MetraByte Corporation
440 Myles Standish Blvd.
Taunton, Mass 02780
(617) 880-3000

USERS GUIDE

for the

INTMDB-64

INTELLIGENT METRABUS CONTROLLER

MetraByte P/N: 64860
Release Date: 1/1/88

****

WARRANTY

All products manufactured by MetraByte are warranted against defective materials and workmanship for a period of One Year from the date of delivery to the original purchaser. Any product found to be defective within the warranty period will, at the option of MetraByte, be repaired or replaced. This warranty does not apply to products which have been damaged by improper use.

****

****

!! WARNING !!

MetraByte Corporation assumes no liability for damages consequent to the use of this product. **This product is not designed with components of a level of reliability suitable for use in life support of other extremely critical systems.**

****

MetraByte Corporation
440 Myles Standish Boulevard.
Taunton, MA. 02780 U.S.A
Phone: (617) 880-3000 Telex: 503989

INTMDB-64 TABLE OF CONTENTS ----------

INTRODUCTION to the MetraBus and the INTMDB-64

The MetraBus Industrial Data Acquisition and Control family is a low cost, high quality system employing a modular approach to real world analog and digital signal interfacing of large numbers of I/O points. It's broad range of computer interface cards allows MetraBus operation with all IBM PC/XT/AT compatibles, VMEbus, and other computers supporting serial communications.

The MetraBus, in its simplest form, consists of an interface card, a 50 conductor ribbon cable, and one or more Analog I/O, Digital I/O, Counter/Timer, Relay, Signal Conditioning, Breadboard, Diagnostic etc. remote boards.

The addition of MetraByte's INTMDB-64 interface card, with it's built-in microcomputer and program storage capability, allows true distributed intelligence for acquisition and control of even the most complex manufacturing processes. It's small size, very low cost, and proven reliability make this the system of choice for versatile, stand-alone acquisition and control applications. The INTMDB-64 was designed to replace the computer as the local MetraBus controller/driver or it may be configured as a local area controller with up to sixteen satellite systems (literally thousands of I/O points). All this from a single computer serial port.

The INTMDB-64 consists of a microcomputer with on-chip BASIC and 16KBytes (standard, expandable to 32KBytes) of EPROM for local program storage. This allows the board to run locally stored programs with or without intervention from a terminal, computer, or other ASCII compatible device. A built-in serial line printer interface and connector allows data and status information to be sent directly to printers, terminals, other computers, modems, process controllers, and other ASCII compatible serial devices.

The INTMDB-64 may be operated from a 12 volt (@ 500 mA) battery for field/remote use or as a power back-up where less than ideal supply voltages are prevalent. MetraBus power monitoring is done by the INTMDB-64 and is available as status bits. Auxiliary +5 or +12 volts supplies in conjunction with a user installed DC to DC converter allows the INTMDB-64 to handle power-on/off situations.

The INTMDB-64 contains the equivalent of a system Time-Out in the form of a programmable (via BASIC-52) WATCH-DOG timer. It is essentially a fail-safe device forcing a INTMDB-64/MetraBus RESET in the event of non-activity and, therefore, avoiding a system-wide "hang". The timer is jumper configurable for either 1 or 10 second cycles and, when enabled, is always reset upon MetraBus Read/Write cycles. If the WATCH-DOG timer is enabled and no MetraBus activity occurs for the selected period, the INTMDB-64 is reset as if a power-on reset occurred. The power-on reset condition would normally be configured to establish a fail-safe situation and await control commands to proceed. A pushbutton reset (jumper ENABLE/DISABLE) is also provided for ease of program development when on-going activity must be manually halted.

Up to 16 INTMDB-64 boards may be "chained" such that commands from a master (usually a PC type computer or terminal) are passed down the line to each board and responses relayed back up the line. A protocol system is used on each board so that only commands addressed properly are accepted. See multiple system block diagram "Computer With Satellite INTMDB-64 Systems".

BASIC language support is via the INTEL 8052AH microcontroller employing MCS BASIC-52 with an 8KByte (ROM) interpreter. MCS BASIC-52 is a powerful implementation of "standard" BASIC. Incorporated into this version of BASIC-52 are a multitude of additonal commands specific to control environments, taking advantage of the 8052AH architecture, and advanced MetraBus capability.

Program generation is accomplished via a serial (RS-232 or RS-422) link from a terminal or computer to the INTMDB-64. Programs may be written directly into INTMDB-64 RAM (8KBytes standard, expandable to 16KBytes) or may be stored on diskette and subsequently downloaded to the INTMDB-64. Once RAM resident, the programs may be run, tested, and modified at will.

