OPERATIONS MANUAL
LBC-486Plus
LBC-586Plus

REVISION HISTORY

P/N 403-0259-000

ECO Number	Date Code	Rev Level
ORIGINATED	970422	C
97-36	970602	C1
97-78	970829	C2
97-105	971204	C3
98-01	980107	C4
98-18	980311	C5
98-57	980807	C6
98-86	980817	C7
99-30	990609	D
99-83	991206	E

TABLE OF CONTENTS

1 General Information

1.1 Features

- 486DX4 at 100MHz or 5X86 at 133 MHz
- 100% PC-AT Compatible
- Up to 32 Mbytes of user installable FPM or EDO DRAM
- Optional 256K L2 Cache
- Solid State Disk Support of up to 12MB
- PCI High-Resolution VGA controller for CRT or Flat Panel usage
- PCI IDE Controller
- NE2000 Compatible 10BaseT, AUI, Ethernet Controller
- Four 16550 Compatible Serial ports with optional RS422, RS485, J1708 interfaces
- Bi-directional Parallel printer port supports EPP and ECP modes
- 48 Digital I/O lines with 24 line event sense capability
- Dual Floppy Disk interface
- 16-Bit PC/104 Expansion Bus
- Watchdog Timer with Power-fail reset

1.2 General Description

The LBC-486/586Plus is a small, high-performance, embeddable computer system on a single board. It integrates a number of popular I/O options including VGA, Ethernet, Solid-State Disk, and High-Density Parallel I/O. Four PC compatible serial ports are standard, as are the floppy, hard disk, and parallel printer interfaces. The LBC-Plus is populated with either a 100 MHz AMD DX4 processor or the AMD 5x85 133 MHz processor. Up to 32Mbytes of user installable SIMM memory is supported. An optional 256KB level two cache is also available. A full 16-bit PC/104 expansion bus is provided for further expansion to an entire industry of add-on peripherals including sound and speech modules, SCSI controllers, Analog I/O modules, and literally hundreds of other options available from WinSystems and a variety of vendors supporting the PC/104 standard. An onboard silicon disk array supports disks up to 2 megabytes in size and can utilize SRAM, PEROM or EPROM as the disk media. Boot capability is provided onboard and a set of utilities and drivers are provided to make the silicon disk based system very user friendly. Alternately, the M-Systems DiskOnChip FLASH modules may be populated, supporting disk sizes ranging from 1 Megabyte to 12 Megabytes.

1.3 Specifications

1.3.1 Electrical

Bus Interface : PC/104 8-Bit or 16-Bit expansion bus

System Clock : Jumper programmable from 4MHz to 50MHz

Interrupts : TTL Level input

VCC : +5V +/- 5% at 2.0A typical with a 133MHz 5X86 processor with 16M DRAM
1.8A typical with a 100MHz DX4 processor and 16M DRAM

VCC1 : +12V +/-5% (Not required. PC/104 Expansion, Flat-Panel, or AUI use only)

VCC2 : -12V +/-5% (Not required. PC/104 Expansion or FLat-Panel use only)

1.3.2 Memory

Addressing : 32 Megabyte addressing

BIOS ROM : 128K OTPROM

Memory SIMM Socket : 72-pin Fast Page Mode or EDO DRAM in sizes from 1M to 32M

SSD Memory : Two 32-pin JEDEC standard sockets support 4-Mbit SRAM, 4-Mbit PEROM, 4-Mbit EPROM, 8-Mbit EPROM, or one M-Systems 32-Pin DOC (DiskOnChip) module.

1.3.3 Mechanical

Dimensions : 5.75 X 8.0 X 0.60 inches (without PC/104 modules or cables)

PC-Board : FR4 Epoxy Glass with 4 signal layers and 2 power planes with screened component legend, and plated through holes.

Jumpers : 0.025" square posts on 0.10" centers

Connectors :

Multi I/O :	50 pin RN type IDH-50LP
COM3/COM4 :	20-pin RN type IDH-20LP
Floppy Disk :	34 pin RN type IDH-34-LP
Fixed Disk :	40 pin RN type IDH-40-LP
Digital I/O :	Two 50 pin RN type IDH-50-LP
10BaseT :	RJ45

Ethernet AUI :	16 pin RN type IDH-16-LP
CRT :	10 pin RN type IDH-10-LP
Flat Panel :	50 pin RN type IDH-50-LP
Power/Reset :	8 pin in-line Molex
PC/104 Bus :	64 Pin SAMTEC type ESQ-132-12-G-D 40 Pin SAMTEC type ESQ-120-12-G-D

1.3.4 Environmental :

Operating Temperature :	-40° to +70° C
Non-condensing relative humidity :	5% to 95%

2 LBC-PLUS Technical Reference

2.1 Introduction

This section of the manual is intended to provide sufficient information regarding the configuration and usage of the LBC-Plus board. WinSystems maintains a Technical Support group to help answer questions regarding configuration, usage, or programming of the board. For answers to questions not adequately addressed in this manual, contact Technical Support at (817) 274-7553 between 8AM and 5PM Central Time.

2.2 ALI 1487/1489 Chipset

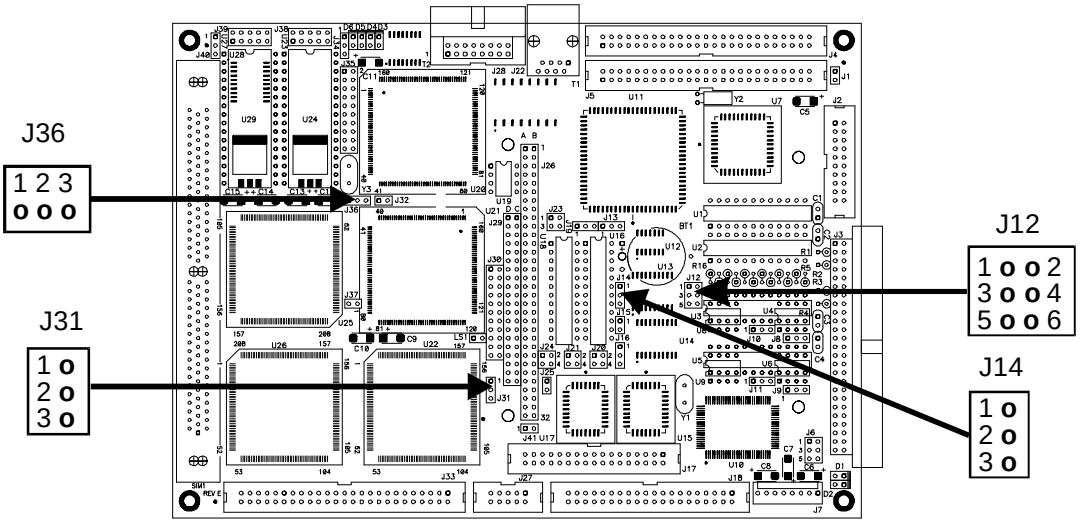
The LBC-Plus utilizes the ALI FINALI-486 Chipset which provides a highly-integrated, high-performance backbone for full PC/AT compatibility. The Chipset contains the logic for DRAM and bus state control as well as the standard complement of 'AT' class peripherals, including :

8 DMA Channels compatible with PC/AT 8237A DMA controllers
15 interrupt inputs compatible with master/slaved 8259 interrupt controllers
Three 8254 compatible timer/counter channels
A PC-AT compatible real time clock/calendar with CMOS RAM
A PCI BUS IDE interface
A PC/AT compatible keyboard interface

These functional units are 100% PC/AT compatible and are supported by the AWARD BIOS and setup. Users desiring to access these internal peripherals directly should refer to any manufacturers generic literature on the equivalent discrete component.

There are a number of internal registers within the Finali-486 chipset that are used by the BIOS for control and configuration. Refer to the I/O map in Appendix A for port usage to avoid conflicts when adding external I/O devices.

2.3 CPU Speed Selection



The LBC-Plus uses a Crystal controlled frequency synthesizer to control the CPU clock rate. The jumper block at J12 allows for the selection of any of 8 CPU base clock frequencies ranging from 8 MHz to 100 MHz.

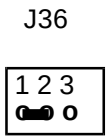
The table below gives all of the possible CPU clock speeds available by jumpering J12.

CPU Speed	J12 1-2	J12 3-4	J12 5-6
8 Mhz	ON	ON	ON
16 MHz	ON	ON	OFF
33 MHz	ON	OFF	ON
40 MHz	ON	OFF	OFF
50 MHz	OFF	ON	ON
66 MHz	OFF	ON	OFF
80 MHz	OFF	OFF	ON
100 MHz	OFF	OFF	OFF

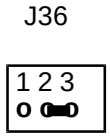
NOTE : The LBC-Plus board will be jumpered at the factory for the rated speed of the installed processor. Jumpering J12 to any speed in excess of the rated speed may result in CPU overheating, misoperation, and possible destruction of the CPU. Failures of CPUs which have been operated above their rated speed or temperature are not covered under the WinSystems standard product warranty.

2.3.1 Clock Multiplier Select

486DX4 and 5X86 processors actually run at a multiple of the base oscillator frequency. The jumper block at J36 allows selection of the multiplier as shown here :



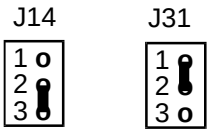
3X



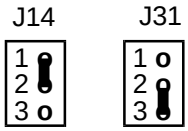
2X - 486
4X - 5X86

2.4 PCI Clock Select

The PCI bus clock source must be selected using jumper blocks at J14 and J31. The CPUCLK source may be selected any time the CPU base frequency is less than or equal to 33MHz. For any CPU base frequency in excess of 33MHz the CPUCLK/2 selection must be made.



CPUCLK



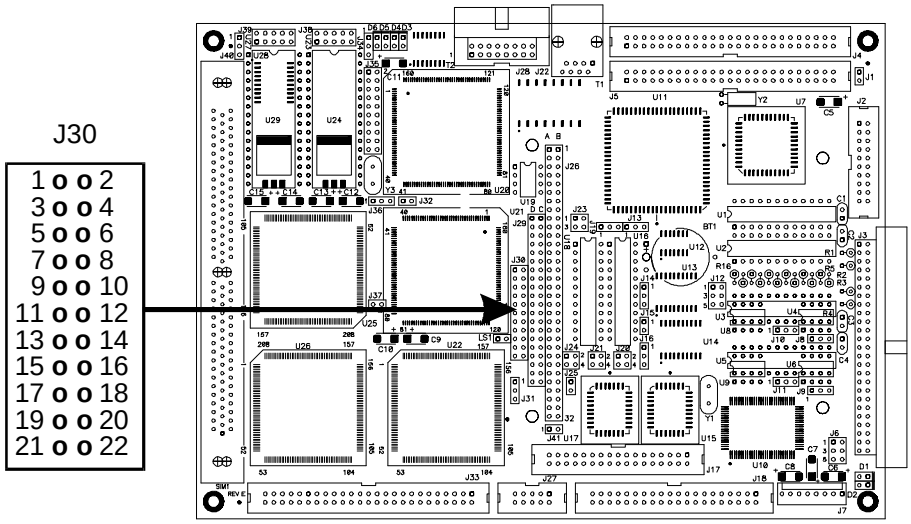
CPUCLK/2

2.5 Memory Installation

The LBC-Plus utilizes user installable 72-pin standard SIMMs. SIMM modules should be a minimum speed of 70nS and X32 architecture is preferred as there is no support for the parity bits provided by X36 bit modules. A single SIMM socket is provided which can support DRAM sizes from 1MB to 32MB.

Installation is accomplished with power off by angling the SIMM module approximately 30 degrees from vertical and inserting the fingers into the connector (It may be necessary to remove any device installed in the U27 socket.). The SIMM module is keyed slightly off-center and cannot be inserted backwards without extreme force. Once the fingers are in the socket, the module is then rotated to the vertical until the retaining clips snap into place. Removal is the reverse process. Pull the retaining clips outward and the SIMM module, once released, should rotate back to an appropriate removal angle.

2.6 Interrupt routing

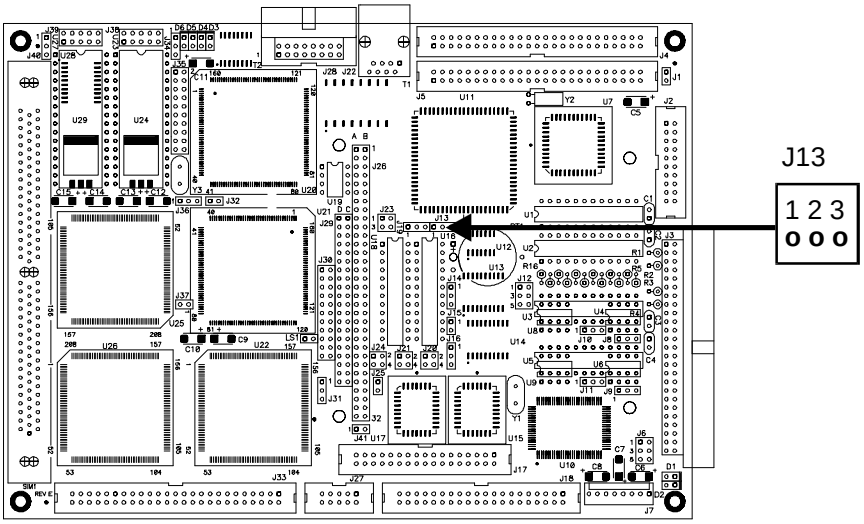


All interrupts on the LBC-Plus are routed to their respective PC/104 bus pins. Onboard peripherals, serial, parallel, and disk are routed to their typical usage interrupts using the jumper block at J30. This block allows disconnecting or rerouting of the onboard interrupts. The layout for the J30 header and the default jumper settings are shown here.

J30

COM4	1	2	IRQ9
ENET	3	4	IRQ10
COM3	5	6	IEQ11
COM4	7	8	IRQ12
WS16C48	9	10	IRQ15
IDE	11	12	IRQ14
LPT	13	14	IRQ7
FLOPPY	15	16	IRQ6
COM3	17	18	IRQ5
COM1	19	20	IRQ4
COM2	21	22	IRQ3

2.7 Real Time Clock/Calendar



The LBC-Plus contains an onboard Clock/Calendar within the ALI1487 chip. This clock is fully compatible with the MC146818A used in the original PC-AT computers. This clock has a number of features including periodic and alarm interrupt capabilities. In addition to the time and date keeping functions, the system configuration is kept within the CMOS RAM contained in the clock section. This RAM holds all of the setup information regarding hard and floppy disk types, video type, shadowing, wait states, etc. Refer to the section on the AWARD BIOS Setup for what is configured via the CMOS RAM.

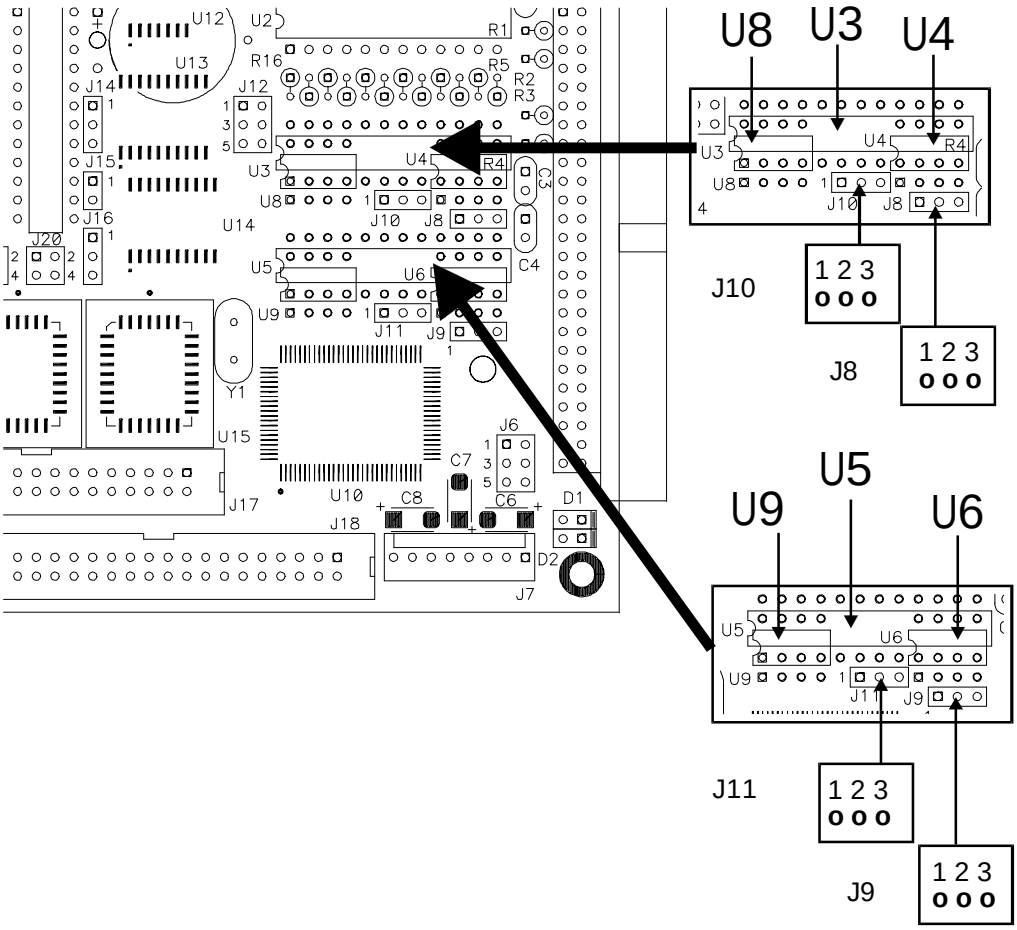
It may become necessary at some time to make the CMOS RAM forget its current configuration and to start fresh with factory defaults. This may be accomplished by removing power from the board . Then remove the jumper from pins 1-2 on J13 and place on pins 2-3 for 30 seconds. Replace the jumper on J13 pins 1-2, power-up, and reconfigure the CMOS settings as desired.

NOTE : J13 is the master battery enable jumper. Removing the jumper removes battery power from the entire board including the SSD array. Be sure that any data contained in battery backed SRAM is backed up before removing the battery jumper. J13 must be jumpered 2-3 in the clear position if a battery is not installed.

2.8 Keyboard Interface

The LBC-Plus contains an onboard PC-AT style keyboard controller. Keyboard connection is made through the Multi-I/O connector at J3. An adapter cable P/N CBL-162-1 is available from WinSystems to make ready access to all of the devices terminated at the Multi-I/O connector. Users desiring custom connections should refer to the Multi-I/O connector pin definitions given later in this manual.

2.9 Serial Interface



The LBC-Plus provides four 16550 compatible RS-232 serial ports at the following addresses :

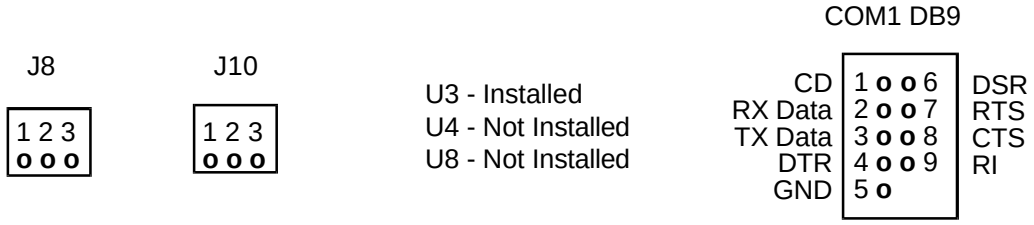
COM1	3F8H	at IRQ 4
COM2	2F8H	at IRQ 3
COM3	3E8H*	at IRQ 5**
COM4	2E8H*	at IRQ 9**

*COM ports 3 and 4 can be enabled or disabled individually via the jumper block at J23. When J23 pins 1-2 are jumpered, COM3 is enabled. When J23 pins 3-4 are jumpered, COM4 is enabled.

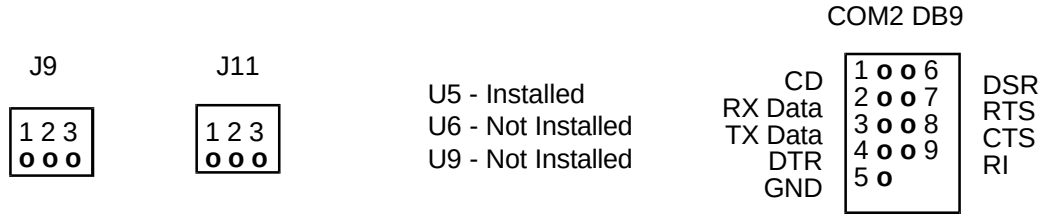
**The interrupts are not disconnected when COM3 or COM4 are disabled. Use the interrupt routing block described earlier to disconnect the default interrupts if desired.

The two primary serial ports, COM1 and COM2 are configurable for RS-422, RS-485 or J1708, with the addition of optional driver ICs. The configuration options for each of the supported modes are shown on the following pages.

COM1 - RS-232

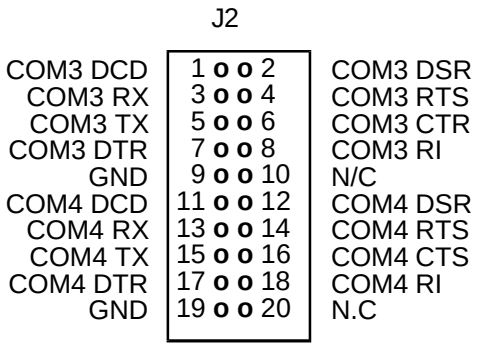


COM2 - RS-232



COM3/COM4 - RS-232

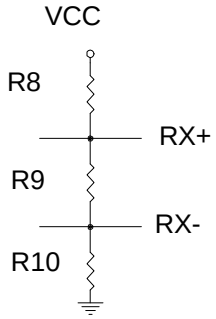
COM3 and COM4 are RS-232 only and are terminated at J2. An adapter cable is available from WinSystems (P/N CBL- 173-1), which adapts J2 to two standard DB9M connectors. The pin definitions for J2 are shown here :



2.9.1 RS-422 Mode Configuration

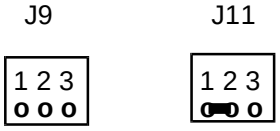
RS-422 levels are supported on both COM1 and COM2 with the installation of the optional “Chip Kit”, WinSystems part number CK-75176-2. This kit provides the driver ICs necessary for a single channel of RS-422. If two channels of RS-422 are required then two kits will be needed. RS-422 is a 4-wire point-to-point full-duplex interface allowing much longer cable runs than are possible with RS-232. The differential transmitter and receiver twisted pairs offer a high degree of noise immunity. RS-422 usually requires the lines be terminated at both ends. This termination can be accomplished either on the cable or by installing resisters on the board in locations reserved for them. The method for determining the correct resistor values is beyond the scope of this document but it is recommended that trial values of 100 ohms be used in all three locations at the receiver end. The following illustration shows the correct mode jumpering, driver IC installation, I/O connector pin definitions, and termination resistor locations for each of the channels when used in RS-422 mode.

COM1 - RS-422

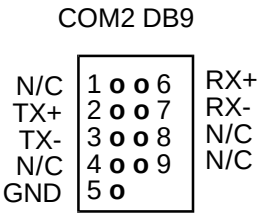


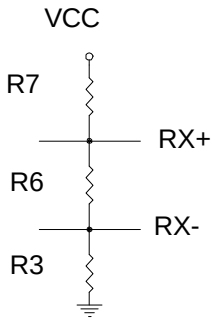
RS-422 NOTE : When used in RS-422 mode, the transmitter must be enabled by setting the RTS bit in the Modem Control Register (Bit1).

COM2 - RS-422



U5 - Not Installed
U6 - Installed
U9 - Installed



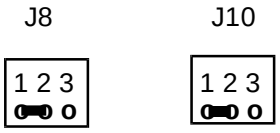


RS-422 NOTE : When used in RS-422 mode, the transmitter must be enabled by setting the RTS bit in the Modem Control Register (Bit1).

2.9.2 RS-485 Mode Configuration

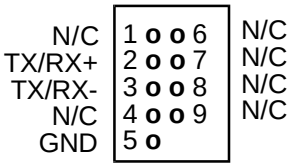
The RS-485 Multi-drop interface is supported on both channels with the installation of the optional "Chip Kit", WinSystems part number CK-75176-2. A single kit is sufficient to configure both channels for RS-485. RS-485 is a 2-wire multi-drop interface where only one station at a time talks (transmits) while all others listen (receive). RS-485 usually requires the twisted pair be terminated at each end of the run. The required termination values are dependent upon a number of factors including: line impedance, line length, etc. A good trial value is 100 ohms in all three resistor locations. The following illustrations show the correct jumpering, driver IC installation, I/O connector pinout, and termination resistor locations for each of the channels when used in RS-485 mode.

