defined as DLNKST : disable link status monitoring)
DPAUSE	DPAUSE=1 disables automatic Pause Operation. DPAUSE=0 enables Pause Operation. When the controller receives a Control Frame with a Pause opcode, the next frame to be transmitted will be suspended until requested Pause Time expires. (AM79C960 defines this bit a s Disable Automatic Polarity Correction which is used by the on-chip MAU. Since the controller does not include an MAU, this bit is redefined)
10	Reserved, R/W as zero. (AM79C960 defines this bit a s MENDEC Loopback which is used by the on-chip MAU. Since the controller does not include an MAU, this bit has no meaning)
9	Reserved , R/W as zero (AM79C960 defines this bit a s Low Receive Threshold/Transmit Mode Select which is used by the on-chip MAU. Since the controller does not include an MAU, this bit has no meaning)
8	Reserved, R/W as zero. (AM79C960 defines this bit a s PORTSEL[1] which selects the on-chip MAU. Since the controller does not include an MAU, this bit has no meaning).
7	Reserved, R/W as zero. (AM79C960 defines this bit a s PORTSEL[0] which selects the on-chip MAU. Since the controller does not include an MAU, this bit has no meaning).
6	R/W as zero. (The AM79C960 uses this bit to enable Internal Loopback)
DRTY	DRTY=1 disables transmit retries. Only 1 attempt will be made to transmit a TX frame. DRTY=0 enables up to 16 TX retries. Read/Write only when STOP is set.
4	Reserved R/W as zero. (The AM79C960 uses this bit to Force a Collision. This is only used for chip testing purposes and is not implemented on the controller)
DTXFCS	DTXFCS=1 disables transmit CRC generation. DTXFCS=0 enables transmit CRC generation, the FCS will be appended to the frame. Also see the ADD_FCS bit in TMD1. Read/Write only when STOP is set.
2	Reserved, R/W as zero. (The AM79C960 uses this bit and bits 4 and 10 to select various loopback modes. The controller supports PHY through MII interface)
DTX	DTX=1 disables transmit operation. Accesses to the transmit ring buffers will stop. DTX=0 enables the Buffer Management Unit to access the TX ring buffers. Read/Write only when STOP is set.
DRX	DRX=1 disables receive operation. Accesses to the receive ring buffers will stop. DRX=0 enables the Buffer Management Unit to access the RX ring buffers. Read/Write only when STOP is set.

Register 77 - CSR16: Initialization Address Register, IADR[15:0]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

IADR 15:0	Lower 16 bits of the Initialization address register. Bits 2:0 must be 0. This register is the same as CSR1. READ/WRITE only when STOP=1.

Register 78 - CSR17: Initialization Address Register, IADR[23:16]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	AD23	AD22	AD21	AD20	AD19	AD18	AD17	AD16

15:8	Reserved. R/W as zero.
IADR 23:16	Upper 8 bits of the Initialization address register. This register is the same as CSR2. READ/WRITE only when STOP=1.

Register 79 - CSR24: Base Address of RX ring[15:0]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	BA14	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0

BADR 15:0	Lower 16 bits of the Base Address of RX ring. CSR24[0] is the LSb and CSR25[7] is the MSb. Bits [2:0] are assumed to be 000. READ/WRITE only when STOP is set.

Register 80 - CSR25: Base Address of RX ring[23:16]]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16

BADR 16:23	Upper 8 bits of the Base Address of RX ring.
31-24	R/W as zeroes.

Register 81 - CSR30: Base Address of TX ring[15:0]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	BA14	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0

BADR 15:0	Lower 16 bits of the Base Address of TX ring. CSR30[0] is the LSb and CSR31[7] is the MSb. Bits [2:0] are assumed to be 000. READ/WRITE only when STOP is set.

Register 82 - CSR31: Base Address of TX ring[23:16]]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16

BADR 16:23	Upper 8 bits of the Base Address of TX ring.
31-24	R/W as zeroes.

Register 83 - CSR46: Poll Time Counter, POLL

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

15:0	This is the value in the Poll-Time Counter. When this counter overflows, a Poll of the TX and RX descriptor rings is performed and the counter is reloaded with CSR47. The Poll Time Counter increments on every CLK when CSR0 STRT =1. READ/WRITE only when STOP=1.

Register 84 - CSR47: Polling Interval, POLLINT

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

15:0	This register contains the value which is loaded into the Poll Time Counter when the counter overflows. The value is the two's complement of the desired interval. Each count is one CLK period. The least significant 4 bits, [3:0], are not used and assumed to be zero. The register is really 17 bits wide with bit[16] assumed to be a one. The following value is loaded into CSR46: {1, POLLINT[15:4],000}. Thus, the default value of 0000 corresponds to 32K clocks = 1.3ms @CLK=50 MHz. READ/WRITE only when STOP=1.

Register 85 - CSR76: Receive Ring Length, RXLEN

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

15:0	This register contains the length of the RX ring in two's-complement form. It is assigned the value from the RLEN field during initialization. This register can also be programmed manually to allow ring length of any size. Only the low 12 bits are used, bits [15:12] are assumed to be ones. Thus, the maximum ring length is 4K buffers. READ/WRITE only when STOP=1. Set to 0xFFFF on reset (one buffer) The AM79C960 has a full 16-bit register here. More than 4K buffers are impractical, so the controller only implements 12 bits.

Register 86 - CSR78: Transmit Ring Length, TXLEN

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

15:0	This register contains the length of the TX ring in two's-complement form. It is assigned the value from the TLEN field during initialization. This register can also be programmed manually to allow ring length of any size. Only the low 12 bits are used, bits [15:12] are assumed to be ones. Thus, the maximum ring length is 4K buffers. READ/WRITE only when STOP=1. Set to 0xFFFF on reset (one buffer) The AM79C960 has a full 16-bit register here. More than 4K buffers are impractical, so the controller only implements 12 bits.

Register 87 - CSR88-89: Chip ID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV	REV	REV	REV	PN	PN	PN	PN	PN	PN	PN	PN	PN	PN	PN	PN

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

31-28	Chip Revision register, read only
27-12	16-Bit code for the Part Number = 0000000000000000, read only.
11-1	Manufacturer ID = 01001111111, read only.
0	Always 1 read only.

Register 88 - CSR112: Missed Frame Count, MFC

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

15-8	R/W as zero. Note that the AM79C960 had a full 16 bit value for this field.
7-0	Counts the number of missed frames. When this counter overflows, it sets the MFCO(CSR4.9) bit. Cleared by STOP.

Register 89 - CSR114: Receive Collision Count, RXCC

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

15-8	R/W as zero. Note that the AM79C960 had a full 16 bit value for this field.
7-0	Counts the number of receive collisions both regular and late. When this counter overflows, it sets the RCVCCO (CSR4.5) bit. Cleared by STOP.

7.3.6 MII Port Register

The MIIP register is accessed (directly) at offset 0x18. Bits MDC and MDO of this register drive MDC and MDIO signals of the MII bus. The MII management frame is generated by a sequence of writes to MDC, MDI and MDOE bits.

Register 90 - MII Port Register, MIIP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FDEN	0	0	0	0	0	0	MD1	MDOE	0	0	0	0	0	MDC	MDO

FDEN	FDEN=1 sets the controller to Full-Duplex mode FDEN=0 sets the controller to Half-Duplex mode. This bit is used internally by the controller, and is not a part of MII management interface. However, the value written into FDEN bit must match the mode an external PHY is set to. FDEN is cleared by RESET.
14-8	Reserved. Read as zeros.
MDI	Read only, this bit indicates actual logical value on MII MDIO bi-directional signal. During a Management Read Frame, this bit reads serial data sent from PHY to the controller.
MDOE	This bits controls the Output-Enable Pin of IO buffer driving MDIO signal. MDOE=1 the controller is driving MDIO signal. MDOE=0 the controller doesn’t drive MDIO (MDIO is Z or driven by PHY) Note: If MDOE=1 and MDI reads different value than MDO , then there is a contention on MDIO signal (programming sequence error). MDOE is cleared by RESET.
6-2	Reserved. Read as zeros.
MDC	This bit drives the clock signal (MDC) of the MII management interface. Every transition of this bit causes a transition of the MDC signal. The MII MDC clock is not required to be periodic. A value written into MDC bit should remain unchanged for at least 160 ns (8 processor clocks @ CLK=50 MHz) and a minimum period of 400 ns has to be ensured (20 processor clocks @ CLK=50 MHz) MDC can be 0 or 1 for an unlimited time. MDC is cleared by RESET.

MDO If MDOE=1 then MDO bit drives the data signal (MDIO) of the MII interfaces. MDO must not be changed at the same time when a 1 is being written to MDC bit. MDO is cleared by RESET.

7.3.7 Media Independent Interface (MII) Operation

With respect to the 7-layer communications model, the Physical Layer 1 is referred to as PHY. A typical PHY device is the LXT971A Dual-Speed Fast Ethernet Transceiver.

The PHY device implements the Media Independent Interface (MII) as defined in the IEEE 802.3 standard. The MII consists of a data interface and a management interface.

Separate channels are provided for transmitting data from the Ethernet controller to the PHY (E_TXD), and for passing data received from the line (E_RXD) to the Ethernet controller. Each channel has its own clock, data bus, and control signals.

Nine signals are used to pass received data to the Ethernet controller:
E_RXD<3:0>
E_RX_CLK
E_RX_DV
E_RX_ER
E_COL
E_CRS

Seven signals are used to transmit data from the Ethernet controller:
E_TXD<3:0>
E_TX_CLK
E_TX_EN
E_TX_ER

The PHY supplies both clock signals as well as separate outputs for carrier sense and collision. Data transmission across the MII is normally implemented in 4-bit-wide nibbles.

8 CAN Controller

8.1 CAN Overview

There are two completely independent CAN peripheral controllers. Both CAN controllers are 100% Bosch CAN 2.0B compliant. The technical highlights are the following:

- 3 programmable acceptance filters
  - Message filter covers: ID, IDE, RTR, 16 DATA bits
  - Each filter has its own enable flag
- Transmit Path
  - 3 Tx message holding registers with internal priority arbiter
  - Message abort command
- Receive FIFO
  - 4 message deep receive FIFO
  - FIFO status indicator
- Bus coupler
  - Intel style interface module
  - Full synchronous zero wait-states interface
  - Status and configuration interface
- Programmable Interrupt Controller
- Listen only mode
- CANbus analysis functions
  - Arbitration lost capture
  - Error event capture
  - Actual frame reference pointer
- Programmable CANbus physical layer interface

8.2 CAN Operation

The CAN controller appears to the microprocessor as a memory-mapped I/O device.

Each peripheral has 256 bytes of I/O address space allocated to it. The starting I/O address for each peripheral depends on the Peripheral Chip Select (PCS) line assigned to it. PCS1 is assigned to CAN0 and PCS2 is assigned to CAN1. The Base I/O address is determined by the Peripheral Base Address (PBA) and the offset. The PBA is determined via the PACS register.

Table 32 - CAN I/O Address

Select Line	Assigned To	Base I/O Address
PCS1	CAN0	PBA + 0100h
PCS2	CAN1	PBA + 0200h

The Peripheral Chip Selects are programmed through two registers—the Peripheral Address Chip Select (PACS) register and the PCS and MCS Auxiliary (MPCS) register. The Peripheral Address Chip Select (PACS) register determines the base address, the ready condition, and the wait states for the PCS3–PCS0 outputs.

The MPCS register determines whether PCS chip selects are active during memory or I/O bus cycles and specifies the ready and wait states for the PCS6–PCS5 outputs. The CAN devices must be mapped into the I/O space.

The PCS pins are not active on reset. The PCS pins are activated as chip selects by writing to the PACS and MPCS registers.

Peripheral Address Chip Select Register, PACS (20-bit mode)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A19	A18	A17	A16	A15	A14	A13	A12	A11	1	1	1	1	R2	R1	R0
0	0	0	0												

A19:11	Upper address bits of the starting address of the Peripheral Base Address (PBA). Allows programmer to set the PBA at any 2K byte address boundary. Since we're operating only in I/O space which is limited to 64 Kbytes, A19-A16 must always be zero.
R2:0	Ready generation select. See Table 5 - MMCS Wait State and Ready Select.

Peripheral Address Chip Select Register, PACS (24-bit mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	R2	R1	R0
0	0	0	0	0	0	0	0								

A23:11	Upper address bits of the starting address of the Peripheral Base Address (PBA). Allows programmer to set the PBA at any 2K byte address boundary. Since we're operating only in I/O space which is limited to 64 Kbytes, A23-A16 will always be zero.
R2:0	Ready generation select. See Table 5 - MMCS Wait State and Ready Select.

8.2.1 CAN Interrupts (INT1 – INT2)

The CAN controllers use the following interrupts.

Table 33 - CAN Interrupt Assignments

Interrupt Signal	Assignment	Vector Type	Vector Address	Default Priority
INT1	CAN0	13	34H	7
INT2	CAN1	14	38H	8

8.2.2 CAN Interrupt Control Registers

These are the control registers for the external interrupt sources. Each control register contains the level-trigger mode bit, mask bit and programmable priority level. The mask bits in these control registers are the same mask bits as in the mask register. Modifying them in the control register will also modify the corresponding mask bits in the mask register, and vice versa. This register is read/write

Register 91 - External Interrupt Control Registers, I1CON & I2CON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	LTM	MSK	PRI2	PRI1	PRI0

LTM	The level-trigger mode bit will set the respective interrupt source for either level triggered mode (LTM=1), or edge triggered mode (LTM=0). When set for level-triggered mode an interrupt is generated whenever the external interrupt signal is high. In edge triggered mode an interrupt request is generated when the external interrupt signal makes a low to high transition. In both cases, the level must remain high until the interrupt is acknowledged.
MSK	Setting this bit is will mask the respective interrupt requests. Resetting this bit will enable the respective interrupts.
PR2:0	These bits represent the programmable priority level for the respective interrupt source. Bit combination 000\b has the highest priority while 111\b has the lowest priority.

