SPIRIT-30 SYSTEM

TECHNICAL REFERENCE MANUAL

Sonitech International Inc
83 Fullerbrook Road
Wellesley, MA 02181 , USA

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

CHAPTER 2: HARDWARE DESCRIPTION

CHAPTER 3: LIBRARY DESCRIPTION

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 : INTRODUCTION

1.1 OVERVIEW

The SPIRIT-30 system offers 33 MFLOPS of performance for PC XT/AT, PS/2, Apple, SUN (VME bus), Q-bus and the Multibus II and is embraced by a complete application development environment. The SPIRIT-30 board, along with comprehensive development software (SPIRIT-EDSP) and DSP program library (DSPL) is well suited for embedded applications for image analysis, graphics, numerical computation, control systems, telecommunications and other high performance applications.

The heart of the SPIRIT-30 is the Texas Instrument's TMS320C30 Digital Signal Processor (DSP) with 60 ns instruction cycle time, 2Kx32 words of internal RAM, single cycle floating-point multiply/accumulate and an on chip DMA controller.

1.2 HARDWARE AND SOFTWARE SYSTEM REQUIREMENTS

The following hardware and software environment is required by the SPIRIT-30 :

- IBM XT/AT/386 or compatibles
- DOS 3.0 or later
- Hard disk
- 640K memory
- Hercules or IBM compatible EGA graphics

1.3 FEATURES

The SPIRIT-30 offers the following features :

* Texas Instrument's high performance floating-point Digital Signal Processing chip (TMS320C30)

* Universal Host Interface permits connection of various buses

* Single slot IBM XT/AT direct plug-in solution

* Up to 128 Kbytes of fast dual access RAM (25 nsec)

* Capability to expand entire memory range, up to 64 Mbytes

Figure 1.0 shows the *SPIRIT-30 Functional Block Diagram*. As shown, the PC XT/AT/386 bus and the universal host connector bus become a common bus after the front end interface to the respective host computer. This new bus incorporates two write only ports and one read only port. Port A is used to read the data from the SPIRIT memory. Port B is used to write address and data information to the SPIRIT and Port C is used to write control information to the SPIRIT. The new bus and the DSP connect to the dual access fast SRAM array. Upto 32Kx32 of dual access memory is available on board. Additional dual access memory is available via the memory expansion board. Besides the dual access memory the DSP also has 8Kx32 of dedicated memory on the peripheral memory expansion. The two serial ports of the DSP are brought out to the two connectors on the side panel.

Figure 1.1 shows the *SPIRIT-30 Interface Block Diagram*. As shown the SPIRIT-30 is a direct plug in board for the PC XT/AT/386 machines. For other host systems such as VME bus, MAC, PS/2, Q bus and Multibus, the SPIRIT interfaces through a Digital I/O (DIO) board and connects to the DIO via a 37 pin cable. This cable carries power, GND and other signals for the SPIRIT. For these hosts, the SPIRIT is packaged in a small box which sits by the side of the host computer.

1.4 PROCESSOR

The processor used for application acceleration is the TMS320C30. This processor offers the following features :

* 60-ns single-cycle instruction execution time
* One 4K x 32-bit single-cycle dual-access on chip ROM block
* Two 1K x 32-bit single cycle dual-access on chip RAM block
* 64 x 32- bit instruction cache
* Up to 16M x 32-bit of addressable memory space
* 40/32-bit floating point/integer multiply and ALU
* 32 bits barrel shifter
* On chip Direct Memory Access (DMA) controller for concurrent I/O and CPU operation
* Parallel multiply and ALU instructions in single cycle
* Zero overhead loops with single cycle branches
* Two serial ports to support 8/16/32 bits transfers
* Two 32 bits timers

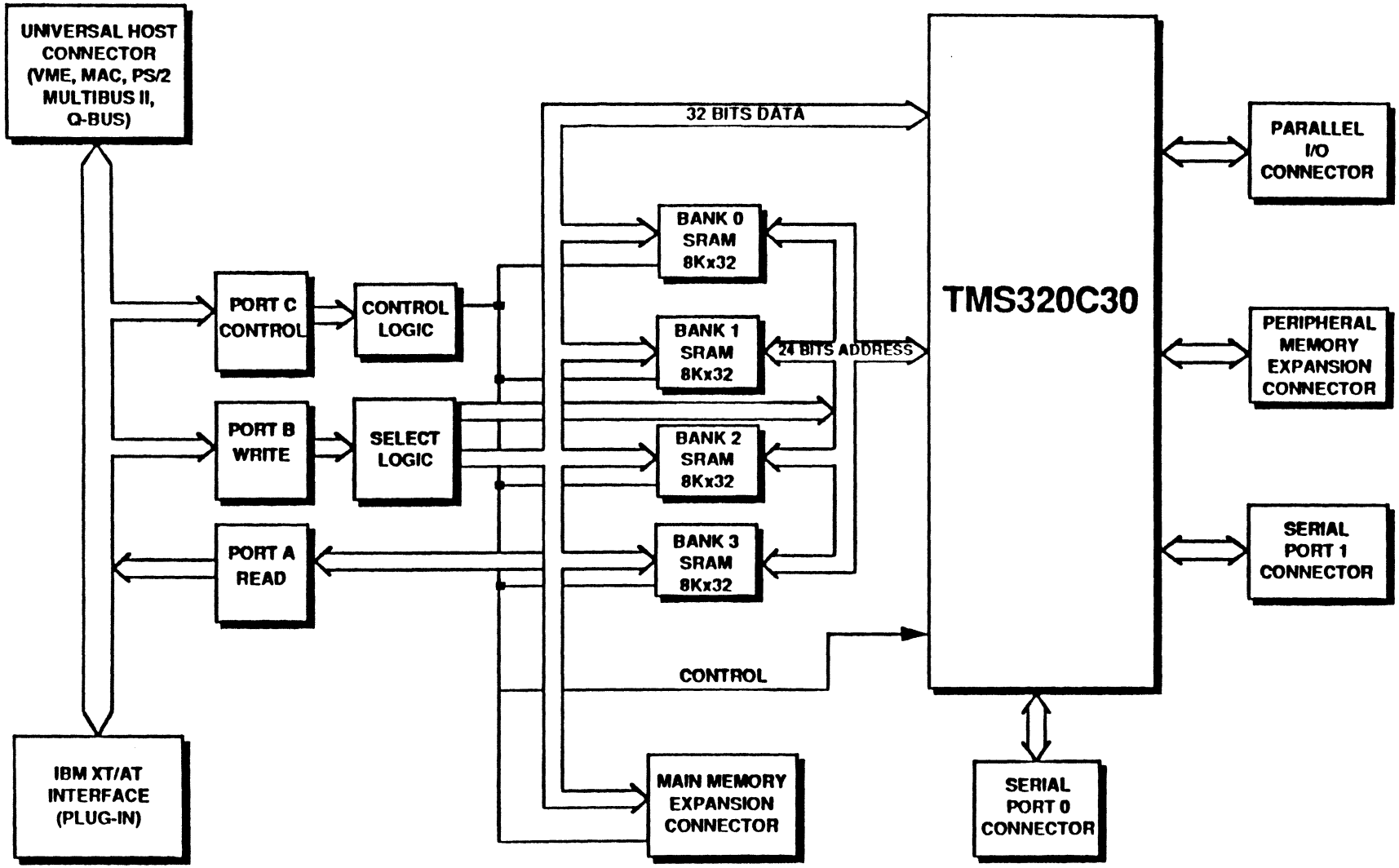
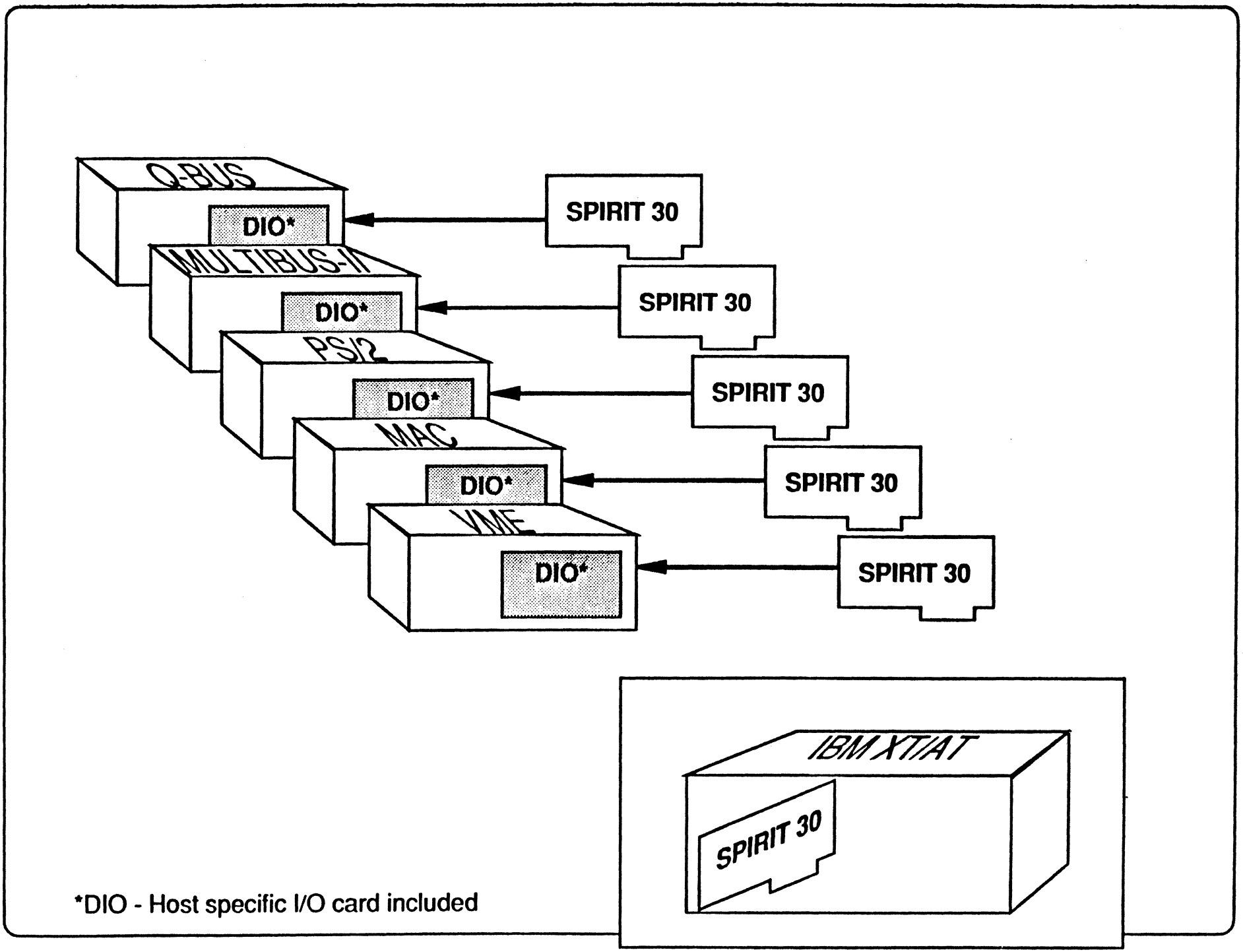