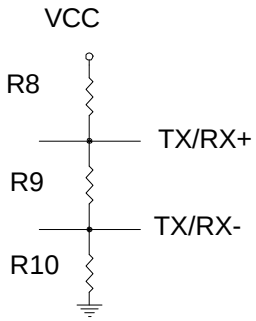
COM1 - RS-485



U3 - Not Installed
U4 - Installed
U8 - Not Installed

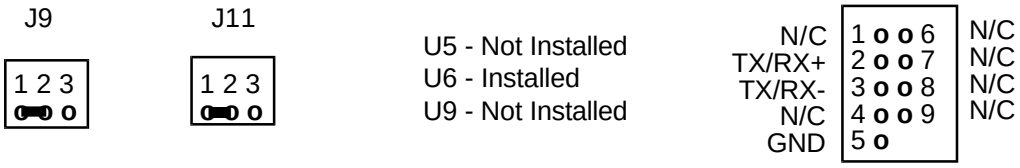
COM1 DB9

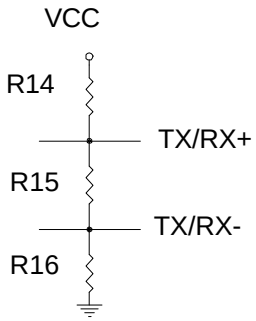




RS-485 NOTE : Because RS-485 uses a single twisted-pair, all transmitters are connected in parallel. Only one station at a time may transmit or have its transmitter enabled. The transmitter Enable/Disable is controlled in software using bit 1 in the Modem Control Register (RTS). When RTS is set, the transmitter is enabled, and when cleared (the normal state) the transmitter is disabled and the receiver is enabled. Note that it is necessary to allow some minimal settling time after enabling the transmitter before transmitting the first character. Likewise, following a transmission, it is necessary to be sure that all characters have been completely shifted out of the UART (Check Bit 6 in the Line Status Register) before disabling the transmitter to avoid chopping off the last character.

COM2 - RS-485



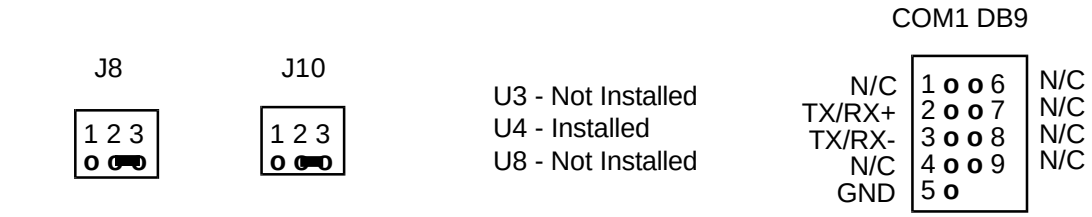


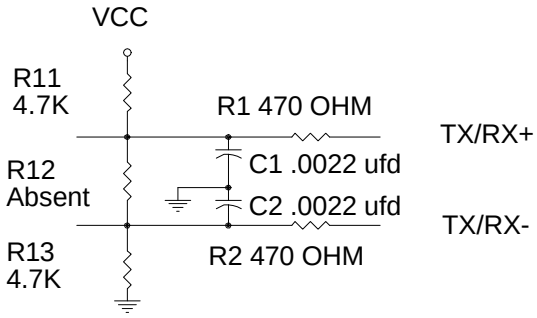
RS-485 NOTE : Because RS-485 uses a single twisted-pair, all transmitters are connected in parallel. Only one station at a time may transmit or have its transmitter enabled. The transmitter Enable/Disable is controlled in software using bit 1 in the Modem Control Register (RTS). When RTS is set, the transmitter is enabled, and when cleared (the normal state) the transmitter is disabled and the receiver is enabled. Note that it is necessary to allow some minimal settling time after enabling the transmitter before transmitting the first character. Likewise, following a transmission, it is necessary to be sure that all characters have been completely shifted out of the UART (Check Bit 6 in the Line Status Register) before disabling the transmitter to avoid chopping off the last character.

2.9.3 SAE J1708 Configuration

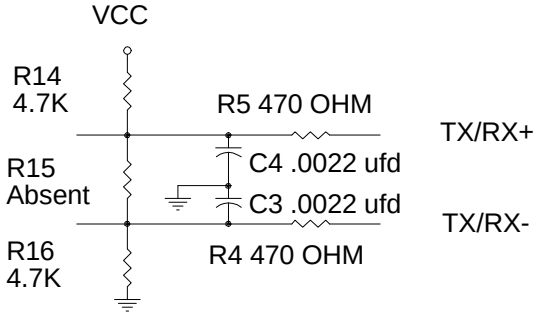
The Society of Automotive Engineers (SAE) J1708 interface is a variation of the RS-485 interface which is used for "Serial Data Communications between Microcomputer Systems in Heavy Duty Vehicle Applications". It is beyond the scope of this document to go into detail on the J1708 specification. The LBC-Plus may be user configured for J1708 by the addition of the CK-75176-2 "Chip Kit". One "Chip Kit" is sufficient to configure both channels for J1708. The illustrations that follow show the correct jumpering, driver IC installation, I/O connector pin definitions, and the termination network details for each of the channels when used in J1708 mode.

COM1 - J1708

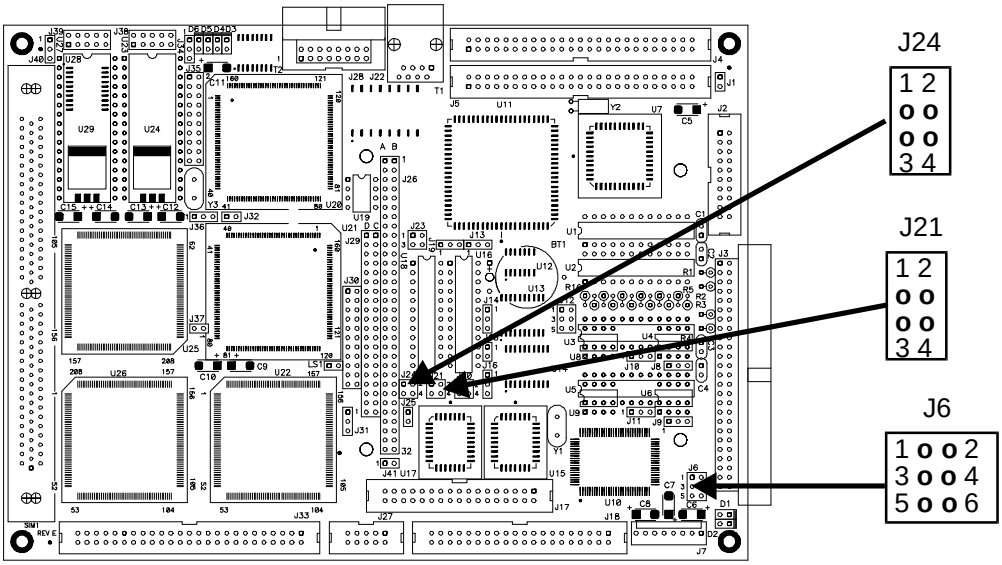




COM2 - J1708



2.10 Parallel Printer Port



The LBC-Plus supports a fully bi-directional parallel printer port capable of EPP and ECP operations. The parallel port is mapped at 278H and is terminated at the Multi-I/O connector J3. The pin definitions for the parallel port DB25 connector when using the CBL-162-1 cable are shown below :

STROBE	1 **o** **o** 14	AUTOFD
PD0	2 **o** **o** 15	ERROR
PD1	3 **o** **o** 16	INIT
PD2	4 **o** **o** 17	SLIN
PD3	5 **o** **o** 18	GND
PD4	6 **o** **o** 19	GND
PD5	7 **o** **o** 20	GND
PD6	8 **o** **o** 21	GND
PD7	9 **o** **o** 22	GND
ACK	10 **o** **o** 23	GND
BUSY	11 **o** **o** 24	GND
PE	12 **o** **o** 25	GND
SLCT	13 **o**	

2.10.1 Parallel Port Mode Selection

The parallel port mode is selected via the jumper block at J6 per the following table :

	SPP Mode	EPP Mode	ECP Mode	EPP/ECP Mode
J6 Jumpering	3-5 4-6	3-5 2-4	1-3 4-6	1-3 2-4

2.10.2 ECP DMA Configuration

When the parallel port is used in an ECP configuration, the jumper blocks at J21 and J24 are used to select the desired DMA channel as shown here :

2.11 Speaker/Sound Interface

The LBC-Plus utilizes a high-impedance piezo type device for audio output. BIOS beep codes, error signaling, or user defined tones can be presented via this device.

2.12 PC/104 Bus Interface

The LBC-Plus supports I/O expansion through the standard PC/104 connectors at J26 and J29. The LBC-Plus supports both 8-bit and 16-bit PC/104 modules. The PC/104 connector pin definitions are provided on the following page for reference purposes.

2.13 Floppy Disk Interface

The LBC-Plus supports up to 2 standard 3 1/2" or 5 1/4" PC compatible floppy disk drives. The drives are connected via the I/O connector at J17. Note that the interconnect cable to the drives is a standard floppy I/O cable used on desk-top PCs. The cable must have the twisted section prior to the drive A position. The pin definitions for the J17 connector are shown here :

J17

GND	1	o o	2	RPM/LC
GND	3	o o	4	N/C
GND	5	o o	6	N/C
GND	7	o o	8	INDEX
GND	9	o o	10	MTR0
GND	11	o o	12	DRV1
GND	13	o o	14	DRV0
GND	15	o o	16	MTR1
GND	17	o o	18	DIR
GND	19	o o	20	STEP
GND	21	o o	22	WDATA
GND	23	o o	24	WGATE
GND	25	o o	26	TRK0
GND	27	o o	28	WPRT
GND	29	o o	30	RDATA
GND	31	o o	32	HDSEL
GND	33	o o	34	DSKCHG

J26

GND	B1	A1	IOCHK
RESET	B2	A2	BD7
+5V	B3	A3	BD6
IRQ9	B4	A4	BD5
-5V	B5	A5	BD4
DRQ2	B6	A6	BD3
-12V	B7	A7	BD2
0WS	B8	A8	BD1
+12V	B9	A9	BD0
GND	B10	A10	IOCHRDY
MEMW	B11	A11	AEN
MEMR	B12	A12	SA19
IOW	B13	A13	SA18
IOR	B14	A14	SA17
DACK3	B15	A15	SA16
DRQ3	B16	A16	SA15
DACK1	B17	A17	SA14
DRQ1	B18	A18	SA13
REFRESH	B19	A19	SA12
SYSCLK	B20	A20	SA11
IRQ7	B21	A21	SA10
IRQ6	B22	A22	SA9
IRQ5	B23	A23	SA8
IRQ4	B24	A24	SA7
IRQ3	B25	A25	SA6
DACK2	B26	A26	SA5
TC	B27	A27	SA4
BALE	B28	A28	SA3
+5V	B29	A29	SA2
OSC	B30	A30	SA1
GND	B31	A31	SA0
GND	B32	A32	GND

J29

GND	C0	D0	GND
SBHE	C1	D1	MEMCS16
LA23	C2	D2	IOCS16
LA22	C3	D3	IRQ10
LA21	C4	D4	IRQ11
LA20	C5	D5	IRQ12
LA19	C6	D6	IRQ15
LA18	C7	D7	IRQ14
LA17	C8	D8	DACK0
MEMR	C9	D9	DRQ0
MEMW	C10	D10	DACK5
SD8	C11	D11	DRQ5
SD9	C12	D12	DACK6
SD10	C13	D13	DRQ6
SD11	C14	D14	DACK7
SD12	C15	D15	DRQ7
SD13	C16	D16	VCC
SD14	C17	D17	MASTER
SD15	C18	D18	GND
KEY	C19	D19	GND

2.14 IDE Hard Disk Interface

The LBC-Plus supports standard IDE fixed disks through the I/O connector at J18. A red activity LED is present at D1. The pin definitions for J18 are shown here :

J18

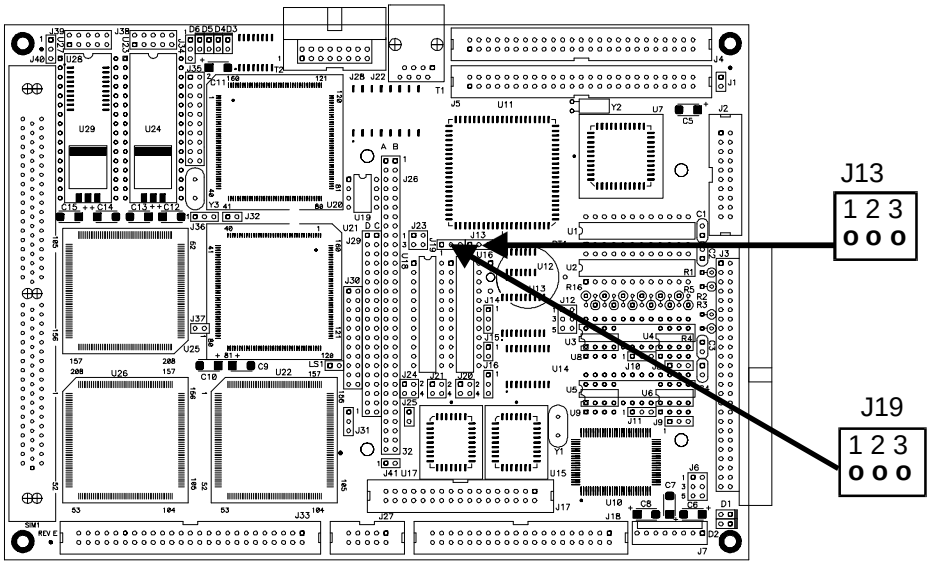
RESET	1	o o	2	GND
D7	3	o o	4	D8
D6	5	o o	6	D9
D5	7	o o	8	D10
D4	9	o o	10	D11
D3	11	o o	12	D12
D2	13	o o	14	D13
D1	15	o o	16	D14
D0	17	o o	18	D15
GND	19	o o	20	N/C
GND	21	o o	22	GND
IOW	23	o o	24	GND
IOR	25	o o	26	GND
N/C	27	o o	28	ALE
N/C	29	o o	30	GND
INTRQ	31	o o	32	IOCS16
A1	33	o o	34	N/C
A0	35	o o	36	A2
HDCS0	37	o o	38	HDCS1
N/C	39	o o	40	GND

2.15 Watchdog Timer Configuration

The LBC-Plus board features a power-on voltage detect and power-down/power brown-out reset circuit to protect memory and I/O from faulty CPU operation during periods of illegal voltage levels. This supervisor circuitry also features a watchdog timer which can be used to guard against software lockups. An internal timer with a period of 1.5 seconds will, when enabled, reset the CPU if the watchdog has not been serviced within the allotted time. There are three watchdog modes available on the LBC-Plus. With no jumper installed on J19, the watchdog is totally disabled and can never reset the CPU. When J19 is jumpered on pins 2-3, the watchdog circuit is permanently enabled and timing begins immediately with power-on. This mode is NOT compatible with the AWARD BIOS or with MS-DOS but is available for directly embedded code that takes the place of the BIOS. The watchdog must be accessed every 1.5 seconds or a reset will occur. Petting in this mode is accomplished by writing to I/O port 1EFH with any value.

An alternate mode of operation is via software enable/disable control. This mode is set by jumpering J19 pins 1-2. In this mode the watchdog timer powers-up disabled and must be enabled in software before timing will begin. Enabling is accomplished by writing a 1 to I/O port 1EEH. Writing a 0 to I/O port 1EEH will disable the watchdog. After enabling, petting may be accomplished by writing any value to I/O port 1EFH at least every 1.5 seconds or a reset will occur. This mode of operation can be used with the BIOS or DOS provided that the watchdog is disabled before making any extensive BIOS or DOS calls, especially video or Disk IO calls which could exceed the 1.5 seconds allowed. The draw-

back to this mode is that a lockup during the time the watchdog is disabled will not allow for auto-recovery and will require an external reset.



2.16 Status LED

A green LED is populated on the board at D2 which can be used for any application specific purpose. The LED can be turned on in software by writing a 1 to I/O port 1EDH. The LED can be turned off by writing a 0 to 1EDH.

2.17 Battery Select Control

An onboard 200mAh nominal capacity, lithium coin-cell battery is provided for the CMOS Clock/Calendar and for battery backing-up Solid State Disk SRAMs. A master battery enable jumper is provided at J13. When J13 is jumpered pins 1-2, battery power is supplied to the Clock/Calendar and to the individual jumper blocks for battery backup of SSD SRAMs. When J13 is jumpered pins 2-3, the battery is totally disconnected and no current will be drawn from it. Battery life is highly dependent upon duty cycle as there is no current drawn from the battery when +5 volts is applied to the board. Both storage and operational temperatures play a prominent factor in battery life. High temperatures will shorten battery life significantly. J13 must be jumpered 2-3 in the clear position if a battery is not installed.

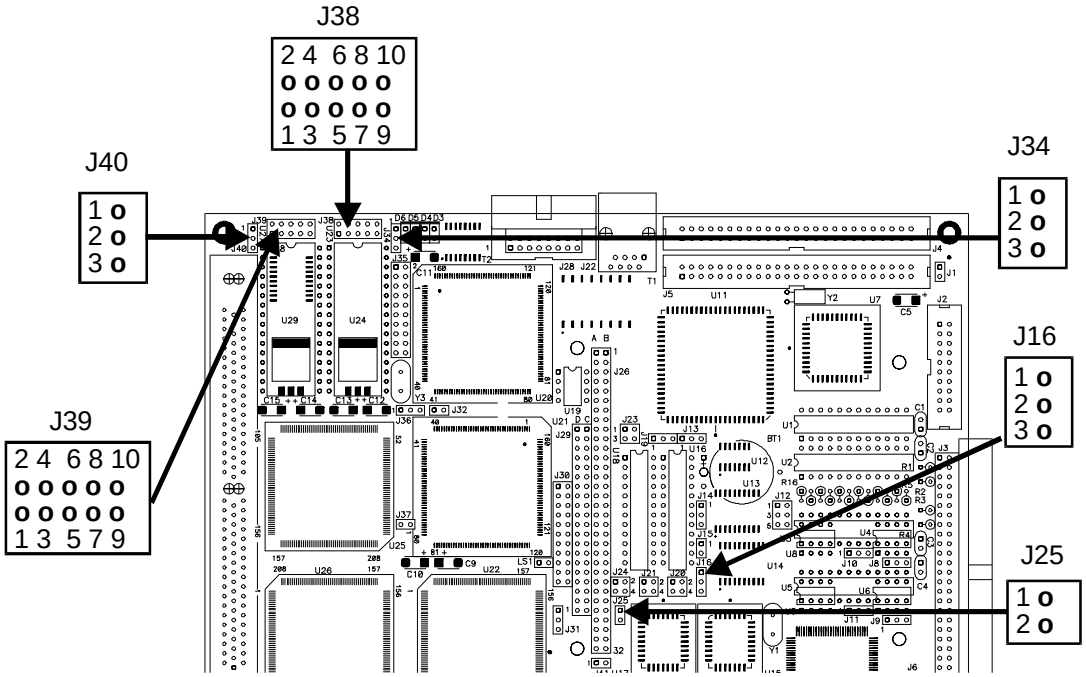
2.18 Power/Reset Connection

Power is supplied to the LBC-Plus via the connector at J7. The pin definitions for J7 are given below. An optional normally-open push-button-reset switch may also be connected to J7 between PBRESET* and ground.

J7

8 o	PBRESET*
7 o	GND
6 o	GND
5 o	GND
4 o	+5V
3 o	+5V
2 o	+12V
1 o	-12V

2.19 Silicon Disk Configuration

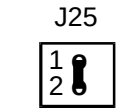


The LBC-Plus supports the use of EPROM, PEROM (Flash), SRAM, and the M-Systems DiskOnChip (DOC) devices to be used as Solid State Disk (SSD) drives. Section 4 of this manual provides the necessary information for the generation and usage of the Silicon drives. This section documents the required hardware configurations for the various type of devices. Two 32-pin JEDEC memory sockets at U27 and U23 are used to contain the RAM, ROM,Flash, or DOC devices used for the disk. The silicon

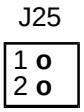
disk array is memory mapped into a 16k byte hole at segment E400H and has an I/O control register at 1ECH.

2.19.1 Silicon Disk Mode

There are two basic modes of Silicon Disk operation on the LBC-Plus. The first uses the onboard BIOS extension and supports the use of 512K or 1M EPROMS, 512K SRAMS, or 512K ATMEL Flash Devices. The second mode uses the M-Systems DiskOnChip device. The mode is controlled via the jumper block at J25 as shown here :



DOC Mode



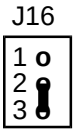
SSD Mode

NOTE : Jumpering for DOC mode with EPROMs, RAMs, of Flash devices installed effectively acts as a disable to the Solid State Disk and similarly when a DOC device is installed and the jumper is selected for standard devices the DOC is disabled.

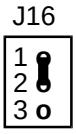
IMPORTANT NOTE : To insure Windows 95 compatibility, J25 must be jumpered 1-2.

2.19.2 Device Size Selection

The onboard Solid State Disk array supports either 512K EPROMs, SRAMs or FLASH device or 1M EPROM devices. The device size selection is made at J16 as shown here :



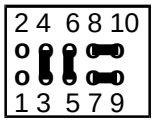
512K or DOC Device



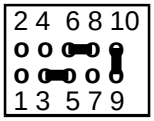
1M Devices

2.19.3 Device Type Selection

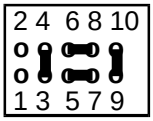
Each of the devices in the array has an individual device type jumper block at the device socket. J39 sets the device type for U27 and J38 sets the device type for U23. The supported device type jumperings are shown here :



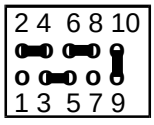
512K X 8
SRAM



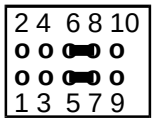
512K X 8
EPROM



512K X 8
PEROM



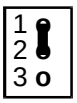
1 MEG X 8
EPROM



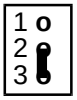
DOC
DEVICE

2.19.4 Battery Backup Selection

When using SRAM devices and nonvolatile operation is desired, battery backup can be selected on a socket-by-socket basis. J40 for U27 and J34 for U23. The illustration below shows the jumpering for battery backup or standard operation.



Battery Backup
Enabled



Battery Backup
Disabled

NOTE : Having the jumper(s) selected for battery backup when using other than low-power-standby SRAMs (such as with EPROMs, or PEROMs) will result in the quick draining of the onboard battery.

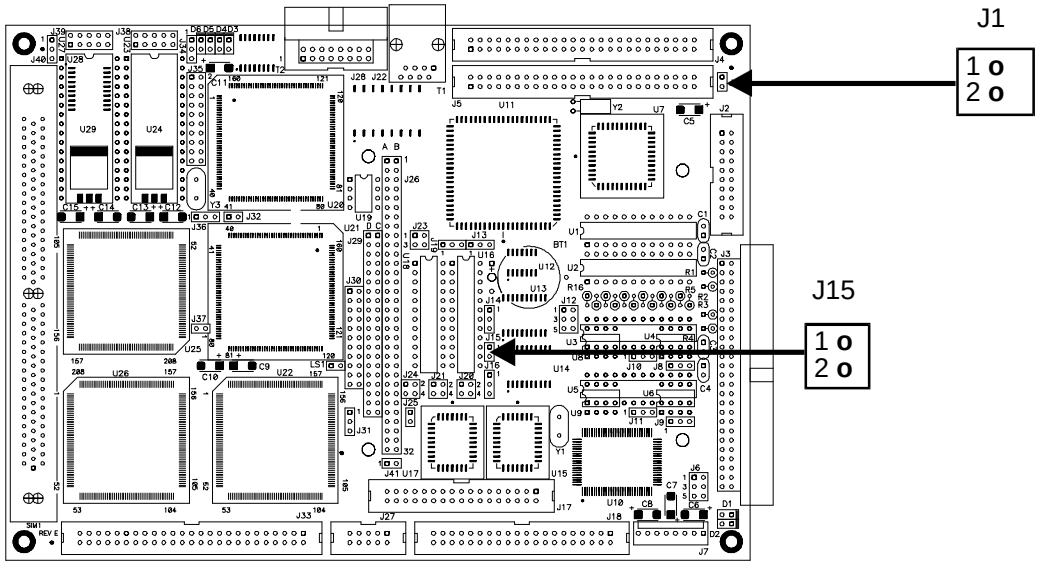
2.19.5 Silicon Disk Notes

1. When installing devices, U27 is the first device in the array and must always contain the first device of a bootable disk.

