

Technical Manual MT488



by

TIDAL

TIDAL ENGINEERING
9 Whippany Rd
Whippany, NJ 07981

By: Craig Borax
Date: July 1997
Rev. Original

Table of Contents

1.0 OVERVIEW.....	1
2.0 SPECIFICATIONS	3
2.1 THE IEEE-488 INTERFACE	3
2.2 RS232 INTERFACE	3
2.3 A/D CONVERTER	3
2.4 D/A CONVERTER SPECIFICATIONS	4
2.5 DIGITAL I/O	4
2.6 SMARTCORE(TM) SPECIFICATIONS	5
2.7 INDICATORS AND CONTROLS.....	6
2.8 PIN DESCRIPTIONS.....	6
2.9 PHYSICAL DESCRIPTION.....	9
3.0 FUNCTIONAL DESCRIPTION.....	10
3.1 IEEE-488 INTERFACE	10
3.2 RS-232 INTERFACE	10
3.3 A/D CONVERTER	10
3.4 D/A CONVERTER	11
3.5 DIGITAL I/O	12
3.6 SMART CORE FEATURES	13
3.7 POWER SUPPLY FEATURES	13
4.0 OPERATING INSTRUCTIONS.....	14
4.1 UNPACKING	14
4.2 SETUP AND CONFIGURATION.....	14
4.3 SOFTWARE INSTALLATION.....	15
4.3.1 <i>Dynamic C</i>	15
4.3.2 <i>Libraries</i>	15
4.3.3 <i>Example Program</i>	15
5.0 PROGRAMMING.....	16
5.1 MT488 SPECIFIC DRIVERS.....	16
<i>ibloc</i> <MT488.LIB>	16
<i>ieee_488</i> <MT488.LIB>.....	16
<i>process_esr_stb</i> <MT488.LIB>	16
<i>ibren</i> <MT488.LIB>.....	16
<i>ibdev</i> <MT488.LIB>	16
<i>ibclr</i> <MT488.LIB>	17
<i>init_488</i> <MT488.LIB>.....	17
<i>ibrd</i> <MT488.LIB>	17
<i>ibwrt</i> <MT488.LIB>.....	17
<i>wait_sync</i> <MT488.LIB>	17
<i>wait4co</i> <MT488.LIB>.....	17
<i>ieee_cmd</i> <MT488.LIB>	17
<i>dspCIC</i> <MT488.LIB>	18
<i>read_a2d</i> <MT488.LIB>.....	18
<i>set_da</i> <MT488.LIB>.....	18
<i>wr_pio</i> <MT488.LIB>	18
<i>rd_pio</i> <MT488.LIB>	18
<i>rs232_send</i> <MT488.LIB>	18

ieee_send <MT488.LIB>18

address_changed <MT488.LIB>18

5.2.2 EXAMPLE PROGRAM SOURCE(LECROYA.C)19

Setup Serial Port interrupt for LCDkey interface.....19

Setup transmit and receive buffers for the RS-232 Interface19

Define user interface codes19

Define general purpose constants.....19

Define Constants for hardware and Chip selects19

Define Global Variables.....19

Define Other RS-232 parameters.....19

Connect libraries20

Define IEEE 488 buffers20

Define GPIB device structure.....20

Digital IO Test Function20

Sine Generation Function.....20

Floating Point Correction function.....21

tprintf function21

Keyboard Input Function22

Main Function.....22

Lecroy command string definitions.....22

Watchdog bypass.....23

Main Loop.....23

5.2 EXAMPLE PROGRAM (LECROYA.C)26

6.0 TROUBLESHOOTING.....31

7.0 DRAWINGS AND SCHEMATICS33

7.1 MT488 PARTS LIST.....33

List of Figures

FIGURE 1, MT488 BLOCK DIAGRAM.....	2
FIGURE 2, SMARTCORE BLOCK DIAGRAM	5
FIGURE 3, IEEE-488 ADDRESS SWITCH	6
FIGURE 4, SMARTCORE BLOCK DIAGRAM	13
FIGURE 5 MT488 INTERFACES.....	14
FIGURE 6, MT488 COMPONENT LAYOUT	36

SAFETY

WARNING

High voltage AC and DC exists on the MT488. Do not open the unit with power on. Failure to observe this warning could result in serious injury or death.

1.0 OVERVIEW

The MT488 is a unique hand-held IEEE 488 controller which incorporates a C programmable Microprocessor and a plethora of input and output capabilities.

The IEEE-488 interface from which the MT488 gets its name, implements the Talker/Listener/Controller functions of the IEEE 488 interface, a.k.a. the General Purpose Interface Bus (GPIB). See Figure 1, MT488 Block Diagram. The IEEE-488 bus is designed to allow up to 15 instruments within a localized area to communicate with each other over a common bus. Each device has a unique address, read from a set of address switches at power up, that it responds to. The MT488 has a five position dip switch used to set the address on the MT488. Data is sent byte serial/bit parallel and consists of device dependent and interface messages.

In addition to the 488 interface the MT488 also supports two RS232 communication ports at 9600 Baud. The RS232 interface is an asynchronous serial interface. Commands are received and data is sent by the MT488 in bit serial format.

The analog input on the MT488 consists of an 11 Channel, 12 Bit A/D (Analog to Digital) converter with 2.5 or 5.0 Volts Full Scale. The A/D converter is isolated from the microprocessor and the communications interfaces with high speed opto-isolators affording a 2500 VAC isolation barrier.

The analog output on the MT488 consists of four 12 Bit D/A (Digital to Analog) converters with 2.5 Volts Full Scale. The D/A converter is isolated from the microprocessor and the communications interfaces with high speed opto-isolators affording a 2500 VAC isolation barrier.

The MT488's digital I/O is capable of 32 Bits of Bi-directional digital Input and output. The four 8 bit ports are also isolated from the microprocessor and the communications interfaces.

The MT488 incorporates a microprocessor module from Z-World Engineering which provides a compact and powerful Central Processing Unit (CPU). The SmartCore features a Z180 CPU running at 9.216 Mhz.

The SmartCore onboard memory consists of:

- a. SRAM, 32K to 128K Battery Backed
- b. EPROM, 32K to 128K
- c. EEPROM, 512 bytes

Other SmartCore peripherals are:

- d. Real-time Clock
- e. Reset and Power Failure Monitor
- f. Serial Interface for program development
- g. Serial Interface for the application.
- h. Two programmable timers.
- i. Two DMA channels.
- j. Memory decoding for external devices (/CS1-/CS6).

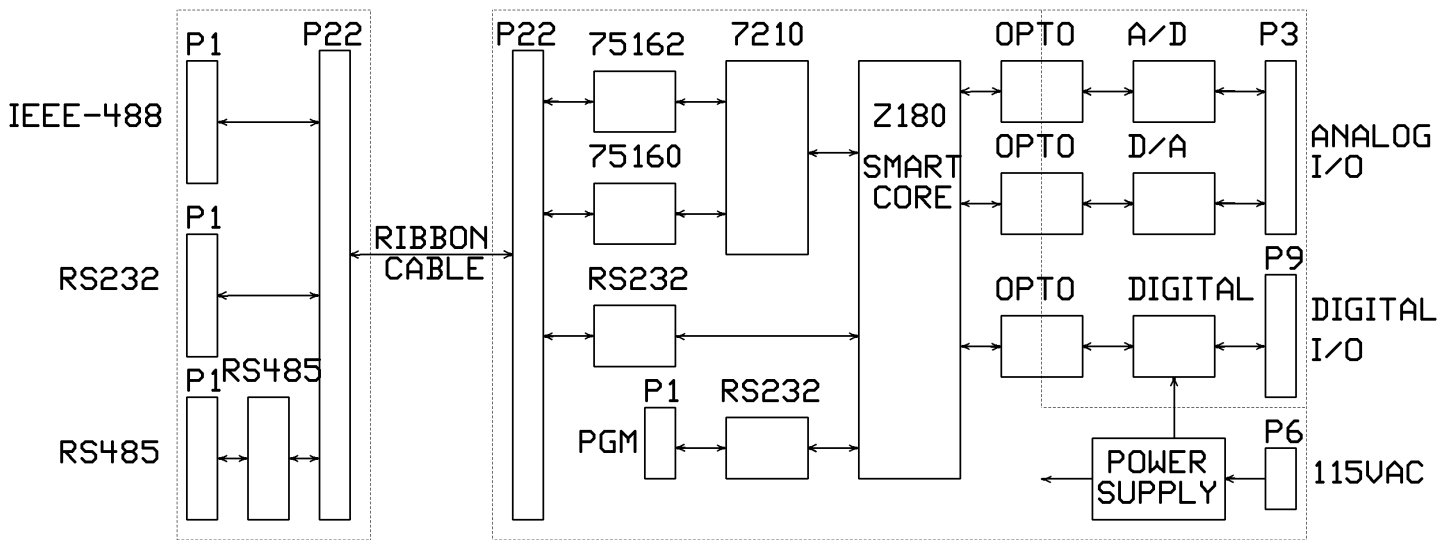


Figure 1, MT488 Block Diagram

2.0 Specifications

2.1 The IEEE-488 Interface

Specification: IEEE-488.1 and IEEE-488.2 (partial)
 Controller: NEC uPD7210 or National Instruments 7210
 Controller Clk: 8 Mhz
 Transceivers: 75160 and 75162 (Texas Instruments)
 Address Capability: 0 to 30
 SRQ generation: ESE,SRE masks (See section 5, programming)

2.2 RS232 Interface

RS232 communications
 Baud Rate: 9600 Baud
 Data Size: 8 bit
 Parity: None
 Stop bits: 1
 Connector Configuration: 9 Pin D-Sub, IBM Pinout

2.3 A/D Converter

Parameter	
Channels	11
Resolution	12 Bit
A/D P/N	TLC2543
Linearity Error	+/-1 LSB max
Gain Error	+/-1 LSB max
Offset Error	+/-1.5 LSB max

Voltage Reference

Parameter	
Reference Voltage	2.50-5.0 VDC
Reference P/N	LM4040-5.0
Reference Accuracy	+/- 0.5% max
Temperature Coeff.	+/- 100 ppm/°C

Isolation: 2500 VAC (Between CPU and I/O)

Note: The A/D converter section uses a separate LM4040 voltage reference which is available in 2.5, 4.096 and 5.0 VDC versions. By changing the voltage reference to a different version full scale voltage ranges of 2.5, 4.096 and 5.0 VDC can be accommodated. The data above is for the 2.5 VDC volt reference.