Figure 1.0 SPIRIT-30 FUNCTIONAL BLOCK DIAGRAM


Q-BUS
DIO*
SPIRIT 30
MULTIBUS-II
DIO*
SPIRIT 30
PS/2
DIO*
SPIRIT 30
MAC
DIO*
SPIRIT 30
VME
DIO*
SPIRIT 30
IBM XT/AT
SPIRIT 30
*DIO - Host specific I/O card included

1.5 MEMORY ARRAY AND EXPANSION

The memory array on the SPIRIT-30 system consists of 4 banks of dual access fast SRAM (25ns access time) of 8kx32 words each providing a total memory of 32kx32 or 128k bytes. This memory is mapped to the primary memory bus of the TMS320C30, starting from 000000h.

Additional memory can be obtained by expanding memory either on the parallel I/O bus or the primary memory bus of the processor. This is done by plugging additional memory expansion daughter boards to the expansion slots of the SPIRIT-30.

A total of 512 words of memory (from address 0c0h to 2c0h) are reserved for internal use and cannot be used by user programs.

1.6 HOST COMPUTER INTERFACE

The SPIRIT-30 board interfaces with a variety of host computers including IBM XT/AT/386, VME bus, Q-bus, PS/2 and multibus II. The board is port mapped into the I/O address space of the host and occupies 3 consecutive addresses. The base address is user selectable via switches.

The SPIRIT-30's auto incrementing address feature provides high speed block data transfers. The autoincrement feature can be disabled by writing to the Control register for applications which require polling on a single memory address.

1.6.1 IBM XT/AT/386 interface

The SPIRIT-30 board directly plugs into any IBM XT/AT/386 and 100% compatible computers. The base address is user selectable through the 8 dip switch settings (SW2) to avoid conflicts with other adapters and to allow multiple SPIRIT-30s to operate in a single host.

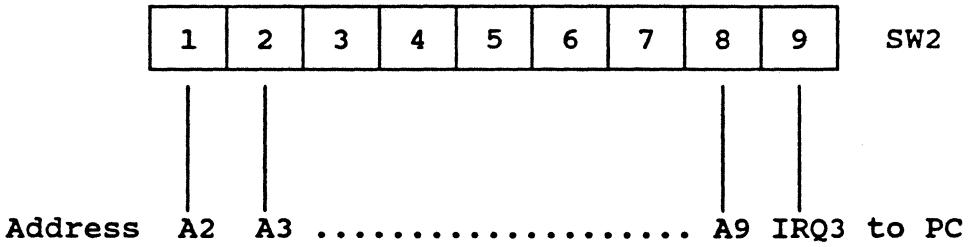
PC Host configuration switch settings :

The SPIRIT-30 board has one DIP block with 9 switches which allow the user to

- Select the base address

- Disable / enable interrupt to PC host

The switch block is labelled SW2 and is located right above the PC bus fingers. The base address can be changed with the switch settings 1 to 8 as the Address lines A2 to A9 of the PC are connected to these settings.



If the switch setting is depressed on the ON side then it indicates 'on' or a logic '1'. Similarly if the switch setting is depressed on the OFF side then it indicates 'off' or a logic '0'.

Following is a list of the I/O ports used by the SPIRIT-30 board :

base address - READ PORT
base address + 1 - WRITE PORT
base address + 2 - CONTROL PORT

For the PC hosts the SPIRIT-30 boards are factory installed with base address of 31Ch, hence the factory installed port addresses are ,

READ PORT - 31Ch
WRITE PORT - 31Dh
CONTROL PORT - 31Eh

These addresses are achieved by setting A2 through A9 to the following values :

A9 - 1 A8 - 1 A7 - 0
A6 - 0 A5 - 0 A4 - 1
A3 - 1 A2 - 1

1.6.1 Other host system interface

For host systems other than PC XT/AT/386, the SPIRIT-30 system interfaces to the host through a Digital I/O (DIO) card. SPIRIT connects to the DIO card via a 37 pin cable. The SPIRIT-30 board is packaged in a small box and sits by the side of the host. The base address is user selectable in the host I/O space through the jumper switch located on the Digital I/O card. Refer to the user documentation of the DIO card for switch settings.