Name	Mnemonic	Offset
End of Interrupt	EOI	22h
Poll	POLL	24h
Poll Status	POLLST	26h
Interrupt Mask	IMASK	28h
Priority Mask	PRIMSK	2Ah
In-Service	INSERV	2Ch
Interrupt Request	REQST	2Eh
Interrupt Status	INTSTS	30h
INT 0 Control	I0CON	38h
INT 1 Control	**I1CON**	**3Ah**
INT 2 Control	**I2CON**	**3Ch**

8.2.3 In-Service Register (INSERV, offset 2Ch)

This register contains the in-service bits for each of the interrupt sources. An in-service bit is set to indicate that a source's service routine is being executed. When an in-service bit is set the interrupt controller will not generate interrupts from sources with a lower priority than the current interrupt being serviced. This allows interrupt service routines to run with interrupts enabled. This register is read/write.

Register 92 - In-Service Register for CAN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	**I2**	**I1**	I0	D1	D0	0	TMR

SPI	In-service bit for the Serial Peripheral Interface
D3:0	In-service bits for the DMA channels (DMA3:0).
I5:0	**In-service bits for the external interrupt sources (INT5:0).**
SP1:0	In-service bits for the asynchronous serial ports (SP1:0).
TMR	In-service bit for all three timers (TMR2:0).

8.2.4 Interrupt Request Register (REQST, offset 2Eh)

This register contains the interrupt request bits for the Timers, DMA channels and external interrupt sources. This register is read/write except were noted.

Register 93 - Interrupt Request Register for CAN

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	I2	I1	I0	D1	D0	0	TMR

I5:0	Interrupt request bits for the external interrupt sources (INT5:0). These bits represent the actual state of the external interrupt pins, after they have been passed through a two level synchronizer, or a level detector depending on the LTM bit for that respective interrupt line. These bits will be set whenever an external interrupt line make a low to high transition if LTM=0. Otherwise, if LTM=1 these bits will follow the input pins after a two clock delay. Because these bits represent the state of the external interrupt pins they can not be written.

8.3 Internal Registers

The DSTni-LX chip contains two totally independent CAN channels. The operation and access to each device however is identical. The only difference is the starting I/O BASE address for each channel. Both CAN channels have their registers located and fixed in the internal I/O space of the DSTni-LX chip. Both are implemented as true 16-bit devices. Therefore, ***all accesses made to the CAN channel registers must be 16-bit I/O type accesses in the I/O space***. Byte accesses will result in erroneous operation. The CAN channel registers can ***not*** be mapped into the internal "memory" space of the DSTni-LX.

There are a total of 62, 16-bit registers for each CAN channel that enable configuration, control, status, and operational data. The following table is the 16-bit register mapping for both CAN channels of these registers.

8.3.1 Port Map of Internal Registers

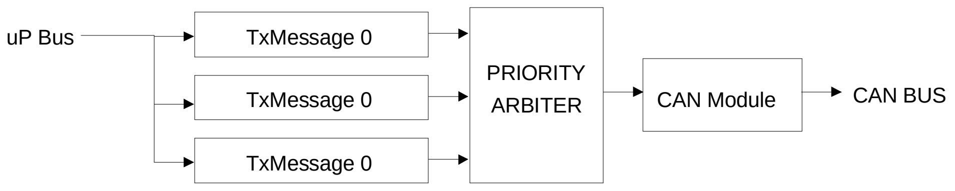
Table 34 - CAN Internal Registers

Address	Type	16-bit Register
0x00	R/W	TxMessage_0: ID, ID28-13
0x02	R/W	ID12-00
0x04	R/W	TxMessage_0: Data, D55-48, D63-56
0x06	R/W	D39-32, D47-40
0x08	R/W	D23-16, D31-24
0x0A	R/W	D07-00, D15-08
0x0C	R/W	TxMessage_0: RTR, IDE, DLC_3-0
0x0E	R/W	TxMessage_0: Control Flags, TXAbort, TRX
0x10	R/W	TxMessage_1: ID, ID28-13
0x12	R/W	ID12-00
0x14	R/W	TxMessage_1: Data, D55-48, D63-56
0x16	R/W	D39-32, D47-40
0x18	R/W	D23-16, D31-24
0x1A	R/W	D07-00, D15-08
0x1C	R/W	TxMessage_1: RTR, IDE, DLC_3-0
0x1E	R/W	TxMessage_1: Control Flags, TXAbort, TRX
0x20	R/W	TxMessage_2: ID, ID28-13
0x22	R/W	ID12-00
0x24	R/W	TxMessage_2: Data, D55-48, D63-56
0x26	R/W	D39-32, D47-40
0x28	R/W	D23-16, D31-24
0x2A	R/W	D07-00, D15-08
0x2C	R/W	TxMessage_2: RTR, IDE, DLC_3-0
0x2E	R/W	TxMessage_2: Control Flags, TXAbort, TRX
0x30	R	RxMessage: ID, ID28-13
0x32	R	ID12-00
0x34	R	RxMessage: Data, D55-48, D63-56
0x36	R	D39-32, D47-40
0x38	R	D23-16, D31-24
0x3A	R	D07-00, D15-08
0x3C	R	RxMessage: RTR, IDE, DLC_3-0,AFI_2-0
0x3E	R/W	RxMessage: Control Flags, Fifo_Lvl_2-0, MsgAval
0x40	R	Transmitter and Receive Error Counter
0x42	R	Error Status
0x44	R/W	Message Level Threshold
0x46	R/W	Interrupts Flags
0x48	R/W	Interrupt Enable Register
0x4A	R/W	CAN mode, Loop_Back, Passive, Run
0x4C	R/W	CAN Bit Rate Div., cfg_bitrate_10-0
0x4E	R/W	CAN tsegs

Address	Type	16-bit Register
0x50	R/W	Acceptance Filter Enable Register, AFE_2-0
0x52	R/W	Acceptance Mask Register 0 (AMR0), ID28-13
0x54	R/W	ID12-00, IDE, RTR
0x56	R/W	D55-48, D63-56
0x58	R/W	Acceptance Code Register 0 (ACR0), ID28-13
0x5A	R/W	ID12-00, IDE, RTR
0x5C	R/W	D55-48, D63-56
0x5E	R/W	Acceptance Mask Register 1 (AMR1), ID28-13
0x60	R/W	ID12-00, IDE, RTR
0x62	R/W	D55-48, D63-56
0x64	R/W	Acceptance Code Register 1 (ACR1), ID28-13
0x66	R/W	ID12-00, IDE, RTR
0x68	R/W	D55-48, D63-56
0x6A	R/W	Acceptance Mask Register 2 (AMR2), ID28-13
0x6C	R/W	ID12-00, IDE, RTR
0x6E	R/W	D55-48, D63-56
0x70	R/W	Acceptance Code Register 2 (ACR2), ID28-13
0x72	R/W	ID12-00, IDE, RTR
0x74	R/W	D55-48, D63-56
0x76	R/W	Arbitration Lost Capture Register (ALCR)
0x78	R/W	Error Capture Register (ECR)
0x7A	R/W	Frame Reference Register (FRR)

8.3.2 TX Message Registers

To avoid priority inversion issues in the transmit path, three transmit buffers are available with a built-in priority arbiter. Upon transmission of a message, the priority arbiter evaluates all pending messages and selects the one with the highest priority. The message priority is re-evaluated after each message abort event such as arbitration loss.



Sending a message

- Write message into one of the transmit message holding registers (TxMessage0/1/2)
- Request transmission by setting the respective TRX flag. This flag remains set as long as the message holding registers contains this message. The content of the message buffer must not be changed while the TRX flag is set!
- The TRX flags remains set as long as the message transmit request is pending.
- The successful transfer of a message is indicated by the respective tx_xfer interrupt and by releasing the TRX flag. Depending on the tx_level configuration settings an additional interrupt source tx_msg is available to indicate that the message holding registers are empty or bellow a certain level.

8.3.3 Removing a message from a transmit holding register

A message can be removed from one of the three transmit holding registers (TxMessage0/1/2) by setting the TxAbort flag.

Use following procedure to remove the contents of a particular TxMessage buffer:

- Set TxAbort to request the message removal
- This flag remains set as long as the message abort request is pending. It is cleared when either the message won arbitration (tx_xmit interrupt active) or the message was removed (tx_xmit interrupt inactive).

8.3.4 TxMessage Registers

The following table shows TxMessage_0 registers. The registers for TxMessage_1 and TxMessage_2 are identical except for the offsets. See the complete register table at the end of this section.

Table 35 - TxMessage_0 Registers

Offset 00h, TxMessage_0:ID28

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13

Offset 02h, TxMessage_0:ID12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	0	0	0

Offset 04h, TxMessage_0:Data 55

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56

Offset 06h, TxMessage_0:Data 39

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D39	D38	D37	D36	D35	D34	D33	D32	D47	D46	D45	D44	D43	D42	D41	D40

Offset 08h, TxMessage_0:Data 23

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D23	D22	D21	D20	D19	D18	D17	D16	D31	D30	D29	D28	D27	D26	D25	D24

Offset 0Ah, TxMessage_0:Data 7

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D07	D06	D05	D04	D03	D02	D01	D00	D15	D14	D13	D12	D11	D10	D09	D08

Offset 0Ch, TxMessage_0:RTR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	RTR	IDE	DLC3	DLC2	DLC1	DLC0

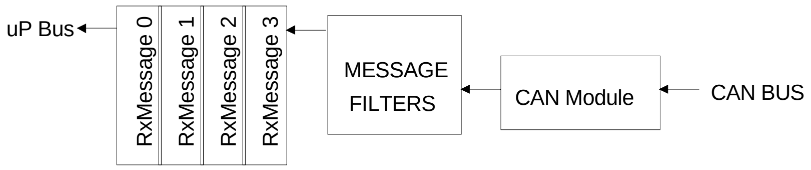
Offset 0Eh, TxMessage_0:Ctrl Flags

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

ID_28 .. ID_0:	Message identifier for both standard and extended messages. Standard messages use ID_28 .. ID_18
D_63 .. D_0:	Message data: Byte 1 is D_63 .. D_56, Byte 2 is D_55 .. D_48 etc.
RTR:	Remote bit
IDE:	Extended identifier bit
DLC_3:DLC_0:	Data length code, invalid values are transmitted as they are, but only 8 data bytes.
TRX:	Writing a ‘ 1’ will initiate a message transmit request. Note: The Tx message buffer must not be changed while TRX is ‘ 1’ ! When the whole message is successfully transmitted, TRX will go low.
TxAbort:	Set this flag to request the removal of the pending message in Tx message buffer. This happens the next time when an arbitration loss occurred. The flag is cleared when the message was removed or when the message won arbitration. The TRX flag is released at the same time as well.

8.3.5 RX Message Registers

A four message deep FIFO stores the incoming messages. Status flags indicate how many messages are stored. Additional flags are provided to determine from which acceptance filter the actual message is coming from.



Procedure for reading received messages

- Wait for rx_msg interrupt.
- MessageReadLoop:
  - read message
  - acknowledge ‘ message read’ by writing a ‘ 1’ to MsgAv register
  - read MsgAv; reading a ‘ 1’ means a new message is available
  - IF MsgAv=1 THEN jump to MessageReadLoop
- Acknowledge rx_msg interrupt by writing a ‘ 1’ to this register location.

This is the recommended procedure for Rx message handling.

8.3.6 RxMessage Registers

The following table shows RxMessage registers. See the complete register table at the end of this section.

Table 36 - RxMessage Registers

Offset 30h, RxMessage:ID28

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13

Offset 32h, RxMessage:ID12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	0	0	0

Offset 34h, RxMessage:Data 55

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56

Offset 36h, RxMessage:Data 39

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D39	D38	D37	D36	D35	D34	D33	D32	D47	D46	D45	D44	D43	D42	D41	D40

Offset 38h, RxMessage:Data 23

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D23	D22	D21	D20	D19	D18	D17	D16	D31	D30	D29	D28	D27	D26	D25	D24

Offset 3Ah, RxMessage:Data 7

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D07	D06	D05	D04	D03	D02	D01	D00	D15	D14	D13	D12	D11	D10	D09	D08

Offset 3Ch, RxMessage:RTR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	AFI _2	AFI _1	AFI _0	0	0	RTR	IDE	DLC _3	DLC _2	DLC _1	DLC _0

Offset 3Eh, RxMessage:Msg Flags

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Fifo2	Fifo1	Fifo0	0	0	0	0	Msg Aval

ID_28 .. ID_0:	Message identifier for both standard and extended messages. Standard messages use ID_28 .. ID_18; ID_17..ID_0 are set to ‘ 1’ .
D_63 .. D_0:	Message data: Byte 1 is D_63 .. D_56, Byte 2 is D_55 .. D_48 etc.
AFI_2 .. AFI_0:	Acceptance filter indicator: These bits indicate which acceptance filter(s) accepted the incoming message. If more than one filter accepted the message than more than one bit is set.
RTR:	Remote bit
IDE:	Extended identifier bit
DLC_3:DLC_0:	Data length code, invalid values are received as they are!
MsgAval:	MsgAval will go high when a new message is available. Writing a ‘ 1’ will clear this flag and indicate that the message has been read. If an other message is available, than this flag is not cleared and the new message from RxMsg1 buffer is accessible.
Rx_Fifo[2:0]:	Rx FIFO status. These two flags indicate how many messages are waiting in the queue. These flags are read-only! “000”: empty “001”: 1/4 full “010”: 1/2 full “011”: 3/4 full “100”: full others: not applicable

8.3.7 Error Count and Status Registers

Offset 40h, Tx / Rx Error Count

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0	TE7	TE6	TE5	TE4	TE3	TE2	TE1	TE0

Offset 42h, Error Status

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	RX96	TX96	ES1	ES0

Offset 44h, Tx / Rx Message Level

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	RL1	RL0	TL1	TL0

RE7-0	These are the Rx_er_cnt bits. The receiver error counter according to the Bosch CAN specification. When in bus off, this counter is used to count the idle states.
TE7-0	These are the Tx_er_cnt bits. The transmitter error counter according to the Bosch CAN specification. When it is greater than 255(dec), it is fixed at 255.

