Tidal Engineering Corporation
www.tidaleng.com

MT488A

IEEE 488 MINI TEST CONTROLLER

Technical Manual

2 Emery Avenue
Randolph, NJ 07869

Tel (973)328-1181 Fax (973)328-2302

DATE: 15 MARCH 2000 REV. 2
P/N TE1148

List of Changes

Original 10 November 1998
Revision 1
12 November 1998
Typographical Corrections
Revision 2
Added List of changes page
Added Wrap around test plug wiring table

Table of Contents

List of Figures

1.0 OVERVIEW

The MT488A 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 MT488A 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, MT488A Block Diagram. The IEEE-4888 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. Data is sent byte serial/bit parallel and consists of device dependent and interface messages.

In addition to the 488 interface the MT488A also supports two RS-232 communication ports at 9600 Baud. The RS-232 interface is an asynchronous serial interface. One port is used for programming the MT488A and the other interfaces to the LCD and keypad interface.

The MT488A’s digital I/O is capable of 32 Bits of bi-directional digital input/output. The four 8 bit ports can be optically isolated from the microprocessor and the communications interfaces.

The MT488A 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, 128K Battery Backed
b. EPROM, 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).

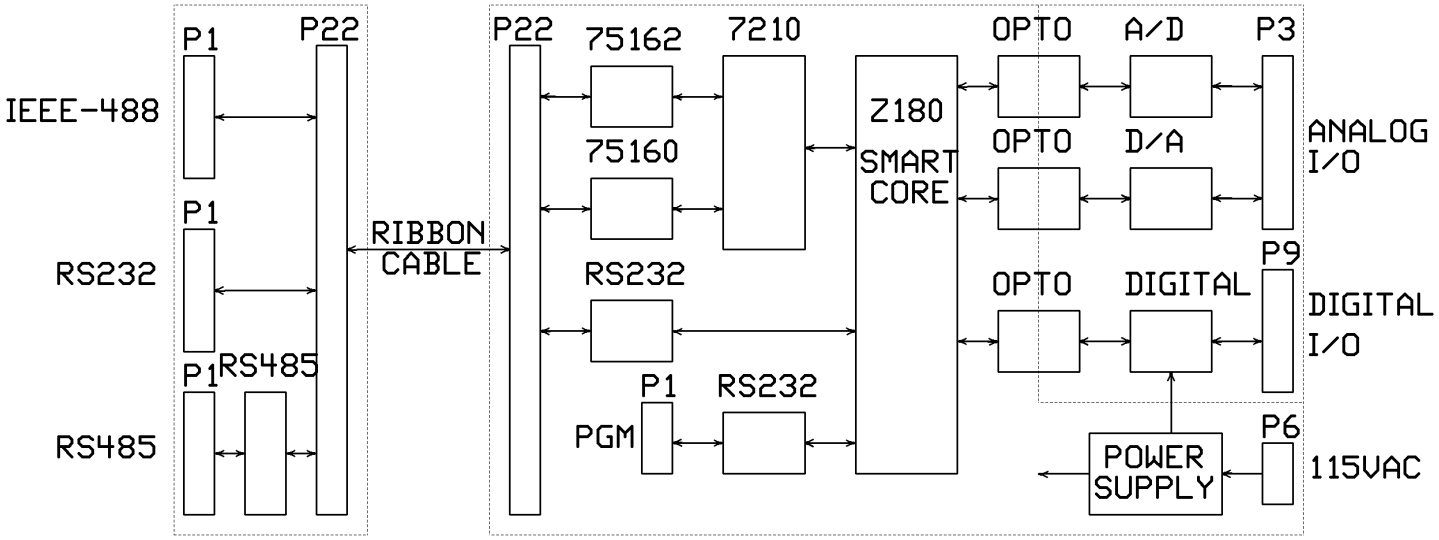


Figure 1, MT488A Block Diagram

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 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)

2.4 SmartCore(tm) Specifications

The MT488A 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).

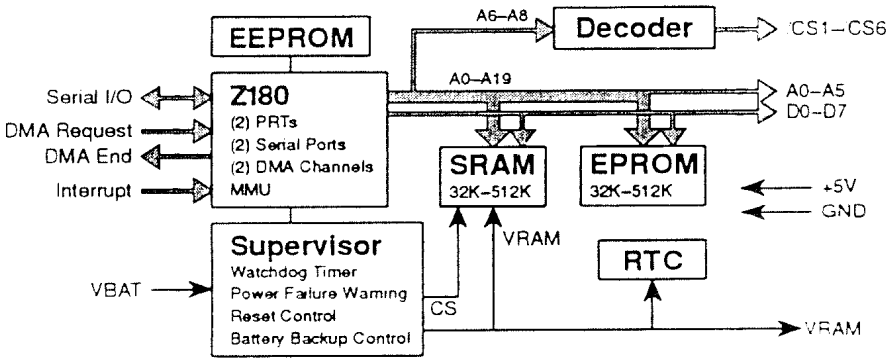


Figure 2, SmartCore Block Diagram

2.5 Indicators and Controls

a. Run/Program Switch

This switch controls the mode of operation of the MT488A and is read during power on. The MT488A will run the stored program in the run position. The MT488A will monitor the programming port for communications with the Dynamic C program.