2. The DiskOnChip option must use the socket at U23. When a DOC is installed, U27 is available as a Secondary Silicon Disk device. See section 4.4 for more information.

2.20 Parallel I/O

The LBC-Plus utilizes the WinSystems WS16C48 ASIC high-density I/O chip mapped at a base address of 120H. The first 24 lines are capable of fully latched event sensing with sense polarity being software programmable. Two 50-pin connectors allow for easy mating with industry standard I/O racks.



2.20.1 Parallel I/O Enable

The parallel features of the LBC-Plus can be enabled or disabled using the jumper block at J15. When J15 is jumpered the parallel I/O is enabled at I/O address 120H. When J15 is open the 16 addresses starting at I/O address 120H are free for use by other devices.

2.20.2 Parallel I/O Connectors

The 48 lines of parallel I/O are terminated through two 50-pin connectors at J4 and J5. The J4 connector handles I/O ports 0-2 while J5 handles ports 3-5. The pin definitions for J4 and J5 are shown on the following page.

J4

Port 2 Bit 7	1 o o 2	GND
Port 2 Bit 6	3 o o 4	GND
Port 2 Bit 5	5 o o 6	GND
Port 2 Bit 4	7 o o 8	GND
Port 2 Bit 3	9 o o 10	GND
Port 2 Bit 2	11 o o 12	GND
Port 2 Bit 1	13 o o 14	GND
Port 2 Bit 0	15 o o 16	GND
Port 1 Bit 7	17 o o 18	GND
Port 1 Bit 6	19 o o 20	GND
Port 1 Bit 5	21 o o 22	GND
Port 1 Bit 4	23 o o 24	GND
Port 1 Bit 3	25 o o 26	GND
Port 1 Bit 2	27 o o 28	GND
Port 1 Bit 1	29 o o 30	GND
Port 1 Bit 0	31 o o 32	GND
Port 0 Bit 7	33 o o 34	GND
Port 0 Bit 6	35 o o 36	GND
Port 0 Bit 5	37 o o 38	GND
Port 0 Bit 4	39 o o 40	GND
Port 0 Bit 3	41 o o 42	GND
Port 0 Bit 2	43 o o 44	GND
Port 0 Bit 1	45 o o 46	GND
Port 0 Bit 0	47 o o 48	GND
+5V	49 o o 50	GND

J5

Port 5 Bit 7	1 o o 2	GND
Port 5 Bit 6	3 o o 4	GND
Port 5 Bit 5	5 o o 6	GND
Port 5 Bit 4	7 o o 8	GND
Port 5 Bit 3	9 o o 10	GND
Port 5 Bit 2	11 o o 12	GND
Port 5 Bit 1	13 o o 14	GND
Port 5 Bit 0	15 o o 16	GND
Port 4 Bit 7	17 o o 18	GND
Port 4 Bit 6	19 o o 20	GND
Port 4 Bit 5	21 o o 22	GND
Port 4 Bit 4	23 o o 24	GND
Port 4 Bit 3	25 o o 26	GND
Port 4 Bit 2	27 o o 28	GND
Port 4 Bit 1	29 o o 30	GND
Port 4 Bit 0	31 o o 32	GND
Port 3 Bit 7	33 o o 34	GND
Port 3 Bit 6	35 o o 36	GND
Port 3 Bit 5	37 o o 38	GND
Port 3 Bit 4	39 o o 40	GND
Port 3 Bit 3	41 o o 42	GND
Port 3 Bit 2	43 o o 44	GND
Port 3 Bit 1	45 o o 46	GND
Port 3 Bit 0	47 o o 48	GND
+5V	49 o o 50	GND

2.20.3 Parallel I/O VCC Enable

The I/O connectors can provide +5 volts to an I/O rack or for miscellaneous purposes by jumpering J1. When J1 is jumpered +5 volts is provided at pin 49 of both J4 and J5. It the user's responsibility to limit current to a safe value (less than 1A) to avoid damaging the CPU board.

2.20.4 WS16C48 Register Definitions

The LBC-Plus uses the WinSystems' exclusive ASIC device, the WS16C48. This device provides 48 lines of digital I/O. There are 17 unique registers within the WS16C48. The following table summarizes the registers and the text that follows provides details on each of the internal registers.

I/O Address Offset	Page 0	Page 1	Page 2	Page 3
00H	Port 0 I/O	Port 0 I/O	Port 0 I/O	Port 0 I/O
01H	Port 1 I/O	Port 1 I/O	Port 1 I/O	Port 1 I/O
02H	Port 2 I/O	Port 2 I/O	Port 2 I/O	Port 2 I/O
03H	Port 3 I/O	Port 3 I/O	Port 3 I/O	Port 3 I/O
04H	Port 4 I/O	Port 4 I/O	Port 4 I/O	Port 4 I/O
05H	Port 5 I/O	Port 5 I/O	Port 5 I/O	Port 5 I/O
06H	Int_Pending	Int_Pending	Int_Pending	Int_Pending
07H	Page/Lock	Page/Lock	Page/Lock	Page/Lock
08H	N/A	Pol_0	Enab_0	Int_ID0
09H	N/A	Pol_1	Enab_1	Int_ID1
0AH	N/A	Pol_2	Enab_2	Int_ID2

Register Details

Port 0-5 I/O - Each I/O bit in each of the 6 ports can be individually programmed for input or output. Writing a '0' to a bit position causes the corresponding output pin to go to a High-Impedance state (pulled high by external 10K ohm resistors). This allows it to be used as an input. When used in the input mode, a read reflects the inverted state of the I/O pin, such that a high on the pin will read as a '0' in the register. Writing a '1' to a bit position causes the output pin to sink current (up to 12mA), effectively pulling it low.

INT_PENDING - This read-only register reflects the combined state of the INT_ID0 through INT_ID2 registers. When any of the lower 3 bits are set, it indicates that an interrupt is pending on the I/O port corresponding to the bit position(s) that are set. Reading this register allows an Interrupt Service Routine to quickly determine if any interrupts are pending and which I/O port has a pending interrupt.

PAGE/LOCK - This register serves two purposes. The upper two bits select the register page in use as shown here :

D7	D6	Page
0	0	Page 0
0	1	Page 1
1	0	Page 2
1	1	Page 3

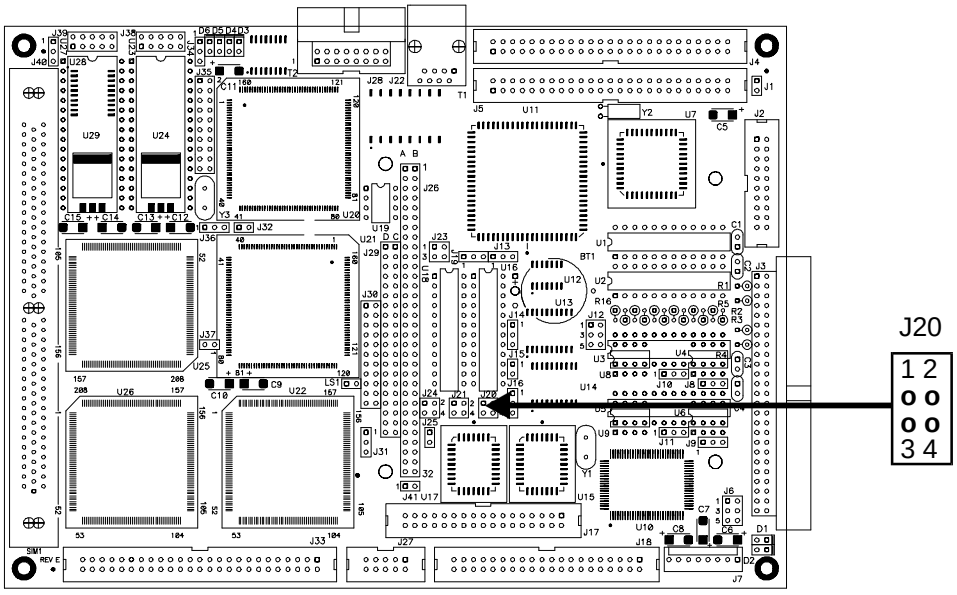
Bits 5-0 allow for locking the I/O ports. A '1' written to the I/O port position will prohibit further writes to the corresponding I/O port.

POL0-POL2 - These registers are accessible when page 1 is selected. They allow interrupt polarity selection on a port-by-port and bit-by-bit basis. Writing a '1' to a bit position selects the rising edge detection interrupts while writing a '0' to a bit position selects falling edge detection interrupts.

ENAB0-ENAB2 - These registers are accessible when page 2 is selected. They allow for port-by-port and bit-by-bit enabling of the edge detection interrupts. When set to a '1' the edge detection interrupt is enabled for the corresponding port and bit. When cleared to a '0' the bit's edge detection interrupt is disabled. Note that this register can be used to individually clear a pending interrupt by disabling and reenabling the pending interrupt.

INT_ID0 - INT_ID2 - These registers are accessible when page 3 is selected. They are used to identify currently pending edge interrupts. A bit when read as a '1' indicates that an edge of the polarity programmed into the corresponding polarity register has been recognized. Note that a write to this register (value ignored) clears ALL of the pending interrupts in this register.

2.21 VGA Configuration



2.21.1 Introduction

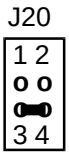
The LBC-Plus uses a third generation CRT/Flat panel VGA controller. It supports standard VGA output as well as a variety of Flat Panel Displays using optional Flat Panel Adapter (FPA) modules. The video on the LBC-Plus uses the Chips and Technologies 6554X series of high performance VGA controllers. The C&T controller supports standard and Super-VGA modes as well as Color and Monochrome panels with 8,9,12,15,16,18, and 24-bit interfaces. WinSystems provides flat panel support through a series of FPA (Flat Panel Adapter) modules. Contact your WinSystems Applications Engineer for the most current list of available FPAs and supported panels.

Details regarding interfacing to specific Flat Panels is not provided in this manual but should be referenced in the documentation accompanying the FPA module. Attempted connection to any flat panel not directly supported by a WinSystems FPA module is at the user's risk and extreme care should be exercised to avoid damaging or destroying the panel.

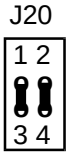
HAZARD WARNING : LCD panels can require a high voltage for the panel backlight. This high-frequency voltage can exceed 1000 volts and can present a shock hazard. Care should be taken when wiring or handling the inverter output. To avoid danger of shock and to avoid damaging fragile and expensive panels, make all connection changes with power removed.

2.21.2 VGA BIOS ROM Type Selection

The LBC-Plus comes standard with a video BIOS expansion ROM populated at U15. Various ROM sizes can be used to support a variety of flat panel configurations each needing its own BIOS image. The FPA adapter modules, when connected to J33, automatically select the correct BIOS image for the panel family the FPA supports. The factory will ordinarily configure the BIOS ROM for the size provided but the illustration below shows the proper jumpering of J20 for the supported ROM sizes.



27C010
27C020
27C040



27C080

2.21.3 CRT Output Connection

Video output to a standard VGA monitor is made via the connector at J27. An adapter cable part number CBL-207-1 is available from WinSystems to adapt from J27 to the standard DB15 VGA connector. The pin definitions for the J27 connector are shown here :

J27

RED	1 o o 2	GND
GREEN	3 o o 4	GND
BLUE	5 o o 6	GND
HSYNC	7 o o 8	GND
VSYNC	9 o o 10	GND

2.21.4 Flat Panel Output Connection

Connection to all flat panels is made via the 50-pin connector at J33. This connector is cabled to the appropriate FPA (Flat Panel Adapter) module which then breaks out the necessary cabling for attachment to the panel itself. The FPA module also supplies any special controls that may be needed for the panel. Refer to the FPA documentation for specific hook-up instructions. The pin definitions for J33 are shown here :

J33

SW0	1 o o 2	SW1
SW2	3 o o 4	SW3
P23	5 o o 6	P22
P21	7 o o 8	P20
P19	9 o o 10	P18
GND	11 o o 12	GND
P17	13 o o 14	P16
P15	15 o o 16	P14
P13	17 o o 18	P12
GND	19 o o 20	GND
P11	21 o o 22	P10
P9	23 o o 24	P8
P7	25 o o 26	P6
GND	27 o o 28	GND
P5	29 o o 30	P4
P3	31 o o 32	P2
P1	33 o o 34	P0
GND	35 o o 36	GND
SHFCLK	37 o o 38	LP
FLM	39 o o 40	M
ENVCC	41 o o 42	ENBKL
ENVEE	43 o o 44	PHS
PVS	45 o o 46	-12V
+12V	47 o o 48	+12V
VCC	49 o o 50	VCC

2.21.5 Video Mode Tables

The LBC-Plus video section supports a number of standard and extended VGA modes. The following tables extracted from the C&T 65540/65545 databook show the video modes along with the required amount of RAM.

Standard Video Modes - VGA Standard

Mode # (Hex)	Display Mode	Colors	Text Display	Font Size	Pixel Resolution	Dot Clock (MHz)	Horizontal Frequency (KHz)	Vertical Frequency (Hz)	Video Memory	CRT CODE
0+, 1+	Text	16	40 X 25 40 X 25 40 X 25	9 X 16 8 X 14 8 X 8	360 X 400 320 X 350 320 X 200	28.322 25.175 25.175	31.5	70	256KB	A,B,C
2+, 3+	Text	16	80 X 25 80 X 25 80 X 25	9 X 16 8 X 14 8 X 8	720 X 400 640 X 350 620 X 200	28.322 25.175 25.175	31.5	70	256KB	A,B,C
4	Graphics	4	40 X 25	8 X 8	320 X 200	25.175	31.5	70	256KB	A,B,C
5	Graphics	4	40 X 25	8 X 8	320 X 200	15.175	31.5	70	256KB	A,B,C
6	Graphics	2	80 X 25	8 X 8	640 X 200	25.175	31.5	70	256KB	A,B,C
7+	Text	Mono	80 X25 80 X 25 80 X 25	9 X 16 9 X 14 9 X 8	720 X 400 720 X 350 720 X 350	28.322	31.5	70	256KB	A,B,C
D	Planar	16	40 X 25	8 X 8	320 X 200	25.175	31.5	70	256KB	A,B,C
E	Planar	16	80 X 25	8 X 8	640 X 200	25.175	31.5	70	256KB	A,B,C
F	Planar	Mono	80 X 25	8 X 14	640 X 350	25.175	31.5	70	256KB	A,B,C
10	Planar	16	80 X 25	8 X 14	640 X 350	25.175	31.5	70	256KB	A,B,C
11	Planar	2	80 X 30	8 X 16	640 X 480	25/175	31.5	60	256KB	A,B,C
12	Planar	16	80 X 30	8 X 16	640 X 480	25.175	31.5	60	256KB	A,B,C
13	Packed Pixel	256	40 X 25	8 X 8	320 X 200	25.175	31.5	70	256KB	A,B,C

CRT Codes

A - PS/2 Fixed Frequency analog CRT or equivalent (31.5/35.5 Khz Horizontal Frequency Specification)
B - Multi-Frequency CRT Monitor (37.5 Khz minimum Horizontal Frequency Specification) (NEC MultiSync 3D or equivalent)
C - Multi-Frequency High-Performance CRT Monitor (48.5 KHZ minimum Horizontal Frequency Specification) MultiSyc 5D or equivalent

Extended Resolution Modes

Mode # (Hex)	Display Mode	Colors	Text Display	Font Size	Pixel Resolution	Dot Clock (MHz)	Horizontal Frequency (KHz)	Vertical Frequency (Hz)	Video Memory	CRT CODE
20	4-Bit Linear	16	80 X 30	8 X 16	640 X 480	25.175	31.5	60	512KB	A,B,C
22	4-Bit Linear	16	100 X 37	8 X 16	800 X 600	40.00	37.5	60	512KB	B,C
24	4-Bit Linear	16	128 X 48	8 X 16	1024 X 768	65.00	48.5	60	512KB	C
24I						44.90	35.5	43	512KB	B,C
30	8-Bit Linear	256	80 X 30	8 X 16	640 X 480	25/175	31.5	60	512K	A,B,C
32	8-Bit Linear	256	100 X 37	8 X 16	800 X 600	40.00	37.5	60	512KB	B,C
34	8-Bit Linear	256	128 X 48	8 X 16	1024 X 768	65.00	48.5	60	1MB	C
34I						44.90	35.5	43	1MB	B,C
40	16-Bit Linear	32K	80 X 30	8 X 16	640 X 480	50.350	31.5	60	1MB	A,B,C
41	16-Bit Linear	64K	80 X 30	8 X 16	640 X 480	50.350	31.5	60	1MB	A,B,C
50	24-Bit Linear	16M	80 X 30	8 X 16	640 X 480	65.00	27.1	51.6	1MB	B.C
60	Text	16	132 X 25	8 X 16	1056 X 400	40.00	30.5	68	256KB	A,B,C
61	Text	16	132 X 50	8 X 16	1056 X 400	40.00	30.5	68	256KB	A,B,C
6A,70	Planar	16	100 X 37	8 X 16	800 X 600	40.00	38.0	60	256KB	B,C
72,75	Planar	16	128 X 48	8 X 16	1024 X 768	65.00	48.5	60	512KB	C
72,75I						44.90	35.5	43	512KB	B,C
78	Packed Pixel	16	80 X 25	8 X 16	640 X 400	25.175	31.5	70	256KB	A,B,C
79	Packed Pixel	256	80 X 30	8 X 16	640 X 480	25.175	31.5	60	512KB	A,B,C
7C	Packed Pixel	256	100 X 37	8 X 16	800 X 600	40.00	37.5	60	512KB	B,C
7E	Packed Pixel	256	128 X 48	8 X 16	1024 X 768	65.00	48.5	60	1MB	C
7EI	Packed Pixel					44.90	35.5	43	1MB	B,C

Support for the modes above is included directly in the BIOS. The 'I' in the mode # column indicates "interlaced"

CRT Codes

A - PS/2 Fixed frequency analog CRT or equivalent (31.5/35.5 Khz Horizontal Frequency Specification)
B - Multi-Frequency CRT Monitor (37.5 Khz minimum Horizontal Frequency Specification) (NEC MultiSync 3D or equivalent)
C - Multi-Frequency High-Performance CRT Monitor (48.5 KHZ minimum Horizontal Frequency Specification) MultiSyc 5D or equivalent

High Refresh Modes

Mode # (Hex)	Display Mode	Colors	Text Display	Font Size	Pixel Resolution	Dot Clock (MHz)	Horizontal Frequency (KHz)	Vertical Frequency (Hz)	Video Memory	CRT CODE
12	Planar	16	80 X 30	8 X 16	640 X 480	31.50	37.5	75	256KB	B,C
30	8-Bit Linear	256	80 X 30	8 X 16	640 X 480	31.50	37.5	75	256KB	C
79	Packed Pixel	256	80 X 30	8 X 16	640 X 480	31.50	37.5	75	512KB	C
6A,70	Planar	16	100 X 37	8 X 16	800 X 600	49.50	46.9	75	512KB	C
32	8-Bit Linear	256	100 X 37	8 X 16	800 X 600	49.50	46.9	75	1MB	C
7C	Packed Pixel	256	100 X 37	8 X 16	800 X 600	49.50	46.9	75	1MB	C

2.22 Ethernet Configuration

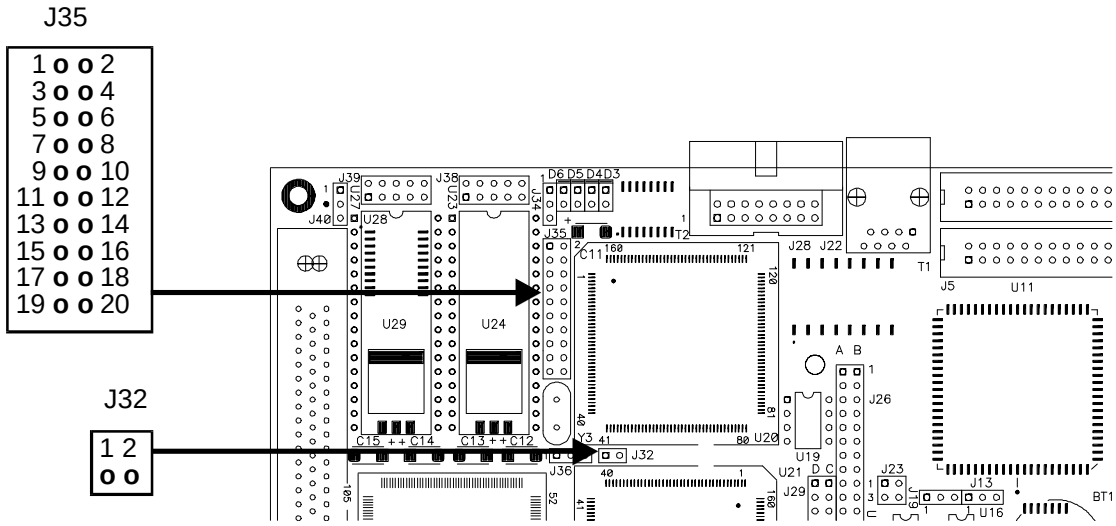
The Ethernet section of the LBC-Plus uses the National 83905 AT/LANTIC Local Area Network Twisted-Pair Interface controller. The AT/LANTIC controller is a CMOS/VLSI device used in the implementation of CSMA/CD local area networks. Supported network interfaces include 10BASE5 or 10BASE2 via an external transceiver connected to the AUI port, and twisted pair Ethernet (10BASE-T) using the onboard transceiver. The AT/LANTIC provides the Ethernet Media Access Control (MAC), Encode-Decode (ENDEC) with an AUI interface, and 10BASE-T transceiver functions in accordance with IEEE 802.3 standards.

This functional block incorporates the receiver, transmitter, collision heartbeat. loopback jabber, and link integrity blocks as defined in the standard. The transceiver when combined with the equalization resistors, transmit/receive filters, and pulse transformers provide a complete physical interface from the AT/LANTIC Controller ENDEC module and the twisted-pair medium.

The integrated ENDEC module allows the Manchester encoding and decoding via a differential transceiver and phase-lock-loop decoder at 10 Mbit/sec. Also included are a collision detect translator and diagnostic loopback capability. The ENDEC module interfaces directly to the transceiver module and provides full IEEE compliant AUI (Attachment Unit Interface) for connection to other media transceivers.

The Media Access Control (MAC) function is provided by the Network Interface Control (NIC) module which provides simple and efficient packet transmission and reception control by means of off-chip memory which can be accessed either through an I/O port or mapped into the system memory map.

An onboard EPROM holds the Ethernet Address and optional configuration information. This allows for “jumperless” configuration using software to configure the board for its operating mode, media type, I/O address, interrupt, etc.



2.22.1 Jumpered vs. Jumperless Mode

The Ethernet section can be configured either through the jumper block at J35 (described later) or via the configuration information stored in the onboard EEPROM. The source for the configuration information is selected via the jumper at J32 as shown here :

Jumpered Mode

Jumperless Mode

NOTE : The choice of "jumpered" or "jumperless" is totally exclusive. This means that if "jumperless" mode is selected, all of the jumpers on J35 are ignored and the configuration information comes totally from the EEPROM. Similarly if "jumpered" mode is selected the information in the EEPROM (except for the Ethernet Address) is ignored.

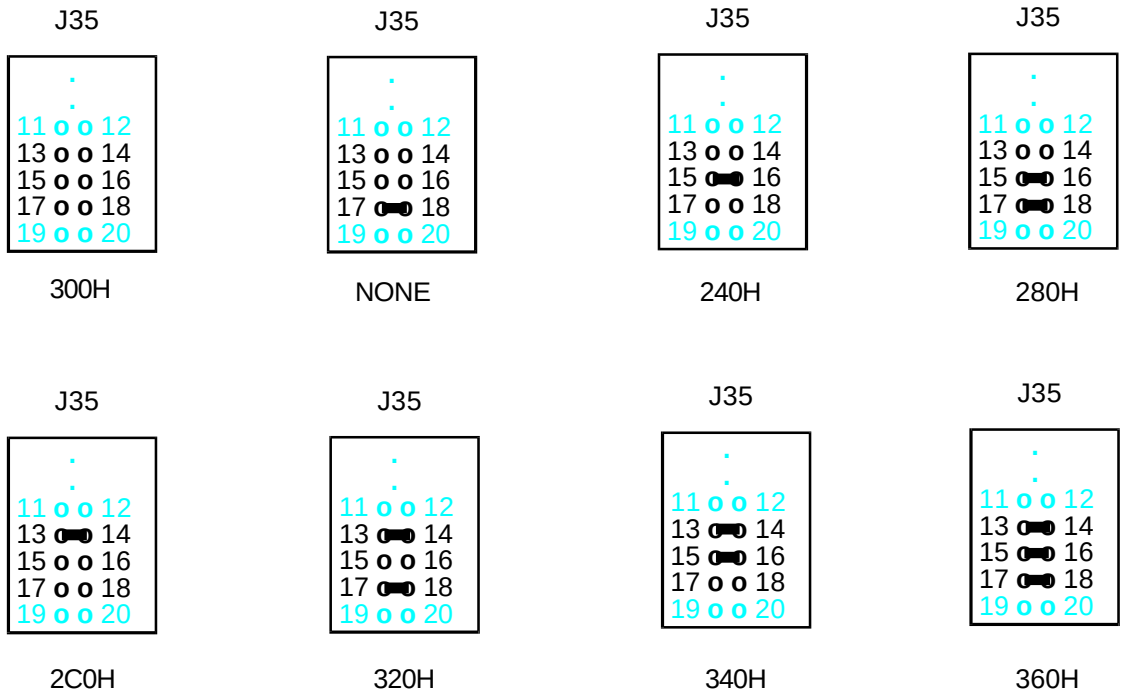
IMPORTANT NOTE : Interrupts must ALWAYS be jumpered manually via the jumper block at J30 (Section 1.6) which matches the "jumpered" or "jumperless" interrupt selection.