The following status information is read only, there is no set/reset possible.

RX96	Rxgte96 or rx > 96. The receiver error counter is greater or equal 96(dec)
TX96	Tx96 or tx > 96. The transmitter error counter is greater or equal 96(dec)
ES1-0	error_stat. The error state of the CAN node: “00”: error active (normal operation) “01”: error passive “1x”: bus off

Special FIFO fill levels are provided to reduce the number of interrupts:

RL1-0	rx_level[1:0]: Sets the rx_msg interrupt threshold: 0: at least 1 message in receive Fifo 1: at least 2 messages in receive Fifo 2: at least 3 messages in receive Fifo 3: at least 4 messages in receive Fifo
TL1-0	tx_level[1:0]: Sets the tx_msg interrupt threshold: 0: all tx buffers are empty 1: minimum 2 empty buffers 2: minimum 1 empty buffer 3: not applicable

8.3.8 Interrupt Flags

The following flags are set on internal events (they activate an interrupt line when enabled). They are cleared by writing a ‘ 1’ to the according flag. Acknowledging the tx_msg interrupt acknowledges all tx_xmit interrupt sources as well. Acknowledging one of the tx_xmit interrupt sources acknowledges the tx_msg interrupt too.

Offset 46h, Interrupt Flags

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_ MSG	TX_ MSG	TX_X MIT2	TX_X MIT1	TX_X MIT0	BUS_ OFF	CRC_ ERR	FORM _ERR	ACK_ ERR	STUF _ERR	BIT_ ERR	RX_ OVR	OVR_ LOAD	ARB_ LOSS	0	0

RX_MSG:	Depending on rx_level, at least one message is available.
TX_MSG:	Depending on tx_level, at least one message buffer is empty.
TX_XMIT2/1/0	Indicates that the respective message was successfully sent.
BUS_OFF:	The CAN has reached the bus off state.
CRC_ERR, FORM_ERR, ACK_ERR, STUF_ERR, BIT_ERR:	Any of the mentioned error occurred while receiving or transmitting a message.
RX_OVR:	Receiver overrun: This Flag indicates when a new message arrived while the receive buffer is full. This Flag is set if either the incoming message overwrites an existing one or if it is discarded.
OVR_LOAD:	An overload condition has occurred.
ARB_LOSS:	The arbitration was lost while sending a message.
INT_ENB	Interrupt Enable.

8.3.9 Interrupt Enable Registers

All interrupt sources are grouped into three groups (traffic, error and diagnostics interrupts). To enable a particular interrupt, set its enable flag to ‘ 1’ .

Offset 48h, Interrupt Enable Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_ MSG	TX_ MSG	TX_X MIT2	TX_X MIT1	TX_X MIT0	BUS_ OFF	CRC_ ERR	FORM _ERR	ACK_ ERR	STUF _ERR	BIT_ ERR	RX_ OVR	OVR_ LOAD	ARB_ LOSS	0	INT_ ENB

rx_msg, tx_msg, tx_xmit2, tx_xmit1, tx_mit0 and rx_ovr:	int1_n group (traffic interrupts)
crc_err, form_err, ack_err, stuff_err, bit_err, bus_off:	int2_n group (error interrupts)
ovr_load, bit_sync, arb_loss:	int3_n group (diagnostic interrupts)
int_ebl:	general interrupt enable.

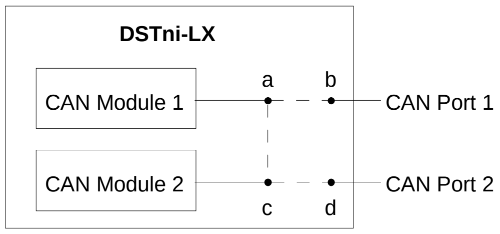
8.3.10 CAN Operating Mode

The CAN modules can be used in different operating modes. By disabling transmitting data, it is possible to us the CAN in listen only mode enabling features such as automatic bit rate detection. The two modules can be used in an on-chip loop-back mode.

Offset 4Ah, Operating Mode

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

RUN:	writing a ‘ 1’ sets the CAN controller into run mode. Read ‘ 1’ when running writing a ‘ 0’ sets the CAN controller into stop mode. Read ‘ 0’ when stopped
PASSIVE:	The output is held at ‘ R’ level. The CAN module is only listening. ‘ 0’ : CAN active ‘ 1’ : CAN passive
LOOP_BACK:	Internal LoopBack Mode: ‘ 0’ : a-b; c-d Default ‘ 1’ : a-c Internal loopback



Note: LoopBack mode register in CAN module 2 is non functional! For proper operation in loop-back mode the configuration of both CAN modules must be the same.

8.3.11 CAN Configuration Registers

These registers set the bit rate and other configuration parameters.

Offset 4Ch, Bit Rate Divisor

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	BR10	BR09	BR08	BR07	BR06	BR05	BR04	BR03	BR02	BR01	BR00

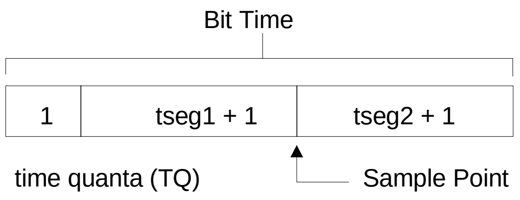
Offset 4Eh, Configuration

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVR_MSG	TS2_2	TS2_1	TS2_0	TS1_3	TS1_2	TS1_1	TS1_0	0	0	0	AUTO_RES	CFG_SJW1	CFG_SJW1	SAMP_MOD	EDGE_MOD

BR10:0	Cfg_bitrate[10:0] Prescaler for generating the time quantum: “00000000000”: Maximum speed (1 TQ = 1 clock cycle) “00000000001”: 1 TQ = 2 clock cycles ... “11111111111”: 1 TQ = 2048 clock cycles
OVR_MSG	overwrite_last_msg ‘ 0’ : Under the same conditions, a new message will be discarded and no rx_msg flag will be set. ‘ 1’ : When the FIFO is full and a new message arrives it overwrites the message in RxMsg3 buffer.
TS2_2:0	Cfg_tseg2: Length -1 of the second time segment. Cfg_tseg2=0 is not allowed; cfg_tseg2=1 is only allowed in direct sampling mode. See diagram below.
TS1_3:0	Cfg_tseg1: Length - 1 of the first time segment (bit timing). It includes the propagation time segment. Cfg_tseg1=0 and cfg_tseg1=1 are not allowed. See diagram below.
AUTO_RES	auto_restart: ‘ 0’ : After bus off, the CAN must be started ‘ by hand’ ‘ 1’ : After bus off, the CAN is restarting automatically after 128 groups of 11 recessive bits.
CFG_SJW1	Cfg_sjw: Synchronization jump width - 1. $sjw tseg1 \leq$ *and* $sjw tseg2 \leq$
SAMP_MOD	sampling_mode: ‘ 0’ : One sampling point is used in the receiver path ‘ 1’ : 3 sampling points with majority decision are used
EDGE_MOD	edge_mode: ‘ 0’ : Edge from ‘ R’ to ‘ D’ is used for synchronization ‘ 1’ : Both edges are used

Setting proper bitrate, tseg1 and tseg2

The following relations exist for bit time, time quanta, time segments 1/2 and the data sampling point.



Bittime = (1+ (tseg1 + 1) + (tseg2 + 1)) x timequanta

timequanta = (bitrate +1) / f_{clk}

e.g., for 1Mbps with f_{clk} = 8Mhz, set bitrate = 0, tseg1 = 3 and tseg2 = 2

Please observe the following conditions for setting tseg1 and tseg2:

tseg1=0 and tseg1=1 are not allowed
tseg2=0 is not allowed; tseg2=1 is only allowed in direct sampling mode.

8.3.12 Acceptance Filter and Acceptance Code Mask

Three programmable Acceptance Mask and Acceptance Code Register (AMR/ACR) pairs are available to filter incoming messages.

The acceptance mask register (AMR) defines whether the incoming bit is checked against the acceptance code register (ACR).

AMR: ‘ 0’ : The incoming bit is checked against the respective ACR. If the incoming bit and the respective ACR are not the same than the message is discarded.

‘ 1’ : The incoming bit is don’ t care.

Offset 50h, Acceptance Filter Enable Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	AFE_2	AFE_1	AFE_0

AFE0, AFE1, AFE2:

Each Acceptance Mask Register can be enabled with this flag.

‘ 0’ : Acceptance filter is disabled

‘ 1’ : Acceptance filter is enabled

If all three message filters are disabled then no messages will be received! To receive all messages then one message filter must be enabled and programmed with all its fields as “don’ t care”.

The following table shows the Acceptance Mask Register for AMR0 and the Acceptance Code Register ACR0. The registers for AMR1/ACR1 and AMR2/ACR2 are identical except for the offsets. See the complete register table at the end of this section.

Offset 52h, Acceptance Mask Register (AMR0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13

Offset 54h, AMR0:ID12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	IDE	RTR	0

Offset 56h, AMR0:Data 55

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56

Offset 58h, Acceptance Code Register (ACR0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13

Offset 5Ah, AMR0:ID12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	IDE	RTR	0

Offset 5Ch, AMR0:Data 55

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56

8.3.13 CANbus Analysis

For advanced analysis of a CAN system, three additional registers are provided. Apart of the arbitration lost and error capture registers, a CANbus frame reference register is provided as well. It contains additional information on the CANbus state as well as the physical Rx and TX pins as well.

Arbitration Lost Capture Register (ALCR)

Always the most recent arbitration loss event with the frame reference pointer is captured.

Offset 76h, Arbitration Lost Capture Register (ALCR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	FR4	FR3	FR2	FR1	FR0	0	0	FRB5	FRB4	FRB3	FRB2	FRB1	FRB0

Error Capture Register (ECR):

Always the most recent error event with the frame reference pointer, rx- and tx-mode and the according error code are captured.

Offset 78h, Error Capture Register(ECR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR2	ERR1	ERR0	FR4	FR3	FR2	FR1	FR0	TX_ MOD	RX_ MOD	FRB5	FRB4	FRB3	FRB2	FRB1	FRB0

Frame Reference Register (FRR)

The FRR contains information of the current bit of the CAN message. A frame reference pointer indicates the current bit position. This enables message tracing on bit level.

Offset 7Ah, Frame Reference Register (FRR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STUFF_ IND	RX_ BIT	TX_ BIT	FR4	FR3	FR2	FR1	FR0	RX_ MOD	TX_ MOD	FRB5	FRB4	FRB3	FRB2	FRB1	FRB0

ERR2:0	Error_code: This is the default: “000” crc_err: “001” form_err: “010” ack_err: “011” stuff_err: “100” bit_err: “101”
STUFF_IND	‘ 0’ : idle ‘ 1’ means a stuff bit is inserted
TX_MOD	‘ 0’ : not in TX mode (receiving or idle) ‘ 1’ : transmitting data
RX_MOD:	‘ 0’ : not in RX mode (transmitting or idle) ‘ 1’ : receiving data

FR4:0	frame_ref_field: This is the frame reference a incoming or outgoing CAN message. It is coded like this: stopped: “00000”; synchronize: “00001”; interframe: “00101”; bus_idle: “00110”; start_of_frame: “00111”; arbitration: “01000”; control: “01001”; data: “01010”; crc: “01011”; ack: “01100”; end_of_frame: “01101”; error_flag: “10000”; error_echo: “10001”; error_del: “10010”; overload_flag: “11000”; overload_echo: “11001”; overload_del: “11010”; other codes are not used!
RX_BIT:	The bit state on the receiver line
TX_BIT:	The bit state on the transmitter line
FRB5:0	frame_ref_bit_nr: A 6 bit vector used for counting the bit numbers in one field.
For example:	When field = “data” = “01010” and “bit_nr” = “000000” and “tx_mode” = ‘ 1’ that means transmitting the first data bit.