1.7 INPUT / OUTPUT

SPIRIT-30 has several I/O connectors available for flexible interface to the outside world. These connectors are listed below along with the functional description :

Table 1.0 - Connectors on the SPIRIT-30 system

P1	- Parallel I/O connector
P2	- Peripheral Memory bus expansion connector
P3	- Emulation connector for XDS1000 as the host
P4A	- Primary Memory bus expansion connector
P4B	- Primary Memory bus expansion connector
P5	- Host connector
P6	- Universal Host connector (37 pin D type)
P11	- SERIAL PORT 0 connector
P12	- SERIAL PORT 1 connector

Figure 1.2 shows these connectors and their positions. Also refer to Figure 1.3 for signal diagram of these connectors.

1.8 BOARD JUMPERS AND CONNECTORS

The SPIRIT-30 system has the following jumpers. Refer to table 1.1 for a list of connectors to figure 1.2 for connector and jumper positions on the board.

Table 1.1 - Jumpers on the SPIRIT-30 system

J1	Plugged in	TMS320C30 operates in Microcomputer Mode
	Removed	TMS320C30 operates in Microprocessor Mode
J2	J2-1 connected to J2-2	EPROM Mode
	J2-2 connected to J2-3	SRAM Mode
J3	J3-1 connected to J3-2	0 wait state BANK0 and 1
	J3-2 connected to J3-3	1 wait state BANK0 and 1

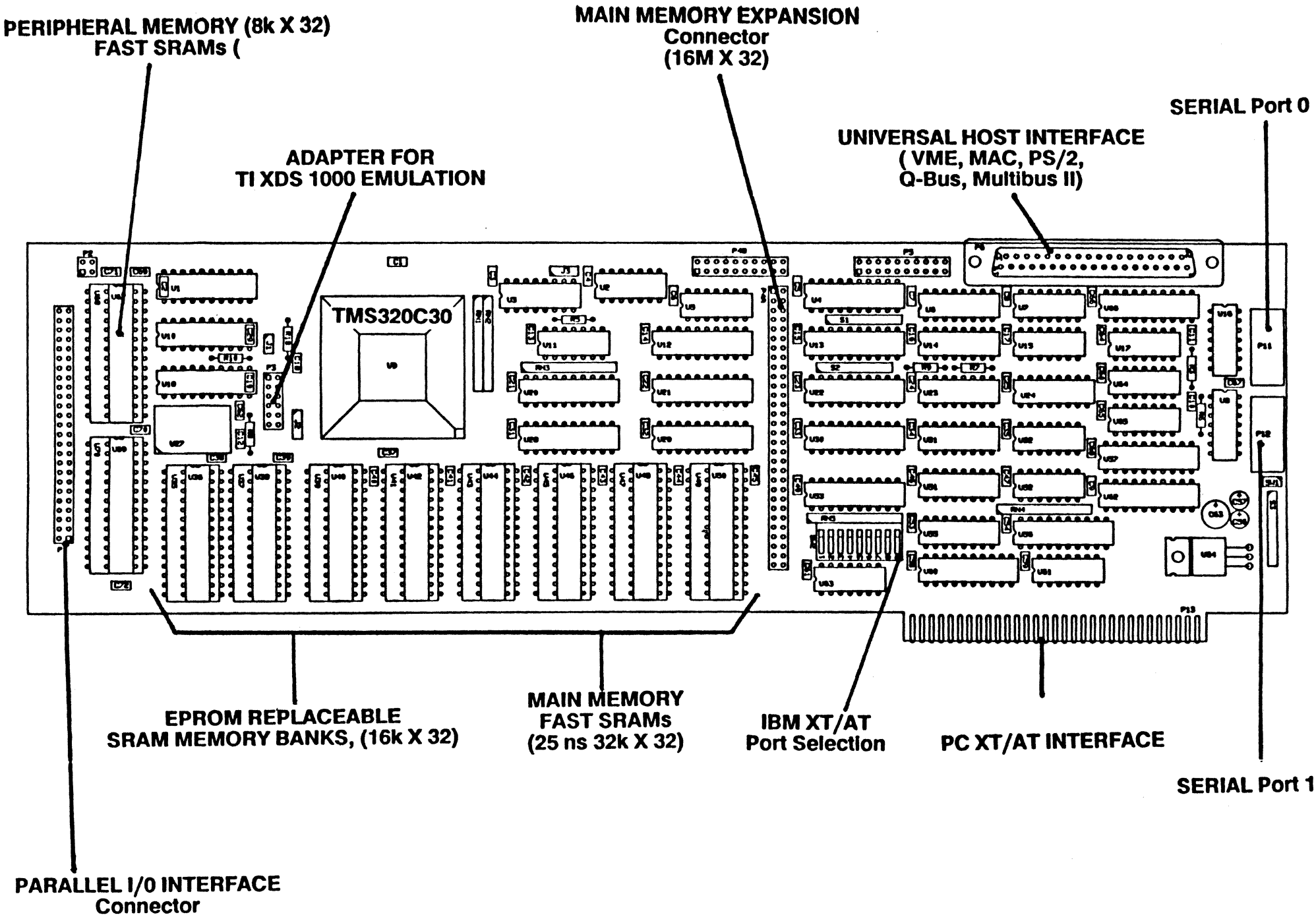


Figure 1.2 SPIRIT-30 BOARD LAYOUT

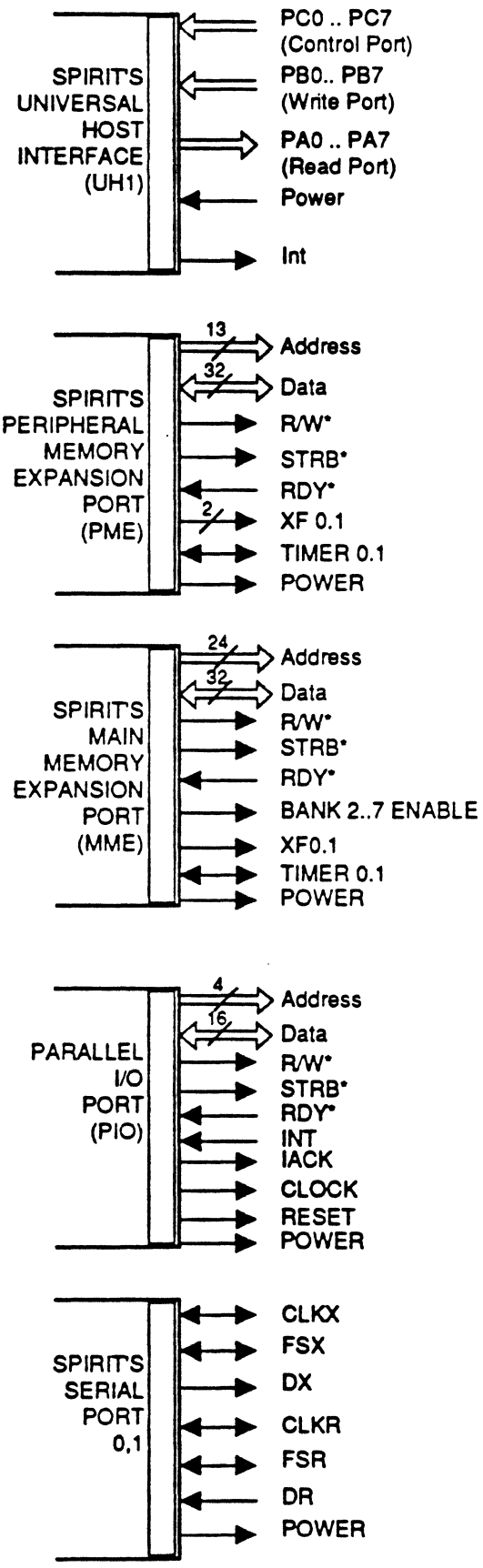


Figure 1.3 SPIRIT-30 CONNECTORS

1.9 DEFAULT JUMPER SETTINGS

The SPIRIT-30 board is factory configured to the following jumpers :

J1 - Removed (Microprocessor mode)
J2 - J2-2 connected to J2-3 (SRAM configuration)
J3 - J3-1 connected to J3-2 (0 wait state BANK0 and 1)

1.10 POWER SUPPLY CONNECTIONS

SPIRIT-30 requires +5. All supplies are regulated by the onboard regulator.