Note: Cycle MT488A power after changing the Run/Program switch setting.

b. Power Switch

This rocker switch controls power to the MT488A.

c. DC Power connector

This connector accepts a 2.5mm DC barrel plug from a wall transfomer producing 9 to 24 VDC.

d. VFD Display

The VFD display is a 4 line X 20 Character Vacuum Fluorescent. Alternatively, a similarly configure LCD display is offered.

e. Membrane Switch Assembly, 4 X 6 array

2.6 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. P2 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

c. P3 Power

Pin	Name/Description
1	7-24 VDC

2	GND

d. P4 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

2.7 Physical Description

Dimensions: 10.20″ X 4.72″ X 1.89″
Weight: 1.5 lb.
Power Requirements: 6 VA, 5 W
Operating temperature: 0 to 50 ^{o}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 MT488A 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 MT488A. Note that the CTS signal (RTS at the IBM) should be driven true to enable communications.

3.3 Digital I/O

The MT488A'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 I^2C bus. Their I^2C 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 I^2C 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 MT488A 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).

3.4 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).

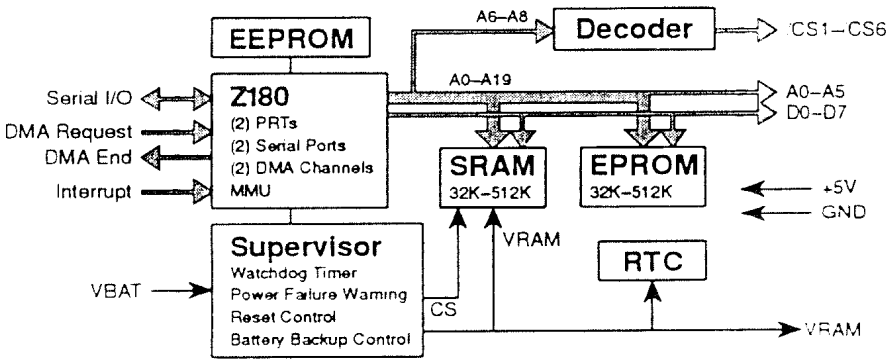


Figure 3, SmartCore Block Diagram

3.5 Power Supply Features

The MT488A features a switching power supply. The power supply provides power to the computer interface circuitry and to the I/O sections.

4.0 Operating Instructions

4.1 Unpacking

If the shipping carton is damaged on receipt call the carrier immediately. Then inspect the MT488A for damaged or loose components. If there is any damage notify your sales agent to obtain RMA authorization.

4.2 Setup and Configuration

The MT488A is ready to operate without any calibration or setup. To begin programming the MT488A for your application 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.