2.4 D/A Converter Specifications

Parameter	
Channels	6 Max
Resolution	12 Bit 0.61mV @ 2.5 VDC f.s.
D/A P/N	LTC1257
Offset Error	+/-5mV max
Total Error	+/-2 LSB max
Output Impedance	300 ohm max
Rated load	2 ma 100pf
Temperature Coeff.	+/- 0.02 LSB/°C max

Voltage Reference

Parameter	
Reference Voltage	2.50 VDC
Reference P/N	LM4040-2.5
Reference Accuracy	+/- 0.5% max
Temperature Coeff.	+/- 100 ppm/°C

Isolation: 2500 VAC (Between CPU and I/O)

Note: Like the A/D section, the D/A converter section uses a separate LM4040 voltage reference which is available in voltage ranges of 2.5, 4.096, 5.0 and 10 volts. By using a different version of the chip different full scale voltage ranges can be accommodated. The data above is for the 2.5 VDC volt reference.

2.5 Digital I/O

Parameter	
Ports	4 eight bit ports (32 bits total)
Type	Quasi Bi-directional
Port P/N	Phillips PCF8574
Output Voltage	5.0 VDC typ.
Input Voltage Range	-0.5 to 5.5 VDC
Output Current Low	10 ma Min.
Output Current High	30 uA Min
Input Voltage Low	1.5 VDC Max
Input Voltage High	3.5 VDC Min

Isolation: 2500 VAC (Between CPU and I/O)

2.6 SmartCore(tm) Specifications

The MT488 incorporates a microprocessor module from Z-World Engineering which provides a compact and powerful Central Processor core module (Figure 2, SmartCore block diagram). The SmartCore(tm) features:

- Z180 CPU running at 9.216 Mhz.
- Static Ram (SRAM), 32K to 128K (Battery Backed) for data storage.
- Erasable Programmable Read Only Memory (EPROM), 32K to 128K for program storage.
- Electrically Erasable PROM (EEPROM), 512 bytes, for non-volatile storage without a battery.
- Real-time Clock, for time calculations.
- Reset and Power Failure Supervisor to provide reset to the microprocessor if power is out of range and to switch in the battery back-up if there is a battery installed.
- Serial Interface for program development, Port 1(Not used with EPROM).
- Serial Interface for the application, Port 0.
- Two programmable timers used as interval timers.
- Two Direct Memory Access (DMA) channels for high speed transfers to internal (Serial Ports) and external I/O.
- Memory decoding for external devices (/CS1-/CS6).

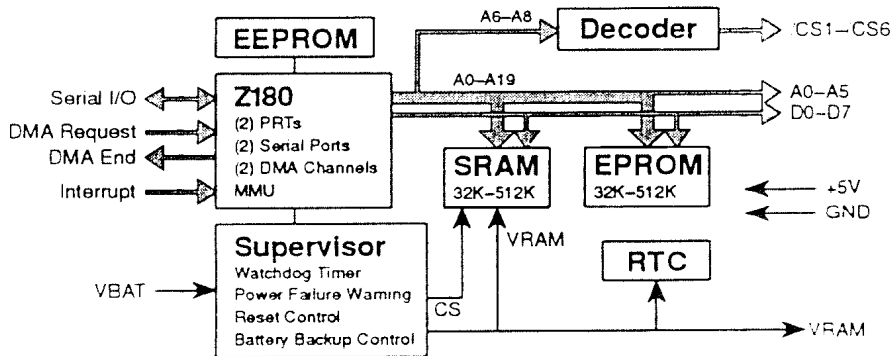


Figure 2, SmartCore Block Diagram

2.7 Indicators and Controls

a. Address switch

The IEEE-488 Address is programmed via the 5 position dip switch on the rear of the MT488. The address is configured as follows:

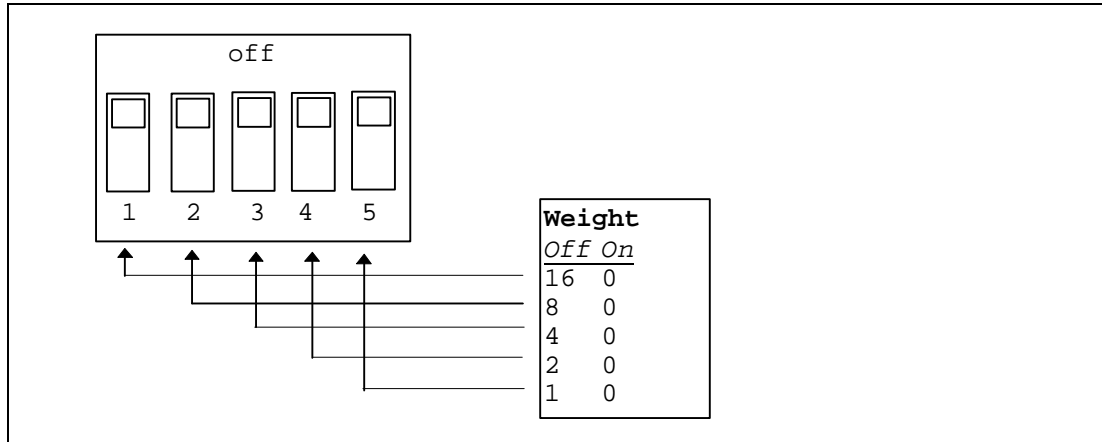


Figure 3, IEEE-488 Address Switch

Note: Cycle MT488 power after changing dip switch setting.

2.8 Pin Descriptions

a. P1 IEEE-488 Connector

Pin	Name/Description	Pin	Name/Description
1	DIO1/Data line	13	DIO5/Data line
2	DIO2/Data line	14	DIO6/Data line
3	DIO3/Data line	15	DIO7/Data line
4	DIO4/Data line	16	DIO8/Data line
5	EOI/End or Identify	17	REN/Remote Enable
6	DAV/Data Valid	18	DAV Gnd
7	NRFD/Not Ready For Data	19	NRFD Gnd
8	NDAC/Not Data Accepted	20	NDAC Gnd
9	IFC/Interface Clear	21	IFC Gnd
10	SRQ/Service Request	22	SRQ Gnd
11	ATN/Attention	23	ATN Gnd
12	Shield	24	Ground

b. P3 D/A and A/D Converter Connector

Pin	I/O	Name/Description	Voltage Level
1	O	CH0/Channel 0 output	0 to 2.50 VDC
14	O	CH1/Channel 1 output	0 to 2.50 VDC
2	O	CH2/Channel 2 output	0 to 2.50 VDC
15	O	CH3/Channel 3 output	0 to 2.50 VDC
3	O	CH4/Channel 3 output	0 to 2.50 VDC
16	O	CH5/Channel 3 output	0 to 2.50 VDC
12	I	CH0/Channel 0 input	0 to 2.50 VDC
13	I	CH1/Channel 1 input	0 to 2.50 VDC
N/C	I	CH2/Channel 2 input	0 to 2.50 VDC
11	I	CH3/Channel 3 input	0 to 2.50 VDC
10	I	CH4/Channel 4 input	0 to 2.50 VDC
9	I	CH5/Channel 5 input	0 to 2.50 VDC
8	I	CH6/Channel 6 input	0 to 2.50 VDC
24	I	CH7/Channel 7 input	0 to 2.50 VDC
6	I	CH8/Channel 8 input	0 to 2.50 VDC
7	I	CH9/Channel 9 input	0 to 2.50 VDC
22	I	CH10/Channel 10 input	0 to 2.50 VDC
5	I	Ground	Reference
18	I	Ground	Reference
19	I	Ground	Reference
20	I	Ground	Reference
21	I	Ground	Reference
23	I	Ground	Reference
25	I	Ground	Reference

c. P4 RS-232 Interface Connector

Pin	I/O	Name/Description	Voltage Level
1	I	Gnd	Ref
2	I	RX/Receive Data	-8 to +8 VDC
3	O	TX/Transmit Data	-8 to +8 VDC
4	I	CTS/Clear to Send	-8 to +8 VDC
5	O	RTS/Request to Send	-8 to +8 VDC
7	I	Gnd	Ref