8.3.14 CAN Register Map

Off-set (Hex)	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TX Msg 0**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x02	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	-	-	-
0x04	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x06	…	D39	D38	D37	D36	D35	D34	D33	D32	D47	D46	D45	D44	D43	D42	D41	D40
0x08	…	D23	D22	D21	D20	D19	D18	D17	D16	D31	D30	D29	D28	D27	D26	D25	D24
0x0a	…	D07	D06	D05	D04	D03	D02	D01	D00	D15	D14	D13	D12	D11	D10	D09	D08
0x0c	…	-	-	-	-	-	-	-	-	-	-	RTR	IDE	DLC_3	DLC_2	DLC_1	DLC_0
0x0e	**TX Msg 0 Ctrl Flags**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TXAbort	TRX
0x10	**TX Msg 1**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x12	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	-	-	-
0x14	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x16	…	D39	D38	D37	D36	D35	D34	D33	D32	D47	D46	D45	D44	D43	D42	D41	D40
0x18	…	D23	D22	D21	D20	D19	D18	D17	D16	D31	D30	D29	D28	D27	D26	D25	D24
0x1a	…	D07	D06	D05	D04	D03	D02	D01	D00	D15	D14	D13	D12	D11	D10	D09	D08
0x1c	…	-	-	-	-	-	-	-	-	-	-	RTR	IDE	DLC_3	DLC_2	DLC_1	DLC_0
0x1e	**TX Msg 1 Ctrl Flags**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TXAbort	TRX
0x20	**TX Msg 2**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x22	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	-	-	-
0x24	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x26	…	D39	D38	D37	D36	D35	D34	D33	D32	D47	D46	D45	D44	D43	D42	D41	D40
0x28	…	D23	D22	D21	D20	D19	D18	D17	D16	D31	D30	D29	D28	D27	D26	D25	D24
0x2a	…	D07	D06	D05	D04	D03	D02	D01	D00	D15	D14	D13	D12	D11	D10	D09	D08
0x2c	…	-	-	-	-	-	-	-	-	-	-	RTR	IDE	DLC_3	DLC_2	DLC_1	DLC_0

Off-set (Hex)	**Register**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0x2e	**TX Msg 2 Ctrl Flags**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	TXAbort	TRX
0x30	**RX Msg**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x32	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	-	-	-
0x34	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x36	…	D39	D38	D37	D36	D35	D34	D33	D32	D47	D46	D45	D44	D43	D42	D41	D40
0x38	…	D23	D22	D21	D20	D19	D18	D17	D16	D31	D30	D29	D28	D27	D26	D25	D24
0x3a	…	D07	D06	D05	D04	D03	D02	D01	D00	D15	D14	D13	D12	D11	D10	D09	D08
0x3c	…	-	-	-	-	-	AFI_2	AFI_1	AFI_0	-	-	RTR	IDE	DLC_3	DLC_2	DLC_1	DLC_0
0x3e	**RX Msg Flags**	-	-	-	-	-	-	-	-	Fifo_Lvl_2	Fifo_Lvl_1	Fifo_Lvl_0	-	-	-	-	MsgAval
0x40	**TX & RX Error Cnt**	rx_er_cnt_7	rx_er_cnt_6	rx_er_cnt_5	rx_er_cnt_4	rx_er_cnt_3	rx_er_cnt_2	rx_er_cnt_1	rx_er_cnt_0	tx_er_cnt_7	tx_er_cnt_6	tx_er_cnt_5	tx_er_cnt_4	tx_er_cnt_3	tx_er_cnt_2	tx_er_cnt_1	tx_er_cnt_0
0x42	**Error Status tatu**	-	-	-	-	-	-	-	-	-	-	-	-	Rxgte96	Txgte96	error_stat_1	error_stat_0
0x44	**TX/ RX Msglevel**	-	-	-	-	-	-	-	-	-	-	-	-	rx_level_1	rx_level_0	tx_level_1	tx_level_0
0x46	**IRQ flags**	rx_msg	tx_msg	tx_xmit2	tx_xmit1	tx_xmit0	bus_off	crc_error	form_error	ack_error	stuff_error	bit_error	rx_ovr	ovr_load	arb_loss	-	-
0x48	**IRQ Enb. Reg.**	rx_msg	tx_msg	tx_xmit2	tx_xmit1	tx_xmit0	bus_off	crc_error	form_error	ack_error	stuff_error	bit_error	rx_ovr	ovr_load	arb_loss	-	int_enable

Off-set (Hex)	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4a	**CAN Mode Mode**	-	-	-	-	-	-	-	-	-	-	-	-	-	Loop_Back	Passive	Run
0x4c	**CAN bit rate Div.**	-	-	-	-	-	cfg_bitrate_10	cfg_bitrate_9	cfg_bitrate_8	cfg_bitrate_7	cfg_bitrate_6	cfg_bitrate_5	cfg_bitrate_4	cfg_bitrate_3	cfg_bitrate_2	cfg_bitrate_1	cfg_bitrate_0
0x4e	**CAN tsegs**	ovr_wrt_msg	cfg_tseg2_2	cfg_tseg2_1	cfg_tseg2_0	cfg_tseg1_3	cfg_tseg1_2	cfg_tseg1_1	cfg_tseg1_0	-	-	-	auto-restart	cfg_sjw_1	cfg_sjw_1	sample_mode	edge_mode
0x50	**AFE Reg.**	-	-	-	-	-	-	-	-	-	-	-	-	-	AFE_2	AFE_1	AFE_0
0x52	**AMR0**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x54	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	IDE	RTR	-
0x56	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x58	**ACR0**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x5a	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	IDE	RTR	-
0x5c	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x5e	**AMR1**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x60	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	IDE	RTR	-
0x62	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x64	**ACR1**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x66	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	IDE	RTR	-
0x68	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x6a	**AMR2**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x6c	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	IDE	RTR	-
0x6e	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56

Off-set (Hex)	Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x70	**ACR2**	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
0x72	…	ID12	ID11	ID10	ID09	ID08	ID07	ID06	ID05	ID04	ID03	ID02	ID01	ID00	IDE	RTR	-
0x74 116d	…	D55	D54	D53	D52	D51	D50	D49	D48	D63	D62	D61	D60	D59	D58	D57	D56
0x76	**ALC Reg**	-	-	-	frame_ref_4	frame_ref_3	frame_ref_2	frame_ref_1	frame_ref_0	-	-	frame_bit_5	frame_bit_4	frame_bit_3	frame_bit_2	frame_bit_1	frame_bit_0
0x78	**ECR**	err_code_2	err_code_1	err_code_0	frame_ref_4	frame_ref_3	frame_ref_2	frame_ref_1	frame_ref_0	TX_Mode	RX_Mode	frame_bit_5	frame_bit_4	frame_bit_3	frame_bit_2	frame_bit_1	frame_bit_0
0x7a	**FRR**	Stuff_Ind	RX_Bit	TX_Bit	frame_ref_4	frame_ref_3	frame_ref_2	frame_ref_1	frame_ref_0	RX_Mode	TX_Mode	frame_bit_5	frame_bit_4	frame_bit_3	frame_bit_2	frame_bit_1	frame_bit_0

8.4 CAN Bus Interface

The DSTni-LX contains two complete CAN controllers, CAN0 and CAN1. Each controller supplies two signal pins, CAN receive (CAN_RX) and CAN transmit (CAN_TX). These signals are routed to interface circuits and a CAN transceiver such as the PCA82C251. From the transceiver, the signals become CAN- and CAN+, which are routed to CAN interface connectors. The CAN transceiver can support DeviceNet or CANopen interface requirements.

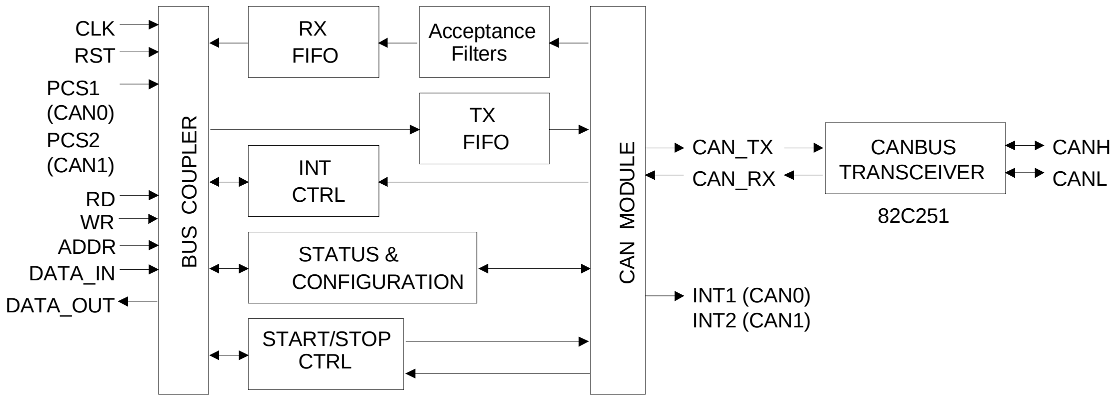


Figure 19 - CAN Bus Interface

8.4.1 Interface Connections

The following sample circuits demonstrate a practical DeviceNet or CANopen interface. The wiring diagram for DeviceNet and CANopen connections are shown below.

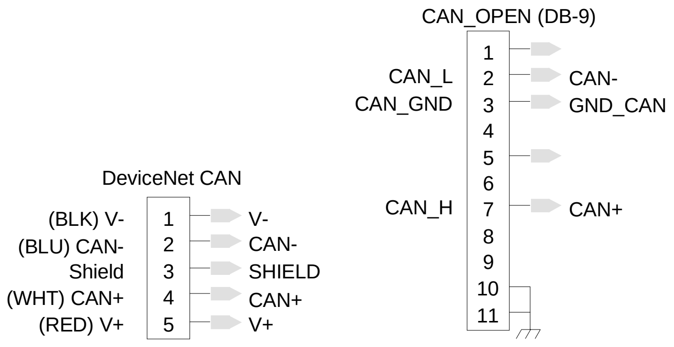
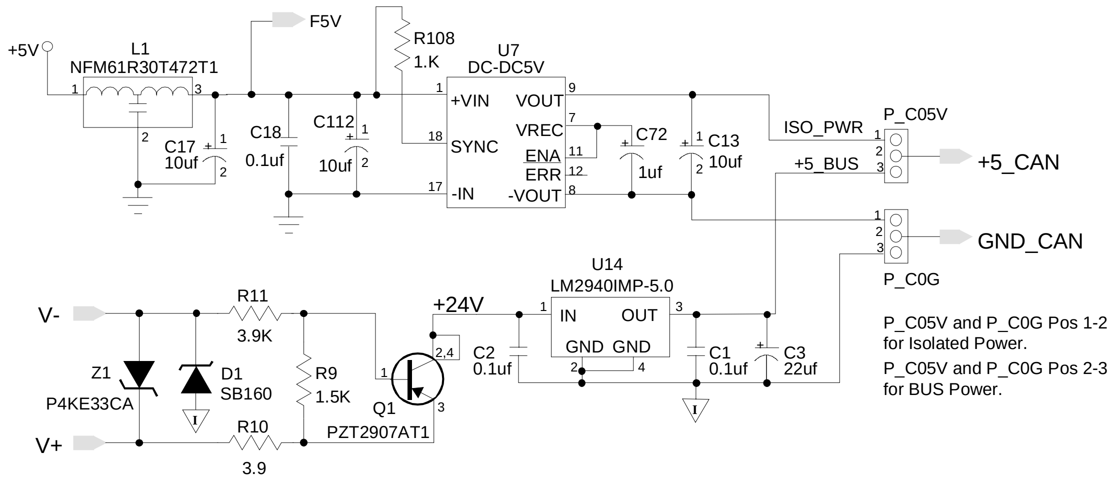


Figure 20 - CAN Connector

DeviceNet can supply network voltage on the V- and V+ pins. This supply can be used to operate the transceiver and interface circuits. In the circuit below, V- and V+ signals are combined to form +24, which is then connected to a regulator to generate the +5_BUS signal for the transceiver circuits.

You can also provide local isolated power for the transceiver circuits, as required when using CANopen. If you are using both DeviceNet and CANopen, use the jumpers to select between bus power (+5_BUS) or isolated power (ISO_PWR). The jumpers P_C05V and P_C0G will then provide +5_CAN and GND_CAN to the transceiver circuits.



Note: Diagrams are for tutorial purposes only and may not reflect the actual circuit on the evaluation module. Always refer to the reference schematic diagrams included with the evaluation module.

Figure 21 - Power for CAN

The transceiver converts CAN- and CAN+ signals to RXD and TXD signals and vice versa. To protect the DSTni-LX from external electrical noise, the CAN interface circuits are isolated. The following circuits show how the RXD and TXD signals from the transceiver are isolated from the CAN_RX and CAN_TX signals of the DSTni-LX.

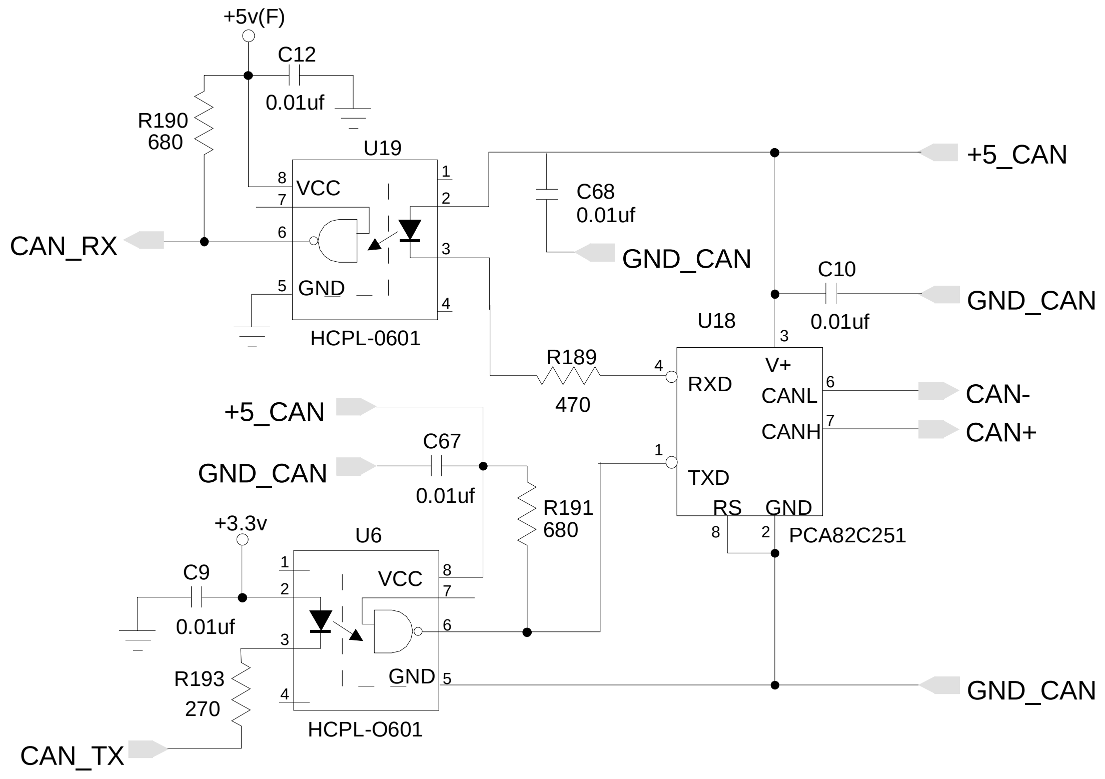


Figure 22 - CAN Transceiver and Isolation Circuits

9 Profibus Controller

Profibus® is the bus system for communication in small cell networks and with field devices. Being based on the European standard EN 50170, Volume 2, Profibus provides the possibility of interfacing standard-conformant components from other manufacturers.