The following sections detail the J35 jumpering when the “jumpered” mode is selected.

2.22.2 I/O Port Selection

The NE2000 section of the LBC-Plus uses 32 consecutive I/O addresses in the CPU’s I/O space. The base address is selected using three pins on the J35 configuration jumper. The choices available are:

240H
280H
2C0H
300H
320H
340H
360H
None



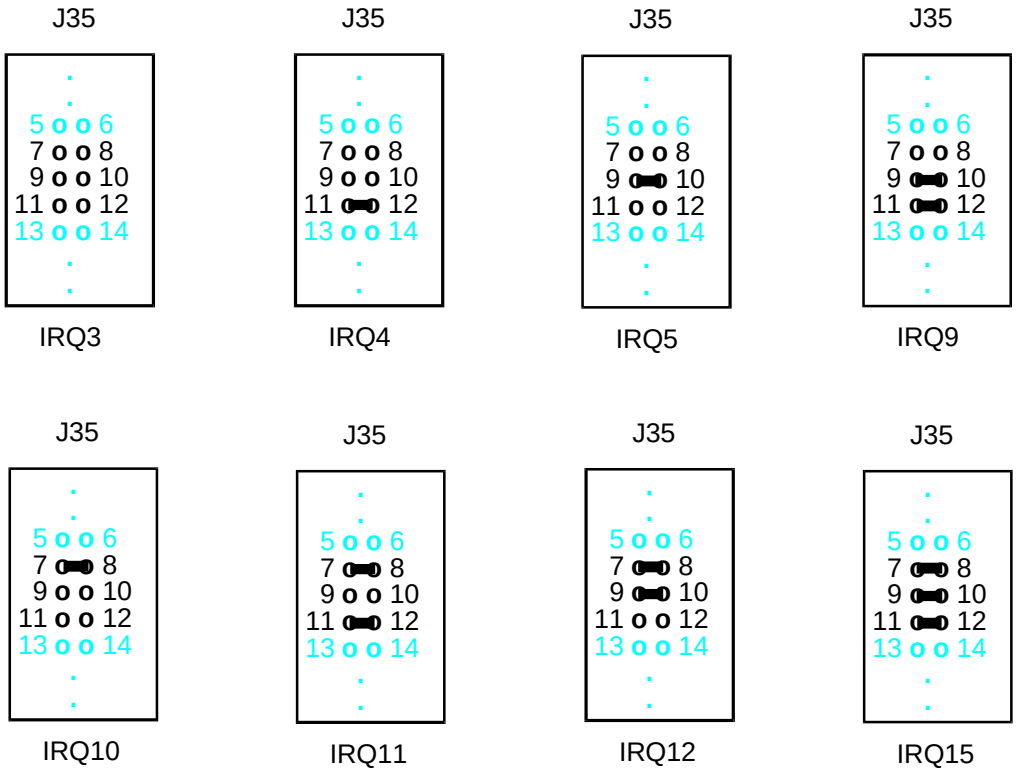
The proper jumpering for each of these choices is shown in the following illustrations.

2.22.3 Interrupt Selection

The NE2000 section needs an interrupt line for signaling various conditions to the software driver. There are 8 possible choices as shown here :

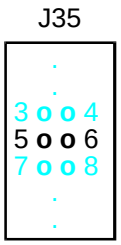
IRQ 3
IRQ 4
IRQ 5
IRQ 9
IRQ 10
IRQ 11
IRQ 12
IRQ 15

The proper jumpering for the three relevant jumper positions corresponding to the available interrupt choices are shown here :

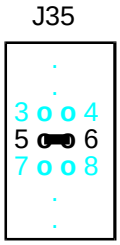


2.22.4 I/O vs. Shared Memory Mode

The Ethernet buffer RAM can be accessed in either of 2 ways. In the typical NE2000 compatible mode, the RAM is accessed through the NIC via I/O ports. An alternate access scheme is available using the shared memory mode. In this mode it is software compatible with the WD8013EBT from Standard Microsystems (formerly Western Digital). In this mode a 32K window in the PC adapter space is used to access packet memory. The address of this window is controlled by the driver. For NE2000 compatibility the I/O mode should be selected. The jumpering for each of the access modes is shown below :



I/O Mode



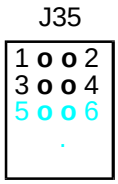
Shared Memory Mode

2.22.5 Media Type Selection

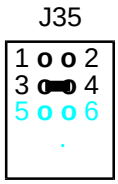
The media type is also jumper selectable via 2 pins on J35. The available choices are :

Twisted-pair 10BASE-T J22
Thin Ethernet Coax[1]
AUI[2] J28
Twisted-pair 10BASE-T Reduced Squelch[3]

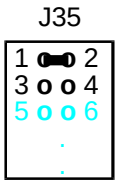
The J35 jumpering for each of the options is shown below.



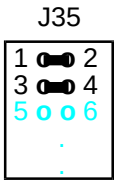
10 BASE-T



THIN ETHERNET[1]



AUI



Non-Spec 10 BASE-T[2]

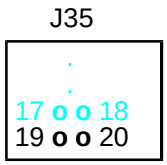
[1] The thin Ethernet mode is not usable with the LBC-Plus. If thin Ethernet is required, it is necessary to select the AUI mode and use an external transceiver.

[2] The AUI is connected via J28. An adapter cable, WinSystems part number CBL-147-1, is available which terminates in a standard DB15 connector.

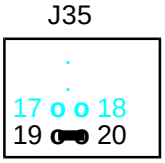
[3] The non-spec twisted-pair mode with reduced squelch levels allows the use of longer cable lengths than specified in the twisted-pair specification, or the use of cable with higher losses.

2.22.6 Compatible Vs. Enhanced Mode

The NE2000 section uses two 32K byte buffer RAMs on board. In compatible mode, only 8K of each RAM (total of 16K) is accessible to the driver. When the Enhanced mode is chosen the full 32K is available from each RAM. This enhanced mode is generally supported by the supplied AT/LANTIC drivers but may not be usable with generic NE2000 software or drivers. When in doubt, choose the compatible mode. The J35 jumpering for the compatible and enhanced modes are shown here :



Compatible Mode



Enhanced Mode

2.22.7 Status LEDs

There are 4 LEDs installed on the LBC-Plus used to give a visual indication of Ethernet status. The color, location, and general description of each LED is given here :

D3	RED	Collision
D4	GREEN	Transmit
D5	GREEN	Receiver
D6	YELLOW	Link

2.22.8 Boot ROM Selection

The LBC-Plus supports the use of the remote boot feature available from NOVELL, QNX, and some other operating systems by allowing provisions for a user installed BIOS extension ROM into U23. Only a 27C010 EPROM device is supported in this mode although only 32K is available for the code. If the BIOS extension is supplied in a smaller device it will have to be reprogrammed into a 27C010 device. In order to use this BIOS extension, the board must be configured for the DOC mode. (See section 2.19 for details). It is also possible to select the address where the BIOS ROM will appear by configuring the SSD Socket Relocation option in the BIOS features menu of CMOS Setup.

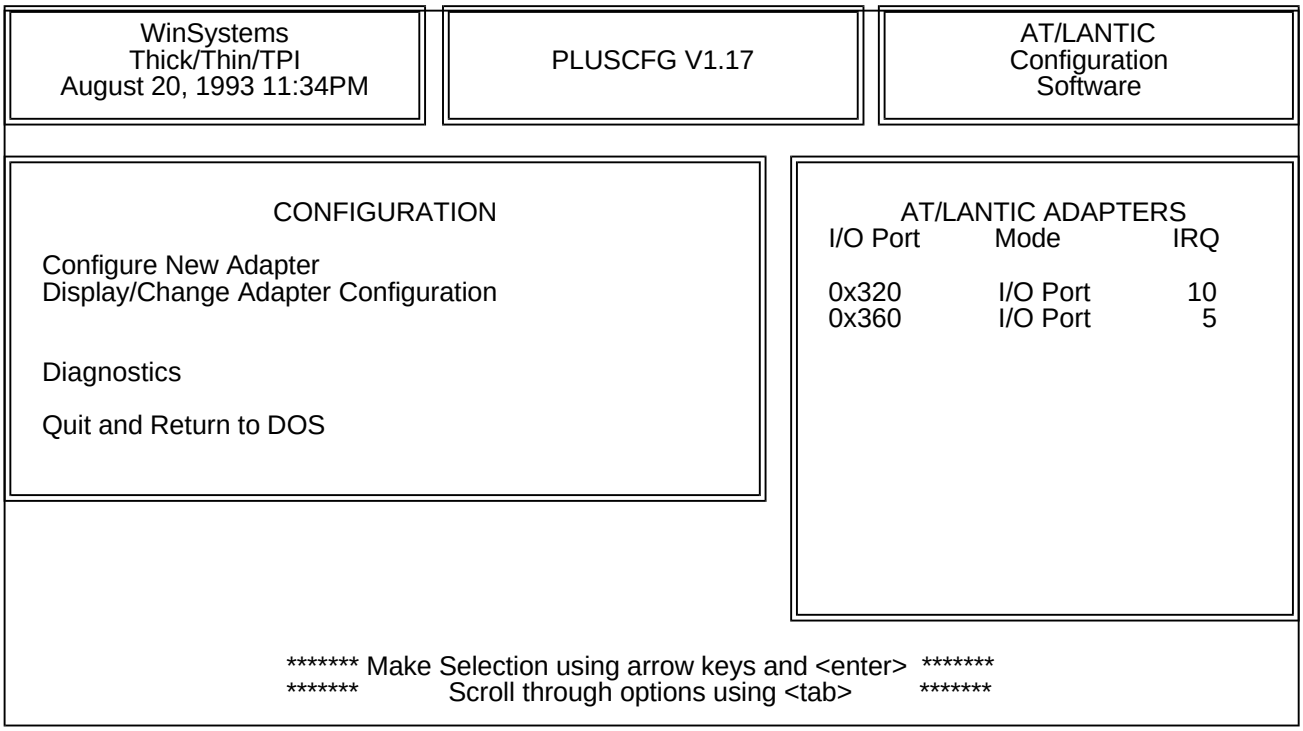
NOTE : The Ethernet BOOT ROM support and onboard SSD support are mutually exclusive, only one or the other may be used. Contact WinSystems for boards that support this feature.

2.22.9 PlusCfg Configuration Utility

When "jumperless" mode is selected (Section 2.22.1), the configuration is made via software which is then saved to the onboard EEPROM. PLUSCFG.EXE along with MESSAGE.MSG can be run from the provided floppy or can be copied to a hard disk. From the DOS command line PLUSCFG.EXE is executed by typing :

```
pluscfg [Enter]
```

The configuration program will load and display the basic menu and configuration screen. If any AT/LANTIC, or NE2000 adapters are recognized, they will be displayed in a window on the right side of the screen as shown here :



From the main menu choose the desired function. Each of the main menu options will be discussed in the following sections.

2.22.10 Configure New Adapter

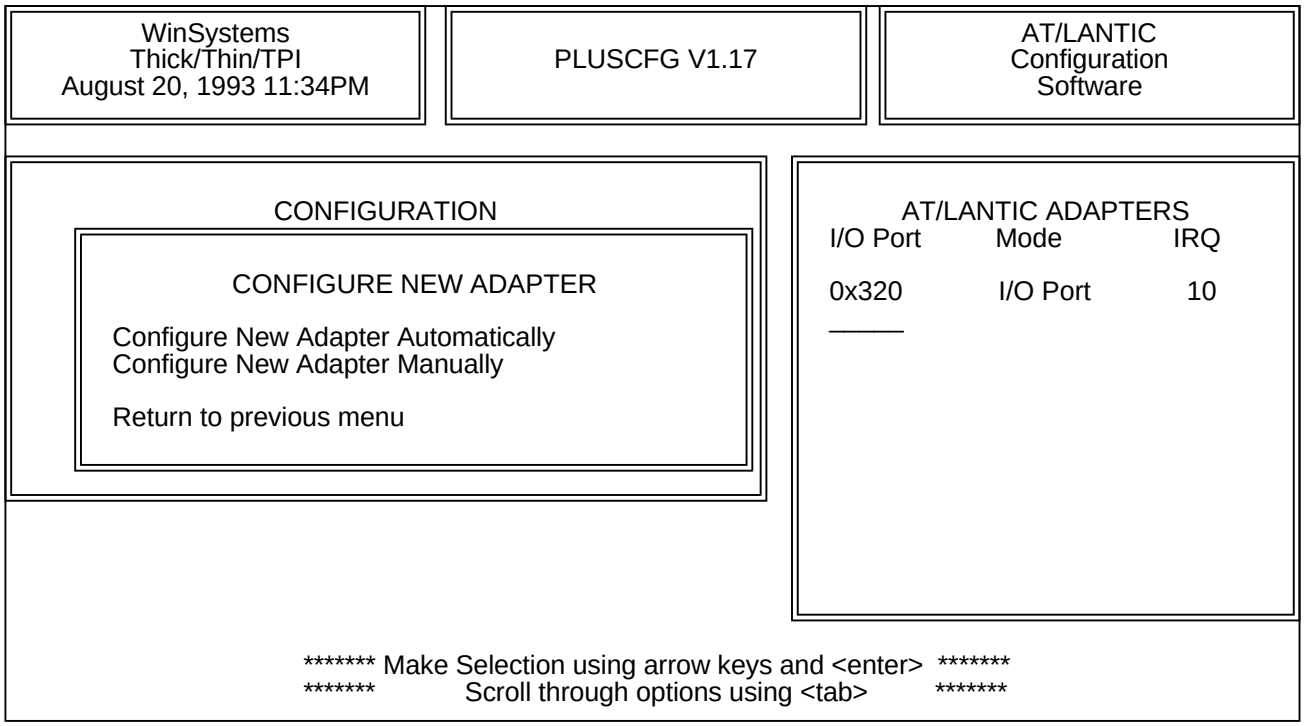
This screen is used to configure an installed adapter that is not present in the window on the right side of the screen. Typically this would be a board that had it's I/O port set to “None”.

Two choices are provided to configure the new adapter.

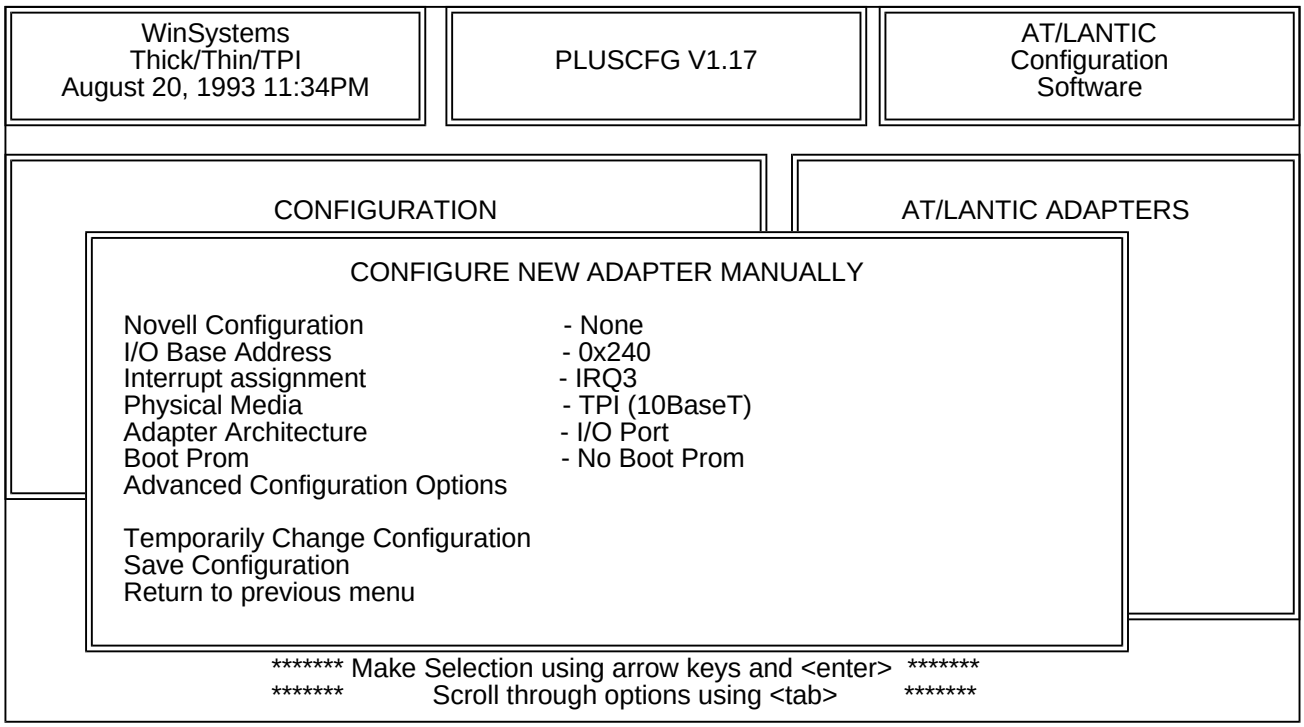
The “Configure New Adapter Automatically” will search out an unconfigured adapter, if present, survey the system, and make automatic choices for I/O address and interrupts for what it believes are free for use. The system will then display a series of configuration options to the user. These include :

Adapter Architecture - I/O Port or Shared Memory
Select Cable Interface - Thin Ethernet or Thick Ethernet or 10BASE-T

The second prompt will only be present if there is no active cable attached or if the program is unable to determine the media type.



The “Configure New Adapter Manually” presents a screen similar to the one shown below :



Use the up and down arrow keys and tab key to change the displayed configuration to what is desired and then select “Save Configuration” to program the EEPROM with the selected choices.

NOTE : PLUSCFG will not allow selection of I/O ports, interrupts, or memory addresses that it believes are being used by other hardware in the system. If PLUSCFG refuses to allow a desired selection for what you know are valid choices, it will be necessary to use the “jumpered” mode, described earlier, for configuration.

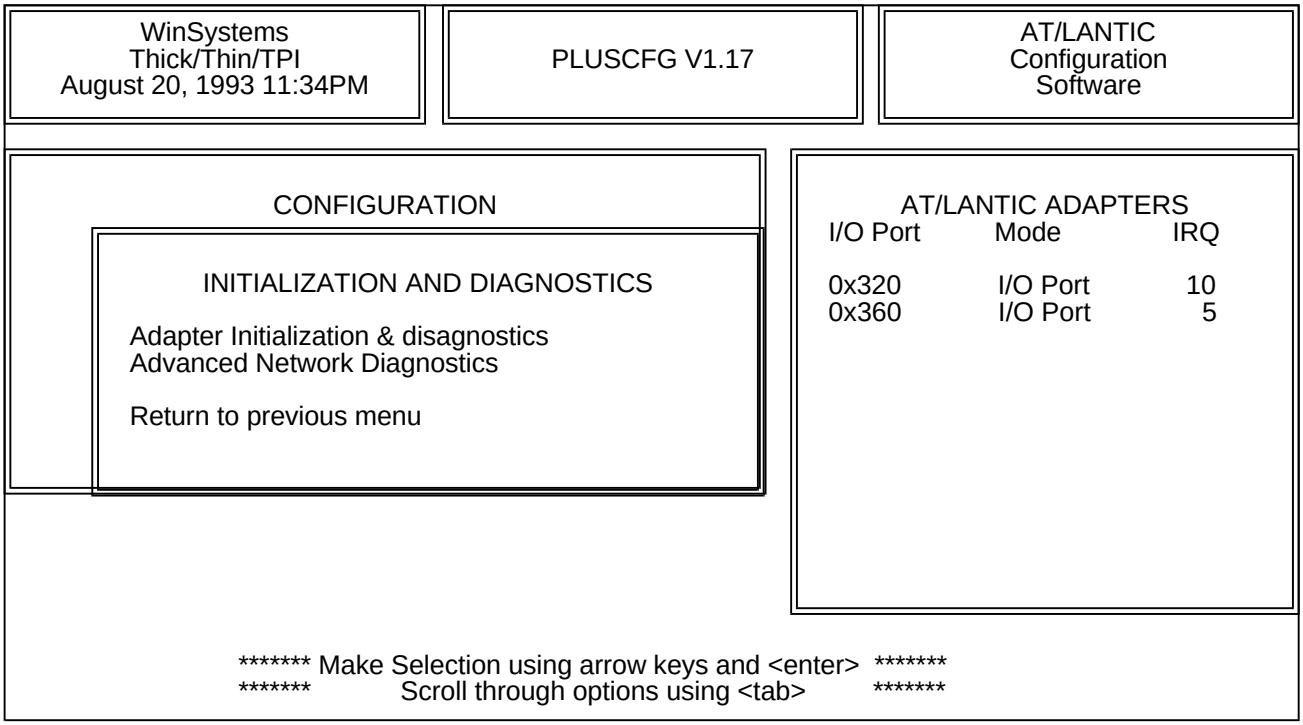
2.22.11 Display/Change Adapter Configuration

This option of the main menu presents the same screen as shown for “Configure New Adapter Manually”. Use the up and down arrow keys and the Tab key to alter the configuration as desired and then select “Save Configuration” to program the EEPROM with the new information.

NOTE : PLUSCFG will not allow selection of I/O ports, interrupts, or memory addresses that it believes are being used by other hardware in the system. If PLUSCFG refuses to allow a desired selection for what you know are valid choices, it will be necessary to use the “jumpered” mode, described earlier, for configuration.