WARNING

High voltage AC and DC exists on the MT488. Do not touch the unit with power on. Failure to observe this warning could result in serious injury or death.

d. P6 Power

Pin	Name/Description
1	120 VAC
2	GND
3	120 VAC

g. P9 Digital I/O Connector

Pin	I/O	Name/Description	Voltage Level
1	O	+5 VDC Power	+5 VDC
20	I/O	D0-LSB Data 0 I/O	+5 VDC
2	I/O	D0 Data 1 I/O	+5 VDC
21	I/O	D0 Data 2 I/O	+5 VDC
3	I/O	D0 Data 3 I/O	+5 VDC
22	I/O	D0 Data 4 I/O	+5 VDC
4	I/O	D0 Data 5 I/O	+5 VDC
23	I/O	D0 Data 6 I/O	+5 VDC
5	I/O	D0-MSB Data 7 I/O	+5 VDC
24	I/O	D1-LSB Data 0 I/O	+5 VDC
6	I/O	D1 Data 1 I/O	+5 VDC
25	I/O	D1 Data 2 I/O	+5 VDC
7	I/O	D1 Data 3 I/O	+5 VDC
26	I/O	D1 Data 4 I/O	+5 VDC
8	I/O	D1 Data 5 I/O	+5 VDC
27	I/O	D1 Data 6 I/O	+5 VDC
9	I/O	D1-MSB Data 7 I/O	+5 VDC
28	I/O	D2-LSB Data 0 I/O	+5 VDC
10	I/O	D2 Data 1 I/O	+5 VDC
29	I/O	D2 Data 2 I/O	+5 VDC
11	I/O	D2 Data 3 I/O	+5 VDC
30	I/O	D2 Data 4 I/O	+5 VDC
12	I/O	D2 Data 5 I/O	+5 VDC
31	I/O	D2 Data 6 I/O	+5 VDC
13	I/O	D2-MSB Data 7 I/O	+5 VDC
32	I/O	D3-LSB Data 0 I/O	+5 VDC
14	I/O	D3 Data 1 I/O	+5 VDC
33	I/O	D3 Data 2 I/O	+5 VDC
15	I/O	D3 Data 3 I/O	+5 VDC
34	I/O	D3 Data 4 I/O	+5 VDC
16	I/O	D3 Data 5 I/O	+5 VDC
35	I/O	D3 Data 6 I/O	+5 VDC
17	I/O	D3-MSB Data 7 I/O	+5 VDC
36	O	Power Return	0 VDC

2.9 Physical Description

Dimensions: 6.5" x 9.5" x 1.5"

Weight: 2 lb

Power Requirements: 6 VA, 5 W

Operating temperature: 0 to 50 °C.

3.0 Functional Description

3.1 IEEE-488 Interface

The IEEE-488 bus is designed to allow up to 15 instruments within a localized area to communicate with each other over a common bus. Each device has a unique address, read from a set of address switches at power up, that it responds to.

In general, data may be sent by one device (the talker) and received by any number of listeners. The IEEE-488 interface is managed by the NEC uPD7210 type Application Specific IC (ASIC). The interface consists of eight data lines (DIO0-DIO8) and eight handshake/control lines. These signals are handshaking lines DAV, NRFD, NDAC and control lines IFC, ATN, SRQ, REN and EOI.

When the ATN line is true, all devices listen for the command coming over the bus. When the ATN is false, only addressed devices can participate in transfers.

The Service Request Line (SRQ) allows any device on the bus to request service and "interrupt" the controller.

The End or Identify line (EOI) is signaled by the talker when the last data byte has been put on the bus. The IEEE-488 interface (henceforth referred to as the 488) is implemented by an NEC uPD7210 specific controller and two 488 specific transceivers.

3.2 RS-232 Interface

In addition to the 488 interface the MT488 supports RS232 communications at 9600 Baud.

The UART (Universal Asynchronous Receiver/Transmitter) is an integrated peripheral on the Z180 Microprocessor. The Z180 has two UART ports, 0 and 1. The UART channel 0 is used for this application. The UART signals are passed through the 5 Volt only RS232 transceiver U85. U85 has an onboard charge pump circuit to generate the +/- 9 volts required by the RS232 interface. The RS232 port is also interrupt driven and generates an interrupt when it requires attention.

The RS232 interface connects externally through a 9 Pin D-Sub-miniature connector (P4) PCB mounted on the MT488. Note that the CTS signal (RTS at the IBM) should be driven true to enable communications.

3.3 A/D Converter

The Analog input on the MT488 consists of an 11 Channel, 12 Bit A/D (Analog to Digital) converter with an input range of 0 to 2.5 Volts Full Scale. The A/D converter is isolated from the microprocessor

and the communications interfaces with high speed opto-isolators affording a 2500 VAC isolation barrier.

The A/D converter reference inputs are provided by a precision 2.5 VDC source from National Semiconductor, LM4040 (U22). The A/D converter's full scale voltage can be changed to 4.096 or 5.0 VDC by substituting the appropriate version of the LM4040. Note that the A/D can only accept 5.0 VDC max whereas the D/A converter can accept up to 10 VDC.

The A/D Converter's analog inputs are protected by diode clamps to +5 VDC and ground and an input resistor networks RN9-RN11. The input resistors in conjunction with the capacitor networks CN1 and CN2, provide a low pass filter function to reduce input noise on the A/D converter.

3.4 D/A Converter

The Analog output on the MT488 consists of up to six (6) Channels of 12 Bit D/A (Digital to Analog) conversion with a range of 0 to 2.5 Volts Full Scale. The D/A converter is isolated from the microprocessor and the communications interfaces with high speed opto-isolators affording a 2500 VAC isolation barrier.

The D/A Converter ICs, U11-U16, are LTC1257 CMOS device Linear Technology. The converter interfaces via a four wire serial interface which simplifies optical isolation.

These digital signals are manipulated by the microprocessor via the 8 bit addressable latch, U61 and are isolated by U42 opto-isolator.

The D/A converter reference inputs are provided by a precision 2.5 VDC source from National, LM4040 (U21).

The D/A Converter's analog outputs are protected by an output resistor network, RN8.

3.5 Digital I/O

The MT488's digital I/O provides up to 32 Bits of Bi-directional digital input/output. The four 8 bit ports are isolated from the microprocessor and the communications interfaces with high speed opto-isolators affording a 2500 VAC isolation barrier.

The digital I/O port IC's, U51-U53, are Phillips PCF8574 chips. These parts are remote 8 bit I/O expanders for the I²C bus. Their I²C interface is a two wire serial interface which simplifies optical isolation.

The interface signals consist of:

- a. Serial Clk SCL (U51-14)
- b. Serial Data SDA (U51-15)

The I²C interface incorporates a serial addressing system whereby data can be written to or read from a specific port among up to 8 ports which are connected in parallel. The port address is configured by three pins on each chip, A0 (Pin 1), A1 (Pin 2) and A2 (Pin 3). These three address pins allow 8 devices to be individually addressed on the same serial bus. In the MT488 system U51 is at address 0, U52 is at address 1, U53 is at address 2 and U54 is at address 3.

Each pin on these ports can sink up to 10 ma and source up to 30 uA. Additional source current can be provided by the pull up resistor networks RN21 thru RN24. The ports are quasi bi-directional which means they can be used as inputs or outputs. The microprocessor can read any pin by first setting the port to high, which engages a high impedance pull-up, and then the processor read the pins.

The digital signals, SDA and SCL are manipulated by the microprocessor via the 8 bit addressable latch, U61 and are isolated by opto-isolator U47. The interrupt and read data are isolated by opto-isolator U48. The read data is connected to input port U71 (74HC541).

3.6 Smart Core Features

The SmartCore(tm) features a Z180 CPU running at 9.216 Mhz. (Figure 4, SmartCore block diagram).

The SmartCore onboard memory consists of:

- a. SRAM, 32K to 128K Battery Backed.
- b. EPROM, 32K to 128K .
- c. EEPROM, 512 bytes.

Other SmartCore peripherals are:

- d. Real-time Clock
- e. Reset and Power Failure Supervisor.
- f. Serial Interface for program development.
- g. Serial Interface for the application.
- h. Two programmable timers.
- i. Two DMA channels.
- j. Memory decoding for external devices (/CS1-/CS6).

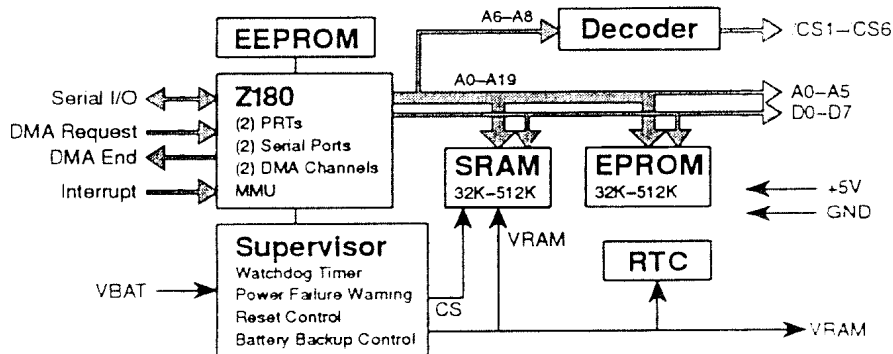


Figure 4, SmartCore Block Diagram

3.7 Power Supply Features

The MT488 features an off line switching power supply. The power supply provides power to the computer interface circuitry and to the I/O sections.