Profibus-FMS (layer 7) is generally used for communication between automation devices and field devices. Profibus-DP is designed for fast, cyclic data exchange with field devices. Profibus fulfils the requirements for simple cabling for electrical and optical data transmission, and allows long distances to be covered.

The Profibus Peripheral Controller emulates the Siemens ASPC2 (Advanced Serial Profibus Controller) controller which can operate as either Master or Slave. This controller has its own DMA controller, and uses the hold/hold acknowledge signals to access the Turbo 186 system bus. These accesses must be shared with the Ethernet Controller.

9.1 Interfacing to the ASPC2

The Profibus software interfaces to the ASPC2 peripheral device using a memory based table called the System Control Block (SCB).   The Profibus software defines this table, and loads a pointer to the table into the SCB BASE HW and SCB BASE LW registers of the ASPC2. The SCB is the beginning node for a number of doubly linked lists that are used for send requests and receive requests. Each node is known as an Application Block (APB). The Profibus software defines and initializes the APB, and the ‘chains in’ the APB to the desired list for the ASPC2 to process. Once the ASPC2 has processed a specific APB, it then unlinks it from its present list and then ‘chains in’ the APB to one of the CON/IND lists. The ASPC2 also creates an appropriate interrupt when the APB service has been completed.

With the SCB in a common memory shared by two independent processes, it is necessary that the two processes do not modify the SCB, or any of the Profibus data areas at the same time. Such an occurrence would most likely cause one or both of the processes to fail. As an example, when chaining in a new request, the Profibus software must not be interrupted, and the ASPC2 must not be permitted to access the SCB to insure that the doubly linked pointer integrity is maintained.

In the discrete implementation that uses the 80186 processor, RAM, and the ASPC2 chip, the Hold request from the ASPC2 drives a small amount of additional logic that holds off the ASPC2 during the times that the processor is updating the SCB and other related data structures.   The output of this logic drives the Hold input of the 80186 processor. When the 80186 processor honors the Hold request by asserting the Hold Acknowledge, the processor ceases to execute instructions, and is essentially stalled until the ASPC2 finishes its use of the RAM, and the ASPC2 releases its Hold request. Because the processor has been stalled by the Hold request, it cannot access either the RAM or the ASPC2 registers. This does satisfy the above requirements.

With the DSTni-LX, this approach would also stop any other processes that were also present, such as Ethernet, CAN, a serial protocol, etc. This was deemed to be an inappropriate solution if one wants to maintain the high performance and throughput that the DSTni-LX is capable of achieving.

In order to resolve this resource issue, a hardware semaphore has been designed into the DSTni-LX. This semaphore accepts requests from both the 80186 processor and the ASPC2 to access the SCB and its associated data areas. Requests are granted when the semaphore is not presently assigned. If the semaphore has been assigned, the grant is delayed until the other user has cancelled its request. The hardware design insures that simultaneous requests will always result in the semaphore being assigned to only one of the requestors.

The ASPC2 requests the semaphore by asserting its Hold request signal. The output of the semaphore drives the Hold acknowledge input signal to the ASPC2 to indicate that the semaphore has been assigned to the ASPC2.

The Profibus software requests the semaphore by setting Bit 4 of the Profibus Extended Control (PEC) Register. The software must then read the PEC register, and check Bit 5 for a logic 1, indicating that the semaphore has been assigned to the Profibus Software. Once Bit 5 is a logic 1, the Profibus software may safely access the SCB and any of its associated data areas. Once the software has completed its accesses, then the semaphore is released by clearing Bit 4 of the PEC. It is very important that the SCB and its data areas be accessed quickly, to avoid holding off the ASPC2 for too great a time period (20usec or less).

Because the DMA and the ASPC2 registers share an internal data bus, it is also necessary to obtain the semaphore before accessing the ASPC2 registers. This insures that the DMA and any register accesses cannot occur simultaneously. Such an occurrence would definitely cause a system failure.

While the addition of the hardware semaphore does place additional requirements on the Profibus software, it does permit the DSTni-LX communication chip to also provide high performance solutions for CAN, Ethernet, and serial protocol designs at the same time.

9.2 Profibus Operation

The Profibus controller appears to the microprocessor as a memory-mapped I/O device.

Each peripheral has 256 bytes of I/O address space allocated to it. The starting I/O address for each peripheral depends on the Peripheral Chip Select (PCS) line assigned to it. PCS3 is assigned to the Profibus Controller. The Base I/O address is determined by the Peripheral Base Address (PBA) and the offset. The PBA is determined via the PACS register.

Table 37 - Profibus I/O Address

Select Line	Assigned To	Base I/O Address
PCS3	Profibus	PBA + 0300h

The Peripheral Chip Selects are programmed through two registers—the Peripheral Address Chip Select (PACS) register and the PCS and MCS Auxiliary (MPCS) register. The Peripheral Address Chip Select (PACS) register determines the base address, the ready condition, and the wait states for the PCS3–PCS0 outputs.

The MPCS register determines whether PCS chip selects are active during memory or I/O bus cycles and specifies the ready and wait states for the PCS6–PCS5 outputs. The ProfiBus device must be mapped into the I/O space.

The PCS pins are not active on reset. The PCS pins are activated as chip selects by writing to the PACS and MPCS registers.

Peripheral Address Chip Select Register, PACS (20-bit mode)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A19	A18	A17	A16	A15	A14	A13	A12	A11	1	1	1	1	R2	R1	R0
0	0	0	0												

A19:11	Upper address bits of the starting address of the Peripheral Base Address (PBA). Allows programmer to set the PBA at any 2K byte address boundary. Since we're operating only in I/O space which is limited to 64 Kbytes, A19-A16 must always be zero.
R2:0	Ready generation select. See Table 5 - MMCS Wait State and Ready Select.

Peripheral Address Chip Select Register, PACS (24-bit mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	R2	R1	R0
0	0	0	0	0	0	0	0								

A23:11	Upper address bits of the starting address of the Peripheral Base Address (PBA). Allows programmer to set the PBA at any 2K byte address boundary. Since we're operating only in I/O space which is limited to 64 Kbytes, A23-A16 will always be zero.
R2:0	Ready generation select. See Table 5 - MMCS Wait State and Ready Select.

9.2.1 Profibus Interrupt

The Profibus controller uses the following interrupt.

Table 38 - Profibus Interrupt

Interrupt Signal	Assignment	Vector Type	Vector Address	Default Priority
INT3	Profibus	15	3Ch	9

9.2.2 Profibus Interrupt Control Register

This is the control register for the external interrupt source. The control register contains the level-trigger mode bit, mask bit and programmable priority level. The mask bits in this control register are the same mask bits as in the mask register. Modifying them in the control register will also modify the corresponding mask bits in the mask register, and vice versa. This register is read/write

Register 94 - External Interrupt Control Register, I3CON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	LTM	MSK	PRI2	PRI1	PRI0

LTM	The level-trigger mode bit will set the respective interrupt source for either level triggered mode (LTM=1), or edge triggered mode (LTM=0). When set for level-triggered mode an interrupt is generated whenever the external interrupt signal is high. In edge triggered mode an interrupt request is generated when the external interrupt signal makes a low to high transition. In both cases, the level must remain high until the interrupt is acknowledged.
MSK	Setting this bit is will mask the respective interrupt request. Resetting this bit will enable the respective interrupt.
PR2:0	These bits represent the programmable priority level for the respective interrupt source. Bit combination 000\b has the highest priority while 111\b has the lowest priority.

Name	**Mnemonic**	**Offset**
End of Interrupt	EOI	22h
Poll	POLL	24h
Poll Status	POLLST	26h
Interrupt Mask	IMASK	28h
Priority Mask	PRIMSK	2Ah
In-Service	INSERV	2Ch
Interrupt Request	REQST	2Eh
Interrupt Status	INTSTS	30h
INT 3 Control	**I3CON**	**3Eh**

9.2.3 In-Service Register (INSERV, offset 2Ch)

This register contains the in-service bits for each of the interrupt sources. An in-service bit is set to indicate that a source's service routine is being executed. When an in-service bit is set the interrupt controller will not generate interrupts from sources with a lower priority than the current interrupt being serviced. This allows interrupt service routines to run with interrupts enabled. This register is read/write.

Register 95 - In-Service Register for Profibus

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	I2	I1	**I3**	D1	D0	0	TMR

SPI	In-service bit for the Serial Peripheral Interface
D3:0	In-service bits for the DMA channels (DMA3:0).
I5:0	**In-service bits for the external interrupt sources (INT5:0).**
SP1:0	In-service bits for the asynchronous serial ports (SP1:0).
TMR	In-service bit for all three timers (TMR2:0).

9.2.4 Interrupt Request Register (REQST, offset 2Eh)

This register contains the interrupt request bits for the Timers, DMA channels and external interrupt sources. This register is read/write except were noted.

Register 96 - Interrupt Request Register for Profibus

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SPI	D3	D2	I5	SP0	SP1	I4	I3	I2	I1	**I3**	D1	D0	0	TMR

I5:0	Interrupt request bits for the external interrupt sources (INT5:0). These bits represent the actual state of the external interrupt pins, after they have been passed through a two level synchronizer, or a level detector depending on the LTM bit for that respective interrupt line. These bits will be set whenever an external interrupt line make a low to high transition if LTM=0. Otherwise, if LTM=1 these bits will follow the input pins after a two clock delay. Because these bits represent the state of the external interrupt pins they can not be written.

9.2.5 Profibus Extended Control (PEC, offset 8Eh)

This register provides extended features to the standard Profibus controller. The Profibus controller's DMA channels only supports 20-bits of addressing. When the DSTni-LX is operating in 24-bit mode, the Profibus DMA controller may need to address memory beyond the first megabyte. This register allows software to give the DMA controller access to the full 24-bit address range.

The register defaults to 0 upon chip reset or power-up (Page 0 and Profibus held in reset).

Register 97 - Profibus Extended Control, PEC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P3	P2	P1	P0	0	0	0	0	0	0	SEMGR	SEMRQ	0	0	DPV2	RST

P3:P0	Profibus DMA Page select. Selects the A23-A20-bits of the 24-bit address range for the Profibus DMA controller. This allows the Profibus DMA controller address base to be set for any one of 16 1 Megabyte pages of RAM. The Profibus DMA controller will provide the lower 20-bits of address.
SEMRQ	Set to 1 by software to request access to Profibus registers and shared memory regions. This initiates a request for control over a hardware semaphore between the CPU and the Profibus controller. After setting this bit, the CPU must wait for SEMGR to be 1 before accessing Profibus registers or memory regions shared between the CPU and the Profibus controller. Once the CPU is finished with its access, it must release the semaphore by clearing SEMRQ. **Note:** The CPU must not hold this semaphore for more than 20 microseconds. Holding the semaphore longer than this will impact the operation of the Profibus controller.
SEMGR	This read-only bit must be polled by the CPU after SEMRQ is set and before accesses may be made to Profibus registers or memory shared between the CPU and the Profibus controller. Once the CPU reads a "1" in this bit position, it may proceed with its Profibus accesses.
Bit 3-2	Reserved, Set to 0.

DPV2 This bit is used to control the multiplexing of 4 I/O pins on the DSTni-LX. When this bit is cleared (0), the four pins are connected to serial port 1. When this bit is set, the 4 pins are connected to the ASPC2 signals required for DPV2 mode of operation. The following table shows the affected signals.

Table 39 - DPV2 Mode pin assignments

DPV2 = 0	DPV2 = 1
SP1_RXD	ASPC2 Diag2
SP1_TXD	ASPC2 Diag3
SP1_RTS_N	ASPC2 Diag4
SP1_CTS_N	ASPC2_CTS_N

RST Profibus Reset: 0 = Profibus is held in reset, 1 = Profibus enabled. After chip reset or power-up the Profibus is held is reset until software writes a 1 to this bit.