Any BASIC-52 program(s), up to a maximum of 32KBytes (16KBytes standard, 16 KBytes optional) may be transferred to EPROM for non-volatile storage using a single command. INTMDB-64 resident programs may be automatically executed upon power-up.

MCS BASIC-52 and MetraBus command enhancements contain many unique features that would otherwise require assembly language programming. For example, the XBY and DBY operators allow READ/WRITE access to both external and internal memory respectively. The CBY operator is used to read program memory. The MBIN and MBOUT statements are used to input or output data from the MetraBus. The RESET statement clears the MetraBus. Most of the special function registers of the 8052AH may be accessed using MCS BASIC-52 and allows user access to timer or interrupt modes within the constructs of a BASIC program. An accurate, interrupt driven, real-time clock having a 5 mSec resolution is also implemented in MCS BASIC-52. This clock can be enabled or disabled and used to generate interrupts. Interrupts can also be generated when messages are received by means of the master/slave protocol system.

This manual covers operation of the INTMDB-64 and MetraBus specific implementation of the Intel 8052-BASIC microcontroller. Additionally, users should consult the following technical reference support documents. Note that the MCS BASIC-52 User Manual has been included with this manual and is extensively referenced throughout using the convention of square brackets, i.e [Introduction, p. 1].

MetraBus	MetraBus Manual, MetraByte Corporation.
SCM Modules	SCM-Series Manual, MetraByte Corporation.
BASIC-52	MCS BASIC-52 User Manual, Intel Corp. #2700100-002

QUICK and EASY

The INTMDB-64 was designed with a minimum of switches and jumpers making hardware configuration both quick and easy. This section will cover the essentials of configuring the board and allows you to actually run several simple example programs.

A word of caution. This "Quick and Easy" is exactly that and is not intended as a substitute for the descriptions, explanations, and examples contained elsewhere. The INTMDB-64 is a highly sophisticated, versatile, and quite powerful local controller that, like any microcomputer, is at its best when thoroughly exercised. A solid understanding of BASIC-52, the MDBINT-64, and the MetraBus cards that constitute your system will stand you in good stead when attempting some of the more rigorous acquisition and control techniques that you will surely want to try. We are interested in the applications that you develop using this controller and the MetraBus. If you feel that you have a particularly interesting application/program and would like to share it with others, please call (617) 880-3000 or write our Marketing Department.

Default Configuration

The following switch and jumper settings have been set at the factory prior to shipment. However, they should be checked before attempting to operate the INTMDB-64.

SW3	ON (allows EPROMs to be programmed)
J6	+5V (local power from MetraBus)
J7	10 SEC (watch-dog timer)
J9	PB ON (allows manual reset push-button)
SW2	ALL OFF (BASIC-52 default without protocol)

MetraBus Interface

The following boards constitute a minimum MetraBus system. Connect them using the 50 pin MetraBus ribbon cable:

POWER SUPPLY CARD (PWR-100 or equivalent)

METRABUS CARD (MDG-8 Diagnostic Card for this Example)

INTMDB-64 CARD

Communication Configuration

Any standard ASCII compatible terminal supporting either RS-232 or RS-422 may be used with the INTMDB-64. The terminal should be configured for the following protocol parameters and then connected to the INTMDB-64 via connector J3 for RS-232 and J2 for RS-422.

2400 Baud (optional due to auto-baud rate start-up routine)
7 data bits, 1 parity bit, and 1 stop bit
parity bit equal zero (SPACE)

Power the terminal and the MetraBus power supply. Depress the SPACE bar on the terminal to access BASIC-52. The 8250AH-BASIC initializes the hardware and executes an Auto-BAUD search routine (SW2 must be all OFF). The INTMDB-64 will answer the <SPACE> with the message *MCS-51(tm) BASIC V1.1*.

NOTE: The following conventions are used: **BOLD type** represents output from the terminal (user entered) while normal type represents INTMDB-64 response.

The sign-on message should look like this:

```
*MCS-51(tm) BASIC V1.1*
READY
>
```

If the above message does not appear, check all cables and configuration set-up for both the terminal and the INTMDB-64 again, then depress the <SPACE BAR>.

Preliminary Programming

Once you have signed-on, type the following simple program (<cr> means depress the "Enter" key):

```
>10 FOR I=1 TO 10      <cr>
>20 PRINT I            <cr>
>30 NEXT I             <cr>
```

Once entered, it is good practise to LIST the program and check for any typographical errors. This is done using the "LIST" command:

```
>LIST                  <cr>
```

You should see the following listing on your terminal screen:

```
10   FOR I=1 TO 10
20   PRINT I
30   NEXT I

READY
>
```

If errors are evident, simply re-enter the program and list it again. Once you have an error free listing, RUN the program using "RUN". The response should be as follows:

```
>RUN                   <cr>

1
2
3
4
5
6
7
8
9
10

READY
>
```

Other programs may be written within the context of BASIC-52 and, in fact, you are *encouraged to experiment and become familiar with BASIC-52. Once comfortable with* both the INTMDB-64 and BASIC, the MetraBus may be addressed and exercised.