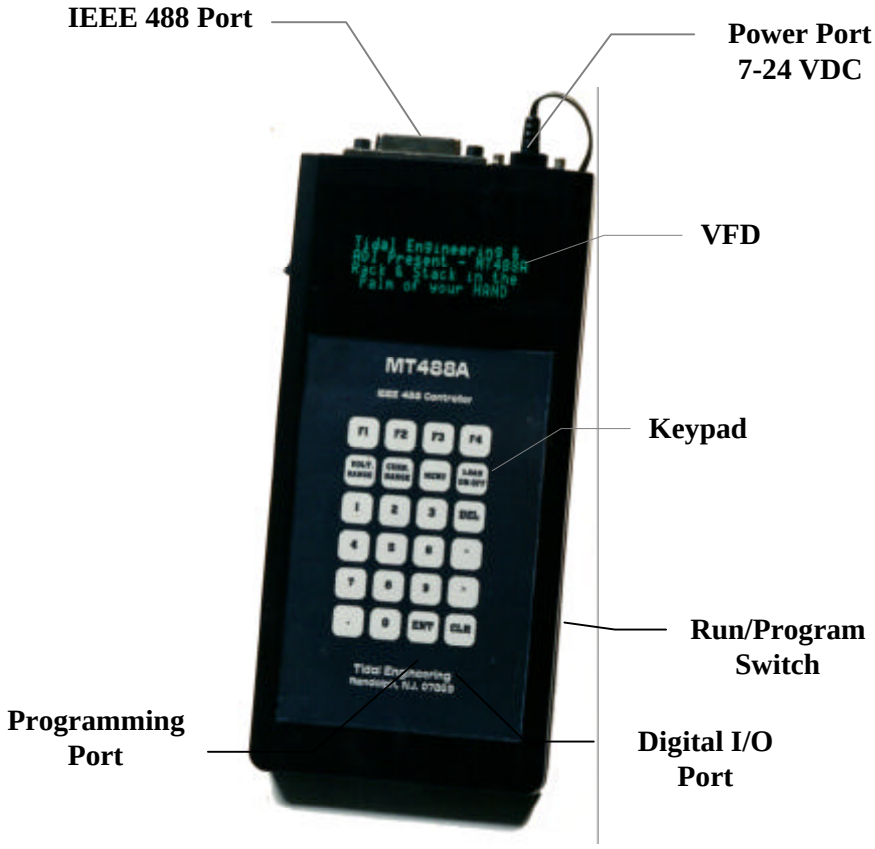


Figure 4 MT488A Interfaces

4.3 Software Installation

4.3.1 Dynamic C

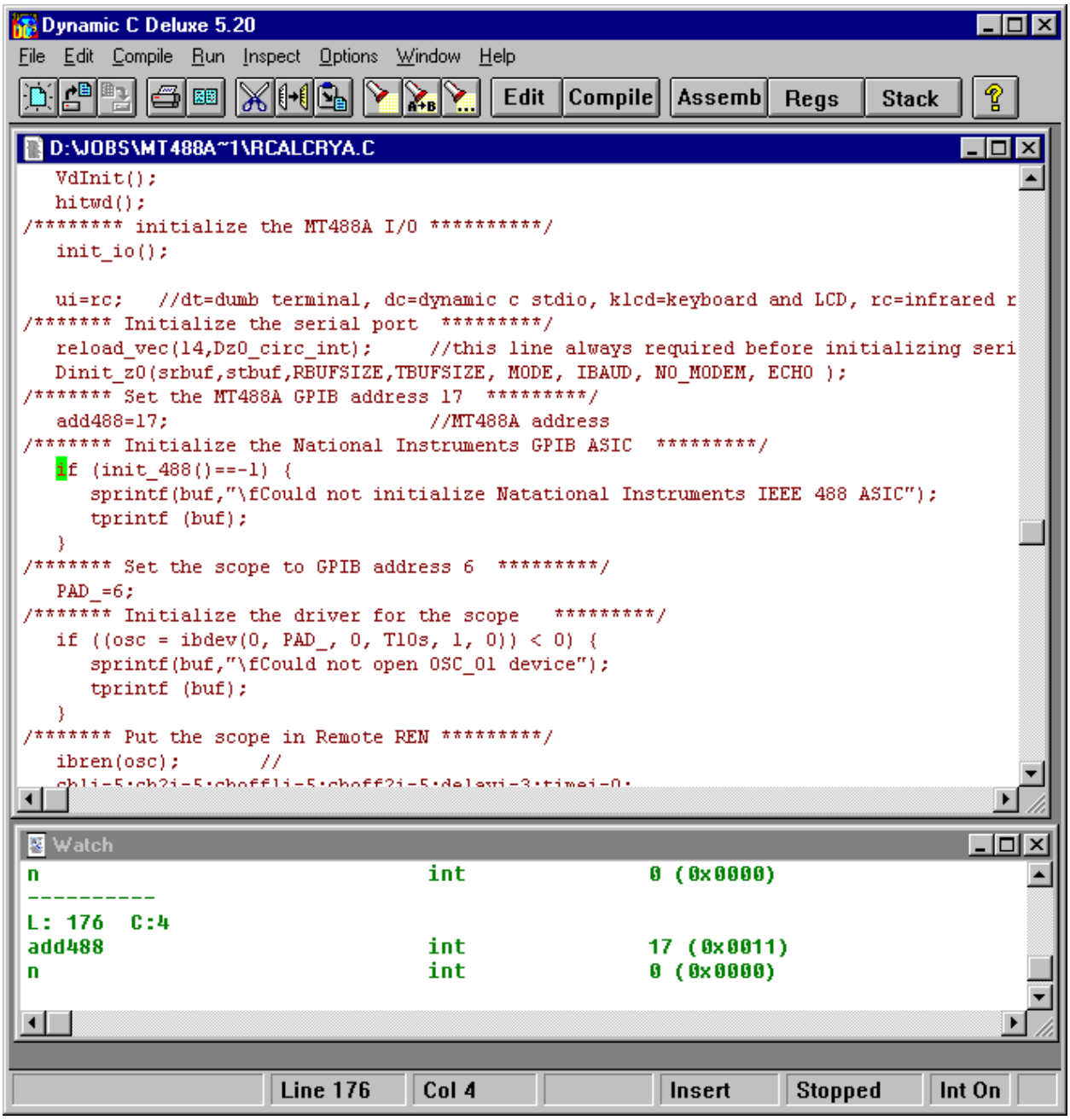
a) Install Dynamic C per the installation directions included with your Dynamic C documentation.

4.3.2 Libraries

a) Copy the MT488AV2 Library (MT488AV2.LIB) from the supplied disk to your Dynamic C library subdirectory (\LIB).
b) Add the MT488AV2.LIB library entry to the LIB.DIR file in your Dynamic C sub-directory using notepad or other text processor.

4.3.3 Example Programs

a) Copy the example programs from the supplied disk to an example directory on your hard drive.


Dynamic C Deluxe 5.20
File Edit Compile Run Inspect Options Window Help
Edit Compile Assemb Regs Stack
D:\JOBS\MT488A~1\RCALCRYA.C
VdInit();
hitwd();
/******** initialize the MT488A I/O **********/
init_io();
ui=rc; //dt=dumb terminal, dc=dynamic c stdio, klcd=keyboard and LCD, rc=infrared r
/******* Initialize the serial port *********/
reload_vec(14,Dz0_circ_int); //this line always required before initializing seri
Dinit_z0(srbuf,stbuf,RBUFSIZE,TBUFSIZE, MODE, IBAUD, NO_MODEM, ECHO);
/******* Set the MT488A GPIB address 17 *********/
add488=17; //MT488A address
/******* Initialize the National Instruments GPIB ASIC *********/
if (init_488()==-1) {
sprintf(buf,"\fCould not initialize Natational Instruments IEEE 488 ASIC");
tprintf (buf);
}
/******* Set the scope to GPIB address 6 *********/
PAD_=6;
/******* Initialize the driver for the scope *********/
if ((osc = ibdev(0, PAD_, 0, T10s, 1, 0)) < 0) {
sprintf(buf,"\fCould not open OSC_01 device");
tprintf (buf);
}
/******* Put the scope in Remote REN *********/
ibren(osc); //
Watch
n int 0 (0x0000)
L: 176 C:4
add488 int 17 (0x0011)
n int 0 (0x0000)
Line 176
Col 4
Insert
Stopped
Int On

5.0 Programming

The Dynamic C libraries included with the MT488A development package have interfaces for all of the SmartCore standard peripherals including EEPROM, Serial Ports, Real time clock, etc.