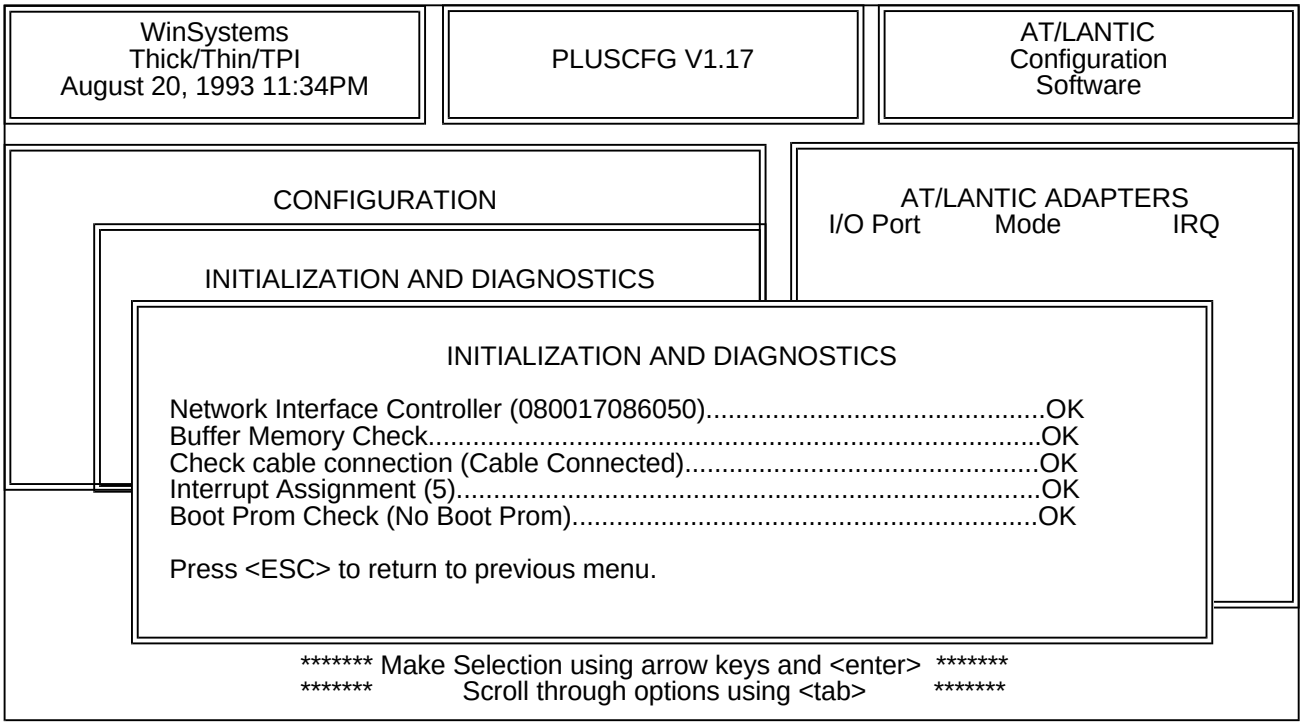
2.22.12 Diagnostics

This third choice from the main menu allows the selection from the diagnostics sub-menu as shown in this screen :



2.22.13 Adapter Initialization and Diagnostics

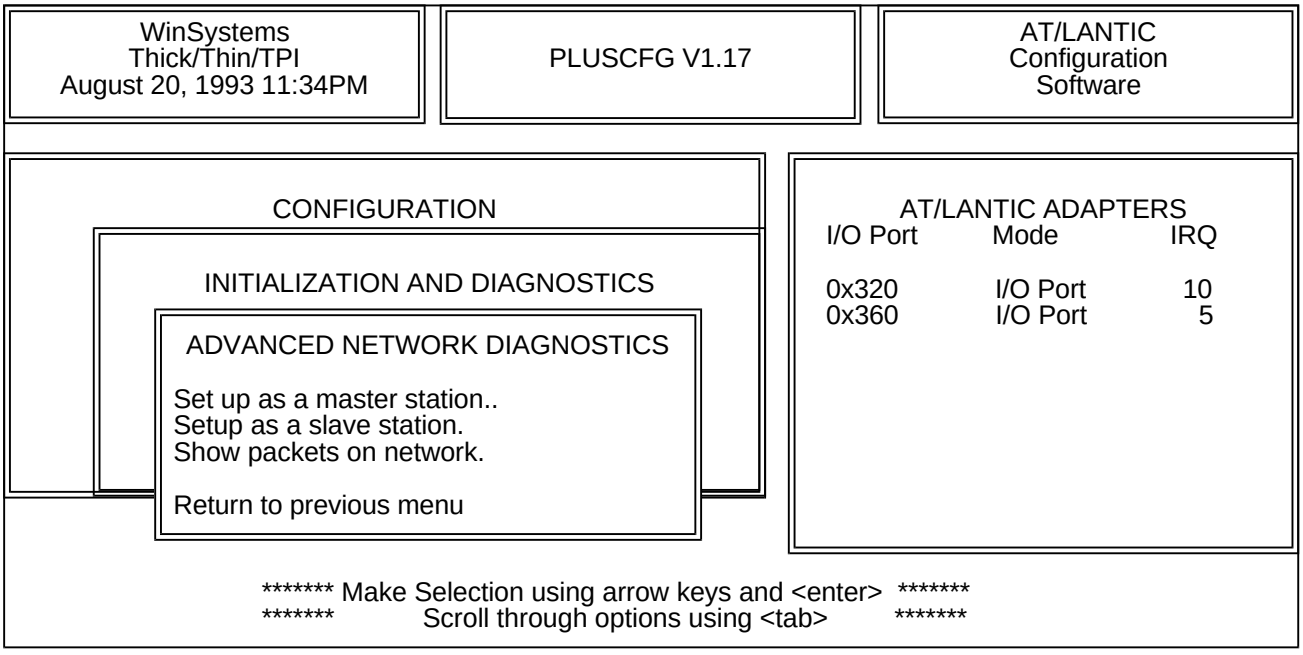
This choice initializes the selected adapter and confirms I/O address, interrupt, media type, etc. The adapter should be connected to the network cable at this time. A sample screen is shown on the following page :



NOTE : The Initialization and Diagnostics must be run and must pass before any of the Advanced diagnostics can be executed.

2.22.14 Advanced Network Diagnostics

The Advanced Network Diagnostics screen is shown here :



Three choices are provided for Advanced Network Diagnostics

2.22.15 Setup as a Master Station

This enables the board under test to be set up as the Master. The master will initiate testing. The Slave must be enabled prior to starting the Master.

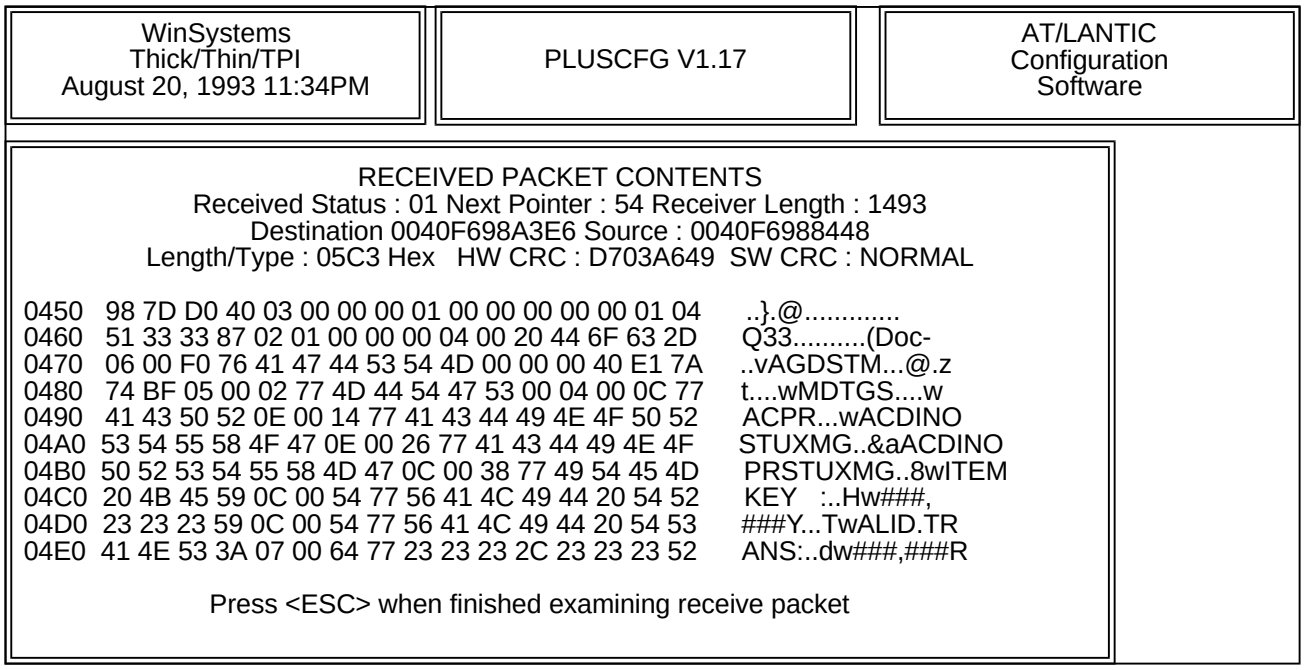
The system will then request a packet repetition length and after entered will begin the test.

2.22.16 Setup as Slave Station

This choice should be made for a known good board. It will echo back across the network all packets initiated by the Master.

2.22.17 Show Packet on Network

This option displays in HEX and ASCII packets as they are received from the network. A sample screen is shown here :



2.22.18 Quit and Return to DOS

This main menu option exits PlusCfg and returns you to the DOS prompt.

2.23 Multi I/O Connector

The I/O to the primary serial channels, the printer port, and keyboard are all terminated via the connector at J3. An adapter cable, part number CBL-162-1, is available from WinSystems to adapt to the conventional I/O connectors. The pin definitions for J3 are shown here :

J3

COM1 - DCD	1	2	COM1 - DSR
COM1 - RXD	3	4	COM1 - RTS
COM1 - TXD	5	6	COM1 - CTS
COM1 - DTR	7	8	COM1 - RI
COM1 - GND	9	10	COM2 - DCD
COM2 - DSR	11	12	COM2 - RSX
COM2 - RTS	13	14	COM2 - TXD
COM2 - CTS	15	16	COM2 - DTR
COM2 - RI	17	18	COM2 - GND
LPT - STROBE	19	20	LPT - AUTOFD
LPT - PD0	21	22	LPT - ERROR
LPT - PD1	23	24	LPT - INIT
LPT - PD2	25	26	LPT - SLCTIN
LPT - PD3	27	28	LPT - GND
LPT - PD4	29	30	LPT - GND
LPT - PD5	31	32	LPT - GND
LPT - PD6	33	34	LPT - GND
LPT - PD7	35	36	LPT - GND
LPT - ACK	37	38	LPT - GND
LPT - BUSY	39	40	LPT - GND
LPT - PE	41	42	LPT - GND
LPT - SLCT	43	44	KEYBD - GND
KEYBD - GND	45	46	KEYBD - GND
KEYBD - KDATA	47	48	KEYBD - CLK
KEYBD - +5V	49	50	KEYBD - +5V

2.23.1 Jumper/Connector Summary

Connector/ Jumper	Description	Page Reference
J1	Parallel I/O VCC Enable jumper	2-22
J2	COM3/COM4 I/O Connector	2-7
J3	Multi-I/O Connector	2-41
J4	Parallel I/O ports 0-2	2-22
J5	Parallel I/O ports 3-5	2-22
J6	Parallel port mode select	2-13
J7	Power/Reset Connector	2-18
J8	COM1 mode select jumper	2-7
J9	COM2 mode select jumper	2-7
J10	COM1 mode select jumper	2-7
J11	COM2 mode select jumper	2-7
J12	CPU Speed Select jumper	2-2
J13	Master battery enable jumper	2-17
J14	PCI Bus Clock select jumper	2-3
J15	Parallel I/O Enable jumper	2-21
J16	SSD Array size select jumper	2-19
J17	Floppy disk I/O connector	2-14
J18	IDE I/O Connector	2-16
J19	Watchdog timer configuration jumper	2-16
J20	VGA BIOS size select jumper	2-25
J21	ECP mode DMA select jumper	2-14
J22	10BASET I/O connector	N/A
J23	COM3/COM4 Enable Disable Select jumper	2-6
J24	ECP DMA select jumper	2-14
J25	SSD/DOC Mode select jumper	2-19
J26	PC/104-8 connector	2-15
J27	Video output connector	2-26
J28	Ethernet AUI connector	N/A
J29	PC/104-16 Connector	2-15
J30	Interrupt routing header	2-4
J31	PCI Bus clock select jumper	2-3
J32	NE2000 Mode select jumper	2-30
J33	FPA50 Panel adapter I/O connector	2-26
J34	U23 VBAT select jumper	2-20
J35	NE2000 Configuration jumper	2-31
J36	CPU Clock multiplier select	2-3
J37	SMI Interrupt	N/A
J38	U23 Device type select jumper	2-20
J39	U27 Device type select jumper	2-20
J40	U27 VBAT Select Jumper	2-20

3 Award BIOS Configuration

3.1 General Information

The LBC-Plus comes equipped with a standard AWARD BIOS with Setup in ROM that allows users to modify the basic system configuration. This type of information is stored in battery-backed CMOS RAM so that it retains Setup information when power is turned off.

3.2 Entering Setup

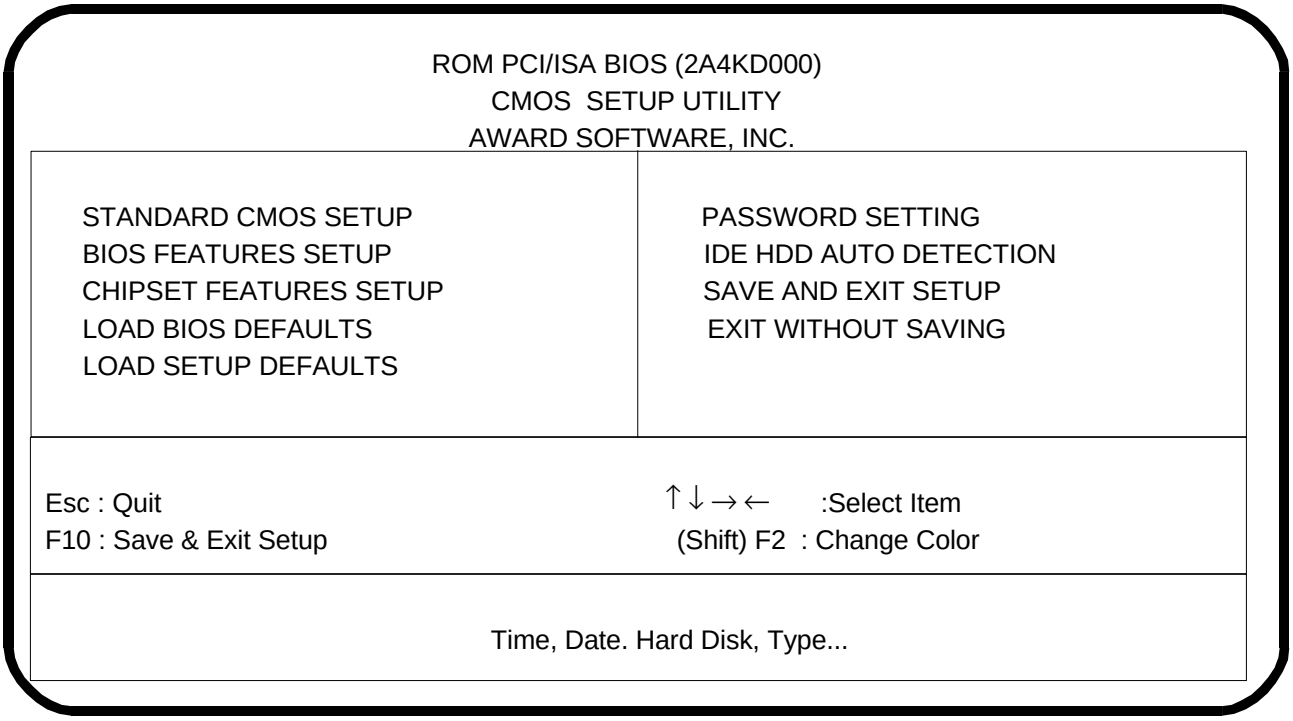
To enter setup, power on the computer and press the DEL key immediately after the message “Press DEL to Enter Setup” appears on the lower left of the screen. If the message disappears before you respond and you still wish to enter setup, restart the system by turning it OFF and then ON or by pressing the RESET button, if so equipped, or by pressing the CTRL, ALT and DEL key simultaneously. Alternately, under certain error conditions of incorrect setup the message :

“Press F1 to continue or DEL to Enter Setup”

may appear. To Enter Setup at that time press the DEL key. To attempt to continue, ignoring the error condition, press the F1 key.

3.3 Setup Main Menu

The main menu screen is displayed on the following page. Each of the options will be discussed in this section. Use the arrow keys to highlight the desired selection and press ENTER to enter the submenu or to execute the function selected.



3.4 Standard CMOS Setup

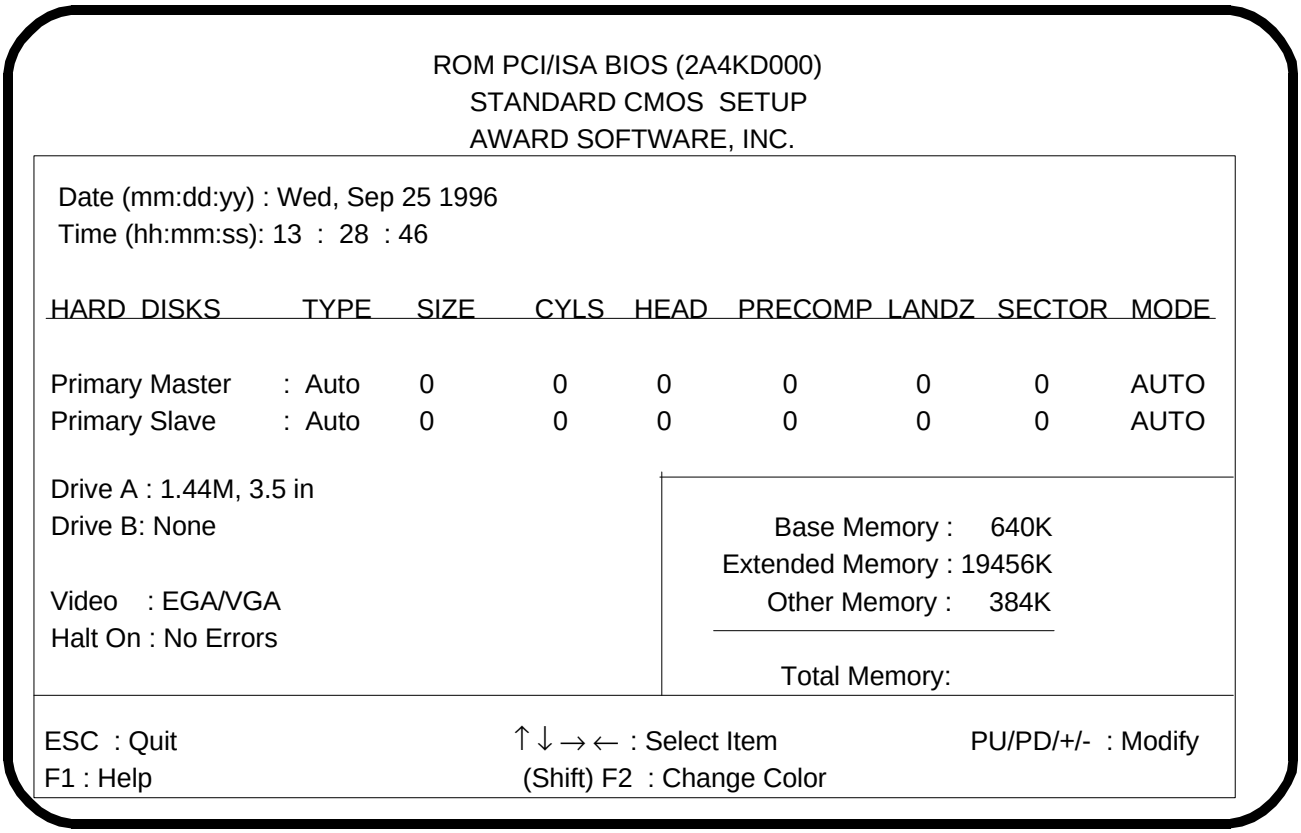
The items in the Standard CMOS Setup menu are divided into several categories. Each category may include one or more setup items. Use the arrow keys to highlight the item and then use the PgUp, PgDn, +.-. keys to select the desired value for the item.

Date

The date format is <day>,< date>,< month>, <year>
day = The day, from Sun to Sat, determined by the BIOS and is display only
Date = the date, from 1 to 31 (or the maximum for the current month)
month = the month, JAN through DEC
year = The year, from 1900 to 2099

Time

The time is hour,minute,second. The time is calculated on the 24-hour, military-time clock such that 1:00PM is 13:00:00.



Drive C / Drive D type

This category identifies the type of hard disk C or hard disk D that has been installed in the system. There are 46 predefined types and a user definable type. Types 1-46 are shown in the following table.

Type	Size	Cylinders	Heads	Sectors	Precomp	Landzone
1	10	306	4	17	128	305
2	20	615	4	17	300	615
3	30	615	6	17	300	614
4	62	940	8	17	512	940
5	46	940	6	17	512	940
6	20	615	4	17	None	615
7	30	462	8	17	256	511
8	30	733	5	17	None	733
9	112	900	15	17	None	901
10	20	820	3	17	None	820
11	35	855	5	17	None	855
12	49	855	7	17	None	855
13	20	306	8	17	128	319

14	42	733	7	17	None	733
15		Reserved				
16	20	612	4	17	0	663
17	40	977	5	17	300	977
18	56	977	7	17	None	977
19	59	1024	7	17	512	1023
20	30	733	5	17	300	732
21	42	733	7	17	300	732
22	30	306	5	17	300	733
23	10	977	4	17	0	336
24	40	1024	5	17	None	976
25	76	1224	9	17	None	1023
26	71	1224	7	17	None	1223
27	111	1224	11	17	None	1223
28	152	1024	15	17	None	1223
29	68	1024	8	17	None	1023
30	93	918	11	17	None	1023
31	83	925	11	17	None	1023
32	69	1024	9	17	None	926
33	85	1024	10	17	None	1023
34	102	1024	12	17	None	1023
35	110	1024	13	17	None	1023
36	119	1024	14	17	None	1023
37	17	1024	2	17	None	1023
38	136	1024	16	17	None	1023
39	114	918	15	17	None	1023
40	40	820	6	17	None	820
41	42	1024	5	17	None	1023
42	65	1024	5	26	None	1023
43	40	809	6	17	None	852
44	61	809	6	26	None	852
45	100	776	8	33	None	775
46	203	684	16	38	None	685

Press PgUp or PgDn to select a numbered hard disk type, or type the number and press ENTER. Most manufacturers supply type information with their drives that can be used to help identify the proper drive type. Modern IDE drives seldom fall into the predefined types and are usually best handled with the “auto” or “user” types. The “auto” mode, reads the hard disk type information from the drive at

boot time and uses it to access the drive. The “user” mode allows for either manual or automatic entry, via the setup option “IDE Auto Detect” of the drive parameters.

If you decide to create the user type manually, you must supply the required parameters as to Cylinder count, Head count, Precomp Cylinder, Landing Zone Cylinder, and number of sectors per track.

On Hard disks larger than 528MB, it will be necessary to choose the Logical Block Addressing mode (LBA) if you wish the drive to be accessible as a single drive letter.

If there is not hard disk installed, be sure to select “None”.

Drive A type/Drive B type

This category identifies the type of floppy drives attached as Drive A: or Drive B:. The choices are as follows :

NONE 360K, 5.25 in.
1.2M, 5.25 in.
720K, 3.5 in
1.44M, 3.5 in.

Video

This category specifies the type of video adapter used for the primary system monitor that matches your video display board and monitor. The available choices are :

EGA/VGA
CGA40
CGA80
MONO

The LBC-Plus has built-in VGA support so EGA/VGA should be selected.

Error Halt

This category determines whether the system will halt if a non-fatal error is detected during the power up self test. The choices are :

No Errors : The system will not be stopped for any error that may be detected.

All Errors : Whenever the BIOS detects a non-fatal error, the system will be stopped and a prompt will appear.

All, but Keyboard : The system will not stop for a keyboard error, it will stop for all other errors.

All, but diskette : The system will not stop for disk errors. All others will result in a prompt.

All but Disk/Key : All errors except diskette or keyboard will result in a halt and a prompt.

Memory

This category is display only and is determined by the BIOS POST (Power On Self Test).

Base Memory

The POST routines in the BIOS will determine the amount of base (conventional) memory installed in the system The value of the base memory is typically 640K for systems with a Megabyte of memory or greater.

Extended Memory

The BIOS determines how much extended memory is present during the POST. This is the amount of memory located above 1MB in the CPU's memory address space.

Other Memory

This refers to memory located in the 640K to 1024K address space. This is memory that can be used for different applications. DOS may use this area to load device drivers and TSRs to keep as much base memory free as possible for application programs. The most common use of this area is for shadow RAM.

3.5 Bios Features Setup

Virus Warning

This option when enabled, protects the boot sector and partition table of the hard disk against unauthorized writes through the BIOS. Any attempt to alter these areas will result in an error message and a prompt to authorize the activity.