SPIRIT-30 can be powered in one of three ways :

1) Power from the host - The host digital I/O card supplies +12V and GROUND lines. +12 Volts is used to generate +5 volts using a regulator.

2) Power from the IBM XT/AT/386 - The +12 and GND lines of the PC are shorted on the board to the same lines from the host on connector P6.

CHAPTER 2 : HARDWARE DESCRIPTION

2.1 HOST INTERFACE

SPIRIT-30 system interfaces with a variety of host buses including IBM XT/AT, Nubus, Q-bus, VME and Multibus II. For the PC XT/AT/386 SPIRIT-30 is a single card solution and directly plugs into the PC, where as for other bus interfaces a digital I/O card is used. The SPIRIT-30 is PORT mapped into the I/O address space of the host computer and occupies 3 consecutive addresses. The address, data and control information between the host computer and the SPIRIT-30 system is exchanged via these 3 ports.

Two of these PORTS are used as WRITE only and the third PORT is used as READ only. The 3 PORTS are designated A, B and C.

PORT A - READ only
PORT B - WRITE only
PORT C - WRITE only

For writing to the SPIRIT-30 memory, the address information is first written to the SPIRIT-30 onboard address counters through ports B and C followed by the data on port B. For reading from the SPIRIT-30 memory, the address information is first written to the SPIRIT-30 onboard address counters through ports B and C followed by the data read on port A.

The SPIRIT-30's auto incrementing address feature provides high speed block data transfers. The autoincrement feature can be disabled by writing to the Control register for applications which require polling on a single memory address,

2.1.1 IBM PC XT/AT/386 Interface

Three ports in the I/O address space of the PC are used for the interface. The base address for the SPIRIT-30 board is jumper selectable in the entire user I/O space of the PC. Address lines A2 to A9 are decoded to provide the base address. The base address can be set to any I/O address by the dip switch (SW2).

To write or read from SPIRIT-30, address is written to the onboard counters and data is written to, or read from the latches.

SPIRIT-30 can interrupt the PC through the IRQ3 input of the PC bus. The DIP switch jumper 9 can be used to enable / disable the interrupt to the PC host.

2.1.2 Universal Host Interface

The SPIRIT-30 board can be used with various host systems and buses (see figure 1.1). The interface of the SPIRIT-30 to various host buses is through the 37 pin D type connector (P6) located on the top right corner of the board. A digital I/O card is used as the front end interface to various host system. The SPIRIT-30 is housed such that it can be located close to the host system. The SPIRIT-30 board is connected to the digital I/O card by a 37 wire ribbon cable. The cable carries all signals including the GND and Power to and from the SPIRIT-30. The digital I/O card has 3 PORTS of 8 bits each, which are used to interface with the SPIRIT-30 board. These ports carry address and data information to and from the SPIRIT-30.

The digital I/O cards are included with the SPIRIT-30 configurations. These cards are sourced from established vendors.

SPIRIT-30 can interrupt the host through the INTIN input of the Digital I/O card.

2.2 STATUS REGISTER

SPIRIT-30 board has a status register, located at the write port B. A read at this PORT B address enables this buffer to be read by the PC. Following status lines, from the DSP, can be monitored through this register :

Bit 0 - XF0
Bit 1 - XF1
Bit 2 - TCLK0
Bit 3 - TCLK1

Bits 4 to 7 are unused

This status register is particularly useful in user applications where the host computer has to poll on a memory location for a software flag. Instead of polling on the software flag (in memory), the user program can set the hardware flag (XF0, XF1, TCLK0, TCLK1) of DSP and the PC can periodically poll on this status register monitoring the state of the these flags. This scheme of hardware flag minimizes the DSP interruptions from the PC and hence results in higher processing throuhput.

2.3 MEMORY ARRAY AND EXPANSION

Memory Array on board

SPIRIT-30 has 32K x 32 of on board dual access memory which can be addressed both from the PC as well as from the TMS320C30. The memory is split into 4 banks of 8Kx32 each. Each one of these banks is implemented by 8Kx8 fast (0, 1 wait states) SRAM memories. Additional 8K x 32 onboard memory is available on the peripheral bus of the TMS320C30 for local use by the DSP. A total of 512 words of memory (from address 0c0h to 2c0h) are reserved for debugger.

Address lines A13..A15 are decoded to generate 8 bank select signals BANK0 to BANK7. Out of these only 4 signals (BANK0 to BANK3) are used to select the four onboard banks. BANK4 to BANK7 signals are available for memory expansion. The following table 2.0 outlines the various banks and their mapping.

Table 2.0 - Banks and the address mapping

BANK	ADDRESS MAPPING	SIZE
BANK 0	000000 - 001FFF	8K (Onboard Standard)
BANK 1	002000 - 003FFF	8K (Onboard Standard)
BANK 2	004000 - 005FFF	8K (Onboard Optional)
BANK 3	006000 - 007FFF	8K (Onboard Optional)
BANK 4	008000 - 009FFF	8K (Expansion board)
BANK 5	00A000 - 00BFFF	8K (Expansion board)
BANK 6	00C000 - 00DFFF	8K (Expansion board)
BANK 7	00E000 - 00FFFF	8K (Expansion board)

Generation of RDY* to TMS320C30 (WAIT state generation)

RDY* signal input to TMS320C30 is used to indicate that the external device is not ready with the data. In this case the TMS320C30 implements WAIT states.

WAIT state generation circuitry is implemented to generate either 0 or 1 wait state only. Zero (0) wait state devices, which tend to be fast, can usually respond immediately with a ready indication. Wait state devices may simply delay their select signals appropriately to generate a ready.

Memory Expansion

SPIRIT-30 has capability for expansion of complete memory space of the TMS320C30. Memory expansion is available for both the Primary bus and the Peripheral bus.

Primary Memory expansion

A total of 16M words (32 bits) of memory can be accessed via this expansion. All address and associated bus signals are brought to the header P4/A and P4/B. See Figure 1.2 for location of these connectors.

Header P4/B contains the extra address lines and decoded BANK select signals, thus eliminating the need for further decoding for up to total of 64Kx32 memory expansion. (only BANK 0 to BANK3 are used on SPIRIT-30, BANK 4 to BANK 7 can be used by the memory expansion board.

Peripheral Memory expansion

Upto 16K words (32 bits) of memory can be expanded on the expansion I/O bus by the use of two strobe signals, namely MSTRB* and IOSTRB*. Of the 16K memory space, 8K memory is available onboard the SPIRIT-30 on MSTRB* signal. The memory space available via the IOSTRB* signal is usually used to connect I/O devices to the SPIRIT, hence the IOSTRB*, 4 address lines, 16 data lines and other control signals are brought out to the connector P1. See Figure 1.2 for location of this connector. Also refer to section 2.5.2 for more details.

2.4 STAND ALONE OPERATION (USING EPROM)

SPIRIT-30 can be operated in a stand-alone mode without any host system interface. Bank 0 and Bank 1 (thus a total of 16Kx32) can be replaced with EPROMs containing the bootstrap / system program. Thus SPIRIT-30 can be used for embedded applications.

While replacing BANK 0 and BANK 1 with EPROMs jumper J2 should be connected between pins 1 and 2. This will disable WRITEing to the EPROMs.