The MT488A specific drivers provide a high level interface to all of the MT488A specific peripherals.

The following are the device specific commands from the MT488AV2.LIB file.

Tidal Engineering Copyright 1997-1998
IEEE 488 and peripheral driver routines for the MT488A Minitest controller.

5.1 MT488A IEEE 488 User Functions

ibloc <MT488A.LIB>

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

ibren <MT488A.LIB>

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

ibdev <MT488A.LIB>

```
SYNTAX: int ibdev(int boardID, int pad, int sad, int tmo, int eot, int eos)
KEYWORDS: GPIB,IEEE 488
DESCRIPTION: Opens a GPIB device.  Write paramters to the device structure as
follows.  Only pad is used currently.
    device[devices].pad=pad;
    device[devices].sad=sad;
    device[devices].eot=eot;
    device[devices].tmo=tmo;
    device[devices].eos=eos;
RETURN VALUE: The handle to the device, else -1 if failed. (15 devices
maximum).
```

ibclr <MT488A.LIB>

```
SYNTAX: void ibclr(int ud);
KEYWORDS:   device clear, GPIB,IEEE 488
DESCRIPTION: Clears a GPIB device.  Ud is the device handle assigned by ibdev.
RETURN VALUE: None.
```

init_488 <MT488A.LIB>

```
SYNTAX: int init_488()
KEYWORDS:   Initialize GPIB,IEEE 488
DESCRIPTION: Initializes the MT488A GPIB controller.
RETURN VALUE: None.
```

ibrd <MT488A.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. Sets ibsta is there is an error during this operation.
```

ibwrt <MT488A.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.
   This operation is synchronous, it doesn't return until the transfer is
   complete or a time-out has occurred
RETURN VALUE: None. Sets ibsta is there is an error during this operation.
```

ieee_send <MT488AV2.LIB>

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

5.2 MT488A IEEE 488 Internal Functions

The following functions are not directly called by the user program.

wait_sync <MT488AV2.LIB>*

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

wait4co <MT488AV2.LIB>*

```
SYNTAX: int wait4co(void)
KEYWORDS: wait GPIB command
DESCRIPTION: Wait for GPIB command to be finished
RETURN VALUE:  Returns 0) for Timeout
               -1 for Error
               1 if successful
```

ieee_cmd <MT488AV2.LIB>

```
SYNTAX: int ieee_cmd(int cmd)
KEYWORDS: send GPIB command
DESCRIPTION: Send a GPIB command
RETURN VALUE:  Returns 0) for Timeout
               -1 for Error
               1 if successful
```

ieee_488 <MT488A.LIB> *

```
SYNTAX: void ieee_488();
KEYWORDS:   GPIB Interrupt, Interrupt
```

DESCRIPTION: Interrupt handler for IEEE 488 (GPIB) interface. Should not be called directly by user program, used only by library.
RETURN VALUE: None.

dspCIC <MT488AV2.LIB>

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

5.3 MT488A Peripheral Functions

eef_rd <MT488AV2.LIB>

SYNTAX: float eef_rd(int address)
KEYWORDS: eef_rd float EEPROM
DESCRIPTION: Does a read of a float from the EEPROM, 4 bytes used. Address requested is int address.
RETURN VALUE: Returns the float value.

eef_wr <MT488AV2.LIB>

SYNTAX: void eef_wr(int address, float data)
KEYWORDS: eef_wr float EEPROM
DESCRIPTION: Does a write of a float to the EEPROM, 4 bytes used. . Address to be written is int address. The data is float data.
RETURN VALUE: none

read_a2d <MT488AV2.LIB>

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

dac_wr <MT488AV2.LIB>

SYNTAX: void dac_wr (int address,int data)
KEYWORDS: dac_wr, Digital to Analog Converter
DESCRIPTION: Writes the data to the addressed dac channel
RETURN VALUE: none

wr_pio <MT488AV2.LIB>

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

rd_pio <MT488AV2.LIB>

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

rs232_send <MT488AV2.LIB>

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

hand_C64 <MT488AV2.LIB>*

SYNTAX: void hand_C64 ()
KEYWORDS: hand_C64
DESCRIPTION: Hand shakes with C64 peripheral controller
RETURN VALUE: none

i2c_write <MT488AV2.LIB>

SYNTAX: void i2c_write(int data)
KEYWORDS: i2c_write
DESCRIPTION: Does an I2C write through the C64 peripheral controller
RETURN VALUE: none

i2c_read <MT488AV2.LIB>

SYNTAX: void i2c_read(int flag)
KEYWORDS: i2c_read
DESCRIPTION: Does an I2C read through the C64 peripheral controller. The flag is set to 1 if more bytes will be read. Flag is set to 0 if this is the last byte to be transferred.
RETURN VALUE: none