CPU Internal Cache

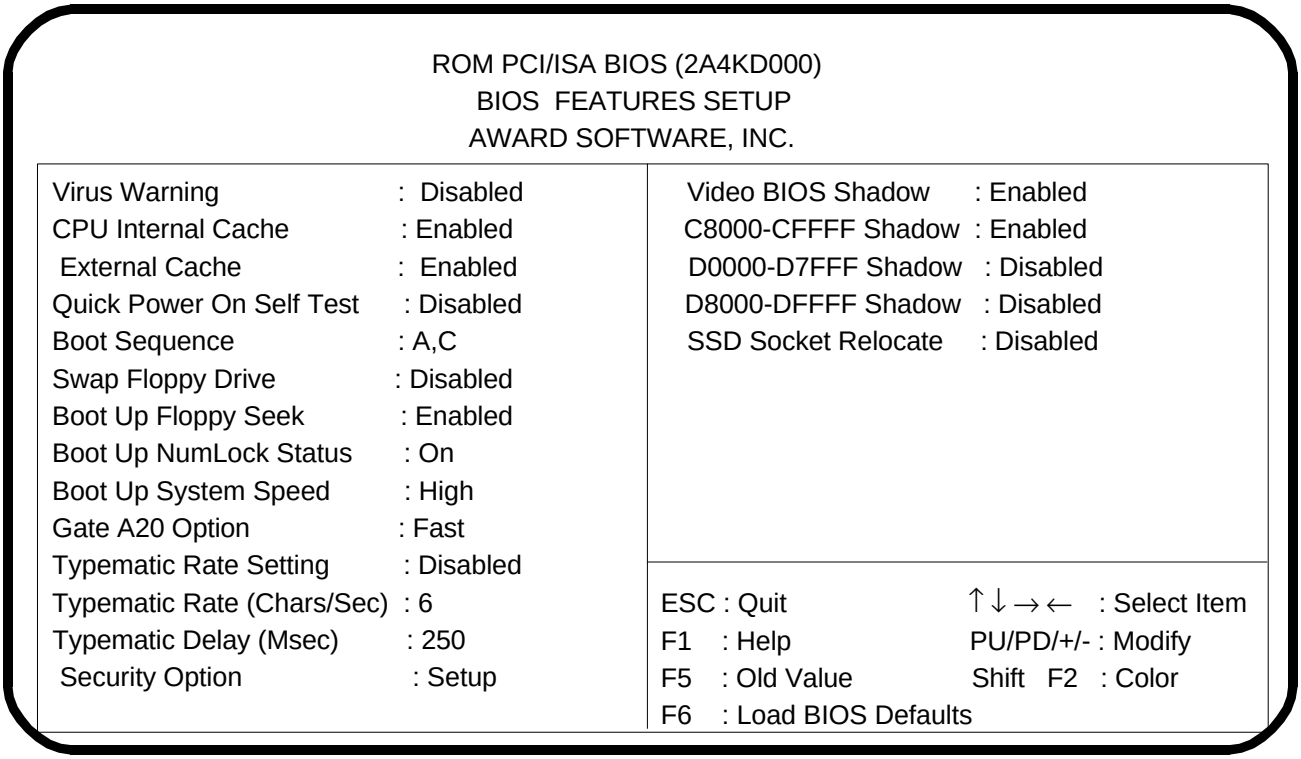
This option, when enabled, provides maximum performance by caching instructions and data using the on-chip cache of the 486 or 586 processor.

External Cache

This option, when enabled, further enhances performance by caching recently used instructions and data into fast SRAM.

Quick Power ON Self Test

This option, when enabled, speeds up the POST during power up. If it is enabled, the BIOS will shorten and/or skip some items during POST.



Boot Sequence

This option determines the boot attempt sequence for the fixed disk and floppy disk. The choices are:

C,A The system will attempt Hard disk boot first
A,C The system will attempt Floppy disk boot first

Swap Floppy Drive

This option allows for swapping of the A: and B: floppy drives without actually relocating the drives on the cable.

Boot Up Floppy Seek

During POST, when this option is enabled, the BIOS will determine if the floppy drive is 40 track or 80 tracks, If disabled, no seek test will be performed and no error can be reported.

Boot Up Numlock Status

This allows user selection of the Numlock state at boot time.

Boot Up System Speed

This option allows specification of the processor speed at boot time. The options are :

HIGH
LOW

Gate A20 Option

This option allows for the selection of the source for the gate A20 signal. The choices are :

Normal - Sourced from the keyboard controller
Fast - Sourced from the Chipset

Typematic Rate Setting

This option enables or disables the typematic rate programming at boot time. Typematic is the auto-repeat function for the keyboard.

Typematic Rate

When the typematic rate setting is enabled the typematic repeat speed is set via this option. The supported rates are :

6 characters per second
8 characters per second
10 characters per second
12 characters per second
15 characters per second
20 characters per second
24 characters per second
30 characters per second

Typematic Delay

When typematic rate setting is enabled, this option specifies the time in milliseconds before auto-repeat begins. The supported values are :

250 mS
500 mS
750 mS
1000 mS

Security Option

This option allows you to limit access to the system and setup, or just to setup. The choices are :

System - The system will not boot and access will be denied if the correct password is not entered at the prompt.

Setup - The system will boot, but access to Setup will be denied if the correct password is not entered at the prompt.

NOTE : To disable security, select "Password Setting" at the Setup Main Menu and then you will be asked to enter a password. Do not type anything, just hit ENTER. Once the security is disabled, the system will boot and you can enter Setup freely.

Shadowing Options

When shadowing for a particular address range is enabled, it instructs the BIOS to copy the BIOS located in ROM into DRAM. This shadowing from an 8-bit EPROM into fast 32-bit DRAM results in a Multi-magnitude increase in performance. The main BIOS is shadowed automatically but there are other areas that may be selected for shadowing as shown here :

Video BIOS Shadow - C000-C7FFF EGA/VGA BIOS ROM
C8000-CFFFF
D0000-D7FFF
D8000-DFFFF

SSD Socket Relocation

The LBC-Plus supports an optional BIOS extension ROM in U27. This feature was implemented primarily to support network boot ROMs but can be used for other purposes by knowledgeable users.

Only 128K X 8 devices (27C010 type) are supported, although only 32K of the device may be used for code. The board is configured for the DOC mode as documented in the SSD configuration section of this manual. This setup option allows for the BIOS extension to be located at any of several addresses as shown here :

Disabled
C8000
CC000
D0000
D4000
D8000
DC000
E0000

Note : This mode of usage is mutually exclusive of any onboard silicon disk usage. Further note that 'Disabled' must be selected when **any** SSD mode is desired.

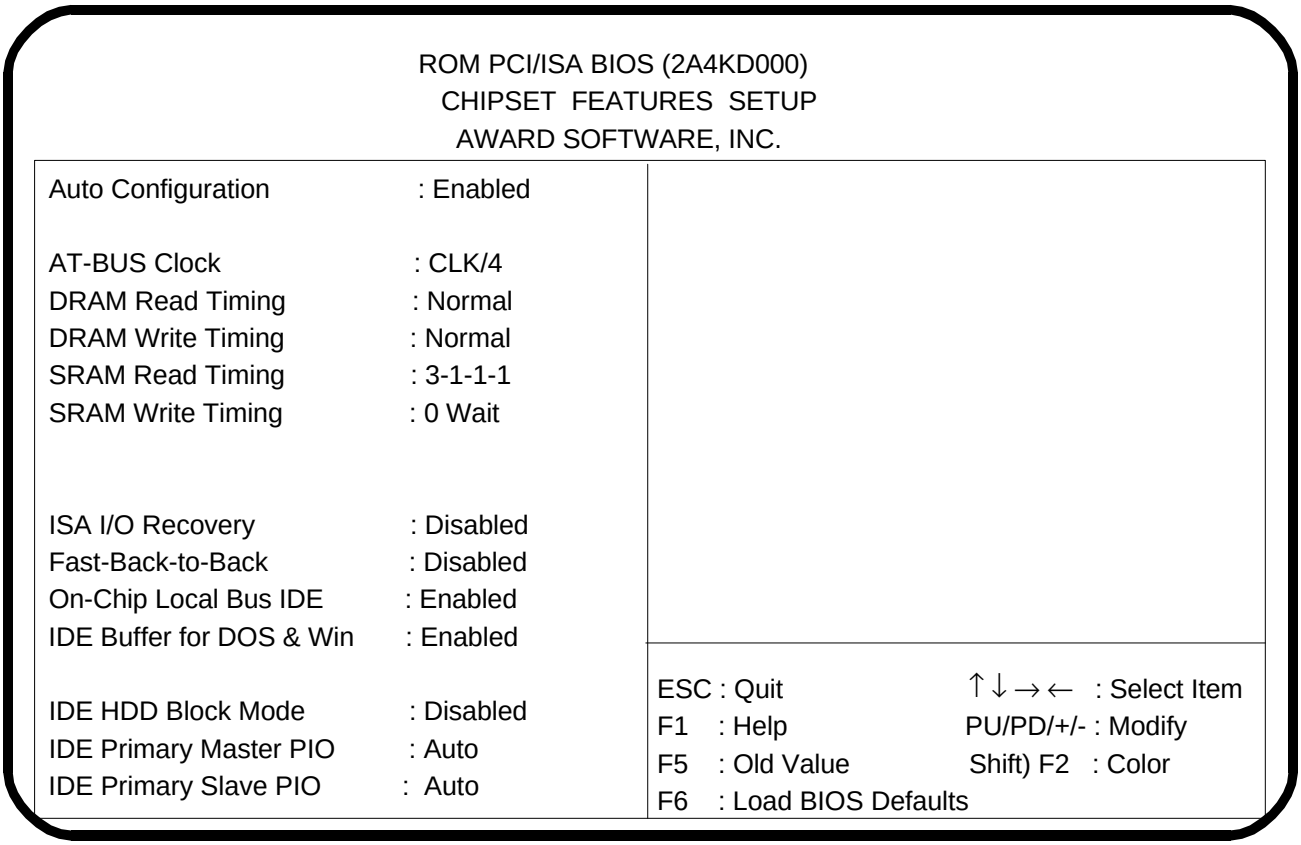
3.6 Chipset Features Setup

The options in this section control the chipset programming at boot time. In most cases, the default settings should be used unless you have a clear understanding of the significance of the change. It is possible using these options to create a system that will either not boot or is very unstable or unreliable. If this should occur, there are two methods to return the system to a stable configuration. If the system works well enough to get into Setup, simply choose the “Load BIOS Defaults” option and then select “Save and Exit Setup” to restore factory defaults. If the system will not run well enough to run Setup, it will be necessary to remove the battery source temporarily until the CMOS memory decays. Refer to Section 2.X for details on reinitializing the CMOS RAM.

Each of the options for the Chipset Features Menu will be briefly discussed in the sections that follow.

Auto Configuration

This option, when enabled, instructs the BIOS to auto-select the proper AT Bus Clock, the DRAM read timing, the DRAM write timing, the SRAM read timing, and the SRAM write timing base upon the calculated CPU speed. The default is “Enabled”.



AT-BUS Clock

This option is available when the "Auto Configuration" option is disabled. This allows selection of the speed of the AT-BUS clock. This clock is any of 5 sub-multiples of the processor oscillator or is fixed at 7.19MHz. The choices available are:

7.19MHz
CLK/3
CLK/4
CLK/5
CLK/6
CLK/8

DRAM Read Timing

This option controls the read timing to the DRAM array. The available options are shown here :

slow
normal - default
fast
fastest

DRAM Write Timing

This option controls the write timing to the DRAM array. The available options are shown here :

slow
normal - default
fast
fastest

SRAM Read Timing

This option allows for selection of the timing patterns used to access the Cache RAM. The available choices are :

2-1-1-1
3-1-1-1 Default
3-2-2-2
4-2-2-2

SRAM Write Timing

This option controls the number of wait-states to be inserted during cache write operations. The choices are :

0 Wait
1 Wait Default

ISA I/O Recovery

The CPU and local bus are much faster than the standard for the ISA bus. Selecting enabled for this option allows additional time for I/O devices to respond to the system. The default is disabled.

Fast Back-To-Back

When enabled, consecutive write cycles targeted to the same slave become fast back-to-back on the PCI bus. The default setting is disabled.

On Chip Local Bus IDE

This option when enabled allows usage of the onboard PCI Bus IDE controller. The default is enabled.

IDE Buffer for DOS & Win

Select Enabled to increase throughput to and from IDE devices by using the on-chip read-ahead and post-write IDE buffers. Note that the use of the buffer may cause some slow IDE devices to appear even slower. The default is Enabled.

IDE HDD Block Mode

Block mode is also called block transfer, multiple command, or multiple sector read/write. If the IDE hard drive supports block mode, select Enabled for the automatic detection of the optimal number of block read/writes per sector that the drive can support. The default is disabled.

IDE Primary Master PIO

There are 5 transfer modes available for hard disk IDE transfers ranging from mode 0 to mode 4 with each successive mode providing an increased level of performance. Selecting “Auto” will allow the BIOS to automatically select the optimum transfer mode for the master hard disk. The default setting is Auto.

IDE Primary Slave PIO

Like the previous item, this option allows for the selection of any one of 5 IDE transfer modes or an Auto selection which allows the BIOS to auto-optimize the IDE transfers to/from the slave hard disk. The default setting is Auto.

3.7 Load BIOS Defaults

This main-menu option will cause the CMOS RAM to be loaded with the default values assigned by the factory. These are usually considered safe values and do not necessarily represent the highest performance values.

3.8 Load Setup Defaults

This option will cause the CMOS RAM to be loaded with default setup values assigned by the factory. These are usually values that were determined to give a higher level of performance along with reliable operation.

3.9 Password Setting

This option allows the setting of the security password. Pressing enter at the password prompt disables the security function completely.

3.10 IDE HDD Auto Detection

This function allows modern IDE fixed disks to be used to their maximum potential by interrogating the driver as to its preferred configuration of tracks,heads, and sectors; and automatically loading these parameters into a “user defined” hard disk type.

3.11 Save & Exit Setup

This function writes all changes to CMOS RAM and restarts the system.

3.12 Exit without Saving

This option exits setup without saving any changes made and then restarts the system.

4 LBC-PLUS Silicon Disk Reference

4.1 Introduction

WinSystems provides silicon disk support for the LBC-Plus using four different media types depending upon the needs of the application.

1. The LBC-Plus provides support for a bootable ROMDISK with a size of up to 1.44 Megabytes. A simple disk imaging technique allows for the easy creation and maintenance of ROMDISKs. Since the bootable ROMDISK is an exact image of a bootable floppy diskette, all testing and debugging can be accomplished using a floppy drive. Once the application is ready for ROM, it's a simple matter to use the MKDISK utility to create the EPROM files necessary for the bootable ROMDISK equivalent of the functioning floppy diskette.

2. In applications requiring occasional program or data updates PEROM, (Flash) disks of up to 1 Megabyte may be used as the boot media. Onboard support is provided for the formatting, reading, and writing of the Floppy drive emulating PEROMs.

3. For Applications needing to log data, update the application, or for convenience during development, battery backed SRAM may be used as the boot media with a size of up to 1 Megabyte.

4. The LBC-Plus supports the M-Systems Disk-On-Chip Devices (DOC). These are single chip devices containing the BIOS Extension, TFFS Flash File System, and a Flash array ranging in size from 1 Megabyte to 12 Megabytes. These devices emulate a hard disk at the BIOS level.

4.2 ROMDISK Usage

MKDISK is a menu-driven utility for creating the ROM image(s) duplicating the desired floppy diskette. MKDISK is invoked at the DOS command line with :

```
MKDISK
```

Select the USSD mode from menu number 1. The other menu options are used with other WinSystems' Silicon Disk systems and are NOT compatible with the LBC-Plus board.

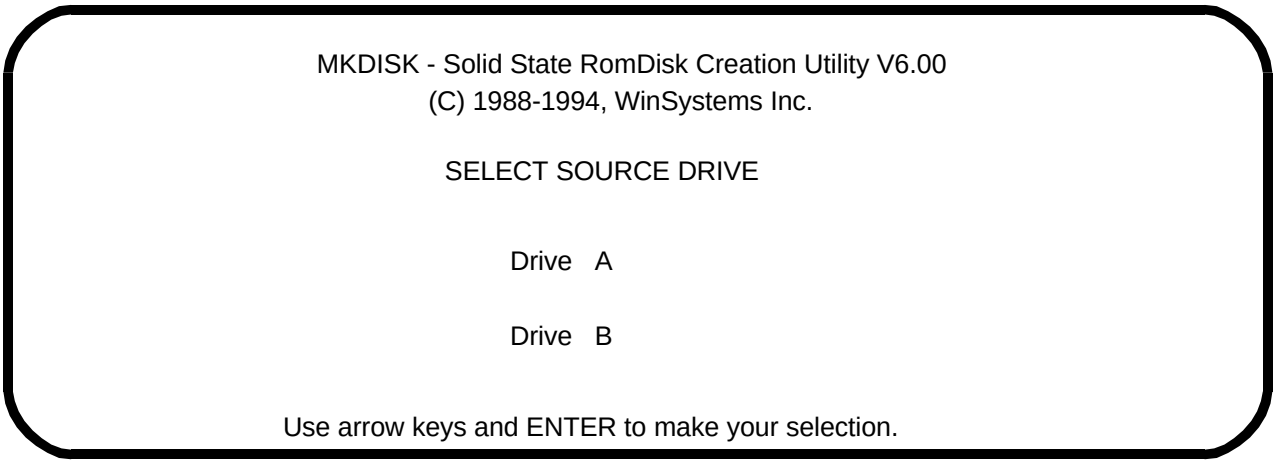
```
MKDISK - Solid State RomDisk Creation Utility V6.00
(C) 1988-1994, WinSystems Inc.

SELECT SSD TYPE

Paged Memory Mode (SSD-XT)
Extended Memory Mode (SSD-AT)
V53 Expanded Memory Mode
I/O Mapped Silicon Disk (USSD)
sx386 On Board ROMDISK
SBC53sx Expanded Memory Mode
SAT-V40 Expanded Memory Mode

Use arrow keys and ENTER to make your selection.
```

MKDISK - Main Menu

From menu number 2 select the appropriate source disk size and type.

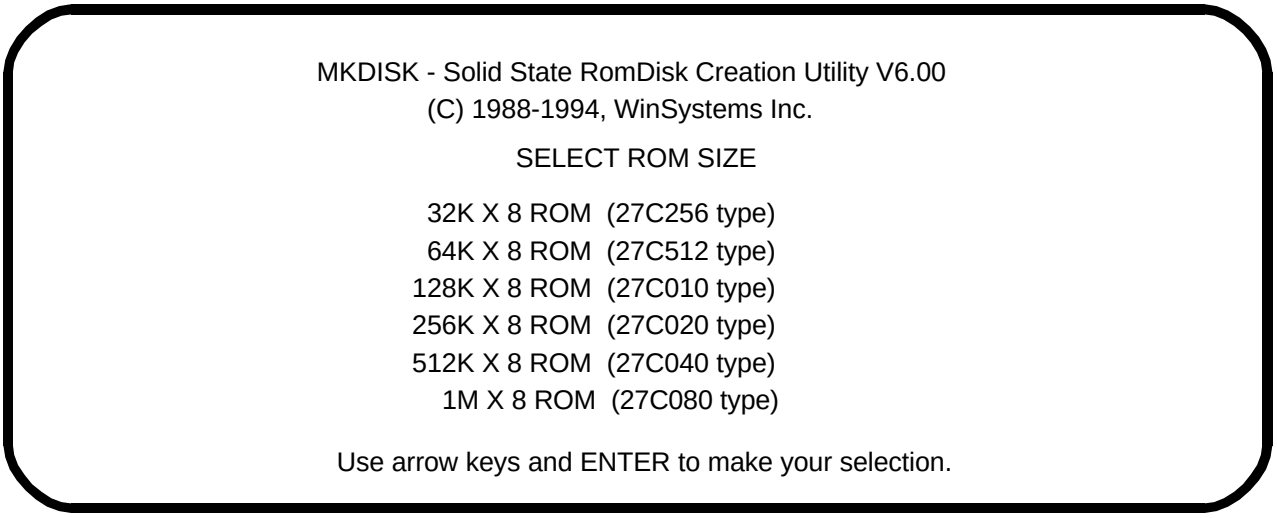
```
MKDISK - Solid State RomDisk Creation Utility V6.00
(C) 1988-1994, WinSystems Inc.

SELECT SOURCE DISK TYPE

160 KB   5 1/4 Single  Sided  8   Sectors 40  tracks
180 KB   5 1/4 Single  Sided  9   Sectors 40  tracks
320 KB   5 1/4 Double Sided  8   Sectors 40  tracks
360 KB   5 1/4 Double Sided  9   Sectors 40  tracks
720 KB   3 1/2 Double Sided  9   Sectors 80  tracks
720 KB   5 1/4 Double Sided  9   Sectors 80  tracks
954 KB   3 1/2 Double Sided  9   Sectors 53  tracks
960 KB   5 1/4 Double Sided  15 Sectors 64  tracks
1.2  Meg 5 1/4 Double Sided  15 Sectors 80  tracks
1.4  Meg 3 1/2 Double Sided  18 Sectors 80  tracks

Use arrow keys and ENTER to make your selection.
```

MKDISK - Drive type Menu



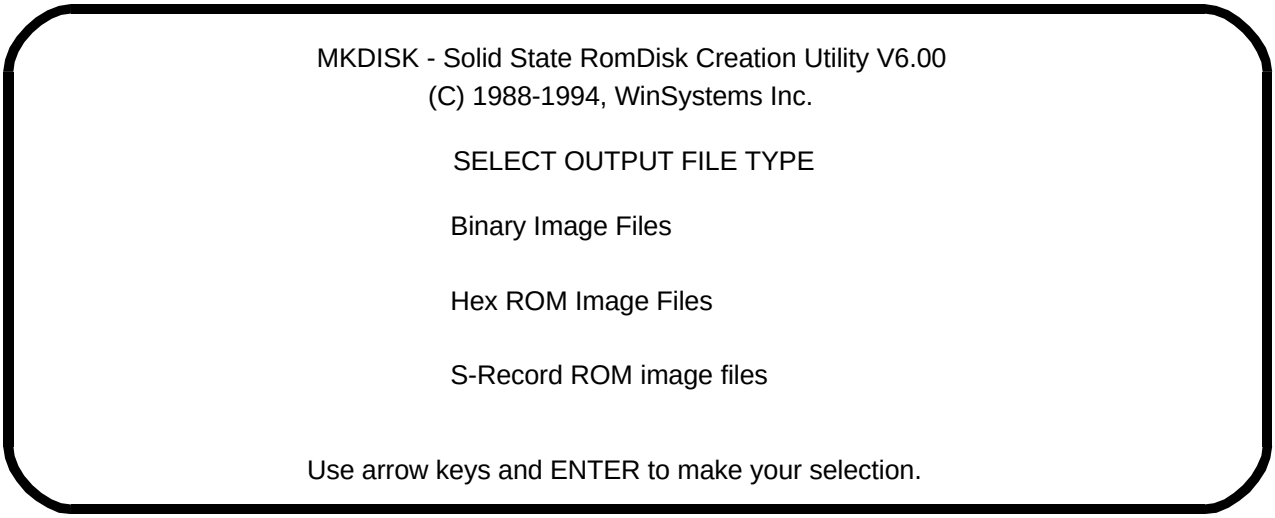
MKDISK - Drive Menu

Select the source drive as appropriate.



MKDISK - ROM type Menu

From menu number 4, select the appropriate EPROM size for the ROMDISK. EPROM size smaller than 512K are not usable with the LBC-Plus, but are provided as choices for use with other WinSystems' silicon disk devices.



MKDISK - Output Menu

From menu number 5, select the appropriate ROM image file format that your EPROM programmer accepts. Selecting the Binary ROM image file format will result in the smallest files. MKDISK will then read the specified floppy diskette and create a ROMx.HEX or ROMx.S19 where x is the ROM number in the sequence (starting with 1) and the extension (.BIN, .HEX, .S19) indicates the output format for Binary, Hex, and Motorola respectively. If more than one file is created, it means that the disk will span more than a single EPROM. Once the ROM(s) have been created using the image files, install the ROM(s), jumper for correct ROM size, and enable the Silicon Disk boot option. The next power up should result in a boot from the A: Silicon Disk. The actual floppy drive, if present, will then be available as drive B:.

4.3 Bootable RAMDISK/FLASHDISK Usage

The LBC-Plus supports bootable RAMDISKs and FLASHDISKs of up to 1 Megabyte in size. 512K X 8 Static RAMs/PEROMs can be installed in the board beginning at U27. One or two RAMs/PEROMs can be installed and the device jumpers should be appropriately set as described in section 2.20. After powerup, it is necessary to configure the silicon disk for the actual size of the drive using the SSDINIT utility. SSDINIT is invoked at the DOS command line with :

```
SSDINIT [A: | B: ] disk_size[K | M]
```

the K or M arguments are optional and are actually ignored. Values below 32 are assumed to be in Megabytes while values above 32 are assumed to be in Kilobytes. An example might help to clarify. To prepare a 1Meg Flash or SRAM disk for formatting, type :

```
SSDINIT B: 1M
```

The disk is now prepared for formatting. The system must be rebooted prior to formatting with the simple DOS command :

```
format b: /s/u
```

After the next reset the formatted silicon disk will boot as the A: drive. If it is ever necessary to by-pass the silicon disk boot in order to reformat or to boot the actual floppy drive, or the hard disk, simply press the <CTRL><ALT><LSHIFT> keys simultaneously immediately following display of the BIOS configuration BOX. The message :

```
Silicon Disk Boot Aborted by User
```

will be displayed and the system will boot from one of the available boot drives.

IMPORTANT NOTE : The FLASHDISK is fully writeable at all times but is not recommended for continuous updating or data logging. The onboard BIOS implements a simple FAT based file system (identical to a floppy disk) with no wear leveling implemented. The PEROMs can and will wear out with excessive write cycles. ATMEL specifies at least 10,000 write cycles.