2.5 RESET CIRCUITRY

SPIRIT-30 reset circuitry resets the DSP, and the logic on the board.

There are three ways to RESET the DSP:

1) On power up DSP and the board are reset or
2) DSP can be reset using and external switch SW1, or
3) DSP can also be reset from the host through the control bits of PORT C.

2.6 ANALOG I/O INTERFACE

SPIRIT-30 system has been designed to interface to a variety of Analog I/O subsystems. The data acquisition can be done either through the serial or the parallel interface.

Sonitech has a variety of data acquisition cards available with various flavors. SPIRIT-30 system also interfaces to data acquisition cards from Data Translation, Metra Byte among other vendors.

2.6.1 Serial I/O

SPIRIT-30 system offers capability for analog I/O through the two serial port of the TMS320C30. Sonitech offers various data acquisition modules with variable sampling rates, 12-16 bits resolution and other specifications. Contact the factory for more information on these options.

Note that SPIRIT-30 can also interface with standard industry Analog I/O cards through serial interface. The two serial ports of the DSP are available on connectors P11 and P12. Refer to Table 2.1 for pin assignment of these connectors.

Table 2.1 - 10 Pin Serial Port Signal Description

PIN #	NAME	DIRECTION	DESCRIPTION
1	GND	-	Digital GND
2	CLKR	I/O	Serial In Clock
3	DR	I	Serial Data In to DSP
4	FSR	I	Serial In Frame Sync
5	+5V	-	+5 Volts
6	INT*	I	Interrupt 2 or 3 to DSP
7	FSX	I/O	Serial Out Frame Sync
8	DX	O	Serial Data Out from DSP
9	CLKX	I/O	Serial Out Clock
10	GND	-	Digital GND

I = Input to the SPIRIT-30
O = Output of SPIRIT-30
* = Active low signal

The polarity of FSX and FSR is programmble

2.6.2 Parallel I/O

SPIRIT-30 has the standard connector P1 to communicate with several Analog I/O cards. This interface is connected to the Expansion I/O bus of the DSP. Of the 32 data lines of the DSP only 16 (D0-D15) are brought to the connector. This interface allows SPIRIT-30 to interface with high speed data acquisition cards.

All signals to and from the connector are buffered. Data is buffered by 74HCT245 and other signals including Address and control signals are buffered by 74HCT244. Refer to Table 2.2 for pin assignment of this connector.

Note that only 4 address lines (out of 13) are brought to the connector the addresses and are not decoded, hence there will be mirror images of the address in blocks of 16.

Table 2.2 - 50 pin Parallel Port pin Description

PIN #	NAME	DIRECTION	DESCRIPTION
1	Vcc	O	+5 V
2	GND	-	GND
3	NC	-	NO CONNECTION
4	H1	O	H1 CLOCK FROM DSP (16 MHz)
5	NC	-	NO CONNECTION
6	NC	-	NO CONNECTION
7	NC	-	NO CONNECTION
8	NC	-	NO CONNECTION
9	NC	-	NO CONNECTION
10	NC	-	NO CONNECTION
11	GND	-	GND
12	CLOCK	O	BUFFERED CLOCK (8HHZ)
13	RESET*	O	RESET TO I/O
14	XF0	O	FLAG FROM DSP
15	NC	-	NO CONNECTION
16	IACK*	O	INTERRUPT ACKNOWLEDGE
17	NC	-	NO CONNECTION
18	INT1*	I	INTERRUPT 1 TO DSP
19	NC	-	NO CONNECTION
20	RDY*	I	I/O READY INPUT TO DSP
21	NC	-	NO CONNECTION
22	IOSTRB*	O	I/O DEVICE ENABLE
23	R/W*	O	READ/WRITE SIGNAL
24	GND	-	GND
25	NC	-	NO CONNECTION
26	H3	O	H3 CLOCK FROM DSP (16 MHz)
27	NC	-	NO CONNECTION
28	GND	-	GND
29	A3	O	ADDRESS LINE 3
30	A2	O	ADDRESS LINE 2
31	A1	O	ADDRESS LINE 1
32	A0	O	ADDRESS LINE 0
33	GND	-	GND
34	D15	I/O	DATA LINE 15 (MSB)
35	D14	I/O	DATA LINE 14
36	D13	I/O	DATA LINE 13
37	D12	I/O	DATA LINE 12
38	D11	I/O	DATA LINE 11

PIN #	NAME	DIRECTION	DESCRIPTION
39	D10	I/O	DATA LINE 10
40	D9	I/O	DATA LINE 9
41	D8	I/O	DATA LINE 8
42	D7	I/O	DATA LINE 7
43	D6	I/O	DATA LINE 6
44	D5	I/O	DATA LINE 5
45	D4	I/O	DATA LINE 4
46	D3	I/O	DATA LINE 3
47	D2	I/O	DATA LINE 2
48	D1	I/O	DATA LINE 1
49	D0	I/O	DATA LINE 0 (LSB)
50	GND	-	GND

I = Input to the SPIRIT-30
O = Output of SPIRIT-30
* = Active low signal

2.7 MULTIPROCESSING

SPIRIT-30 can be used in several multiprocessing configurations.

Up to three SPIRIT-30 systems can be directly connected through the two serial ports. To connect more SPIRIT-30 systems, the parallel I/O bus can also be used to connect one SPIRIT-30 system to its neighbor. The two systems can then be synchronized using interlocked instructions provided by the DSP instruction set.

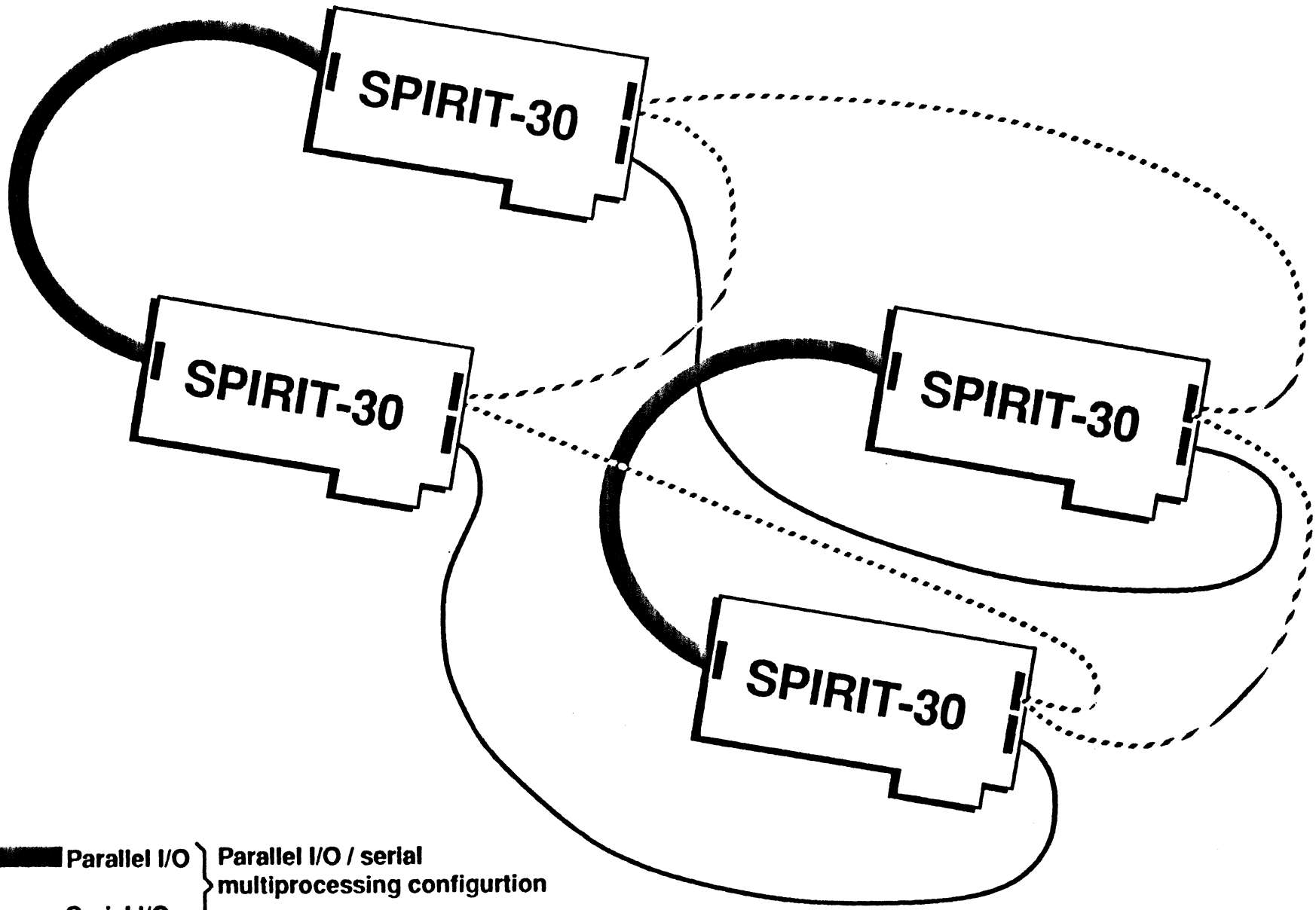
Two of the several possible multiprocessing configuration are shown in block diagram in Figure 2.0. One configuration is completly connected, in which two neighbouring TMS320C30s are connected via the two serial links (serial ports 0 and 1) and the opposite TMS320C30s are connected via the parallel link. The other configuration is the Token Ring in which all DSPs communicate through one serial port via message passing scheme. In this configuration one DSP signals to the next in line via packets of information with the leading and trailing headers.