i2c_start <MT488AV2.LIB>

SYNTAX: void i2c_start()
KEYWORDS: i2c_start
DESCRIPTION: Does an I2C start through the C64 peripheral controller
RETURN VALUE: none

i2c_stop <MT488AV2.LIB>

SYNTAX: void i2c_stop()
KEYWORDS: i2c_stop
DESCRIPTION: Does an I2C stop through the C64 peripheral controller
RETURN VALUE: none

adc_rd <MT488AV2.LIB>

SYNTAX: int adc_rd(int ad_add)
KEYWORDS: adc_rd Analog to Digital Conversion
DESCRIPTION: Does an ADC read through the C64 peripheral controller. The selected channel is ad_add (typically 0 thru 2).
RETURN VALUE: The integer value of the result 0 to 4095 for 12 bit converter.

dio_wr <MT488AV2.LIB>

SYNTAX: void dio_wr(int add, int dd)
KEYWORDS: dio_wr Write to digital I/O Port
DESCRIPTION: Does a write to digital I/O Port through the C64 peripheral controller. Add is the port address, 0 thru 3. Dd is the data, 0 thru 255
RETURN VALUE: none

rd_pio <MT488AV2.LIB>

SYNTAX: int rd_pio(int ad_add)
KEYWORDS: rd_pio Read from digital I/O Port
DESCRIPTION: Does a Read from digital I/O Port through the C64 peripheral controller. Ad_dd is the port address, 0 thru 3.
RETURN VALUE: The value is returned (0 thru 255)

rd_pic <MT488AV2.LIB>

SYNTAX: int rd_pic(int address)
KEYWORDS: rd_pic C64
DESCRIPTION: Does a read from the C64 peripheral controller. Address requested is int address.
RETURN VALUE: Returns the value of the memory location or port addressed.

wr_pic <MT488AV2.LIB>

SYNTAX: void wr_pic(int address, int data)
KEYWORDS: wr_pic C64
DESCRIPTION: Does a write to the C64 peripheral controller. . Address to be written is int address. The data to be written is int data.
RETURN VALUE: none

5.4 MT488A Other Functions

delay_ms <MT488AV2.LIB>

SYNTAX: void delay_ms(int ms)
KEYWORDS: delay_ms
DESCRIPTION: Delays ms milliseconds (approximate based on 9.18MHz processor)
RETURN VALUE: none

init_io <MT488AV2.LIB>

SYNTAX: void init_io()
KEYWORDS: init_io
DESCRIPTION: Sets up the I/O in the MT488A. Initializes wait states. N Clears the C64 controller.
RETURN VALUE: none