The power supply features:

- 2500 VAC Isolation
- 3 outputs +5,+8 and +22 VDC
- EMI suppression

4.0 Operating Instructions

4.1 Unpacking

If the shipping carton is damaged on receipt call the carrier immediately. Then inspect the MT488 for damaged or loose components. If there is any damage notify your sales agent to obtain RMA authorization.

4.2 Setup and Configuration

The MT488 is ready to operate without any calibration or setup. To begin programming the MT488 for your application you'll need to put the Run/Program switch in the Program position and apply power (See figure 5). Then connect up the programming port to your IBM PC compatible computer and start Dynamic C on your PC.

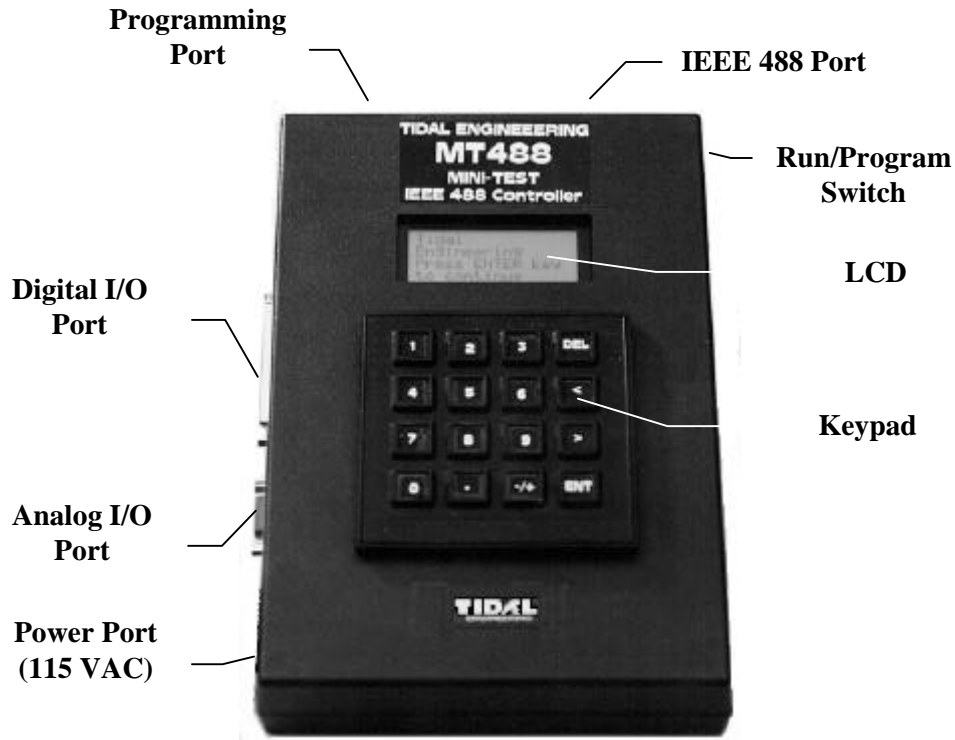


Figure 5 MT488 Interfaces

4.3 Software Installation

4.3.1 Dynamic C

- a) Install Dynamic C per the installation directions included with your Dynamic C documentation.

4.3.2 Libraries

- a) Copy the MT488 Library (MT488.LIB) from the supplied disk to your Dynamic C library subdirectory (\LIB).
- b) Add the MT488 library to the LIB.DIR library list in your Dynamic C subdirectory using DOSedit or other text processor.

4.3.3 Example Program

- a) Copy Lecroya.C from the supplied disk to a directory on your hard drive.

5.0 Programming

5.1 MT488 Specific Drivers

The Dynamic C libraries included with the MT488 development package have interfaces for all of the SmartCore standard peripherals including EEPROM, Serial Ports, Real time clock, etc.

The MT488 specific drivers provide a high level interface to all of the MT488 specific peripherals.

The following are the device specific commands from the MT488.LIB file

Tidal Engineering Copyright 1997
IEEE 488 and peripheral driver routines for the MT488 Minitest controller.

ibloc <MT488.LIB>

SYNTAX: int ibloc(ud);
 KEYWORDS: ib, local
 DESCRIPTION: Set the specified GPIB device to local
 RETURN VALUE: None.

ieee_488 <MT488.LIB>

SYNTAX: void ieee_488();
 KEYWORDS: GPIB Interrupt, Interrupt
 DESCRIPTION: Interrupt handler for IEEE 488 (GPIB) interface
 RETURN VALUE: None.

process_esr_stb <MT488.LIB>

SYNTAX: void process_esr_stb(int esr, int *stb, int ese, int sre)
 KEYWORDS: event status
 DESCRIPTION: Processess event status register for IEEE 488 (GPIB) interface
 RETURN VALUE: None.

ibren <MT488.LIB>

SYNTAX: int ibren(int ud);
 KEYWORDS: remote enable, GPIB, IEEE 488
 DESCRIPTION: remote enable the specified GPIB device
 RETURN VALUE: None.

ibdev <MT488.LIB>

SYNTAX: int ibdev(int boardID, int pad, int sad, int tmo, int eot, int eos)
 KEYWORDS: remote enable, GPIB, IEEE 488
 DESCRIPTION: Opens a GPIB device
 RETURN VALUE: None.

ibclr <MT488.LIB>

SYNTAX: void ibclr(int ud);
KEYWORDS: device clear, GPIB,IEEE 488
DESCRIPTION: Clears a GPIB device
RETURN VALUE: None.

init_488 <MT488.LIB>

SYNTAX: int init_488()
KEYWORDS: Initialize GPIB,IEEE 488
DESCRIPTION: Initializes the MT488 GPIB controller.
RETURN VALUE: None.

ibrd <MT488.LIB>

SYNTAX: int ibrd(int ud, char *buffer, long cnt)
KEYWORDS: GPIB read
DESCRIPTION: Read data from the specified device at address ud
Data stored at buffer
cnt is the number of characters to rd.

RETURN VALUE: None.

ibwrt <MT488.LIB>

SYNTAX: int ibwrt(int ud, char *buffer, long cnt)
KEYWORDS: GPIB write
DESCRIPTION: Write data to the specified device at address ud
Data is stored at buffer
cnt is the number of characters to rd.
RETURN VALUE: None.

wait_sync <MT488.LIB>

SYNTAX: void wait_sync(void)
KEYWORDS: wait GPIB
DESCRIPTION: Wait for GPIB data to be finished
RETURN VALUE: None.

wait4co <MT488.LIB>

SYNTAX: int wait4co(void)
KEYWORDS: wait GPIB command
DESCRIPTION: Wait for GPIB command to be finished
RETURN VALUE: None.

ieee_cmd <MT488.LIB>

SYNTAX: int ieee_cmd(int cmd)
KEYWORDS: send GPIB command
DESCRIPTION: Send a GPIB command
RETURN VALUE: None.

dspCIC <MT488.LIB>

SYNTAX: void dspCIC()
KEYWORDS: Controller-in-Charge
DESCRIPTION: Become Controller in charge
RETURN VALUE: None.

read_a2d <MT488.LIB>

SYNTAX: int read_a2d(int chan)
KEYWORDS: ADC, Analog to Digital Converter
DESCRIPTION: Read Analog to Digital Converter
RETURN VALUE: ADC result

set_da <MT488.LIB>

SYNTAX: char set_da()
KEYWORDS: DAC, Digital to Analog Converter
DESCRIPTION: Set Digital to Analog Converter
RETURN VALUE: none

wr_pio <MT488.LIB>

SYNTAX: void wr_pio(int port);
KEYWORDS: pio, parallel data port
DESCRIPTION: Write to digital port
RETURN VALUE: none

rd_pio <MT488.LIB>

SYNTAX: int rd_pio(int port);
KEYWORDS: pio, parallel data port
DESCRIPTION: Read digital port
RETURN VALUE: none

rs232_send <MT488.LIB>

SYNTAX: void rs232_send();
KEYWORDS: RS232 data
DESCRIPTION: Send data out the RS-232 port
RETURN VALUE: none

ieee_send <MT488.LIB>

SYNTAX: void ieee_send();
KEYWORDS: IEEE 488 data, GPIB
DESCRIPTION: Send data out the IEEE 488 port
RETURN VALUE: none

address_changed <MT488.LIB>

SYNTAX: void address_changed ()
KEYWORDS: IEEE 488 address changed
DESCRIPTION: IEEE 488 address change
RETURN VALUE: none

5.2.2 Example Program Source(Lecroya.c)

Setup Serial Port interrupt for LCDkey interface

```
Define the interrupt vector and parameters for
the Serial communications to the Keypad/LCD
#define INT_VEC SER1_VEC Dz1_circ_int
// #INT_VEC SER0_VEC Dz0_circ_int // use this line when using port 1 for Dynamic C
programming port
#define IBAUD 9600/1200 // baud rate
// with modem either 2400 or 1200
// without modem => 19200,9600, 4800, etc
```