2.8 EMULATION USING XDS1000

SPIRIT-30 is provided with the 12 pin header P12, as recommended by Texas Instruments (page 13-26 of the TMS320C30 users guide), to allow real time emulation via a serial scan path.

For more information refer to pages 13-26 and B9 of the TMS320C30 users guide.

Figure 2.0 SPIRIT-30 MULTIPROCESSING CONFIGURATIONS



CHAPTER 3 : LIBRARY DESCRIPTION

The following is a list of the library functions which are provided with the SPIRIT-30 system. These functions can be used for development of embedded applications.

dsp_reset()
Routine to "reset" TMS320C30.

pc_dsp_int()
Routine to interrupt TMS320C30 from the PC.

dsp_dl_long_array()
Routine to download an array of long integers from the PC's memory to the SPIRIT-30 memory.

dsp_up_long_array()
Routine to upload an array of long integers from the SPIRIT-30 memory to the PC's memory.

dsp_dl_int_array()
Routine to download an array of signed integers from the PC's memory to the SPIRIT-30 memory.

dsp_up_int_array()
Routine to upload an array of signed integers from the SPIRIT-30 memory to the PC's memory.

dsp_dl_unsigned_array()
Routine to download an array of unsigned integers from the PC's memory to the SPIRIT-30 memory.

dsp_up_unsigned_array()
Routine to upload an array of unsigned integers from the SPIRIT-30 memory to the PC's memory.

dsp_fast_dl_array()
Routine to fast download an array of unsigned integers from the PC's memory to the SPIRIT-30 memory.

mem_init()
Routine to initialise the memory of SPIRIT-30 with 0.

dsp_dl_exec()
Routine to download executable file to SPIRIT-30.

get_laddr()
Routine to find address of a global variable in a SPIRIT-30 resident program.

set_board_addr()
Routine to set the I/O base address of SPIRIT-30 ports.

Function name

dsp_reset - Routine to "reset" TMS320C30.

void dsp_reset();

Arguments

None

Return value

None

Processing

Routine asserts the 'reset' signal to the TMS320C30. This causes the TMS320C30 to set its program counter to 0h and commence execution from that location.

Example

```
#include "spirit.h"

main()
{
        dsp_reset();
}
```

This example asserts the 'reset' signal to the TMS320C30.

Related functions

None

Function name

pc_dsp_int - Routine to interrupt the TMS320C30 from the PC.

void pc_dsp_int();

Arguments

None

Return value

None

Processing

Routine asserts the 'INT0' signal to the TMS320C30.

Example

```
#include "spirit.h"

main()
{
        pc_dsp_int();
}
```

This example asserts the 'INT0' signal to the TMS320C30.

Related functions

None

Function name

dsp_dl_long_array - Routine to download an array of long integers (32 bits) from the PC's memory to the SPIRIT-30 external memory.

void dsp_dl_long_array(long address, unsigned count, long far *array);

Arguments

address long integer whose 24 least significant bits represent SPIRIT-30 memory address. Valid address range is 0h to XXXXXXh where XXXXXXh is the memory (in bytes) available on SPIRIT-30 external to TMS320C30.

count number of values to be downloaded.

array array of long integers to be downloaded to the SPIRIT-30 memory.

Return value

None

Processing

Routine downloads an array of long integers (32 bits) from the PC's memory to the SPIRIT-30 external memory. Note that no range checking is done on either the address range or the count.

Example

```
#include "spirit.h"

main()
{
        long spirit_addr;
        unsigned int count;
        long far array[256];

        spirit_addr = 0x0C0L;
        count = 256;

        dsp_dl_long_array(spirit_addr,count,array);
}
```

This example downloads 256 '32-bit words' starting from the location pointed to by "array" to 256 locations of the SPIRIT-30 external memory starting from C0h.

Related functions

dsp_up_long_array()

Function name

dsp_up_long_array - Routine to upload an array of long integers (32 bits) from SPIRIT-30 external memory to the PC's memory.

void dsp_up_long_array(long address, unsigned count, long far *array);

Arguments

address long integer whose 24 least significant bits represent SPIRIT-30 memory address. Valid address range is 0h to XXXXXXh where XXXXXXh is the memory (in bytes) available on SPIRIT-30 external to TMS320C30.

count number of values to be uploaded.

array array of long integers receiving data from the SPIRIT-30 memory.

Return value

None

Processing

Routine uploads an array of long integers (32 bits) to the PC's memory from the SPIRIT-30 external memory. Note that no range checking is done on either the address range or the count.

Example

```
#include "spirit.h"

main()
{
        long spirit_addr;
        unsigned int count;
        long far array[256];

        spirit_addr = 0x0C0L;
        count = 256;

        dsp_up_long_array(spirit_addr,count,array);
}
```

This example uploads 256 '32-bit words' to the memory area starting from the location pointed to by "array" from 256 locations of the SPIRIT-30 external memory starting from C0h.

Related functions

dsp_dl_long_array()

Function name

dsp_dl_int_array - Routine to download an array of signed integers (16 bits) from the PC's memory to the SPIRIT-30 external memory.

void dsp_dl_int_array(long address, unsigned count, int far *array);

Arguments

address long integer whose 24 least significant bits represent SPIRIT-30 memory address. Valid address range is 0h to XXXXXXh where XXXXXXh is the memory (in bytes) available on SPIRIT-30 external to TMS320C30.

count number of values to be downloaded.

array array of signed integers to be downloaded to the SPIRIT-30 memory.

Return value

None

Processing

Routine downloads an array of integers (16 bits) from the PC's memory to the SPIRIT-30 external memory. Note that no range checking is done on either the address range or the count. The integers are sign extended to 32 bits.