4.4 Non-Bootable RAMDISK Usage

A non bootable RAMDISK is often desired in conjunction with a bootable ROMDISK, FLASH-DISK, or rotational media. It can then be used for program updates, parameter storage, or data logging applications. a non-bootable RAMDISK uses the WinSystems Universal Solid State Disk Driver (USSD) which is loaded via the boot media's CONFIG.SYS file with the entry :

```
device = ussd.sys /mod:p /pad:1ec  /seg:e400 /psz:16 /inc:1 /spg:xx /dsz:yy
```

where the YYY in /DSZ:YYY is the size of the disk in Kilobytes and the XXX in the /SPG:XXX is the starting page address in the array for this silicon disk. This hexadecimal value is actually the count of 16K byte blocks preceding the start of the RAMDISK. The simplest approach is to use the table below to determine the correct /SPG value.

Total RAM/ROM/FLASH Prior to this socket	/SPG Value
None	/SPG:80
512K	/SPG:A0
1M	/SPG:C0

A couple of examples might help to illustrate. Suppose we're booting from a floppy or hard disk and we have installed two 512K X 8 SRAMs. In order to create the desired 1 Meg RAMDISK we would need the line :

```
device = ussd.sys /mod:p /seg:e400 /psz:16 /inc:1 /pad:1ec /spg:80 /dsz:1024
```

which would indicate that we wish to create a disk of 1M (1024K) starting at the beginning of the array.

For another example assume that we want a 512K ROMDISK and a 512K RAMDISK. We would create our bootable floppy with the CONFIG.SYS line :

```
device = ussd.sys /mod:p /pad:1ec /seg:e400 /psz:16 /inc:1 /spg:a0 /dsz:512
```

This would create the 512K RAMDISK which will be preceded by the 512K of ROMDISK. We would create our ROMDISK as previously described and place the EPROM into U27. We would install our 512K SRAM into U23.

A final example would be to use the non-bootable RAMDISK in U27 as a secondary silicon disk to a DISK-ON-CHIP. In this case we are booting from the DISK-ON-CHIP and will include the following CONFIG.SYS command line :

```
device = ussd.sys /mod:p /seg:e000 /psz:16 /inc:1 /pad:1ec /spg:80 /dsz:512
```

this will create a 512K RAMDISK in the silicon disk socket U27.

NOTE : USSD, as is the convention with DOS installable disk devices, creates a drive with the **NEXT AVAILABLE** drive letter. Drives A: and B: are always reserved for the physical floppy drive or the BIOS supported bootable Silicon Disk. In a system without a hard disk, the next available drive letter would be C:. In a system with one or more hard drive partitions, the silicon disk created with USSD will be the first available letter following any other drive letters already in use. Also note that it is never necessary to format a disk created with USSD. The disks are self formatting using the size and address information provided on the CONFIG.SYS invocation line. During initialization, USSD examines the silicon disk to determine if a disk already exists which matches the parameters specified. If so, no action is taken and the disk is used as is. If there is not a disk of the type and size specified, it is created.

4.5 Non-Bootable FLASHDISK Usage

The ATMEL 5 Volt Flash Parts (29C040/29C040A) may also be used as a non bootable drive in a manner nearly identical to the RAMDISK usage described in the previous section. The only change when using USSD for the ATMEL PEROMs is the addition of the /EPT:256 parameter to the CONFIG.SYS line which installs the USSD driver. An example using a 512K EPROM for a ROMDISK and a 512K PEROM device would need the line :

```
device = ussd.sys /mod:p /pad:1ec /seg:e400 /psz:16 /inc:1 /spg:a0 /dsz:512 /ept:256
```

in the CONFIG.SYS file on the floppy to be imaged. This invocation will create a 512K Flash disk in the second socket of the array. Refer to the previous section on non-bootable RAMDISK usage for additional details regarding the USSD driver.

4.6 DiskOnChip Usage

The LBC-Plus supports the M-Systems DISK-ON-CHIP (DOC) flash device in sizes ranging from 1MB to 12MB. The DOC device contains a BIOS extension, the TFFS (Tiny Flash File System), and the Flash memory all in a single 32-pin device. The DOC, unlike the other WinSystems' SSD support for the LBC-Plus, emulates a hard disk rather than a floppy disk. The DOC can be used as a secondary hard disk to a physical IDE drive or it can be the only hard disk in the system.

The DOC is installed into the socket at U23. Refer to the section 2.19 for correct device jumpering and enabling of the DOC.

4.6.1 DOC Initialization

The DOC is initialized in an identical fashion to a fixed disk. DOS is booted (from floppy or hard disk), FDISK is run on the DOC drive (be sure to get the right drive), the system is rebooted and then the DOC is formatted using the DOS format command.

If the /S switch was used during formatting and there is no other fixed disk device specified or attached to the system the DOC will become the boot device. If a hard disk is present, the DOC will become a secondary fixed disk.

5 WS16C48 Programming Reference

5.1 Introduction

This section provides basic documentation for the included I/O routines. It is intended that the accompanying source code equip the programmer with a basic library of I/O functions for the WS16C48 or can serve as the basis from which application specific code can be derived.

5.2 Function Definitions

This section briefly describes each of the functions contained in the driver. Where necessary, short examples will be provided to illustrate usage. Any application making use of any of the driver functions should include the header file “uio48.h”, which includes the function prototypes and the needed constant definitions.

Note that all of the functions utilize the concept of “bit_number”. The “bit_number” is a value from 1 to 48 (1 to 24 for interrupt related functions) that correlates to a specific I/O pin. Bit_number 1 is port 0 bit 0 and continues through to bit_number 48 at port 5 bit 7.

INIT_IO - Initialize I/O, set all ports to input

Syntax

```
void init_io(unsigned io_address);
```

Description

This function takes a single argument :

io_address - The I/O address of the WS16C48 chip.

There is no return value. This function initializes all I/O pins for input (Sets them high), disables all interrupt settings, and sets the image values.

READ_BIT - Reads an I/O port Bit

Syntax

int read_bit(int bit_number);

Description

This function takes a single argument :

bit_number - This is a value from 1 to 48 that indicates the I/O pin to read from.

This function returns the state of the I/O pin. A '1' is returned if the I/O pin is low and a '0' is returned if the pin is high.

WRITE_BIT - Write a 1 or 0 to an I/O pin

Syntax

void write_bit(int bit_number, int value);

Description

This function takes two arguments

bit_number - This is value from 1 to 48, which is the bit to be acted upon.

Value - is either 1 or 0.

This function allows for writing of a single bit to either a '0' or a '1' as specified by the second argument. There is no return value and other bits in the I/O port are not affected.

SET_BIT - Set the specified I/O Bit

Syntax

void set_bit(int bit_number);

Description

This function takes a single argument :

bit_number - a value between 1 and 48 specifying the port bit to be set.

This function sets the specified I/O port bit. Note that setting a bit results in the I/O pin actually going low. There is no return value and other bits in the same I/O port are unaffected.

CLR_BIT - Clear the specified I/O Bit

Syntax

void clr_bit(int bit_number);

Description

This function takes a single argument :

bit_number - a value from 1 to 48 indicates the bit number to clear.

This function clears the specified I/O bit. Note that clearing the I/O bit results in the actual I/O pin going high. This function does not affect any bits other than the one specified.

ENAB_INT - Enable Edge Interrupt, Select Polarity

Syntax

void enab_int(int bit_number, int polarity);

Description

This function requires two arguments :

bit_number - A value from 1 to 24 specifying the appropriate bit

polarity - Specifies rising or falling edge polarity detect. The constants RISING and FALLING are defined in “uio48.h”

This function enables the edge detection circuitry for the specified bit at the specified polarity. It does not unmask the interrupt controller, install vectors, or handle interrupts when they occur. There is no return value and only the specified bit is affected.

DISAB_INT - Disable Edge Detect Interrupt Detection

Syntax

void disab_int(int bit_number);

Description

This function requires a single argument :

bit_number - A value from 1 to 24 specifying the appropriate bit.

This function shuts down the edge detection interrupts for the specified bit. There is no return value and no harm is done by calling this function for a bit which did not have edge detection interrupts enabled. There is no affect on any other bits.

CLR_INT - Clear the specified pending interrupt

Syntax

```
void clr_int(int bit_number);
```

Description

This function requires a single argument :

bit_number - The specified bit number from 1 to 24 to reset the interrupt.

This function clears a pending interrupt on the specified bit. It does this by disabling and reenabling the interrupt. The net result after the call is that the interrupt is no longer pending and is rearmed for the next transition of the same polarity. Calling this function on a bit that has not been enabled for interrupts will result in its interrupt being enabled with an undefined polarity. Calling this function with no interrupt pending will have no adverse affect. Only the specified bit is affected.

GET_INT - Retrieve bit number of pending interrupt

Syntax

```
int get_int(void);
```

Description

This function requires no arguments and returns either a '0' for no bit interrupts pending or a value between 1 and 24 representing a bit number that has a pending edge detect interrupt. The function returns with the first interrupt found and begins its search at Port 0 Bit 0 proceeding through to Port 2 Bit 7. It is necessary to use either clr_int() or disab_int() to avoid returning the same bit continuously. This function may either be used in an application's ISR or can be used in the foreground to poll for bit transitions.

5.3 Sample Programs

There are three sample programs in source code form included on the LBC-Plus diskette in the UIO48 directory. These programs are not useful by themselves but are provided to illustrate the usage of the I/O functions provided in UIO48.C.

FLASH.C

This program was compiled with Borland C/C++ version 3.1 on the command line with :

```
bcc flash.c uio48.c
```

This program illustrates the most basic usage of the WS16C48. It uses three functions from the driver code. The io_init() function is used to initialize the I/O functions and the set_bit() and clr_bit() functions are used to sequence through all 48 bits turning each on and then off in turn.

POLL.C

This program was compiled with Borland C/C++ version 3.1 on the command line with :

```
bcc poll.c uio48.c
```

This program illustrates additional features of the WS16C48 and the I/O library functions. It programs the first 24 bits for input, arms them for falling edge detection, and then polls using the library routine get_int() to determine if any transitions have taken place.

INT.C

This program was compiled with Borland C/C++ version 3.1 on the command line with :

```
bcc int.c uio48.c
```

This program is identical in function to the “poll.c” program except that interrupts are active and all updating of the transition counters is accomplished in the background during the interrupt service routine.

Summary

The source code for all three programs as well as the I/O routines are included on the accompanying diskette. The source code is also provided in printed form in APPENDIX F. These I/O routines along with the sample program provided should provide for a good basis on which to build an application using the features of the WS16C48.

6 APPENDIX A - I/O Port Map

The following is a list of PC I/O ports. Addresses marked with a '-' are not used on the LBC-Plus but their use should be carefully qualified so as not to conflict with other I/O boards. I/O addresses marked with a '+' are used on the LBC-Plus board and are unique to the WinSystems design. I/O Addresses marked with '**' are generally unused and should be the basis for the first choices in I/O address selection.

Hex Range	Usage
000-00F	8237 DMA #1
**010-01F	FREE
020-021	8259 PIC #1
+022-023	Finali 486 Chipset Registers
**024-03F	FREE
040-043	8254 Timer
**044-05F	FREE
060-06F	8042 Keyboard Controller
070-071	CMOS RAM/RTC
**072-07F	FREE
080-08F	DMA Page Registers
**090-09F	FREE
0A0-0BF	8259 PIC #2
0C0-0DF	8237 DMA #2
**0E0-0EF	FREE
0F0-0F1	Coprocessor Control
**0F2-11F	FREE
+120-12F	WS16C48 HDIO
**130-1DF	FREE
+1E0-1EF	SSD, Led, Watchdog control
1F0-1FF	Fixed Disk I/O
-200-20F	Joystick port
-210-21F	PCM SSD I/O Ports
-220-22F	Soundblaster I/O ports
**230-237	FREE
-238-23B	BUS Mouse
**240-277	FREE
278-27F	LPT1
**280-2AF	FREE
-2B0-2DF	EGA Video
-2E0-2E7	GPIP Interface
2E8-2EF	COM4
**2F0-2F7	FREE
2F8-2FF	COM2
-300-31F	Prototype Card
-320-32F	XT Hard Disk
**330-377	FREE
-378-37F	Parallel Printer
-380-3AF	SDLC

-3B0-3BB	DMA
-3C0-3CF	EGA
3E8-3EF	COM3
3F0-3F6	Floppy Disk
3F8-3FF	COM1

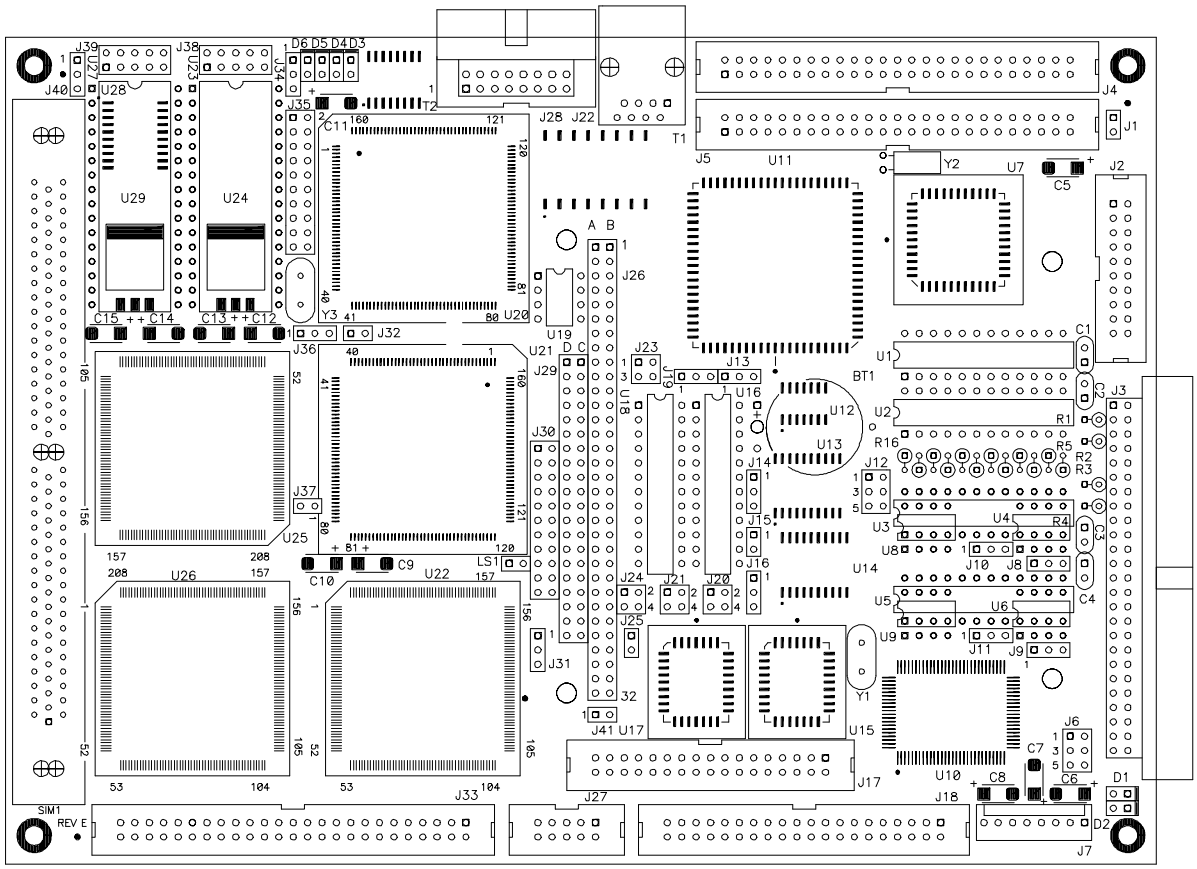
7 APPENDIX B - Interrupt Map

No.	Address	Type	Description
0	00	CPU	Divide by 0
1	04	CPU	Single Step 386 Debug Exception
2	08	CPU	NMI
3	0C	CPU	Breakpoint
4	10	CPU	Overflow
5	14	BIO	Print Screen
		186	Bound Exception
6	18	186	Invalid opcode exception
7	1C	186	Coprocessor unavailable
8	20	Hardware	IRQ0 - 18.2Hz heart beat
		286	LIDT - Double fault exception
9	24	Hardware	IRQ1 - Keyboard interrupt
		286	Coprocessor segment
A	28	Hardware	IRQ2 - XT Reserved, AT-Slaved Controller
		286	Invalid TSS exception
B	2C	Hardware	IRQ3 - COM2
		286	Segment not present
C	30	Hardware	IRQ4 - COM1
		286	Stack fault exception
D	34	Hardware	IRQ5 - XT Hard Disk, AT Free
		286	Protection fault exception
E	38	Hardware	IRQ6 - Floppy Disk Interrupt
		386	Page fault exception
F	3C	Hardware	IRQ7 - LPT1
10	40	BIOS	Video BIOS functions
		286	Coprocessor exception
11	44	BIOS	BIOS Equipment check
		486	Alignment check exception
12	48	BIOS	Memory Size function
13	4C	BIOS	BIOS Disk functions
14	50	BIOS	BIOS serial functions
15	54	BIOS	Cassette/protected mode functions
16	58	BIOS	Keyboard BIOS functions
17	5C	BIOS	BIOS printer functions
18	60	BIOS	SROM Basic Entry point (IBM)
19	64	BIOS	Boot loader function
1A	68	BIOS	BIOS time of day functions
1B	6C	BIOS	Keyboard break vector
1C	70	BIOS	User chained timer tick
1D	74	BIOS	Video Initialization
1E	78	BIOS	Floppy Disk parameter table

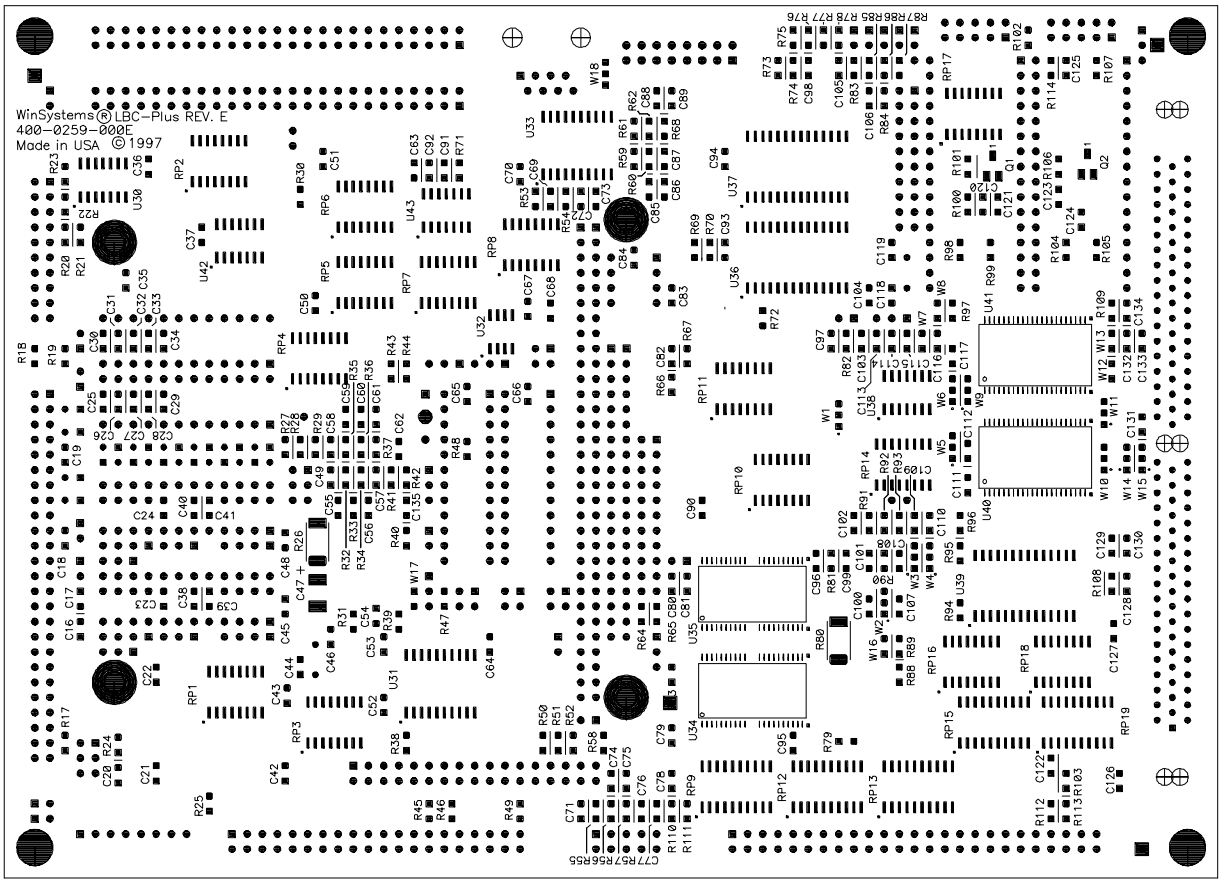
1F	7C	BIOS	CGA graphic character font
20	80	MS-DOS	Program terminate
21	84	MS-DOS	DOS function call
22	88	MS-DOS	Terminate Address
23	8C	MS-DOS	Ctrl-Break Address
24	90	MS-DOS	Fatal Error Vector
25	94	MS-DOS	Absolute disk read
26	98	MS-DOS	Absolute disk write
27	9C	MS-DOS	Terminate
28	A0	MS-DOS	Idle Signal
29	A4	MS-DOS	TTY output
2A	A8	MS-DOS	MS-Net services
2F	BC	MS-DOS	Print Spool
30	C0	MS-DOS	Long jump interface
33	CC	MS-DOS	Mouse functions
3F	FC	MS-DOS	Overlay interrupt
40	100	BIOS	Floppy I/O when fixed disk is present
41	104	BIOS	Fixed disk 1 parameter table
42	108	BIOS	EGA Chain
43	10C	BIOS	EGA Parameter table pointer
44	110	BIOS	EGA graphics character font
4A	128	BIOS	AT Alarm exit address
50	140	BIOS	AT Alarm interrupt
51	144	BIOS	Mouse functions
5A	168	NET	Functions
5B	16C	NET	Boot chain
5C	170	NET	Net BIOS entry
67	19C	MS-DOS	EMS functions
6D	1B4	VGA	VGA Service
70	1C0	Hardware	IRQ8 - Real Time clock
71	1C4	Hardware	IRQ9 - Redirected IRQ2
72	1C8	Hardware	IRQ10 - Unassigned
73	1CC	Hardware	IRQ11 - Unassigned
74	1D0	Hardware	IRQ12 - Unassigned
75	1D4	Hardware	IRQ13 - Unassigned
76	1D8	Hardware	IRQ14 - IDE Fixed Disk
77	1DC	Hardware	IRQ15 - Unassigned
80	200		
F0	3C0	Basic	
F1	3C4		
FF	3FC	Not used	

8 APPENDIX C

LBC-Plus Parts Placement Guide - Top



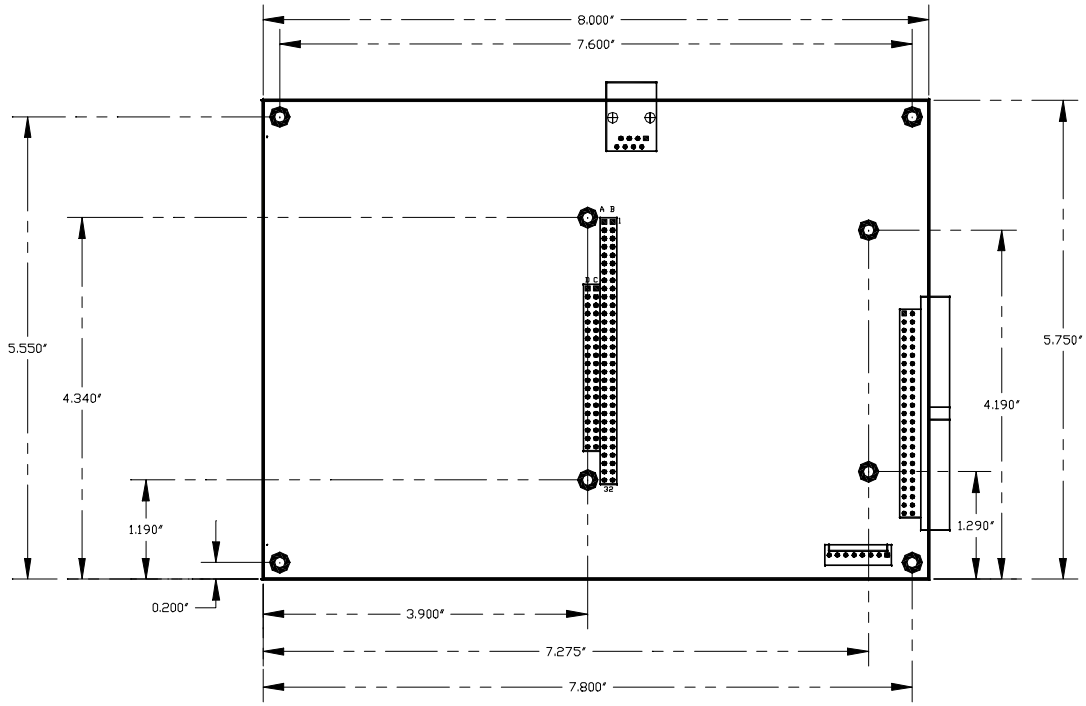
LBC- Plus Parts Placement Guide -Bottom



9 APPENDIX D

LBC-PLUS Mechanical Drawing

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	CHANGED BY



UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES.
TOLERANCE
FRACTIONS ANGLES +/- 1/2°
DECIMALS: .XX ± .03 .XXX ± .010