9.2.6 ASPC2 I/O Register Map

Read		Write		Reset (Hex)	Addr (Hex)
High Byte	**Low Byte**	**High Byte**	**Low Byte**		
TTHOLD$_{15..8}$	TTHOLD$_{7..0}$	TTR$_{15..8}$	TTR$_{7..0}$	0x0000	0x300
Delay Timer$_{15..8}$	Delay Timer$_{7..0}$	INT-MASK_REG$_{15..8}$	INT-MASK_REG$_{7..0}$	0xFFFF	0x302
INT-REQ-REG$_{15..8}$	INT-REQ-REG$_{7..0}$	INT-REQ-REG$_{15..8}$	INT-REQ-REG$_{7..0}$	0x0000	0x304
INT-REG$_{15..8}$	INT-REG$_{7..0}$	INT-ACK-REG$_{15..8}$	INT-ACK-REG$_{7..0}$	------	0x306
Status-REG-L$_{15..8}$	Status-REG-L$_{7..0}$	Mode-REG0$_{15..8}$	Mode-REG0$_{7..0}$	0x0000	0x308
Status-REG-H$_{31..24}$	Status-REG-H$_{23..16}$	Mode-REG1-Set$_{15..8}$	Mode-REG1-Set$_{7..0}$	0x0000	0x30A
-----------	LAS-REG$_{3..0}$	Mode-REG1-Res$_{15..8}$	Mode-REG1-Res$_{7..0}$	------	0x30C
		SCB-BASE-LW$_{15..8}$	SCB-BASE-LW$_{7..0}$	0x0000	0x30E
		SCB-BASE-HW$_{31..24}$	SCB-BASE-HW$_{23..16}$	0x0000	0x310
		TSLOT-REG$_{13..8}$	TSLOT-REG$_{7..0}$	0x0000	0x312
		TID1-REG$_{9..8}$	TID1-REG$_{7..0}$	0x0000	0x314
		TID2-REG$_{9..8}$	TID2-REG$_{7..0}$	0x0000	0x316
		TRDY-REG$_{9..8}$	TRDY-REG$_{7..0}$	0x000B	0x318
		BR-REG$_{10..8}$	BR-REG$_{7..0}$	0x0000	0x31A
		SAP-MAX$_{7..0}$	TS-ADDR-REG$_{6..0}$	0x0000	0x31C
		Token-Err$_{7..0}$	GUD-REG$_{7..0}$	0x0000	0x31E
		TQUI-REG$_{7..0}$	LAY4-Hlen-REG$_{7..0}$	0x0000	0x320
		Resp-Err$_{3..0}$	HSA$_{6..0}$	0x0000	0x322
		Monitor-Select2$_{7..0}$	Monitor-Select1$_{7..0}$	0x0001	0x324
		Mode-REG2$_{5..0}$	Retry Tok$_{3..0}$ Retry Msg$_{3..0}$	0x0000	0x326
		Wait-States$_{15..8}$	Wait-States$_{7..0}$	0x0000	0x328
Tdiff-DP-Reg$_{15..8}$	Tdiff-DP-Reg$_{7..0}$	TDP-REG$_{15..8}$	TDP-REG$_{7..0}$	0x0000	0x32A
			TDP-REG$_{23..16}$	0x0000	0x32C
		Tpsp-Reg$_{9..8}$	Tpsp-Reg$_{7..0}$	0x0000	0x32E
		Buff-Ex_Off$_{15..8}$	Buff-Ex_Off$_{7..0}$	0x0000	0x330
		Trace-Adr$_{10..8}$	Trace-Adr$_{7..0}$	0x07FF	0x332
			Int-Mask-REG$_{23..16}$	0x00FF	0x334
	Int-Req-REG$_{23..16}$		Int-Req-REG$_{23..16}$	0x0000	0x336
	Int-REG$_{23..16}$	Mode-REG4-SET$_{5..0}$	Int-Ack-REG$_{23..16}$	0x0000	0x338
		Mode-REG4-RES$_{5..0}$	Mode-REG3$_{7..0}$	0x0000	0x33A
		SM-Time-Control$_{7..0}$	Isochron-MAS-Adr$_{6..0}$	0xFF7F	0x33C
		Sync-Clock-Cycle$_{3..0}$	Isochron-Group-Ident$_{7..0}$	0x0100	0x33E

The system control block SCB is the interface between the ASPC2 and the controller.

The following lists are chained in the SCB:

- Background Message High List (BMH)
- Background Message Low List (BML)
- Confirmation Indication High List (CON-IND-HIGH) or separate Indication High List (IND-SEP-HIGH)
- Confirmation Indication Low List (CON-IND-LOW) or separate Indication Low List (IND-SEP-LOW)
- Separate Confirmation High List (CON-SEP-HIGH)
- Separate Confirmation Low List (CON-SEP-LOW)
- NOT-OKAY High List (NOT-OK-HIGH)
- NOT-OKAY Low List (NOT-OK-LOW)
- DEFAULT SAP List, SAP Lists (SAP[0..63])

9.2.7 ASPC2 SCB Memory Map

SCB Base Offset	Function	Data Type
0x00	Special Function	next-blk-ptr prev-blk-ptr
0x04	Cyclic Service	next-blk-ptr prev-blk-ptr
0x08	BM-HIGH	next-blk-ptr prev-blk-ptr
0x0C	BM-LOW	next-blk-ptr prev-blk-ptr
0x10	CON-IND-HIGH (IND-SEP-HIGH)	next-blk-ptr prev-blk-ptr
0x14	CON-IND-LOW (IND-SEP-LOW)	next-blk-ptr prev-blk-ptr
0x18	(CON-SEP-HIGH)	next-blk-ptr prev-blk-ptr
0x1C	(CON-SEP-LOW)	next-blk-ptr prev-blk-ptr
0x20	(NOT-OK-HIGH)	next-blk-ptr prev-blk-ptr
0x24	(NOT-OK-LOW)	next-blk-ptr prev-blk-ptr
0x28	Master Watchdog	
0x7C	DEFAULT-SAP	req-next-blk-ptr req-prev-blk-ptr res-next-blk-ptr res-next-blk-ptr req-buf-length req-sa req-all req-fc
0x88	SAP 0-63	req-next-blk-ptr req-prev-blk-ptr res-next-blk-ptr res-next-blk-ptr req-buf-length req-all req-ssap req-fc

9.3 Profibus Interface

The DSTni-LX-002 Controller provides the PROFI_RX, PROFI_TX and PROFI_ENB signals for Profibus control. These signals are tied to interface circuits to and from a Profibus transceiver, to control the flow of information on the bus. The following circuits are typical interface connections for Profibus communications.

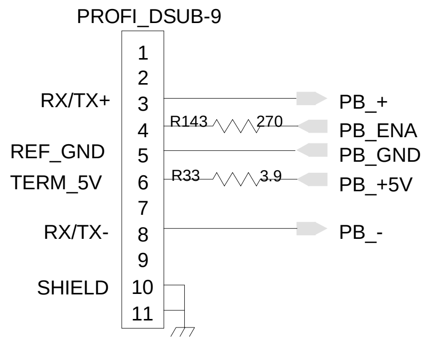


Figure 23 - Profibus 9-Pin DSUB Connector

The following sample Profibus interface circuits provide isolation and signal translation.

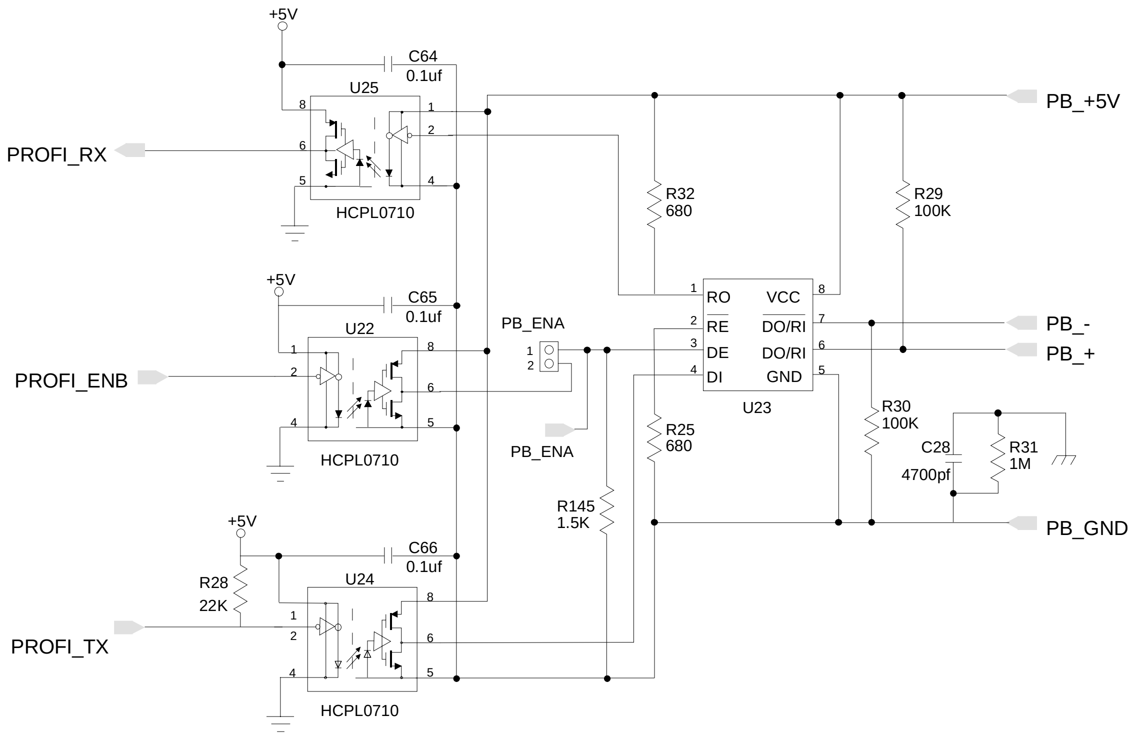


Figure 24 - Profibus Interface Circuit Diagram

9.3.1 Profibus DPV2 Diagnostic Port

These 4 pins are connected to the ASPC2 signals required for DPV2 mode of operation. The following drawing show a typical DPV2 circuit and connector.

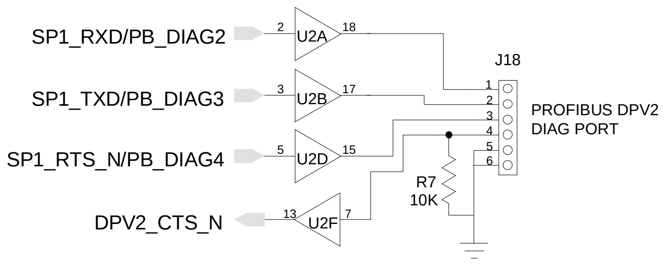


Figure 25 - Profibus DPV2 Port Diagram

10 In-Circuit Emulator Support

10.1 JTAG Interface

The JTAG interface is an integrated JTAG Debugger/In-Circuit Emulator (JTAG DICE) for the DSTni-LX. The interface is a slave controller normally connected to a JTAG debugger. The architecture contains logic for four hardware breakpoints and a 256 word trace buffer. The logic provides for stop, single-step, read/write of memory or I/O, and interrogation of the state of the CPU.

The JTAG interface (IEEE 1149.1) is used by the In-Circuit Emulator to access the hardware registers that support the breakpoint and trace functions. The JTAG interface consists of 4 signals, JTAG Clock (TCK), JTAG Mode (TMS), JTAG Data In (TDI), and JTAG Data Out (TDO).

First Silicon Solutions has developed a System Analyzer, designed to support the special features and integrated peripherals of the DSTni-LX Turbo 186 core. Special “silicon hooks”are integrated into the Turbo 186 core. These On-Chip Instrumentation (OCI™) extensions allow First Silicon Solutions (FS2) to provide a powerful debug tool with advanced features.

The VSA-8X/18X debugger is contained in a compact chassis that connects to the target system using a standard 20 pin AMP debug connector. It requires access to only 4 pins in the core processor. The system runs on a Windows ® 95/98/NT/2000 PC over an IEEE-1284 EPP/ECP high speed parallel port. A graphical, source level debugger program provides an intuitive, easy to use interface.

Turbo 186 core extensions contain 4 event recognizers that can generate triggers to control breakpoints and trace collection. The most basic use of recognizers is for hardware execution breakpoints. Unlike software breakpoints, hardware execution breakpoints can be set in ROM. Like their software counterparts, they stop program execution just prior to an instruction being executed.

11 External Interfaces

11.1 System Bus

The Turbo 186 system bus is available on the external pins. This provides for additional asynchronous memory devices, such as Flash Memory devices or Static RAM Memory devices when a particular application requires more resources than are internal to the chip.

The system bus provides 23 un-multiplexed address lines, ADDR[22:00]. Address Line ADDR[23] is not available as a pin, but is used in the chip select logic that creates the MCS0_n and UMCS_n chip select signals.

There are 16 bi-directional data lines DATA[15:00]. If an 8 bit peripheral is to be added, two external transceivers will be required to perform the 'byte swap' function to create an 8 bit data bus for the peripheral.

The control signals consist of OE_n, WRL_n, WRH_n, and READY_n, and the chip select signals, MCS0_n and UMCS_n. This group of signals will support most types of memories or peripherals that may be added externally.

This bus runs at 20ns with a 48 MHz clock and is a single clock cycle bus. For 0 wait state operation, devices must have less than 20ns access time.

11.2 Interface to Dual Port Memory

The Dual Port RAM memory (DPM) is an 8K x 8 bit device. The 13 address lines are DPA12:DPA00. The Data lines are DPD7:DPD0. The control lines are DPCS_N, DPSS_N, DPWR_N, DPOE_N, DPINT_N, and DPBUSY_N. These signals are shared with PIO, the Parallel I/O pins.

The DPM is typically connected to the PC/ISA bus or the bus of some embedded processor. Creating the DPCS_N signal is done with external hardware typically consisting of either jumper posts or a dip switch and a binary comparator. An external transceiver is needed on the data lines because the PC/ISA drive requirement is 24 ma, and the PC/104 drive requirement is 8 ma, both which are higher than the chip can support.

The DPINT_N is the signal from the DPM that the left side has written to location 1FFFh. The PC or the Embedded Processor may use this signal as an interrupt or as a status bit.

The DPBUSY_N signal is used as a ready input to the PC or the embedded processor. This signal is asserted when both the Turbo 186 and the PC or embedded processor access the same location in the DPM at the same time. When this signal is asserted, the left side of the DPM started its access first, and will complete before the right side. If the right side does not wait for the DPBUSY_N signal to deassert, the data that is read, or attempts to write will be invalid.

All of the external Dual Port RAM signals are first synchronized to the 48Mhz DSTni-LX clock. Thus, all of the signals must have a pulse width of more than 20ns or improper DPRAM operation will occur.

See Dual Port Memory on page 75.

11.3 Interface to Serial Ports

The DSTni-LX provides two asynchronous serial ports. These ports provide a read/write port for full duplex operation with programmable baud rates. Both of the serial channels support RTS/CTS control signals and DMA control. An additional register has been added to support high-speed protocols and signaling for RS-422/485.