Example

```
#include "spirit.h"

main()
{
        long spirit_addr;
        unsigned int count;
        int far array[256];

        spirit_addr = 0x0C0L;
        count = 256;

        dsp_dl_int_array(spirit_addr,count,array);
}
```

This example downloads 256 integer values starting from the location pointed to by "array" to 256 locations of the SPIRIT-30 external memory starting from C0h.

Related functions

dsp_up_int_array()

Function name

dsp_up_int_array - Routine to upload an array of signed integers (16 bits) from SPIRIT-30 external memory to the PC's memory.

void dsp_up_int_array(long address, unsigned count, int far *array);

Arguments

address long integer whose 24 least significant bits represent SPIRIT-30 memory address. Valid address range is 0h to XXXXXXh where XXXXXXh is the memory (in bytes) available on SPIRIT-30 external to TMS320C30.

count number of values to be uploaded.

array array of integers receiving data from the SPIRIT-30 memory.

Return value

None

Processing

Routine uploads an array of integers (16 bits) to the PC's memory from the SPIRIT-30 external memory. The upper 16 bits of SPIRIT-30 memory words are ignored. Note that no range checking is done on either the address range or the count.

Example

```
#include "spirit.h"

main()
{
        long spirit_addr;
        unsigned int count;
        int far array[256];

        spirit_addr = 0x0C0L;
        count = 256;

        dsp_up_int_array(spirit_addr,count,array);
}
```

This example uploads 256 integer values to the memory area starting from the location pointed to by "array" from 256 locations of the SPIRIT-30 external memory starting from C0h.

Related functions

dsp_dl_int_array()

Function name

dsp_dl_unsigned_array - Routine to download an array of unsigned integers (16 bits) from the PC's memory to the SPIRIT-30 external memory.

void dsp_dl_unsigned_array(long address, unsigned count, unsigned far *array);

Arguments

address long integer whose 24 least significant bits represent SPIRIT-30 memory address. Valid address range is 0h to XXXXXXh where XXXXXXh is the memory (in bytes) available on SPIRIT-30 external to TMS320C30.

count number of values to be downloaded.

array array of unsigned integers to be downloaded to the SPIRIT-30 memory.

Return value

None

Processing

Routine downloads an array of unsigned integers (16 bits) from the PC's memory to the SPIRIT-30 external memory. Note that no range checking is done on either the address range or the count. The unsigned integers are extended to 32 bits (upper 16 bits of SPIRIT-30 memory are filled with 0s).

Example

```
#include "spirit.h"

main()
{
        long spirit_addr;
        unsigned int count;
        unsigned int far array[256];

        spirit_addr = 0x0C0L;
        count = 256;

        dsp_dl_unsigned_array(spirit_addr,count,array);
}
```

This example downloads 256 unsigned integer values starting from the location pointed to by "array" to 256 locations of the SPIRIT-30 external memory starting from C0h.

Related functions

dsp_up_unsigned_array()

Function name

dsp_up_unsigned_array - Routine to upload an array of unsigned integers (16 bits) from SPIRIT-30 external memory to the PC's memory.

void dsp_up_unsigned_array(long address, unsigned count, unsigned far *array);

Arguments

address long integer whose 24 least significant bits represent SPIRIT-30 memory address. Valid address range is 0h to XXXXXXh where XXXXXXh is the memory (in bytes) available on SPIRIT-30 external to TMS320C30.

count number of values to be uploaded.

array array of unsigned integers (16 bits) receiving data from the SPIRIT-30 memory.

Return value

None

Processing

Routine uploads an array of unsigned integers (16 bits) to the PC's memory from the SPIRIT-30 external memory. The upper 16 bits of SPIRIT-30 memory words are ignored. Note that no range checking is done on either the address range or the count.

Example

```
#include "spirit.h"

main()
{
        long spirit_addr;
        unsigned int count;
        unsigned int far array[256];

        spirit_addr = 0x0C0L;
        count = 256;

        dsp_up_unsigned_array(spirit_addr,count,array);
}
```

This example uploads 256 unsigned integer values to the memory area starting from the location pointed to by "array" from 256 locations of the SPIRIT-30 external memory starting from C0h.

Related functions

dsp_dl_unsigned_array()

Function name

dsp_fast_dl_array - Routine to fast download an array of unsigned integers (16 bits) from the PC's memory to the SPIRIT-30 external memory.

void dsp_fast_dl_array(long address, unsigned count, unsigned far *array);

Arguments

address long integer whose 24 least significant bits represent SPIRIT-30 memory address. Valid address range is 0h to XXXXXXh where XXXXXXh is the memory (in bytes) available on SPIRIT-30 external to TMS320C30.

count number of values to be downloaded.

array array of unsigned integers to be downloaded to the SPIRIT-30 memory.

Return value

None

Processing

Routine downloads an array of unsigned integers (16 bits) from the PC's memory to the SPIRIT-30 external memory. Note that no range checking is done on either the address range or the count. Unlike the dsp_dl_unsigned_array() routine, it doesn't extend the integers to 32 bits; it leaves the upper 16 bits of the SPIRIT-30 memory unchanged. Hence it is twice as fast as dsp_dl_unsigned_array(). The DSP program making use of this data must treat it as 16-bit numbers, rather than integers (TMS320C30 operates only on 32 bit integers).

Example

```
#include "spirit.h"
main()
{
        long spirit_addr;
        unsigned int count;
        unsigned int far array[256];

        spirit_addr = 0x0C0L;
        count = 256;

        dsp_fast_dl_array(spirit_addr,count,array);
}
```

This example downloads 256 unsigned integer values starting from the location pointed to by "array" to 256 locations of the SPIRIT-30 external memory starting from C0h, leaving upper 16 bits of SPIRIT-30 memory undisturbed.

Related functions

dsp_dl_unsigned_array()

Function name

mem_init - Routine to initialise the memory of SPIRIT-30 with 0.

void mem_init(long address, unsigned count);

Arguments

address long integer whose 24 least significant bits represent SPIRIT-30 memory address. Valid address range is 0h to XXXXXXh where XXXXXXh is the memory (in bytes) available on SPIRIT-30 external to TMS320C30.

count number of memory locations to be initialised.

Return value

None

Processing

Routine downloads 0 to the SPIRIT-30 external memory. Note that no range checking is done on either the address range or the count.

Example

```
#include "spirit.h"

main()
{
        long spirit_addr;
        unsigned int count;

        spirit_addr = 0x0C0L;
        count = 256;

        mem_init(spirit_addr,count);
}
```

This example initialises 256 locations (each 32 bits) of the SPIRIT-30 external memory starting from C0h.

Related functions

dsp_dl_long_array()

Function name

dsp_dl_exec - Routine to download executable file to SPIRIT-30.

int dsp_dl_exec(char *filename);

Arguments

filename DSP executable file.

Return value

0 if successful, -1 if error during load.

Processing

Reads the executable DSP COFF file (prepared by the TMS320C30 linker) and loads the executable code to SPIRIT-30. Also retrieves the symbol table from the COFF file, which can be used by get_laddr() to get address of a global label.

Example

```
#include <stdio.h>
#include "spirit.h"

main()
{
        int status;

        status = dsp_dl_exec("compute.out");
        if (status = = 0)
                printf("OK\n");
        else
                printf("ERROR\n");
}
```

This example downloads the executable code of the DSP COFF file "compute.out" to SPIRIT-30.

Related functions

get_laddr()

Function name

get_laddr - Routine to find address of global label specified.

long get_laddr(char * name);

Arguments

name string ptr to label name.

Return value

Address whose label name matches argument, -1 if no match.

Processing

Routine searches the symbol table of the program which is currently loaded onto SPIRIT-30 for match to "name", returns address corresponding to the label name. A call to this routine must be preceded by a call to dsp_dl_exec(), which retrieves the symbol table from the executable DSP COFF file.