MACHINE FINISH:

CUSTOMER APPROVAL	DATE	
APPROVAL	DATE	
CHECKER	DATE	
DRAFT/DESIGN M.BROWNING	DATE 03/10/97	

WinSystems, Inc.
"THE STD BUS AUTHORITY"

LBC-PLUS
MOUNTING DIMENSIONS

SIZE	CAGE CODE	DWG NO.	REV
C	1AU87	LBCPLUS.DWG	A

SCALE | CAD ID: LBCPLUS.DWG | SHEET 1 OF 1

10 APPENDIX E

WS16C48 I/O Routines and Sample Program Listings

```
/* UIO48.H

        Copyright 1996 by WinSystems Inc.

        Permission is hereby granted to the purchaser of the WinSystems
        UIO cards and CPU products incorporating the UIO device, to distribute
        any binary file or files compiled using this source code directly or
        in any work derived by the user from this file. In no case may the
        source code, original or derived from this file, be distributed to any
        third party except by explicit permission of WinSystems. This file is
        distributed on an "As-is" basis and no warranty as to performance,
        fitness of purposes, or any other warranty is expressed or implied.
        In no case shall WinSystems be liable for any direct or indirect loss
        or damage, real or consequential resulting from the usage of this
        source code. It is the user's sole responsibility to determine
        fitness for any considered purpose.
*/
/*************************************************************************
*       Name     : uio48.h
*
*       Project  : PCM-UIO48 Software Samples/Examples
*
*       Date     : October 30, 1996
*
*       Revision: 1.00
*
*       Author   : Steve Mottin
*
*************************************************************************
*
*       Changes :
*
*       Date              Revision Description
*       ________ ________ ___________________________________________
*       10/30/96 1.00              Created
*
*************************************************************************
*/

#define RISING 1
#define FALLING 0

void init_io(unsigned io_address);
int read_bit(int bit_number);
void write_bit(int bit_number);
void set_bit(int bit_number);
void clr_bit(int bit_number);
void enab_int(int bit_number, int polarity);
void disab_int(int bit_number);
void clr_int(int bit_number);
int get_int(void);
```

```
/* UIO48.C

        Copyright 1996 by WinSystems Inc.

        Permission is hereby granted to the purchaser of the WinSystems
        UIO cards and CPU products incorporating the UIO device, to distribute
        any binary file or files compiled using this source code directly or
        in any work derived by the user from this file. In no case may the
        source code, original or derived from this file, be distributed to any
        third party except by explicit permission of WinSystems. This file is
        distributed on an "As-is" basis and no warranty as to performance,
        fitness of purposes, or any other warranty is expressed or implied.
        In no case shall WinSystems be liable for any direct or indirect loss
        or damage, real or consequential resulting from the usage of this
        source code. It is the user's sole responsibility to determine
        fitness for any considered purpose.
*/
/*************************************************************************
*       Name     : uio48.c
*
*       Project  : PCM-UIO48 Software Samples/Examples
*
*       Date     : October 30, 1996
*
*       Revision: 1.00
*
*       Author   : Steve Mottin
*
*************************************************************************
*
*       Changes :
*
*       Date              Revision Description
*       ________ ________ ___________________________________________
*       10/30/96 1.00              Created
*
*************************************************************************
*/

#include <dos.h>

/* This global holds the base address of the UIO chip */

unsigned base_port;

/* This global array holds the image values of the last write to each I/O
   ports. This allows bit manipulation routines to work without having to
   actually do a read-modify-write to the I/O port.
*/

unsigned port_images[6];

/*=======================================================================
*                               INIT_IO
*
*  This function take a single argument :
*
*
*  io_address  :    This is the base I/O address of the 16C48 UIO Chip
*                                          on the board.
*
*
*  This function initializes all I/O pins for input, disables all interrupt
*  sensing, and sets the image values.
*
*=======================================================================*/

void init_io(unsigned io_address)
{
int x;

        /* Save the specified address for later use */

        base_port = io_address;

        /* Clear all of the I/O ports. This also makes them inputs */

        for(x=0; x < 7; x++)
                outportb(base_port+x, 0);

        /* Clear our image values as well */

        for(x=0; x < 6; x++)
                port_images[x] = 0;

        /* Set page 2 access, for interrupt enables */

        outportb(base_port+7,0x80);

        /* Clear all interrupt enables */

        outportb(base_port+8,0);
        outportb(base_port+9,0);
        outportb(base_port+0x0a,0);

        /* Restore normal page 0 register access */
        outportb(base_port+7,0);
```

```
}

/*=========================================================================
*
*                                READ_BIT
*
*
*  This function takes a single argument :
*
*
*  bit_number    :  The integer argument specifies the bit number to read.
*                   Valid arguments are from 1 to 48.
*
*  return value : The current state of the specified bit, 1 or 0.
*
*  This function returns the state of the current I/O pin specified by
*  the argument bit_number.
*
*=========================================================================*/

int read_bit(int bit_number)
{
unsigned port;
int val;

         /* Adjust the bit_number to 0 to 47 numbering */

         --bit_number;

         /* Calculate the I/O port address based on the updated bit_number */

         port = (bit_number / 8) + base_port;

         /* Get the current contents of the port */

         val = inportb(port);

         /* Get just the bit we specified */

         val = val & (1 << (bit_number % 8));

         /* Adjust the return for a 0 or 1 value */

         if(val)
                  return 1;

         return 0;
}

/*=========================================================================
*
*                                WRITE_BIT
*
* This function takes two arguments :
*
*
*  bit_number :  The I/O pin to access is specified by bit_number 1 to 48.
*
*  val  :   The setting for the specified bit, either 1 or 0.
*
*  This function sets the specified I/O pin to either high or low as dictated
*  by the val argument. A non zero value for val sets the bit.
*
*=========================================================================*/

void write_bit(int bit_number, int val)
{
unsigned port;
unsigned temp;
unsigned mask;


         /* Adjust bit_number for 0 based numbering */

         --bit_number;

         /* Calculate the I/O address of the port based on the bit number */

         port = (bit_number / 8) + base_port;

         /* Use the image value to avoid having to read the port first. */

         temp = port_images[bit_number / 8];/* Get current value */

         /* Calculate a bit mask for the specified bit */

         mask = (1 << (bit_number % 8));

         /* Check whether the request was to set or clear and mask accordingly */

         if(val)           /* If the bit is to be set */
                  temp = temp | mask;
         else
                  temp = temp & ~mask;

         /* Update the image value with the value we're about to write */

         port_images[bit_number / 8] = temp;
```

```
        /* Now actually update the port. Only the specified bit is affected */

        outportb(port,temp);
}

/*=========================================================================
*                                SET_BIT
*
*
*  This function takes a single argument :
*
*  bit_number :  The bit number to set.
*
*  This function sets the specified bit.
*
*=========================================================================*/

void set_bit(int bit_number)
{

        write_bit(bit_number,1);
}

/*=========================================================================
*                                CLR_BIT
*
*
*  This function takes a single argument :
*
*  bit_number :  The bit number to clear.
*
*  This function clears the specified bit.
*
*=========================================================================*/

void clr_bit(int bit_number)
{
        write_bit(bit_number,0);
}

/*=========================================================================
*
*                              ENAB_INT
*
*  This function takes two arguments :
*
*  bit_number  : The bit number to enable intterups for. Range from 1 to 48.
*
*  polarity   : This specifies the polarity of the interrupt. A non-zero
*               argument enables rising-edge interrupt. A zero argument
*               enables the interrupt on the flling edge.
*
*  This function enables within the 16C48 an interrupt for the specified bit
*  at the specified polarity. This function does not setup the interrupt
*  controller, nor does it supply an interrupr handler.
*
*=========================================================================*/

void enab_int(int bit_number, int polarity)
{
unsigned port;
unsigned temp;
unsigned mask;

        /* Adjust for 0 based numbering */

        --bit_number;

        /* Calculate the I/O address based uppon the bit number */

        port = (bit_number / 8) + base_port + 8;

        /* Calculate a bit mask based on the specified bit number */

        mask = (1 << (bit_number % 8));

        /* Turn on page 2 access */

        outportb(base_port+7,0x80);

        /* Get the current state of the interrupt enable register */

        temp = inportb(port);

        /* Set the enable bit for our bit number */

        temp = temp | mask;

        /* Now update the interrupt enable register */

        outportb(port,temp);

        /* Turn on access to page 1 for polarity control */

        outportb(base_port+7,0x40);

        /* Get the current state of the polarity register */
```

```
        temp = inportb(port);              /* Get current polarity settings */

        /* Set the polarity according to the argument in the image value */

        if(polarity)              /* If the bit is to be set */
                temp = temp | mask;
        else
                temp = temp & ~mask;

        /* Write out the new polarity value */

        outportb(port,temp);

        /* Set access back to Page 0 */

        outportb(base_port+7,0x0);

}

/*=========================================================================
*
*                               DISAB_INT
*
* This function takes a single argument :
*
* bit_number : Specifies the bit number to act upon. Range is from 1 to 48.
*
* This function shuts off the interrupt enabled for the specified bit.
*
*=========================================================================*/

void disab_int(int bit_number)
{
unsigned port;
unsigned temp;
unsigned mask;

        /* Adjust the bit_number for 0 based numbering */

        --bit_number;

        /* Calculate the I/O Address for the enable port */

        port = (bit_number / 8) + base_port + 8;

        /* Calculate the proper bit mask for this bit number */

        mask = (1 << (bit_number % 8));

        /* Turn on access to page 2 registers */

        outportb(base_port+7,0x80);

        /* Get the current state of the enable register */

        temp = inportb(port);

        /* Clear the enable bit int the image for our bit number */

        temp = temp & ~mask;

        /* Update the enable register with the new information */

        outportb(port,temp);

        /* Set access back to page 0 */

        outportb(base_port+7,0x0);

}

/*=========================================================================
*
*                               CLR_INT
*
*  This function takes a single argument :
*
*  bit_number : This argument specifies the bit interrupt to clear. Range
*               is 1 to 24.
*
*
*  This function is use to clear a bit interrupt once it has been recognized.
*  The interrupt left enabled.
*
*=========================================================================*/

void clr_int(int bit_number)
{
unsigned port;
unsigned temp;
unsigned mask;


        /* Adjust for 0 based numbering */

        --bit_number;
```

```
        /* Calculate the correct I/O address for our enable register */

        port = (bit_number / 8) + base_port + 8;

        /* Calculate a bit mask for this bit number */

        mask = (1 << (bit_number % 8));

        /* Set access to page 2 for the enable register */

        outportb(base_port+7,0x80);

        /* Get current state of the enable register */

        temp = inportb(port);

        /* Temporarily clear only OUR enable. This clears the interrupt */

        temp = temp & ~mask;              /* clear the enable for this bit */

        /* Write out the temporary value */

        outportb(port,temp);

        /* Re-enable our interrupt bit */

        temp = temp | mask;

        /* Write it out */

        outportb(port,temp);

        /* Set access back to page 0 */

        outportb(base_port+7,0x0);

}


/*=========================================================================
*
*                               GET_INT
*
*  This function take no arguments.
*
*  return value : The value returned is the highest level bit interrupt
*                 currently pending. Range is 1 to 24.
*
*  This function returns the highest level interrupt pending. If no interrupt
*  is pending, a zero is returned. This function does NOT clear the interrupt.
*
*=========================================================================*/

int get_int(void)
{
int temp;
int x;

        /* read the master interrupt pending register, mask off undefined bits */

        temp = inportb(base_port+6) & 0x07;

        /* If there are no interrupts pending, return a 0 */

        if((temp & 7) == 0)
                return(0);

        /* There is something pending, now we need to identify what it is */


        /* Set access to page 3 for interrupt id registers */

        outportb(base_port+7,0xc0);

        /* Read interrupt ID register for port 0 */

        temp = inportb(base_port+8);


        /* See if any bit set, if so return the bit number */

        if(temp !=0)
        {
                for(x=0; x <=7; x++)
                {
                        if(temp & (1 << x))
                        {
                                outportb(base_port+7,0);  /* Turn off access */
                                return(x+1);              /* Return bitnumber with active int */
                        }
                }
        }

        /* None in Port 0, read port 1 interrupt ID register */

        temp = inportb(base_port+9);

        /* See if any bit set, if so return the bit number */
```

```
        if(temp !=0)
        {
                for(x=0; x <=7; x++)
                {
                        if(temp & (1 << x))
                        {
                                outportb(base_port+7,0);  /* Turn off access */
                                return(x+9);              /* Return bitnumber with active int */
                        }
                }
        }

        /* Lastly, read status of port 2 int id */

        temp = inportb(base_port+0x0a);       /* Read port 2 status */

        /* If any pending, return the appropriate bit number */

        if(temp !=0)
        {
                for(x=0; x <=7; x++)
                {
                        if(temp & (1 << x))
                        {
                                outportb(base_port+7,0);  /* Turn off access */
                                return(x+17);             /* Return bitnumber with active int */
                        }
                }
        }

        /* We should never get here unless the hardware is misbehaving but just
            to be sure. We'll turn the page access back to 0 and return a 0 for
            no interrupt found.
        */


        outportb(base_port+7,0);
        return 0;
}
```

```
/* FLASH.C

        Copyright 1996 by WinSystems Inc.

        Permission is hereby granted to the purchaser of the WinSystems
        UIO cards and CPU products incorporating the UIO device, to distribute
        any binary file or files compiled using this source code directly or
        in any work derived by the user from this file. In no case may the
        source code, original or derived from this file, be distributed to any
        third party except by explicit permission of WinSystems. This file is
        distributed on an "As-is" basis and no warranty as to performance,
        fitness of purposes, or any other warranty is expressed or implied.
        In no case shall WinSystems be liable for any direct or indirect loss
        or damage, real or consequential resulting from the usage of this
        source code. It is the user's sole responsibility to determine
        fitness for any considered purpose.
*/

#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include "uio48.h"

/* This is where we have our board jumpered to */

#define BASE_PORT 0x200


/* This is an utlra-simple demonstration program of some of the functions
   available in the UIO48 source code library. This program simply sets and
   clears each I/O line in succession. It was tested by hooking LEDs to all
   of the I/O lines and wathching the lit one race through the bits.
*/

void main()
{
int x;


        /* Initialize all I/O bits, and set then for input */

        init_io(BASE_PORT);

        /* We'll repeat our sequencing until a key is pressed */

        while(!kbhit())
        {
                /* We will light the LED attached to each of the 48 lines */
                for(x=1; x <=48; x++)
                {
                        /* Setting the bit lights the LED */
                        set_bit(x);
                        /* The wait time is subjective. We liked 100mS */
                        delay(100);
                        /* Now turn off the LED */
                        clr_bit(x);
                }
        }
        getch();
}
```

```
/* POLL.C

        Copyright 1996 by WinSystems Inc.

        Permission is hereby granted to the purchaser of the WinSystems
        UIO cards and CPU products incorporating the UIO device, to distribute
        any binary file or files compiled using this source code directly or
        in any work derived by the user from this file. In no case may the
        source code, original or derived from this file, be distributed to any
        third party except by explicit permission of WinSystems. This file is
        distributed on an "As-is" basis and no warranty as to performance,
        fitness of purposes, or any other warranty is expressed or implied.
        In no case shall WinSystems be liable for any direct or indirect loss
        or damage, real or consequential resulting from the usage of this
        source code. It is the user's sole responsibility to determine
        fitness for any considered purpose.

*/

#include <stdio.h>
#include <conio.h>
#include "uio48.h"


#define BASE_PORT 0x200

/* This program uses the edge detection interrupt capability of the
   WS16C48 to count transitions on the first 24 lines. It does this
   however, no by using true interrupts but by polling for transitions
   using the get_int() function.

*/


/* Our transition totals are stored in this array */


unsigned int_counts[25];

/* Definitions for local functions */

void check_ints(void);

void main()
{
int x;


        /* Initialize the I/O ports.  Set all I/O pins to input */

        init_io(BASE_PORT);


        /* Initialize our transition counts, and enable falling edge
           transition interrupts.
        */

        for(x=1; x<25; x++)
        {
                int_counts[x] = 0;        /* Clear the counts */
                enab_int(x,FALLING);      /* Enable the falling edge interrupts */

        }

        /* Clean up the screen for our display. Nothing fancy */
        clrscr();

        for(x=1; x<25; x++)
        {
                gotoxy(1,x);
                printf("Bit number %02d ",x);
        }


        /* We will continue to display until any key is pressed */

        while(!kbhit())
        {

                /* Retrieve any pending transitions and update the counts */

                check_ints();

                /* Display the current count values */

                for(x=1; x < 25; x++)
                {
                        gotoxy(16,x);
                        printf("%05u",int_counts[x]);
                }
        }
        getch();
}

void check_ints()
{
int current;
```

```
        /* Get the bit number of a pending transition interrupt */

        current = get_int();

        /* If it's 0 there are none pending */

        if(current == 0)
                return;

        /* Clear and rearm this one so we can get it again */

        clr_int(current);

        /* Tally a transition for this bit */

        ++int_counts[current];
}
```

```
/* INTS.C

         Copyright 1996 by WinSystems Inc.

         Permission is hereby granted to the purchaser of the WinSystems
         UIO cards and CPU products incorporating the UIO device, to distribute
         any binary file or files compiled using this source code directly or
         in any work derived by the user from this file. In no case may the
         source code, original or derived from this file, be distributed to any
         third party except by explicit permission of WinSystems. This file is
         distributed on an "As-is" basis and no warranty as to performance,
         fitness of purposes, or any other warranty is expressed or implied.
         In no case shall WinSystems be liable for any direct or indirect loss
         or damage, real or consequential resulting from the usage of this
         source code. It is the user's sole responsibility to determine
         fitness for any considered purpose.

*/

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include "uio48.h"

#define BASE_PORT 0x200

/* This program like the poll.c sample uses the edge detection interrupt
   capability of the WS16C48 to count edge transitions. Unlike poll.c,
   however this program actually uses interrupts and update all of the
   transition counters in the background.

*/


/* Our transition totals are stored in this global array */

unsigned int_counts[25];

/* Function declarations for local functions */

void check_ints(void);
void interrupt int_handler(void);
void interrupt (*old_handler)(void);

void main()
{
int x;

         /* Initialize the I/O ports. Set all I/O pins to input */

         init_io(BASE_PORT);

         /* Install an interrupt handler for the board */


         /* We disable interrupts whenever we're changing the environment */

         disable();        /* Disable interrupts during initialization */

         /* Get the old handler and save it for later resoration */

         old_handler = getvect(0x0d);       /* Hardwired for IRQ5 */

    /* Install out new interrupt handler */

         setvect(0x0d,int_handler);

         /* Clear the transition count values and enable the falling edge
            interrupts.
         */

         for(x=1; x<25; x++)
         {
                  int_counts[x] = 0;         /* Clear the counts */
                  enab_int(x,FALLING);       /* Enable the falling edge interrupts */
         }


         /* Unmask the interrupt controller */

         outportb(0x21,(inportb(0x21) & 0xdf));      /* Unmask IRQ 5 */

         /* Reenable interrupts */
         enable();


         /* Set up the display */

         clrscr();         /* Clear the Text Screen */

         for(x=1; x<25; x++)
         {
                  gotoxy(1,x);
                  printf("Bit Number %02d  ",x);
         }


         /* We will continuously print the transition totals until a
```

```
            key is pressed */


         /* All of the processing of the transition interrupts, including
            updating the counts is done in the background when an interrupt
            occurs.
         */

         while(!kbhit())
         {
                 for(x=1; x < 25; x++)
                 {
                         gotoxy(16,x);
                         printf("%05u",int_counts[x]);
                 }
         }

         getch();

         /* Disable interrupts while we restore things */

         disable();

         /* Mask off the interrupt at the interrupt controller */

         outportb(0x21,inportb(0x21) | 0x20);        /* Mask IRQ 5 */

         /* Restore the old handler */

         setvect(0x0d,old_handler); /* Put back the old interrupt handler */

         /* Reenable interrupts. Things are back they way they were before we
            started.
    */

         enable();
}


/* This function is executed when an edge detection interrupt occurs */

void interrupt int_handler(void)
{
int current;

         /* Get the current interrupt pending. There really should be one
            here or we shouldn't even be executing this function.
    */

         current = get_int();

         /* We will continue processing pending edge detect interrupts until
            there are no more present. In which case current == 0
    */

         while(current)
         {

                 /* Clear the current one so that it's ready for the next edge */

                 clr_int(current);

                 /* Tally up one for the current bit number */

                 ++int_counts[current];

                 /* Get the next one, if any others pending */

                 current = get_int();
         }

         /* Issue a non-specific end of interrupt command (EOI) to the
            interrupt controller. This rearms it for the next shot.
         */

         outportb(0x20,0x20);      /* Do non-specific EOI */
}
```

WARRANTY

WinSystems warrants that for a period of two (2) years from the date of shipment any Products and Software purchased or licensed hereunder which have been developed or manufactured by WinSystems shall be free of any material defects and shall perform substantially in accordance with WinSystems' specifications therefore. With respect to any Products or Software purchased or licensed hereunder which have been developed or manufactured by others, WinSystems shall transfer and assign to Customer any warranty of such manufacturer or developer held by WinSystems, provided that the warranty, if any, may be assigned. The sole obligation of WinSystems for any breach of warranty contained herein shall be, at its option, either (i) to repair or replace at its expense any materially defective Products or Software, or (ii) to take back such Products and Software and refund the Customer the purchase price and any license fees paid for the same. Customer shall pay all freight, duty, broker's fees, insurance changes and other fees and charges for the return of any Products or Software to WinSystems under this warranty. WinSystems shall pay freight and insurance charges for any repaired or replaced Products or Software thereafter delivered to Customer within the United States. All fees and costs for shipment outside of the United States shall be paid by Customer. The foregoing warranty shall not apply to any Products or Software which have been subject to abuse, misuse, vandalism, accidents, alteration, neglect, unauthorized repair or improper installations.

THERE ARE NO WARRANTIES BY WINSYSTEMS EXCEPT AS STATED HEREIN. THERE ARE NO OTHER WARRANTIES EXPRESS OR IMPLIED INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, IN NO EVENT SHALL WINSYSTEMS BE LIABLE FOR CONSEQUENTIAL, INCIDENTAL, OR SPECIAL DAMAGES INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF DATA, PROFITS OR GOODWILL. WINSYSTEMS' MAXIMUM LIABILITY FOR ANY BREACH OF THIS AGREEMENT OR OTHER CLAIM RELATED TO ANY PRODUCTS, SOFTWARE, OR THE SUBJECT MATTER HEREOF, SHALL NOT EXCEED THE PURCHASE PRICE OR LICENSE FEE PAID BY CUSTOMER TO WINSYSTEMS FOR THE PRODUCTS OR SOFTWARE OR PORTION THEREOF TO WHICH SUCH BREACH OR CLAIM PERTAINS.

WARRANTY SERVICE

All products returned to WinSystems must be assigned a Return Material Authorization (RMA) number. To obtain this number, please call or FAX WinSystems' factory in Arlington, Texas and provide the following information:

1. Description and quantity of the product(s) to be returned including its serial number.
2. Reason for the return.
3. Invoice number and date of purchase (if available), and original purchase order number.
4. Name, address, telephone and FAX number of the person making the request.
5. Do not debit WinSystems for the repair. WinSystems does not authorize debits.

After the RMA number is issued, please return the products promptly. Make sure the RMA number is visible on the outside of the shipping package.

The customer must send the product freight prepaid and insured. The product must be enclosed in an anti-static bag to protect it from damage caused by static electricity. Each bag must be completely sealed. Packing material must separate each unit returned and placed as a cushion between the unit(s) and the sides and top of the shipping container. WinSystems is not responsible for any damage to the product due to inadequate packaging or static electricity.