6.0 Example Programs

6.1 Example Program Source(RCALECRY.C)

```
//This program runs on the MT488A-RC
//the Infrared Remote Controlled IEEE 488 Controller.
/************************************************
    Tidal Engineering  (c) 1998
    www.gti.net/tidaleng
    1-800-877-0510
    RCALECRY.C
    Infrared (IR) Remote Controlled Oscilloscope
    Lecroy 9310
    Scope GPIB address should be set to 6
************************************************/

#use vdriver.lib
#use driver.lib
#use MT488AV2.LIB

/************   Hardware setup  ***************/
#define C64_ADD 7   //from sbc488an
#define C64_DAT 8
#define WR 16       //from sbc488an
/************   Buffer setups and global defines   ***************/
#define TBUFSIZE 384           // size of transmit buffer
#define RBUFSIZE 384           // size of receive buffer
#define ieee_in_len 128
#define ieee_out_len 128
#define false  0
#define true    -1

/************ User Interface codes defined *************/
#define klcd    1           //keyboard and LCD
#define dt      2           //dumb terminal,
#define dc      3           //dynamic c stdio
#define rc      4           //infrared remote control

/************ Communication paramters defined *************/
#define IBAUD 9600/1200    // baud rate
char MODE = 4;                 // 8 data, no parity, 1 stop
char NO_MODEM = 0;         // we don't want modem
char ECHO  = 0;            // we do not want character echo



int sys_cont;
char key[1];

char ui;
int devices,ibsta,add488, stb,esr;
/************ GPIB timeout and device structure *************/
#define to488      2       // timeout in 2 seconds
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;
    char stbuf[TBUFSIZE];       // transmit buffer
    char srbuf[RBUFSIZE];       // receive buffer
    char buf[RBUFSIZE+1];   // dummy buffer for receiving a complete command
```

```
    char s, t,data;
    char tbuf[ieee_out_len], rbuf[ieee_in_len];
/************ Inkey function ***************/
int in_key(char *data){ //n=in_key(key);//n=Dread_z01ch(key);  int Dread_z01ch(char *data);
    int n;
    *data=0;
    n=Dread_z01ch(data);
    if (strlen(data)==0)
        n=0;
    else
        n=1;
    return (n);
}
/************** tprintf (not used in RC) *************/
void tprintf( char *buft) {
    char buff[40], * gg;
    int del;
    gg="\f";
    switch(ui) {
        case klcd:  Dwrite_z0 (buft,strlen(buft)); break;
        case dc:        printf(buft); break;
        case dt:        Dwrite_z0 (buft,strlen(buft)); break;
        default:
            //do nothing when ui=rc
    }
}

/************** Main Function *************/
/************** Main Function *************/
main() {
    int i;
    int osc;
    unsigned int chan;
    int ii,n,j,z,v,PAD_;
    char temp[25];

/*  The following character strings set the Lecroy delay time  */
    char *delay[]={"TRDL 0\0xa",
                "TRDL 20\0xa",
                "TRDL 40\0xa",
                "TRDL 60\0xa",
                "TRDL 80\0xa",
                "TRDL 100\0xa"};

/*  The following character strings set the Lecroy's channel 1 Volts/Division */
    char *ch1[]={   "C1:VDIV 50mV\0xa",
                "C1:VDIV 100mV\0xa",
                "C1:VDIV 200mV\0xa",
                "C1:VDIV 500mV\0xa",
                "C1:VDIV 1V\0xa",
                "C1:VDIV 2V\0xa",
                "C1:VDIV 5V\0xa"};

/*  The following character strings set the Lecroy's channel 2 Volts/Division */
    char *ch2[]={"C2:VDIV 50mV\0xa",
                "C2:VDIV 100mV\0xa",
                "C2:VDIV 200mV\0xa",
                "C2:VDIV 500mV\0xa",
                "C2:VDIV 1V\0xa",
                "C2:VDIV 2V\0xa",
                "C2:VDIV 5V\0xa"};

/*  The following character strings set the Lecroy's channel 1 offset voltage */
    char *choff1[]={"C1:OFST -5V\0xa",
                "C1:OFST -.2V\0xa",
                "C1:OFST -.1V\0xa",
                "C1:OFST 0V\0xa",
                "C1:OFST .1V\0xa",
                "C1:OFST .2V\0xa",
```

```
                "C1:OFST .5V\0xa"};

/*  The following character strings set the Lecroy's channel 2 offset voltage */
    char *choff2[]={"C2:OFST -5V\0xa",
                "C2:OFST -.2V\0xa",
                "C2:OFST -.1V\0xa",
                "C2:OFST 0V\0xa",
                "C2:OFST .1V\0xa",
                "C2:OFST .2V\0xa",
                "C2:OFST .5V\0xa"};

/*  The following character strings set the Lecroy's sweep */
    char *time[]=
                  {"TDIV .01ms\0xa",
                   "TDIV .02ms\0xa",
                   "TDIV .05ms\0xa",
                   "TDIV .1ms\0xa",
                   "TDIV .2ms\0xa",
                   "TDIV .5ms\0xa",
                   "TDIV 1ms\0xa",
                   "TDIV 2ms\0xa",
                   "TDIV 5ms\0xa",
                   "TDIV 10ms\0xa",
                   "TDIV 20ms\0xa",
                   "TDIV 50ms\0xa",
                   "TDIV 100ms\0xa",
                   "TDIV 200ms\0xa",
                   "TDIV 500ms\0xa",
                   "TDIV 1s\0xa",
                   "TDIV 2s\0xa"};

    int ch1i,ch2i,choff1i,choff2i,timei,done1,delayi;

/******** initialize the Dynamic C virtual driver **********/
    VdInit();
    hitwd();
/******** initialize the MT488A I/O **********/
    init_io();

    ui=rc;  //dt=dumb terminal, dc=dynamic c stdio, klcd=keyboard and LCD, rc=infrared remote
/******* Initialize the serial port  *********/
    reload_vec(14,Dz0_circ_int);       //this line always required before initializing serial
port 0
    Dinit_z0(srbuf,stbuf,RBUFSIZE,TBUFSIZE, MODE, IBAUD, NO_MODEM, ECHO );
/******* Set the MT488A GPIB address 17  *********/
    add488=17;                                //MT488A address
/******* Initialize the National Instruments GPIB ASIC  *********/
    if (init_488()==-1) {
        sprintf(buf,"\fCould not initialize Natational Instruments IEEE 488 ASIC");
        tprintf (buf);
    }
/******* Set the scope to GPIB address 6  *********/
    PAD_=6;
/******* Initialize the driver for the scope   *********/
    if ((osc = ibdev(0, PAD_, 0, T10s, 1, 0)) < 0) {
        sprintf(buf,"\fCould not open OSC_01 device");
        tprintf (buf);
    }
/******* Put the scope in Remote REN *********/
    ibren(osc);         //
    ch1i=5;ch2i=5;choff1i=5;choff2i=5;delayi=3;timei=0;
    ibwrt(osc, ch1[ch1i], strlen(ch1[ch1i]));
   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");
    Dreset_z0rbuf();
/******* Main loop ***************************/
```

```
    while(1) {
        /******* Watch for a key press on the IR Remote *********/
        n=in_key(key);  //  *key=getchar(); use this line to enable stdio control
        if (n==1){
            switch(*key) {
                case 'a':   // Step up through Channel 1, V/Div table
                    if(++ch1i<7) ibwrt(osc, ch1[ch1i], strlen(ch1[ch1i]));
                    else ch1i=6;
                    break;
                case 'd':   // Step down through Channel 1, V/Div table
                    if(--ch1i>=0) ibwrt(osc, ch1[ch1i], strlen(ch1[ch1i]));
                    else ch1i=0;
                    break;
                case 'b':   // Step up through Channel 2, V/Div table
                    if(++ch2i<7) ibwrt(osc, ch2[ch2i], strlen(ch2[ch2i]));
                    else ch2i=6;
                    break;
                case 'e':   // Step down through Channel 2, V/Div table
                    if(--ch2i>=0) ibwrt(osc, ch2[ch2i], strlen(ch2[ch2i]));
                    else ch2i=0;
                    break;
                case 'c':   // Step up through sweep table
                    if(++timei!=17) ibwrt(osc, time[timei], strlen(time[timei]));
                    else timei=16;
                    break;
                case 'f':   // Step down through sweep table
                    if(--timei>=0) ibwrt(osc, time[timei], strlen(time[timei]));
                    else timei=0;
                    break;
                case 'g':   // Step up through Channel 1, offset table
                    if(++choff1i<7) ibwrt(osc, choff1[choff1i], strlen(ch1[choff1i]));
                    else choff1i=6;
                    break;
                case 'j':   // Step down through Channel 1, offset table
                    if(--choff1i>=0) ibwrt(osc, choff1[choff1i], strlen(ch1[choff1i]));
                    else choff1i=0;
                    break;
                case 'h':   // Step up through Channel 2, V offset table
                    if(++choff2i<7) ibwrt(osc, choff2[choff2i], strlen(choff2[choff2i]));
                    else choff2i=6;
                    break;
                case 'k':   // Step down through Channel 2, V offset table
                    if(--choff2i>=0) ibwrt(osc, choff2[choff2i], strlen(ch2[choff2i]));
                    else choff2i=0;
                    break;
                case 'i':   // Step up through delay table
                    if(++delayi<6) ibwrt(osc, delay[delayi], strlen(delay[delayi]));
                    else delayi=5;
                    break;
                case 'l':   // Step down through delay table
                    if(--delayi>=0) ibwrt(osc, delay[delayi], strlen(delay[delayi]));
                    else delayi=0;
                    break;
                case 'm':   // Auto sweep
                    ibwrt(osc, "TRMD AUTO\0xa", 10);
                    break;
                case 'n':   // Normal Sweep
                    ibwrt(osc, "TRMD NORM\0xa", 10);
                    break;
                case 'o':   // Single seep
                    ibwrt(osc, "TRMD SINGLE\0xa", 12);
                    break;
                default:

                    break;

            }
            delay_ms(250);
            Dreset_z0rbuf();
```

```
        }//if character received, i.e. if (n==1)
    }//While(1) loop
}//Main
```