Setup transmit and receive buffers for the RS-232 Interface

```
#define TBUFSIZE 384 // size of transmit buffer
#define RBUFSIZE 384 // size of receive buffer
#define to488 2 // IEEE 488 timeout 2 seconds
```

Define user interface codes

```
#define klcd 1 // keyboard and LCD
#define dt 2 // dumb terminal,
#define dc 3 // dynamic c stdio
```

Define general purpose constants

```
#define false 0
#define true -1
#define right
#define wrong
```

Define Constants for hardware and Chip selects

```
#define CS_7210 0x4000
#define CS_259 0x4040
#define CS_541 0x4080
#define CS_574 0x40C0
#define aclk CS_259
#define scl CS_259+5
#define sda CS_259+6
#define ad_cs CS_259+1
#define ad_data CS_259+2
```

Define Global Variables

```
int osc; //the id for the oscilloscope
int sbc; //the id for the MT488 or a second GPIB device
int err_cnt; //number of GPIB errors encountered, used to stop trying port
// when count gets high
```

Define Other RS-232 parameters

```
char bofer[45]; //Temporary character buffer
char MODE = 4; //8 data, no parity, 1 stop
char NO_MODEM = 0; //we don't want modem
char ECHO = 1; //we do want character echo
```


Connect libraries

```
#use vdriver.lib
#use mt488.lib
```

Define IEEE 488 buffers

```
#define ieee_in_len 128
#define ieee_out_len 128
```

Define GPIB device structure

```
typedef struct{
    int pad;
    int sad;
    int tmo;
    int eot;
    int eos;
} gbibdev;
gbibdev device[max_dev];
char ieee_in[ieee_in_len], *ieee_in_ptr; //GPIB input buffer and pointer
char ieee_out[ieee_out_len], *ieee_out_ptr; //GPIB output buffer and pointer
char brk[]=" "; *ptr, *ptr2, sdata;
int devices, ibsta, add488, esr, stb, ese, sre, sbc_cmd, din0, din1, din2, test, m_flag;
int da[6];
int d[4]; //digital port variables in memory
int bus; //bus type RS232 or GPIB
unsigned int leds, receive_flag, tx_cntr;
char ui;
```

Digital IO Test Function

```
dio_test() {
    int ii, n, k, t, u, v, w;
    while(1) {
        for (ii=255; ii>=0; ii--) {
            w=122;
            for (k=3; k>=0; k--) {
                d[k]=ii;
                wr_pio(k);
                t=rd_pio(k);
                u=rd_pio(k);
                v=rd_pio(k);
                if((t!=ii) || (u!=ii) || (v!=ii)) {
                    printf(" error ");
                }
            }
            for (n=0; n<300; n++){ };
            outport(CS_574, ii);
        }
    }
}
```

Sine Generation Function

```
////////////////////////////////////
//sine generates a floating point sine wave and triangle wave, //
// converts them to 12 bits and sends them to the optional D/A //
// converter. //
////////////////////////////////////
sine() {
```

```

int ii;
float a,b;
char temp[25];
int sine[32],tri[32];
//da[0]=2048;da[1]=1024;da[2]=3073;set_da();
for (ii=0;ii<=31;ii++) {
    a=(float)ii;
    b=sin(6.28*(a/31))+1.0;
    sine[ii]=(int)(b*2048.0);
    tri[ii]=128*ii;
}
while(1) {
    for (ii=0;ii<=31;ii++) {
        da[0]=sine[ii];
        da[1]=tri[ii];
        set_da();
    }
}
}

```

Floating Point Correction function

```

////////////////////////////////////
//floatprec converts a floating point string in the ieee_in buffer //
// to 5 significant digits. Z-World atof() function can't handle //
// more than 5 significant digits. //
////////////////////////////////////
void floatprec(void) {
    char *ttt, *ptt;
    int i;
    ptt=strchr(ieee_in,'E');
    if(ptt!=0) {
        ttt=ieee_in+9;
        for (i=0;i<4;i++) {
            * ttt=* ptt;
            ttt++;
            ptt++;
        }
        * ttt= '\0';
    }
}

```

tprintf function

```

////////////////////////////////////
//tprintf() is a modified printf function which can write to the //
// LCD, a dumb terminal or the STDIO terminal of Dynamic C //
// It will count the GPIB errors and shutdown any further GPIB //
// communications. //
////////////////////////////////////
void tprintf( char *buft) {
    char buff[40], * gg;
    int del;
    // 1234567890123456789012345678901234567890
    gg="\f";
    if (strstr(buft,"CMD")!='\0') {
        err_cnt++;
        if (err_cnt>=5) {
            err_cnt=5;
            strcpy(buff,"\f5 CMD errors\nhave occurred\nshutdown 488\n");
        }
    }
}

```

```

    }
}
switch(ui) {
    case klcd:

        Dwrite_z1 (buft,strlen(buft));
        break;
    case dc:      printf(buft); break;
    case dt:      Dwrite_z0 (buft,strlen(buft)); break;
}
}

```

Keyboard Input Function

```

////////////////////////////////////
//input() waits for a character press and then returns.           //
// Used to pause the program while waiting for user input        //
// Works with any of the three terminal modes communications      //
////////////////////////////////////
void input( char *buf) {
    switch(ui) {
        case klcd: while (Dread_z1(buf,ENTER) == 0 ){}; break;
        case dc:   gets(buf); break;
        case dt:   while (Dread_z0(buf,ENTER) == 0 ){}; break;
    }
}

```

Main Function

```

////////////////////////////////////
main() {

    int i;
    float a,b;
    int count;
    char s, t,data,key[1],key2[1];
    char stbuf[TBUFSIZE]; // transmit buffer for serial port
    char srbuf[RBUFSIZE]; // receive buffer for serial port
    char buf[RBUFSIZE+1]; // dummy buffer for receiving a complete command
    char tbuf[ieee_out_len]; // transmit buffer for GPIB port
    char rbuf[ieee_in_len]; // receive buffer for GPIB port
    int commands,osc,da0,da1,adc[11];
    unsigned int chan;
    int sine[32],tri[32]; //sine and triangle buffers
    int ii,n,j,z,v,PAD_;
    char temp[25];
    int ch1i,ch2i,timei,done2,done;
}

```

Lecroy command string definitions

```

char *ch1[]={ "C1:VDIV 50mV\0xa",
              "C1:VDIV 100mV\0xa",
              "C1:VDIV 200mV\0xa",
              "C1:VDIV 500mV\0xa",
              "C1:VDIV 1V\0xa" };

char *ch2[]={ "C2:VDIV 50mV\0xa",
              "C2:VDIV 100mV\0xa",

```

```

        "C2:VDIV 200mV\0xa",
        "C2:VDIV 500mV\0xa",
        "C2:VDIV 1V\0xa"};
char *time[]={ "TDIV 50ms\0xa",
               "TDIV 100ms\0xa",
               "TDIV 200ms\0xa",
               "TDIV 500ms\0xa",
               "TDIV 20ms\0xa"};
char *menu="\fTo Adjust Scope\n1 = ch1,2 = ch2\nPress 3 = time\n4=ADC & 5=DIG";

```

Watchdog bypass

```

VdInit();           // initialize the virtual driver
hitwd();           // enable the virtual watchdog (watchdog will be taken
                  // care of by virtual driver

```

Main Loop

```

while (1) {
    done2=false;
    ui=klcd;        //dt=dumb terminal, dc=dyanamic c stdio
    err_cnt=0;
//Initialize serial port for Keypad/LCD interface
    Dinit_zl(srbuf,stbuf,RBUFSIZE,TBUFSIZE, MODE, IBAUD, NO_MODEM, ECHO );
//Send Hello message to LCD
    tprintf ("\fTidal\nEngineering\nPress ENTER key\nto continue");
//Wait for user to press enter key
    input (buf);
//Initialize GPIB interface now
    if (init_488()==-1) {
//print message if there was an error initializing GPIB
        sprintf(buf,"
            \fCould not initialize Natational Instruments IEEE 488 ASIC");
        tprintf (buf);
    }
//Enter oscilloscope GPIB address
    tprintf ("\fEnter Scope GPIBaddress and\nPress ENTER key\n(default=6)");
    input (buf);
    if ((strlen(buf))==0) {
        PAD_=6;
    }
    else {
        PAD_=atof(buf);
    }
//Initialize driver for the device
    if ((osc = ibdev(0, PAD_, 0, T10s, 1, 0)) < 0) {
        sprintf(buf,"\fCould not open OSC_01 device");
        tprintf (buf);
    }
//Skip this test code for A/D converter
    while(0){
        for (i=0;i<=10;i++){
            adc[i]=read_a2d(i);
        }
        sprintf(bofer,"\f%+4d %+4d %+4d \n%+4d %+4d %+4d\n%+4d %+4d %+4d\nPress
ENTER key",
            adc[0],adc[1],adc[2],adc[3],adc[4],adc[5],adc[6],adc[7],adc[8]);