Example

```
#include <stdio.h>
#include "spirit.h"

main()
{
        long addr;

        addr = get_laddr("_data_array");
        if (addr = = -1L)
                printf("LABEL DOESN'T EXIST\n");
        else
                printf("ADDRESS OF LABEL: %lx",addr);
}
```

This example finds the address of the global label "data_array" declared in DSP 'C' program. Note that a global label declared in a 'C' program must be prefixed by an underscore when given as a parameter to get_laddr().

Related functions

dsp_dl_exec()

Function name

set_board_addr - Routine to set the I/O base address of SPIRIT-30 ports.

void set_board_addr(unsigned board_no);

Arguments

board_no board number of the SPIRIT-30 which is going to be accessed subsequently.

Return value

0 if setting is successful, -1 if there is an error

Processing

Gets the setting of the form "SPIRIT30=XXX;YYY;.." (XXX,..: i/o address in hex) from the environment and sets base address depending on the board_no (address of board '1' is XXXh, board '2' is YYYh,..). If there is an error while parsing environment string, the base address is left unchanged. The default base address is 31Ch. This address is used by all download and upload routines, and dsp_dl_exec() to access the SPIRIT-30. Set_board_addr routine is useful only when more than one SPIRIT-30 board is used with a single host - this routine must be used to switch from one board to another, before using the other routines.

Examples

Let the environment contain the setting "SPIRIT30=310; 320;330".

1)
```
#include "spirit.h"
main()
{
        int retvalue;
        retvalue = set_board_addr(2);
        printf("%d\n",retvalue);
}
```

This example sets the base address of SPIRIT30 ports to 320h; it returns 0.

2)
```
#include "spirit.h"
main()
{
        int retvalue;
        retvalue = set_board_addr(4);
        printf("%d\n",retvalue);
}
```

This example leaves base address unchanged, since 4th board address is not specified in the setting; it returns -1.

Related functions

None

SPIRIT-30
PRODUCT REGISTRATION

****

Please retain this copy for your records and return the next page to us with complete information

****

The Product you have Purchased is:_____
Product serial number is:_____

Technical Support:

You will receive free product updates free technical support for next 3 months. For technical support contact:

Sonitech International, Inc.
83 Fullerbrook Road
Wellesley, MA 02181, USA
Telephone: (617)235-6824, Fax: (617)235-2531

Replacement Policy:

We will replace your software and boards free of charge within 45 days of purchase if they prove defective. You must call or write to us first indicating the problem and we will respond to you by telephone or writing within 3 days after hearing from you.

Registration:

For us to support and to provide you with timely updates on the workstation, please completely fill the form on the next page and mail to:

Sonitech International Inc.
83 Fullerbrook Road
Wellesley, MA 02181, USA

REGISTRATION CARD

We have read the *Sonitech International Inc.'s* program license agreement and agree to abide by the terms and conditions contained therein.

_____ _____
Signature Name of Customer

Name_____
Title_____
Organization_____
Address_____
Address_____
City_____
State_____ Code_____
Country_____
Telephone _____
Fax_____

Purchase Information:

Product Name:_____
Product Serial Number:_____ 129 _____

Software Purchased From:

__ *Sonitech International Inc.*
__ Distributor
Name:_____
Country:_____
Address:_____
__ Other

How did you first hear about our Products:
__ From Distributor
__ Mail
__ Advertising
__ Editorial
__ University

Please comment (on back of this page) about this package. Thank you.

WARRANTY FORM

This product is warranted against defects in materials and workmanship for 45 days. The Warranty is effective from date of purchase. Sonitech International Inc. (Sonitech) will repair or replace, at its option, any product found to be defective during the Warranty period. The Warranty will be invalid if any permanent alterations are made to the circuit card or any of its circuitry, or if the unit has been abused or mishandled. Damage due to static electric discharges will void the Warranty, as will application of fluctuating voltages on the power supply rails.

EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, NO OTHER WARRANTIES, WHETHER EXPRESSED OR IMPLIED, INCLUDING WARRANTIES OF MERCHANTABILITY AN FITNESS FOR A PARTICULAR PURPOSE, SHALL APPLY TO THIS PRODUCT; UNDER NO CIRCUMSTANCES SHALL SONITECH BE LIABLE FOR CONSEQUENTIAL DAMAGES SUSTAINED IN CONNECTION WITH SAID PRODUCT AND NEITHER ASSUMES NOR AUTHORIZES ANY REPRESENTATIVE OR OTHER PERSON TO ASSUME FOR IT ANY OBLIGATION OR LIABILITY OTHER THAN SUCH AS IS EXPRESSLY SET FORTH HEREIN.

Before returning this product, you must receive a Return Material Authorization (RAM). Returned product will not be accepted without this authorization. Unit must be shipped prepaid in its original container, with the power cable. If non-permanent modifications have been made to the unit, restore it completely to its original operating configuration. Permanent modification will void the Warranty.

SOFTWARE LICENSE AGREEMENT

The program(s) delivered with this Agreement are sold only on the condition that the purchaser agrees to the terms and conditions of this Agreement. READ THIS AGREEMENT CAREFULLY. If you do not agree, return the packaged program UNOPENED to your distributor or dealer and your purchase price will be refunded. If you agree, fill out and sign the Registration Form and RETURN to us by mail.

Sonitech International Inc. (hereinafter called "Sonitech") agrees to grant and the Customer agrees to accept, subject to the following terms and conditions, a personal, nonexclusive and nontransferable license to use the propriety program(s) (hereinafter called the "Program") of Sonitech International Inc. delivered with this agreement.

The program includes executable software, object files, and application asp programs. The program does not include data files, sample executables, and demo programs.

License

The license granted hereunder authorizes the Customer to use the Program in machine readable form on any single computer system (hereinafter called the "System"). A separate license is required for each System on which the Program will be use.

Copy

The program may only be copied, in whole or in part, in printed or machine readable form, for the use by the Customer on the System, to understand the contents of the Program, for back-up purposes, or for archive purposes; provided, however, that no more than two (2) printed copies shall be in existence with respect to any Program at any one time without prior written consent of Sonitech International Inc.

Term

This Agreement shall become effective as of the date of shipment of the Program from Sonitech International to the Customer, and shall continue in force until terminated by either party hereto pursuant to terms below:

The customer may terminate this Agreement by returning the Program unopened within 15 days, the Customer of delivery.

Miscellaneous

The rights and benefits of the Customer hereunder shall not be assigned or transferred in any manner whatsoever.

The validity and construction of this Agreement shall be governed by the laws of the State of Massachusetts. The parties shall attempt to settle disputes, controversies or differences which may arise out of or in relation to or in connection with this Agreement.

IMPORTANT NOTICE

Sonitech International Inc. (Sonitech) reserves the right to make changes in the devices or the device specifications identified in this User's Guide and price without notice. Sonitech advises its customers to obtain the latest version of device specification to verify, before placing orders, that the information being relied upon by the customer is current.

In the absence of written agreement to the contrary, Sonitech assumes no liability for Sonitech's applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does Sonitech warrant or represent that any license, either express or implied, is granted under any patent right, copyright, or other intellectual property right of Sonitech covering or relating to any combination, machine, process in which such semiconductor device might be or are used.

EXCEPT TO THE EXTENT PROHIBITED BY APPLICABLE LAW, UNDER NO CIRCUMSTANCES SHALL SONITECH BE LIABLE FOR CONSEQUENTIAL DAMAGES SUSTAINED IN CONNECTION WITH SAID PRODUCT AND NEITHER ASSUMES NOR AUTHORIZES ANY REPRESENTATIVE OR OTHER PERSON TO ASSUME FOR IT ANY OBLIGATION OR LIABILITY OTHER THAT SUCH AS IS EXPRESSLY SET FORTH HEREIN.