11.3.1 EIA/TIA-232

The serial port interface signals can be interfaced with an EIA/TIA-232 Transceiver, similar to the MAX3232 3.3V Transceiver. The MAX3232 has 2 receivers and 2 drivers. The MAX3232 is pin, package, and functionally compatible with the industry-standard MAX242 and MAX232, respectively.

The following circuit diagram illustrates a typical EIA/TIA-232 circuit, including status LED's.

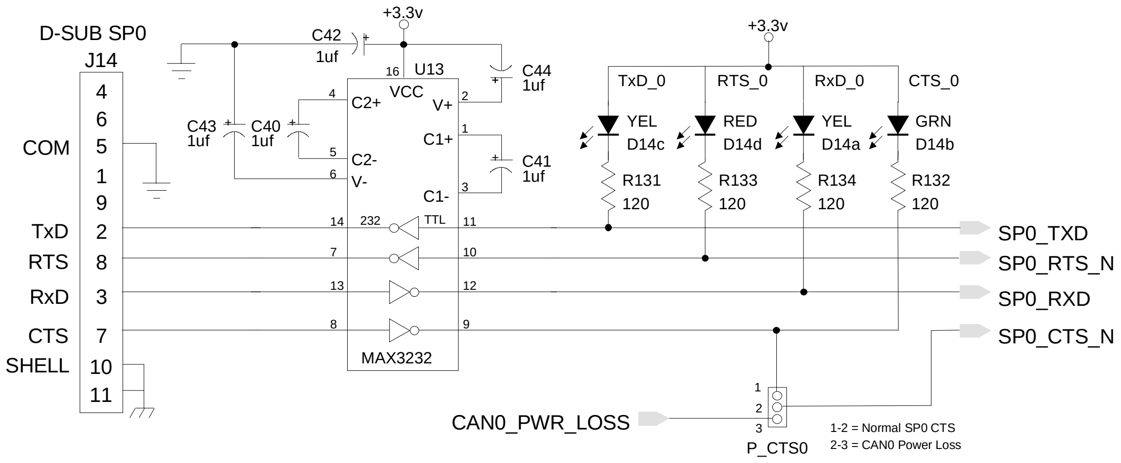


Figure 26 - EIA/TIA-562 Circuit

11.3.2 EIA/TIA-485

The following circuit diagram illustrates a typical EIA/TIA-485 interface.

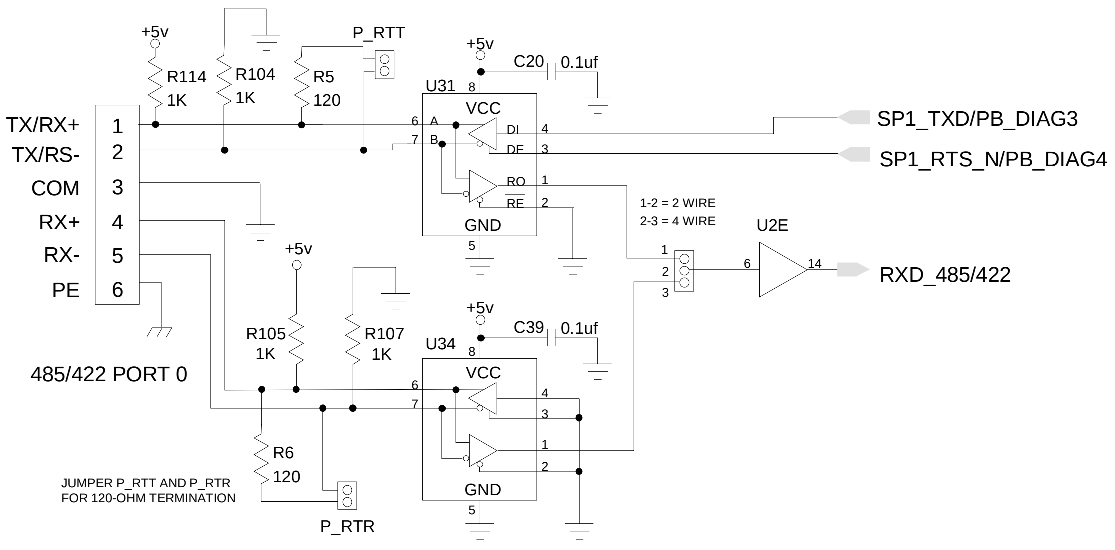


Figure 27 - EIA/TIA-485 Circuit

12 Electrical Specifications

12.1 Operating Conditions

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD} (VCC2)	DC Supply Voltage	Core and Standard I/Os	2.25	2.5	2.75	V
V_{DD3} (VCC)	DC Supply Voltage	3V Interface I/Os	3	3.3	3.6	V
V_I	DC Input Voltage		0		V_{DD3}	V
V_O	DC Output Voltage		0		V_{DD3}	V
V_{I5}	DC Input Voltage	5V Tolerant I/O	0		V_{DD3} + 2.0V	V
V_{O5}	DC Output Voltage	5V Tolerant I/O	0		V_{DD3} + 2.0V	V

12.2 Absolute Maximum Ratings

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_A	Operating Temperature	Industrial	-40		+85	°C
T_{ST}	Storage Temperature		-65		+150	°C

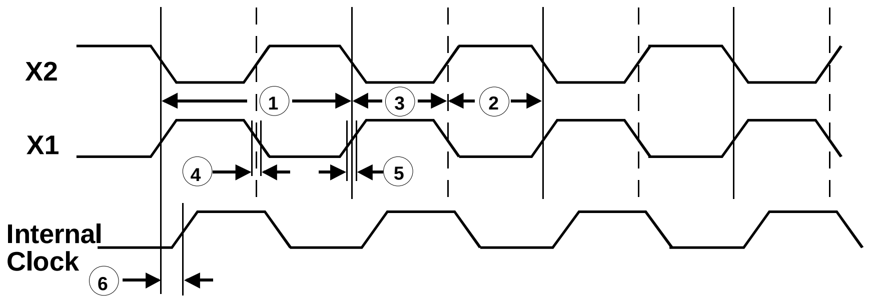
12.3 DC Characteristics

Symbol	Parameter		Min	Max	Unit
V_{IH}	High-level input voltage		2.0	V_{cc}	V
V_{IL}	Low-level input voltage		0	0.8	
V_{OH}	High-level output voltage (I_{OH} = -2.5 mA)	Commercial	2.4		
V_{OL}	Low-level output voltage (I_{OL} = +3.0 mA)			0.4	
V_{OH}	High-level output voltage (I_{OH} = -2.0 mA)	Industrial	2.4		
V_{OL}	Low-level output voltage (I_{OL} = +3.0 mA)			0.4	
$I_{CC(Core)}$	Supply current, 2.5V (Average)			95	mA
$I_{CC\ (I/O)}$	Supply current, 3.3V (Average)			10	mA
I_L	Input or output leakage current ($V_{IN} = V_{CC}$ or GND)		-10	10	uA

12.4 Switching Parameters and Waveforms

12.4.1 Clock Switching Parameters

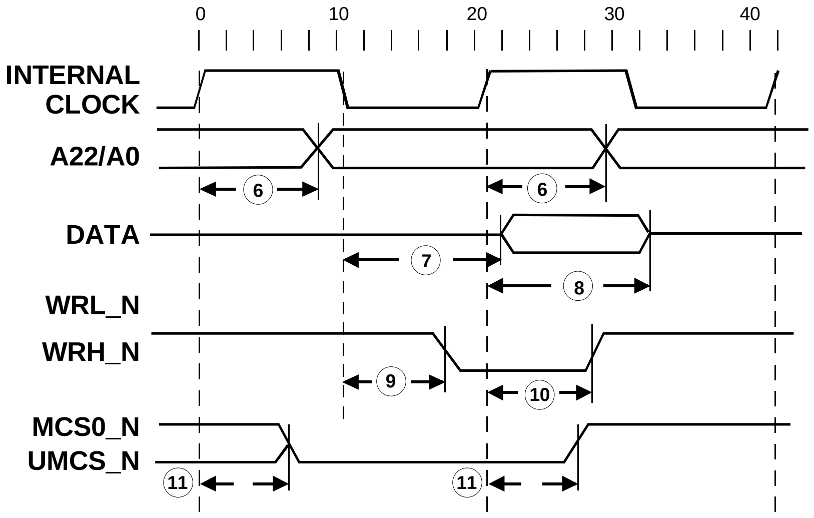
No.	Symbol	Description	Min	Max	Unit
1	t_{CKIN}	X1 Period	20.8		ns
2	t_{CLCK}	X1 Low Time	8		ns
3	t_{CHCK}	X1 High Time	8		ns
4	t_{CKHL}	X1 Fall Time		5	ns
5	t_{CKLH}	X1 Rise Time		5	ns
6	t_{CHICH}	Internal Clock Delay	.3	2	ns



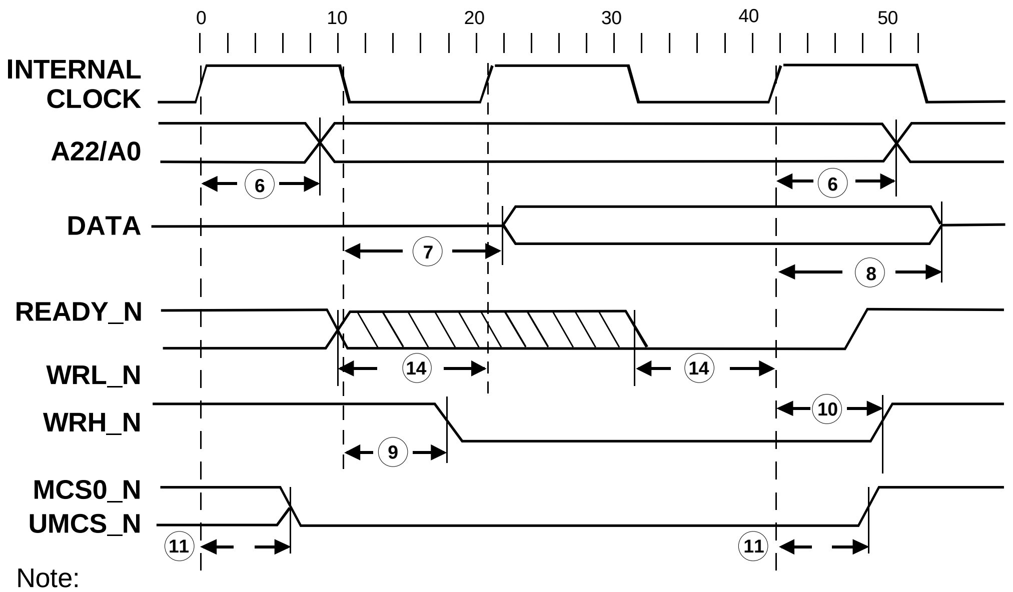
12.4.2 Memory Write Parameters

No.	Symbol	Description	Min	Max	Unit
6	tCHAV	Internal Clock high to Address Available	0	8.23	ns
7	tCLDV	Internal Clock Low to Data Available		11.13	ns
8	tCLDOX	Data Hold Time	0	11.13	ns
9	tCLCTV	Internal Clock Low to WR Low	0	7.12	ns
10	tCHCTX	Internal Clock High to WR High	0	7.12	ns
11	tCHCSV	Internal Clock High to MCS0_N Low	0	6.48	ns
14	TRDYSETUP	Ready_N Setup Time	10		ns

Memory Write



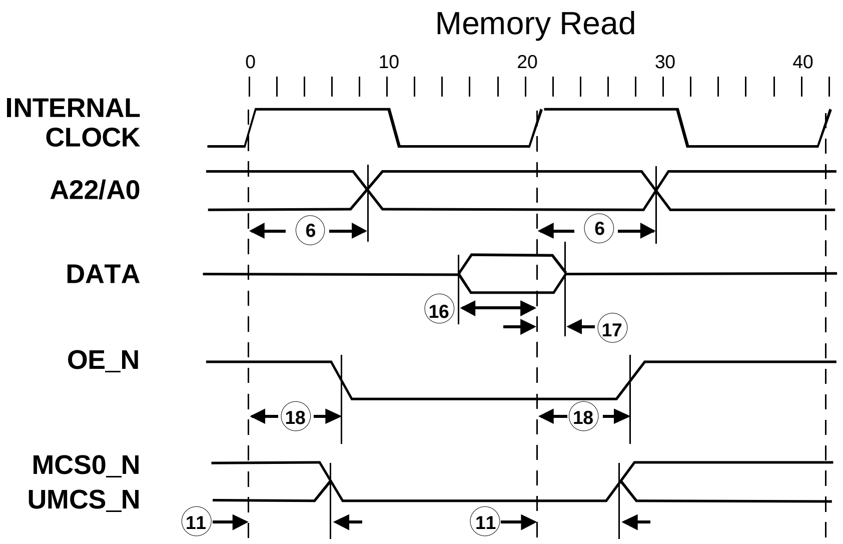
Memory Write 1 Wait State from External Ready



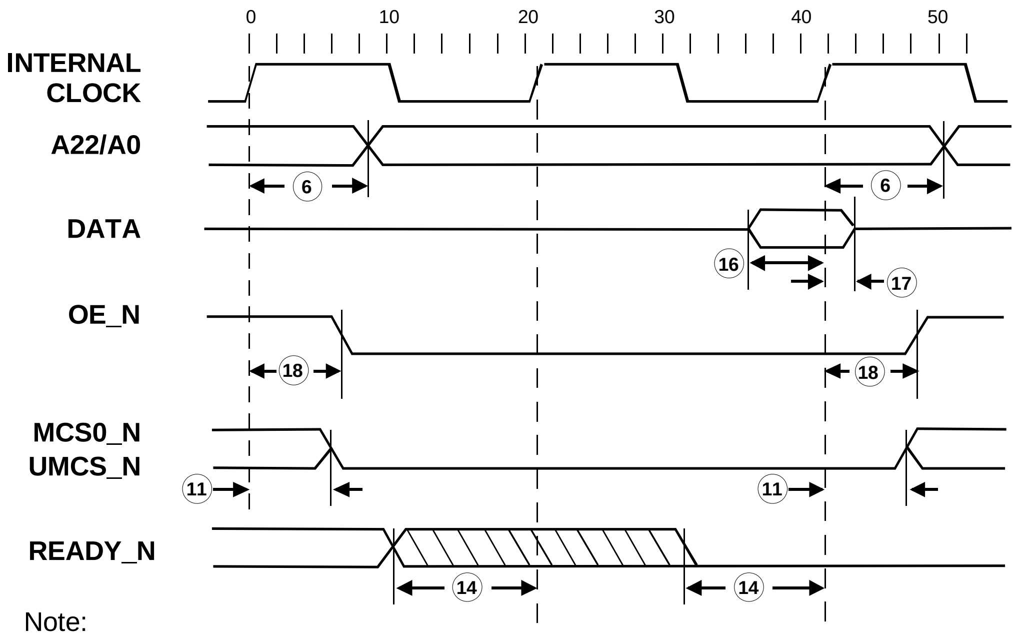
Note:
READY_N is a synchronous input and must be sync'd to the rising edge of X1 clock.