```

```

        tprintf(bofer);
    }
//Enter the address for a second GPIB device, default to 17
tprintf ("\fEnter SBC488\naddress and\nPress ENTER key\n(default=17)");
input (buf);
if ((strlen(buf))==0) {
    PAD_=17;
}
else {
    PAD_=atof(buf);
}
if ((sbc = ibdev(0, PAD_, 0, T10s, 1, 0)) < 0) {
    sprintf(buf, "\fCould not open OSC_01 device");
    tprintf (buf);
}
//Put oscilloscope into local
    ibwrt(osc); //Put Oscillator in REN
        chli=0;ch2i=0,timei=0;
        tprintf (menu);
        ibwrt(osc, ch1[chli], strlen(ch1[chli]));
        if (ibsta & ERR) tprintf("    Could not write to device\n\r");
        ibwrt(osc, ch2[ch2i], strlen(ch2[ch2i]));
        if (ibsta & ERR) tprintf("    Could not write to device\n\r");
        ibwrt(osc, time[timei], strlen(time[timei]));
        if (ibsta & ERR) tprintf("    Could not write to device\n\r");
//Generate a sine wave and triangle wave for D/A demo
    for (ii=0;ii<=31;ii++) {
        a=(float)ii;
        b=sin(6.28*(a/31))+1.0;
        sine[ii]=(int)(b*2048.0);
        tri[ii]=128*ii;
    }
//Start Menu driver loop
while(1) {
//Check for user keypresses now
    n=Dread_zllch(key);
    if (n==1){
        i++;
        switch(*key) {
//Determine which key was pressed
            case '1':
//Send next channel 1 vertical scale
                if(++chli==5) chli=0;
                ibwrt(osc, ch1[chli], strlen(ch1[chli]));
                break;
            case '2':
//Send next channel 2 vertical scale
                if(++ch2i==5) ch2i=0;
                ibwrt(osc, ch2[ch2i], strlen(ch2[ch2i]));
                break;
            case '3':
//Send next channel 1 time scale
                if(++timei==5) timei=0;
                ibwrt(osc, time[timei], strlen(time[timei]));
                break;
            case '4':
//Read and display A/D readings (Channels 0 thru 8)
                done=false;
                while(!done){

```

```

        for (i=0;i<=10;i++){
            adc[i]=read_a2d(i);
        }

        sprintf(bofer, "\f%+4d %+4d %+4d \n%+4d %+4d %+4d\n%+4d %+4d
%+4d\nPress ENTER key",

            adc[0],adc[1],adc[2],adc[3],adc[4],adc[5],adc[6],adc[7],adc[8]);
        tprintf(bofer);
        n=Dread_z11ch(key2);
        if ((n==1)&&(*key2=='\r')){
            done=true;
        }
    }
    break;
case '5':
//Read and display Digital input readings (Channels 0 thru 3)
    done=false;
    while(!done){
        sprintf(bofer, "\f%+3d %+3d %+3d \n%+3d \nPress ENTER key",
            rd_pio(0),rd_pio(1),rd_pio(2),rd_pio(3));
        tprintf(bofer);
        n=Dread_z11ch(key2);
        if ((n==1)&&(*key2=='\r')){
            done=true;
        }
    }
    break;
case 'B':
//Break when F2 is pressed
    done2=true;
    break;

    }
    tprintf (menu);
}
if (done2) break;
}
}
}

```

5.2 Example Program (Lecroya.c)

```

////////////////////////////////////
//  LECROY.C MT488 Demonstration Program          //
//  Controls Lecroy Model 9310 Oscilloscope      //
//                                               //
//  14 June 1996                                //
//                                               //
////////////////////////////////////
// Define the interrupt vector and parameters for
// the Serial communications to the Keypad/LCD
#define INT_VEC SER1_VEC Dz1_circ_int
// #INT_VEC SER0_VEC Dz0_circ_int //use this line when using port 1 for Dynamic C programming port
#define IBAUD 9600/1200 // baud rate
// with modem either 2400 or 1200
// without modem => 19200,9600, 4800, etc
////////////////////////////////////
#define TBUFSIZE 384 // size of transmit buffer
#define RBUFSIZE 384 // size of receive buffer
#define to488 2 // IEEE 488 timeout 2 seconds
////////////////////////////////////
//The following user interface codes are defined
#define klcd 1 //keyboard and LCD
#define dt 2 //dumb terminal,
#define dc 3 //dynamic c stdio
#define false 0
#define true -1
////////////////////////////////////
//Constants for hardware and Chip selects
#define CS_7210 0x4000
#define CS_259 0x4040
#define CS_541 0x4080
#define CS_574 0x40C0
#define aclk CS_259
#define scl CS_259+5
#define sda CS_259+6
#define ad_cs CS_259+1
#define ad_data CS_259+2
////////////////////////////////////
//Global Variable declarations
int osc; //the id for the oscilloscope
int sbc; //the id for the MT488 or a second GPIB device
int err_cnt; //number of GPIB errors encountered, used to stop trying port
// when count gets high
char bofer[45]; //Temporary character buffer
char MODE = 4; //8 data, no parity, 1 stop
char NO_MODEM = 0; //we don't want modem
char ECHO = 1; //we do want character echo

////////////////////////////////////
// Use the following libraries
#include <vdriver.lib>
#include <mt488.lib>
////////////////////////////////////
#define ieee_in_len 128
#define ieee_out_len 128

// Function prototypes
char set_da();
void set_aclk(int state);

////////////////////////////////////
// Variables
typedef struct{
    int pad;
    int sad;
    int tmo;
    int eot;
    int eos;
} gbibdev;
gbibdev device[max_dev];
char ieee_in[ieee_in_len], *ieee_in_ptr; //GPIB input buffer and pointer
char ieee_out[ieee_out_len], *ieee_out_ptr; //GPIB output buffer and pointer
char brk[]=" "; *ptr, *ptr2,sdata;
int devices,ibsta,add488, esr, stb, ese, sre, sbc_cmd,din0,din1,din2,test,m_flag;
int da[6];
int d[4]; //digital port variables in memory
int bus; //bus type RS232 or GPIB
unsigned int leds, receive_flag,tx_cntr;
char ui;

////////////////////////////////////
// Interrupt service routine for Parallel (PIO) Interface change
#define INTO_VEC 0x0038

```

```

#INT_VEC INT0_VEC dio_int;

interrupt void dio_int() {
    din0=rd_pio(0);din1=rd_pio(1);din2=rd_pio(2);
}

////////////////////////////////////
//      Digital IO test
dio_test() {
    int ii,n,k,t,u,v,w;
    while(1) {
        for (ii=255;ii>=0;ii--) {
            w=122;
            for (k=3;k>=0;k--) {
                d[k]=ii;
                wr_pio( k);
                t=rd_pio(k);
                u=rd_pio(k);
                v=rd_pio(k);
                if((t!=ii) || (u!=ii) || (v!=ii)) {
                    printf(" error ");
                }
            }
            for (n=0;n<300;n++){ };
            outport(CS_574, ii);
        }
    }
}

////////////////////////////////////
//sine generates a floating point sine wave and triangle wave, //
// converts them to 12 bits and sends them to the optional D/A //
// converter. //
////////////////////////////////////
sine() {
    int ii;
    float a,b;
    char temp[25];
    int sine[32],tri[32];
    //da[0]=2048;da[1]=1024;da[2]=3073;set_da();
    for (ii=0;ii<=31;ii++) {
        a=(float)ii;
        b=sin(6.28*(a/31))+1.0;
        sine[ii]=(int)(b*2048.0);
        tri[ii]=128*ii;
    }
    while(1) {
        for (ii=0;ii<=31;ii++) {
            da[0]=sine[ii];
            da[1]=tri[ii];
            set_da();
        }
    }
}

////////////////////////////////////
//floatprec converts a floating point string in the ieee_in buffer //
// to 5 significant digits. Z-World atof() function can't handle //
// more than 5 significant digits. //
////////////////////////////////////
void floatprec(void) {
    char *ttt, *ptt;
    int i;
    ptt=strchr(ieee_in,'E');
    if(ptt!=0) {
        ttt=ieee_in+9;
        for (i=0;i<4;i++) {
            * ttt=* ptt;
            ttt++;
            ptt++;
        }
        * ttt= '\0';
    }
}

////////////////////////////////////
//tprintf() is a modified printf function which can write to the //
// LCD, a dumb terminal or the STDIO terminal of Dynamic C //
// It will count the GPIB errors and shutdown any further GPIB //
// communications. //
////////////////////////////////////
void tprintf( char *buft) {
    char buff[40], * gg;
    int del;
    //      1234567890123456789012345678901234567890