6.2 Example Program Source(MTALECRYC.C)

```
//This program runs on the MT488A hand-held IEEE 488 Controller
/************************************************
    Tidal Engineering  (c) 1998
    www.gti.net/tidaleng
    1-800-877-0510
    MTALECRYC.C
    MT488A hand-held example, controls
    Lecroy 9310
************************************************/

#use vdriver.lib
#use driver.lib
#use MT488AV2.LIB

/************   Hardware setup  ***************/
#define C64_ADD 7   //from sbc488an
#define C64_DAT 8
#define WR 16       //from sbc488an
/************   Buffer setups and global defines   ***************/
#define TBUFSIZE 384           // size of transmit buffer
#define RBUFSIZE 384           // size of receive buffer
#define ieee_in_len 128
#define ieee_out_len 128
#define false  0
#define true    -1

/************ User Interface codes defined *************/
#define klcd    1           //keyboard and LCD
#define dt      2           //dumb terminal,
#define dc      3           //dynamic c stdio
#define rc      4           //infrared remote control

/************ Communication paramters defined *************/
#define IBAUD 9600/1200    // baud rate
char MODE = 4;                 // 8 data, no parity, 1 stop
char NO_MODEM = 0;         // we don't want modem
char ECHO  = 0;            // we do not want character echo

int sys_cont;
char key[1];

char ui;
int devices,ibsta,add488, stb,esr;
/************ GPIB timeout and device structure *************/
#define to488       2       // timeout in 2 seconds
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;
    char stbuf[TBUFSIZE];       // transmit buffer
    char srbuf[RBUFSIZE];       // receive buffer
    char buf[RBUFSIZE+1];   // dummy buffer for receiving a complete command
    char s, t,data;
    char tbuf[ieee_out_len], rbuf[ieee_in_len];
    char bofer[45];
/************ Inkey function ***************/
int in_key(char *data){ //n=in_key(key);//n=Dread_z01ch(key);  int Dread_z01ch(char *data);
    int n;
    *data=0;
```

```
    n=Dread_z01ch(data);
    if (strlen(data)==0)
        n=0;
    else
        n=1;
    return (n);
}
/************** tprintf (not used in RC) *************/
void tprintf( char *buft) {
    char buff[40], * gg;
    int del;
    gg="\f";
    switch(ui) {
        case klcd:  Dwrite_z0 (buft,strlen(buft)); break;
        case dc:        printf(buft); break;
        case dt:        Dwrite_z0 (buft,strlen(buft)); break;
        default:
            //do nothing when ui=rc
    }
}
void input( char *buft) {
    switch(ui) {
        case klcd:  while (Dread_z0(buft,ENTER) == 0 ){}; break;
        case dc:        gets(buft); break;
        case dt:        while (Dread_z0(buft,ENTER) == 0 ){}; break;
    }
}

/************** Main Function *************/
/************** Main Function *************/
main() {
    int i;
    int osc;
    unsigned int chan;
    int ii,n,j,z,v,PAD_,done1;
    char temp[25];
    /************** Init Strings for Lecroy 9310*****************/
    char *ch1[]="C1:VDIV 50mV\0xa";
    char *ch2[]="C2:VDIV 50mV\0xa";
    char *time[]="TDIV 50ms\0xa";

    //                  1                   1                   1                   //
    char *menu= "\fMT488A DemonstrationTo Adjust Scope     F1 = ch1, F2 = ch2  F3 = time\n";

    VdInit();                               // initialize the virtual driver
    hitwd();
    init_io();

    done1=false;
    ui=klcd;        //dt=dumb terminal, dc=dyanamic c stdio,klcd=keyboard and LCD

    reload_vec(14,Dz0_circ_int);       //this line always required before initializing serial
port 0
    Dinit_z0(srbuf,stbuf,RBUFSIZE,TBUFSIZE, MODE, IBAUD, NO_MODEM, ECHO );

    tprintf ("\fTidal\nEngineering\nPress ENTER key\nto continue");
    input (buf);
    add488=17;                              //MT488A address
    tprintf ("\fEnter MT488A address and\nPress ENTER key\n(default=17)");
    input (buf);
    if ((strlen(buf))!=0) add488=atof(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);
    }
    ibren(osc);         //Put Oscillator in REN

    tprintf (menu);
    ibwrt(osc, ch1[0], strlen(ch1[0]));
    if (ibsta & ERR) tprintf("     Could not write to device\n\r");
    ibwrt(osc, ch2[0], strlen(ch2[0]));
    if (ibsta & ERR) tprintf("     Could not write to device\n\r");
    ibwrt(osc, time[0], strlen(time[0]));
    if (ibsta & ERR) tprintf("     Could not write to device\n\r");

    while(1) {
        n=in_key(key);//n=Dread_z01ch(key);  int Dread_z01ch(char *data);
        if (n==1){
            i++;
            switch(*key) {
                case 'A':
                    tprintf ("\fEnter volts/div\nfor channel 1\nthen press enter\n");
                    input (buf);
                    sprintf(bofer, "C1:VDIV %sV\0xa",buf);
                    ibwrt(osc, bofer, strlen(bofer));
                    break;
                case 'B':
                    tprintf ("\fEnter volts/div\nfor channel 2\nthen press enter\n");
                    input (buf);
                    sprintf(bofer, "C2:VDIV %sV\0xa",buf);
                    ibwrt(osc, bofer, strlen(bofer));
                    break;
                case 'C':
                    tprintf ("\fEnter ms/div\nfor channel 1\nthen press enter\n");
                    input (buf);
                    sprintf(bofer, "TDIV %s ms\0xa",buf);
                    ibwrt(osc, bofer, strlen(bofer));
                    break;
                default:
                    tprintf ("\fUndefined key press\nPress enter key\nto continue\n");
                    input(buf);
                    break;
            }
            tprintf (menu);
            delay_ms(250);
            Dreset_z0rbuf();
        }
        if (done1) break;
    }
}
```