12.4.3 Memory Read Parameters

No.	Symbol	Description	Min	Max	Unit
6	t_{CHAV}	Internal Clock high to Address Available	0	8.23	ns
16	t_{DVCH}	Data Setup	1.8	5.99	ns
17	t_{CHDX}	Data Hold	0		ns
18	t_{CHRH}	OE_N Inactive Delay	0	6.59	ns
11	t_{CHCSV}	Internal Clock High to MCS0_N Low	0	6.48	ns
14	$T_{RDYSETUP}$	Ready_N Setup Time	10		ns



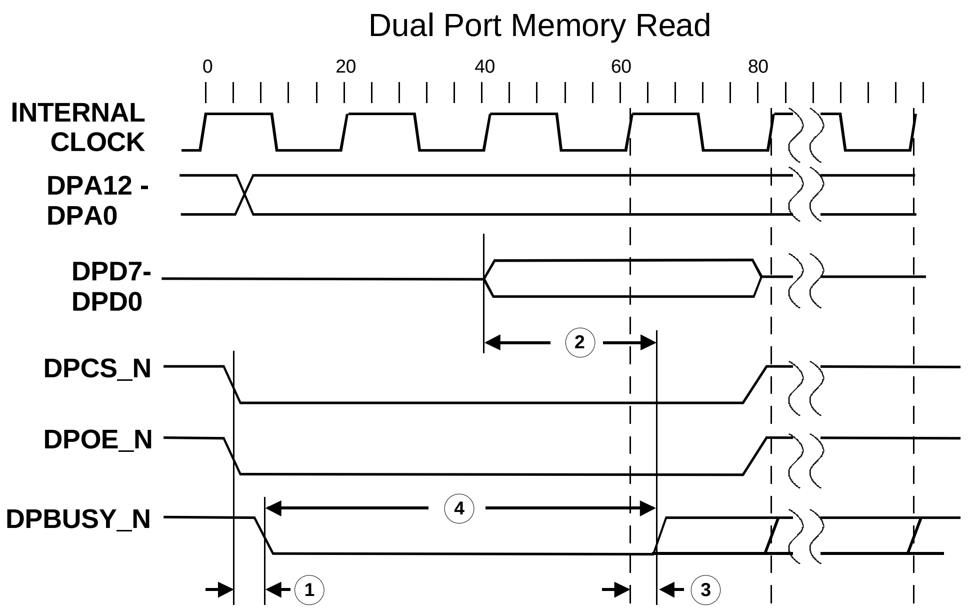
Memory Read with 1 Wait State from External Ready



Note:
READY_N is a synchronous input and must be sync'd to the rising edge of X1 clock.

12.4.4 Dual Port Memory Read Parameters

No.	Description	Min	Max	Unit
1	DPCS_N, DPOE_N low to DPBUSY_N low	6		ns
2	Read Data available to DPBUSY_N high	12		ns
3	Internal Clock high to DPBUSY_N high	6		ns
4	DPBUSY_N low width	0 to 1 Tckck + (2 to 6)*Tckck		

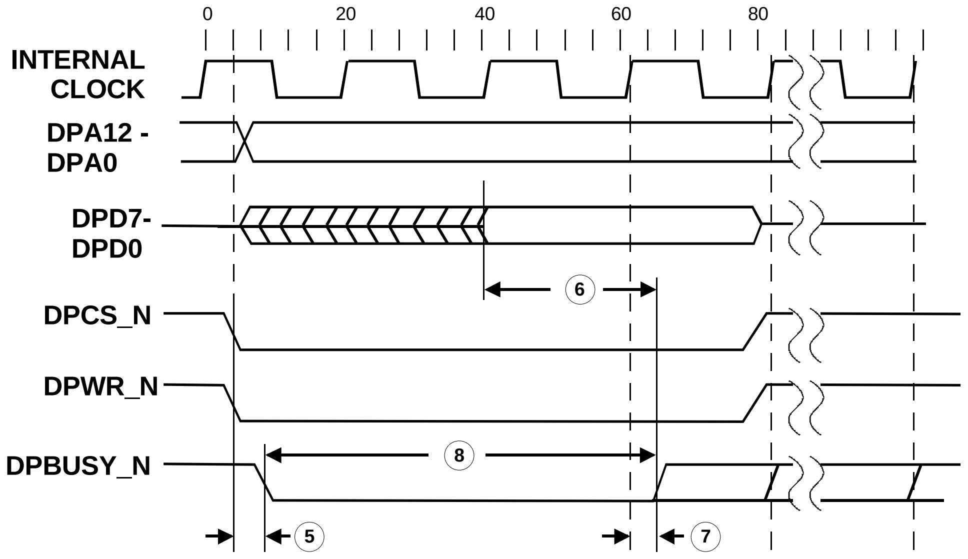


Note: Both DPCS_N and DPOE_N must be low to start a DPM cycle.

12.4.5 Dual Port Memory Write Parameters

No.	Description	Min	Max	Unit
5	DPCS_N, DPWR_N low to DPBUSY_N low	6		ns
6	Wrtie Data setup time to DPBUSY_N high	26		ns
7	Internal Clock high to DPBUSY_N high	6		ns
8	DPBUSY_N low width	0 to 1 Tckck + (2 to 6)*Tckck		

Dual Port Memory Write



Note: Both DPCS_N and DPWR_N must be low to start a DPM cycle.

13 Package Description

160-Pin Package Outline, LQFP, 24x24mm, 1.40mm thick.
JEDEC Registration MS-026, Variation BGA
*160 Lead compliant depopulation of the 176-pin package.

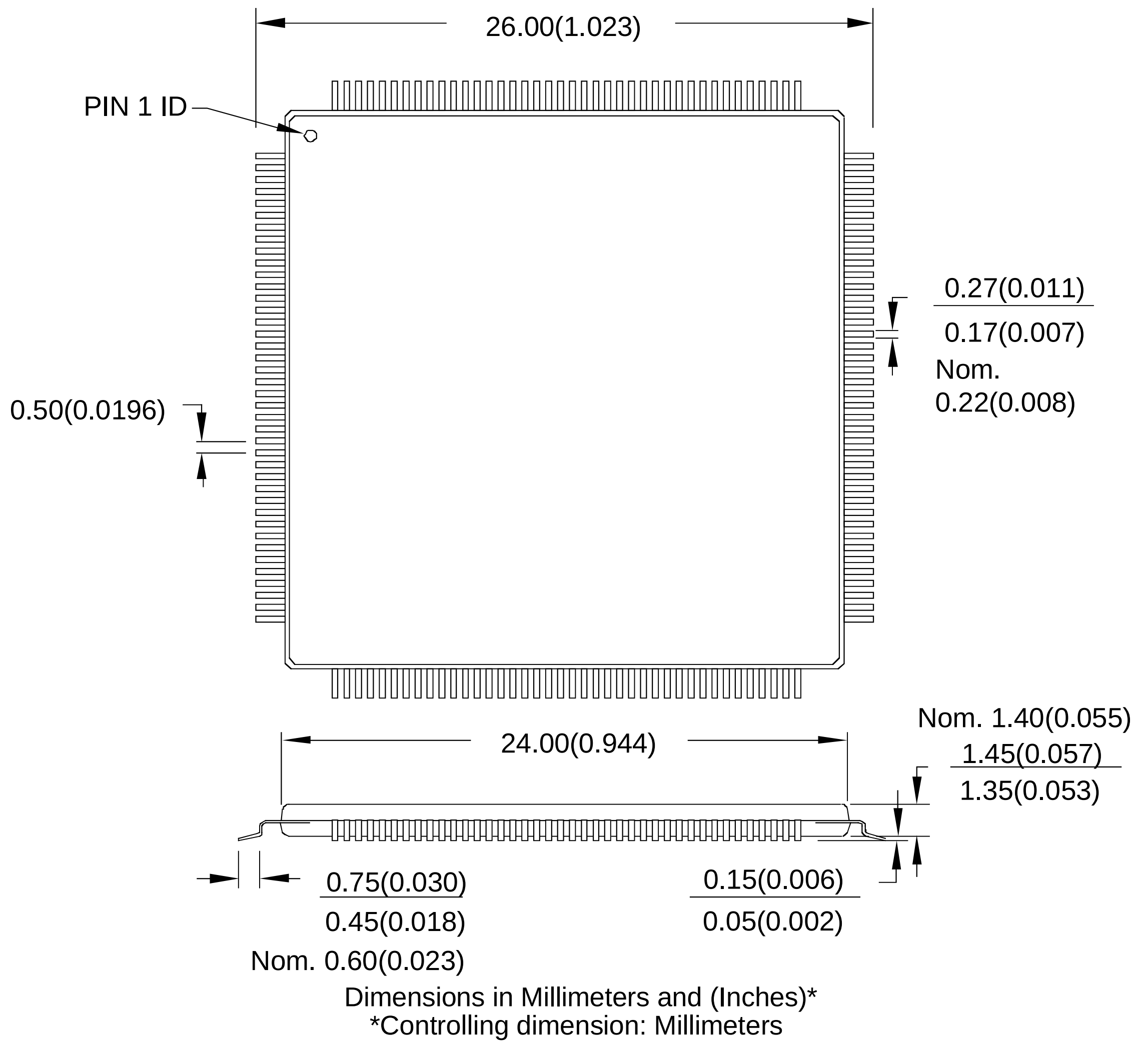
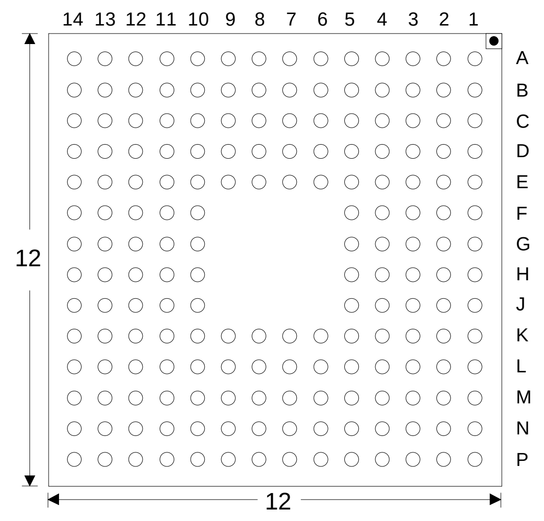
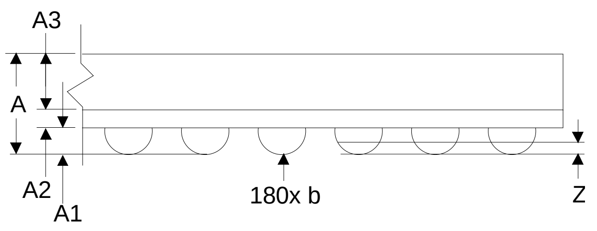


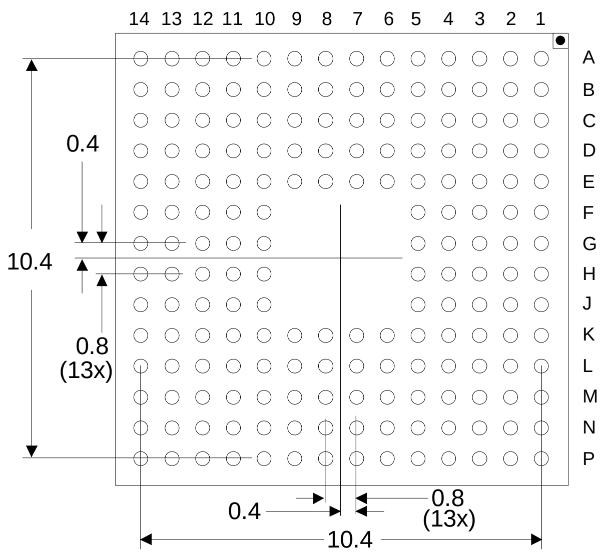
Figure 28 - 160-Lead LQFP Package





DIM	MIN	MAX
A	0.95	1.3
A1	0.25	0.35
A2	0.26	
A3	0.7	
b	0.35	0.45
Z	0.10	

UNIT = MM



LFBGA 180 BALLS, MO-205-D
12X12X1.3 PKG 0.8 PITCH POD

Figure 29 - LX180BGA Package

14 Source References

This section contains a list of sources for hardware and software references in this document.

14.1 Atmel Corporation

14.1.1 AT45DB021A Serial DataFlash®, Rev. 1642B-08/00

The AT45DB021A is a 2.7-volt only, serial interface Flash memory suitable for in-system programming. Its 2,162,688 bits of memory are organized as 1024 pages of 264 bytes each. The device also contains two SRAM data buffers of 264 bytes each.

Index