```



```

gg="\f";
if (strstr(buf,"CMD")!='\0') {
    err_cnt++;
    if (err_cnt>=5) {
        err_cnt=5;
        strcpy(buf,"\f5 CMD errors\nhave occurred\nshutdown 488\n");
    }
}
}
switch(ui) {
    case klcd:
        Dwrite_z1 (buf,strlen(buf));
        break;
    case dc:      printf(buf); break;
    case dt:      Dwrite_z0 (buf,strlen(buf)); break;
}
}

//////////////////////////////////////
//input() waits for a character press and then returns.           //
// Used to pause the program while waiting for user input        //
// Works with any of the three terminal modes communications     //
//////////////////////////////////////
void input( char *buf) {
    switch(ui) {
        case klcd: while (Dread_z1(buf,ENTER) == 0 ){}; break;
        case dc:   gets(buf); break;
        case dt:   while (Dread_z0(buf,ENTER) == 0 ){}; break;
    }
}

//////////////////////////////////////
// show message to the terminal                                   //
//////////////////////////////////////
void showmenu(void){
    int i;
    for( i=0;i<5;i++ ) {
        Dz1send_prompt();
    }
    Dz1send_prompt();
}

//////////////////////////////////////
main() {

    int i;
    float a,b;
    int count;
    char s, t,data,key[1],key2[1];
    char stbuf[TBUFSIZE]; // transmit buffer for serial port
    char srbuf[RBUFSIZE]; // receive buffer for serial port
    char buf[RBUFSIZE+1]; // dummy buffer for receiving a complete command
    char tbuf[ieee_out_len]; // transmit buffer for GPIB port
    char rbuf[ieee_in_len]; // receive buffer for GPIB port
    int commands,osc,da0,dal,adc[11];
    unsigned int chan;
    int sine[32],tri[32]; //sine and triangle buffers
    int ii,n,j,z,v,PAD_;
    char temp[25];
    int ch1i,ch2i,timei,done2,done;

    char *ch1[]={"C1:VDIV 50mV\0xa",
                "C1:VDIV 100mV\0xa",
                "C1:VDIV 200mV\0xa",
                "C1:VDIV 500mV\0xa",
                "C1:VDIV 1V\0xa"};

    char *ch2[]={"C2:VDIV 50mV\0xa",
                "C2:VDIV 100mV\0xa",
                "C2:VDIV 200mV\0xa",
                "C2:VDIV 500mV\0xa",
                "C2:VDIV 1V\0xa"};

    char *time[]={"TDIV 50ms\0xa",
                 "TDIV 100ms\0xa",
                 "TDIV 200ms\0xa",
                 "TDIV 500ms\0xa",
                 "TDIV 20ms\0xa"};

    char *menu="\fTo Adjust Scope\n1 = ch1,2 = ch2\nPress 3 = time\n4=ADC & 5=DIG";

    VdInit(); // initialize the virtual driver
    hitwd(); // enable the virtual watchdog (watchdog will be taken
            // care of by virtual driver

    //init_io();

```

```

while (1) {
    done2=false;
    ui=klcd;        //dt=dumb terminal, dc=dyanamic c stdio
    err_cnt=0;
    Dinit_z1(srbuf,stbuf,RBUFSIZE,TBUFSIZE, MODE, IBAUD, NO_MODEM, ECHO );
    tprintf ("\fTidal\nEngineering\nPress ENTER key\nto continue");

    input (buf);
    if (init_488()==-1) {

        sprintf(buf,"\fCould not initialize Natational Instruments IEEE 488 ASIC");
        tprintf (buf);
    }

    tprintf ("\fEnter Scope GPIBaddress and\nPress ENTER key\n(default=6)");
    input (buf);
    if ((strlen(buf))==0) {
        PAD_=6;
    }
    else {
        PAD_=atof(buf);
    }
    if ((osc = ibdev(0, PAD_, 0, T10s, 1, 0)) < 0) {
        sprintf(buf,"\fCould not open OSC_01 device");
        tprintf (buf);
    }
    while(0){
        for (i=0;i<=10;i++){
            adc[i]=read_a2d(i);
        }

        sprintf(bofer,"\f%+4d %+4d %+4d \n%+4d %+4d %+4d\n%+4d %+4d %+4d\nPress ENTER key",
            adc[0],adc[1],adc[2],adc[3],adc[4],adc[5],adc[6],adc[7],adc[8]);
        tprintf(bofer);
    }
    tprintf ("\fEnter SBC488\naddress and\nPress ENTER key\n(default=17)");
    input (buf);
    if ((strlen(buf))==0) {
        PAD_=17;
    }
    else {
        PAD_=atof(buf);
    }
    if ((sbc = ibdev(0, PAD_, 0, T10s, 1, 0)) < 0) {
        sprintf(buf,"\fCould not open OSC_01 device");
        tprintf (buf);
    }

    ibren(osc);                //Put Oscillator in REN
        chli=0;ch2i=0,timei=0;
        tprintf (menu);
    ibwrt(osc, chl[chli], strlen(chl[chli]));
    if (ibsta & ERR) tprintf("    Could not write to device\n\r");
    ibwrt(osc, ch2[ch2i], strlen(ch2[ch2i]));
    if (ibsta & ERR) tprintf("    Could not write to device\n\r");
    ibwrt(osc, time[timei], strlen(time[timei]));
    if (ibsta & ERR) tprintf("    Could not write to device\n\r");

while(1) {
    n=Dread_zllch(key);
    if (n==1){
        i++;
        switch(*key) {

            case '1':
                if(++chli==5) chli=0;
                ibwrt(osc, chl[chli], strlen(chl[chli]));
                break;
            case '2':
                if(++ch2i==5) ch2i=0;
                ibwrt(osc, ch2[ch2i], strlen(ch2[ch2i]));
                break;
            case '3':
                if(++timei==5) timei=0;
                ibwrt(osc, time[timei], strlen(time[timei]));
                break;
            case '4':
                done=false;
                while(!done){
                    for (i=0;i<=10;i++){
                        adc[i]=read_a2d(i);
                    }

                    sprintf(bofer,"\f%+4d %+4d %+4d \n%+4d %+4d %+4d\n%+4d %+4d %+4d\nPress ENTER key",
                        adc[0],adc[1],adc[2],adc[3],adc[4],adc[5],adc[6],adc[7],adc[8]);
                    tprintf(bofer);
                }
            }
}

```

```
        n=Dread_zllch(key2);
        if ((n==1)&&>(*key2=='\r')){
            done=true;
        }
    }
    break;
case '5':
    done=false;
    while(!done){
        sprintf(bofer,"\f%+3d %+3d %+3d \n%+3d \nPress ENTER key",
            rd_pio(0),rd_pio(1),rd_pio(2),rd_pio(3));
        tprintf(bofer);
        n=Dread_zllch(key2);
        if ((n==1)&&>(*key2=='\r')){
            done=true;
        }
    }
    break;
case 'B':
    done2=true;
    break;
}
tprintf (menu);
}
if (done2) break;
}
}
```

6.0 TROUBLESHOOTING

FAULT CONDITION	PROBABLE CAUSE	CHECK, REMEDY
No IEEE-488 Communications	Address errors Note 3 Termination IEEE-488 Cable MT488 Power Supplies SmartCore Failure 7210 Chip failure	MT488 Dip switch. Valid addresses are 0 thru 30 Control program address Check control program termination, CR, LF and EOI Replace Cable Check power supplies on MT488 (U7). Check data bus for activity Check Interrupt line out of 7210 during transmission.
No RS-232 Communications	Address errors Note 3 Format error RS-232 Cable MT488 Power Supplies SmartCore Failure RS-232 transceiver Chip failure	MT488 Dip switch. Valid address is 31. Terminal program set for 9600,N,8,1 Replace Cable Check power supplies on MT488 (U7). Check data bus for activity Check U85 RX and TX lines during transmission
D/A out voltage error	D/A power supply Opto-Isolator failure Voltage reference D/A IC failure Control software	Check D/A power supply voltage (U1 and U2) Check U41 and U42 outputs while commanding voltage change. Check U21 for 2.50 VDC Replace U11 Check data for errors
A/D Readback error note 1	A/D power supply Opto-Isolator failure Voltage reference A/D IC failure. Control software	Check A/D power supply voltage (U5 and U3) Check U43, U46 and U45 outputs while reading voltage over bus. Check U22 for 2.50 VDC Replace U90 Check data for errors

FAULT CONDITION	PROBABLE CAUSE	CHECK, REMEDY
Digital Readback error	Digital power supply Opto-Isolator failure Control software Port IC failure Voltage levels	Check power supply voltage (U6) Check U48 and U47 outputs while reading digital over bus Check data for errors Replace port IC U51, U52 or U53 Verify input voltage levels are valid (See section 2)
Digital set error	Digital power supply Opto-Isolator failure Control software Port IC failure load current note 2	Check power supply voltage (U6) Check U48 and U47 outputs while reading digital over bus Check data for errors Replace port IC U51, U52 or U53 Verify load current levels are valid (See section 2)

Notes:

1. A/D input voltages outside of specified input range may cause the IC to latch and fail.
2. Digital output ports may require a pull up resistor for certain loads since the source current available is low.
3. Always cycle MT488 power after changing dip switch setting.

7.0 DRAWINGS AND SCHEMATICS

7.1 MT488 Parts List

The MT488 component layout is on Figure 6.