7.0 TROUBLESHOOTING

FAULT CONDITION	PROBABLE CAUSE	CHECK, REMEDY
No IEEE-488 Communications	Address errors Note 3	MT488A Dip switch. Valid addresses are 0 thru 30 Control program address
	Termination	Check control program termination, CR, LF and EOI
	IEEE-488 Cable	Replace Cable
	MT488A Power Supplies	Check power supplies on MT488A (U7).
	SmartCore Failure	Check data bus for activity
	7210 Chip failure	Check Interrupt line out of 7210 during transmission.
No RS-232 Communications	Address errors Note 3	MT488A Dip switch. Valid address is 31.
	Format error	Terminal program set for 9600,N,8,1
	RS-232 Cable	Replace Cable
	MT488A Power Supplies	Check power supplies on MT488A (U7).
	SmartCore Failure	Check data bus for activity
	RS-232 transceiver Chip failure	Check U85 RX and TX lines during transmission
D/A out voltage error	D/A power supply	Check D/A power supply voltage (U1 and U2)
	Opto-Isolator failure	Check U41 and U42 outputs while commanding voltage change.
	Voltage reference	Check U21 for 2.50 VDC
	D/A IC failure	Replace U11
	Control software	Check data for errors
A/D Readback error note 1	A/D power supply	Check A/D power supply voltage (U5 and U3)
	Opto-Isolator failure	Check U43, U46 and U45 outputs while reading voltage over bus.
	Voltage reference	Check U22 for 2.50 VDC
	A/D IC failure.	Replace U90
	Control software	Check data for errors

FAULT CONDITION	PROBABLE CAUSE	CHECK, REMEDY
Digital Readback error	Digital power supply	Check power supply voltage (U6)
	Opto-Isolator failure	Check U48 and U47 outputs while reading digital over bus
	Control software	Check data for errors
	Port IC failure	Replace port IC U51, U52 or U53
	Voltage levels	Verify input voltage levels are valid (See section 2)
Digital set error	Digital power supply	Check power supply voltage (U6)
	Opto-Isolator failure	Check U48 and U47 outputs while reading digital over bus
	Control software	Check data for errors
	Port IC failure	Replace port IC U51, U52 or U53
	load current note 2	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 MT488A power after changing program/run setting.

7.1 Digital I/O Wrap Around Test Connector

The following connector wiring can be used with MTADIGTS.C program to test the operation of the optional digital input output port.

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