P/N 104101B									
ITEM	REV	QTY	REF DES	VALUE	VOLT	MPE P/N	DESCRIPTION	MFG PART NUMBER	MANUFACTURER
1		1	BT1	CR2032	3V		LITHIUM BATTERY, 20 MM	CR2032	PANASONIC
2		1	C1	100	35V			ECE-A1VSF101 (P1231	PANASONIC
3	B	25	C11,C13-C33,C38-C40	0.1	50V	11146			
4		3	C41-C43	330	16V			ECE-A1CFS331 (P1213	PANASONIC
5		1	C51	0.1	1000V	11201		ECQ-E2A104MW	PANASONIC
6		2	C52-C53	0.01	1000V			ECQ-U2A103MW	PANASONIC
7		1	C61	47	25V	12216			
8		1	C81	33	250V			ECE-A2EU330	PANASONIC
9		1	C91	220 pF	1000V	10843			
10		2	CN1-CN2	0.1	50V		CAPACITOR NETWORK, 8 PIN SIP	201CH7X7R104K5	DALE
11		1	D1	1N5230B	4.7V	10990			
12		1	D21	1N4148		10978			
13		2	D41-D42	1N5819		11291			
14		1	D61	HER102		11125			
15	B	6	DB1-DB6	DF06M		11144			
16		1	L1	25 mH		104502	COMMON MODE CHOKE		MAGNAPOWER
17		1	L2	27U				5800-270	J.W. MILLER
18		5	LED0-LED4	L20321			LED, RED, RT ANGLE	5600F1	INDUSTRIAL DEVICES INC
19		1	P1	553811-1			CONNECTOR, 24 PIN, IEEE 488		AMP
20		1	P13	929834-01			CONNECTOR, HEADER, 10 POS, SIL		3M
21		1	P14	929836		11229	CONNECTOR, 10 PIN HEADER, DIL		3 M
22		2	P22/P23	929836		11229	CONNECTOR, 34 PIN HEADER, DIL		3 M
23		1	P3	929836		11229	CONNECTOR, 24 PIN HEADER, DIL		3 M
24		1	P4	P09S-21	P09S-21		CONNECTOR, 9 PIN D-SUB, RT ANGLE	DK: 309F-ND OR A158	NORCOMP OR AMP (DUAL)
25		1	P6	26-48-103			3 POSITION CONNECTOR, .156" SPACING	26-48-1035	MOLEX
26		1	P9	929836		11229			
27		1	Q2	2N2222A		11200			
28		1	R1	2.2K		10040			
29	B	2	R12-R13	10.0K, 1%		10356			
30		6	R14-R19	10K		10048			
31		5	R21-R25	1K		10036			
32		2	R30-R31	56K		10057			
33		2	R41-R42	680		10034			
34		1	R51	220		10028			
35		1	R71	6.8		10010			
36		1	R81	56		10033			
37		1	R91	1K	1/2W	10125			

38		2	R95-R96	1.2K		10037		
39		3	RN9-RN11	10K			RES NETWORK, 4R 8 PIN	CSC08A-03-103
40		1	RN12	1K			RES NETWORK, 7R 8 PIN	CSC08A-01-102
41	B	1	RN2	10K		11274	RES NETWORK, 9R 10 PIN	
42		4	RN21-RN24	470			RES NETWORK, 9R 10 PIN	CSC10A-01-471
43		1	RN3	10K			RES NETWORK, 5R 6 PIN	CSC06A-01-103
44		1	RN31	10K			RES NETWORK, 9R10 PIN	CSC10A-01-103
45		1	RN5	470			RES NETWORK, 9R 10 PIN	CSC10A-01-471
46		1	RN8	100			RES NETWORK, 7R 14 PIN	MDP14-03-101
47		1	SW1	A5005-ND			DIP SWITCH, 5 POSITION, RIGHT ANGLE	
48		1	SW2	CKN-5007			DPDT SLIDE SW, RT ANGLE	1101M2S3AV2BE2 C & K COMPONENTS
49		1	T1	104501		104501	FLYBACK TRANSFORMER	MAGNAPOWER
50		1	U1	LM7815CT		11169		
51		1	U100	SC-Z1B				101-0079 Z-WORLD ENGINEERING
52		1	U101	27C256-90				MICROCHIP
53		1	U110	SE1707			8 MHZ OSC., 14 PIN	SG-51P-8.000MHZ DIGIKEY
54		6	U11-U16	LTC1257CN8			12 BIT DAC	
55		1	U21	LM4040EIZ-X			VOLTAGE REFERENCE	NATIONAL
56		1	U22	LM4040EIZ-X			VOLTAGE REFERENCE	NATIONAL
57		1	U31	PS2501-1			OPTO ISOLATOR	NEC
58		1	U35	TL431		11109		
59		5	U41,U42,U44,U47,U48	HCPL-2531				HP
60		1	U5	LM78L05				LM78L05ACZ NATIONAL
61		4	U51-U54	PCF8574AP				PHILLIPS
62		1	U6	LM7805CT		10914		NATIONAL
63		1	U60	LTC485CN8				LINEAR TECHNOLOGY
64		1	U61	74HC259				MM74HC259N NATIONAL
65		1	U71	74HC541				MM74HC541N NATIONAL
66		1	U81	uPD7210				D7210C NEC OR NATIONAL INST.
67		1	U82	75160				DS75160AN NATIONAL
68		1	U83	75162				SN75162BN TEXAS INSTRUMENTS
69		1	U85	MAX202ECP				MAXIM
70		1	U88	74HC574				MM74HC574N NATIONAL
71		1	U90	TLC2543CN			A TO D CONVERTER, 12 BITS	TEXAS INSTRUMENTS
72		1	U91	TOP200YAI				POWER INTEGRATIONS
73		1	U99	74HC04				MM74HC04N NATIONAL
74		1	XBT1				BATTERY SOCKET, 20 MM	106K KEYSTONE
75		1	ZR1	V145	150V		VARISTOR, 145 VAC	PANASONIC

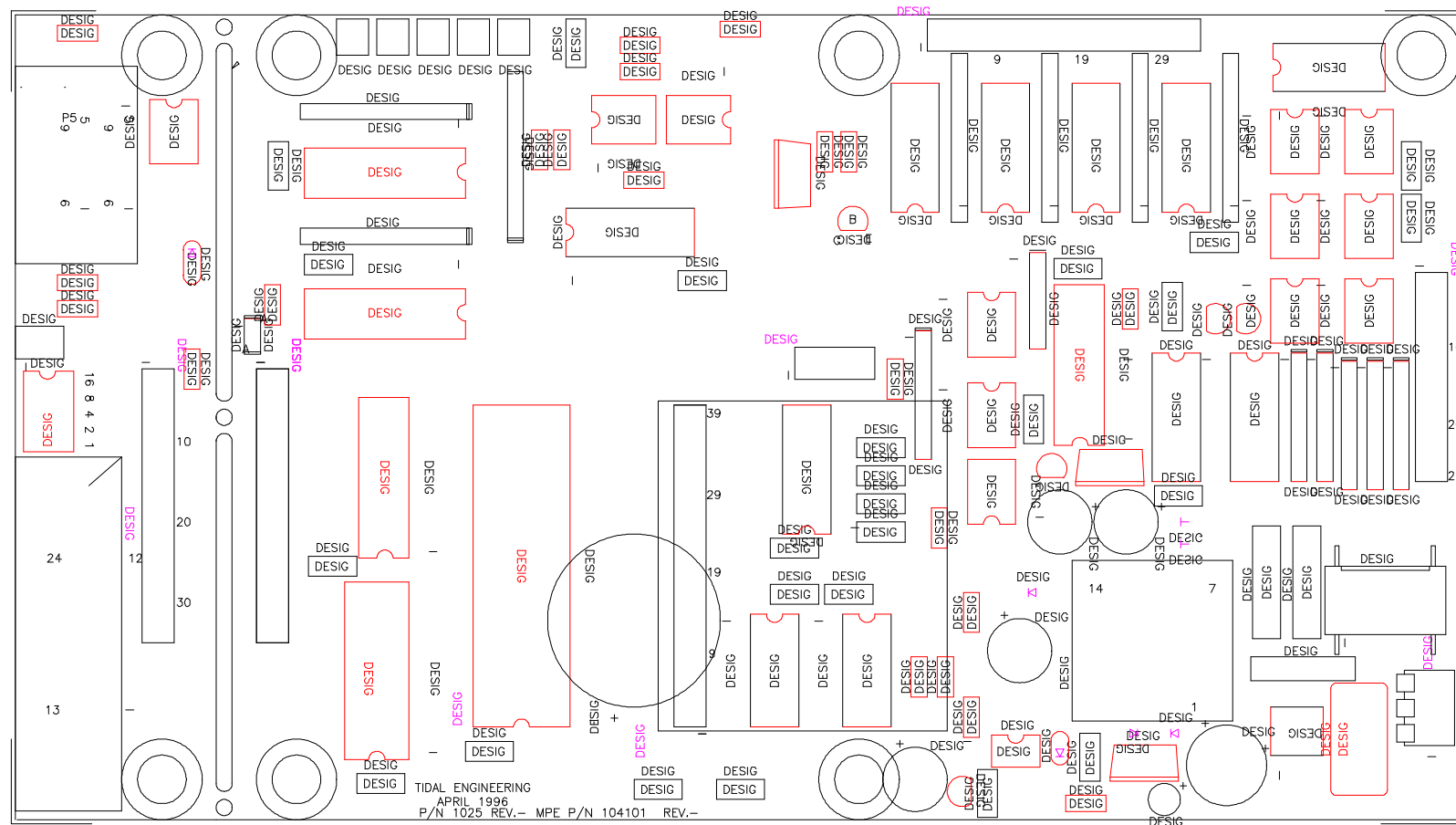


Figure 6, MT488 Component Layout