Clear the existing program from INTMDB-64 memory using the "NEW" command. The following program is written using an MDG-8. An equivalent program using any other MetraBus board may also be used. This program will light the LEDs on the MDG-8 Diagnostic Board.

Set the MDG-8 to Base Address 32 and enter the following program (line 30 is a BASIC delay loop allowing a slow (visual) progression of LED lighting).

```
>NEW                          <cr>

>10 FOR I=1 TO 255            <cr>
>20 XBY(6000H+32)=I           <cr>
>30 FOR J=0 TO 20 : NEXT J    <cr>
>40 NEXT I                    <cr>
>50 GOTO 10                   <cr>

>RUN                          <cr>
```

The LEDs on the MDG-8 board will sequence. (This is actually a binary counting progression from 0 (all LEDs OFF) to 255 (all LEDs ON) and will repeat until the program is interrupted using the standard BREAK character <CTRL-C> or depressing the RESET button on the INTMDB-64.)

NOTE: Use of the RESET button will erase the program in memory and will require execution of the standard sign-on sequence (pressing the <SPACE BAR>).

A <CTRL-C> causes the INTMDB-64 to respond as follows with xxxx being replaced by the program line where execution terminated:

<CTRL-C>

```
STOP - IN LINE xxxx
READY
>
```

Storing Programs in EPROM

Any BASIC-52 program may be stored in EPROM on the INTMDB-64 up to the limits of storage capability. An Additional 16K EPROM may be purchased in the unlikely event that your application program exceeds 16KBytes (standard).

EPROM storage of programs is accomplished using the BASIC-52 **PROG** command. A program number is automatically assigned to the program (a clean EPROM will assign number 1).

```
READY
>PROG        <cr>
1

READY
>
```

In this case, the previous program (MDG-8 LED sequencing) will be stored as program 1 (if you have not erased it). Previously stored programs are retrieved using the "ROM x" command. To test this, you can clear the INTMDB-64 RAM using "NEW" and then RUN the program as follows:

```
READY
>NEW                    <cr>

>ROM 1                  <cr>

READY
>RUN                    <cr>
                        <CTRL-C>
STOP - IN LINE xxx
READY
>
```

The LEDS should be sequencing as before.

Implementing an Auto-Restart Routine

The INTMDB-64 is capable of executing any stored program from a Power-up state (including power restore subsequent to a power failure). This unique capability is similar to the standard AUTOEXEC.BAT file used via MS-DOS. Typing "PROG2" will save the current Communication Protocol (including BAUD rate) and will auto-execute the first program stored in EPROM (LED sequencing).

```
READY
>PROG2                    <cr>

READY
>
```

To test this, remove and restore MetraBus power (or use the RESET button). At power-up the LEDs will begin sequencing. Type <CTRL-C> to halt operations.

```
>RUN                      <cr>
                          <CTRL-C>
STOP - IN LINE xxxx
READY
>
```

NOTE: MCS BASIC-52 contains a minimum level line editor [Line Editor, p. 8] supporting only the DELETE key (remove last character) and the <CTRL-D> character (erase line). For a more elegant solution to this editing problem see the section in this manual on using a personal computer for program generation.

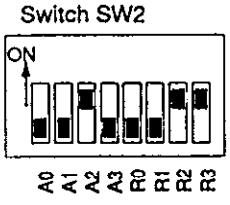
HARDWARE

Overview of Options

The INTMDB-64 may be configured for a wide variety of RESET and Communication options. These various options are set using the User Programs EPROM and/or a single 8-gang DIP switch. These options determine such basic communication parameters as console baud rate, and log-on message, as well as some of the more advanced features of the INTMDB-64 such as auto-execution of a BASIC-52 program after RESET or power-up. This advanced auto-execute feature allows the INTMDB-64 to function as a versatile, stand-alone, microcontroller for field/remote use or wherever a self-contained, intelligent controller is required.

The DIP switch on the INTMDB-64 is labeled as shown below and referred to as a hexadecimal byte with the least significant bit numbered 1 (A0) and the most significant bit labeled 8 (R3). The ON position signifies a 1 (High) while OFF signifies 0 (Low). The upper nibble (4 msb's) is for the various RESET options, RST OPT (R0-R3) and the lower nibble is the address, ADDR (A0-A3).

EXAMPLE DIP SWITCH SETTING



Example Shown: Value = 4 + 64 + 128 = 196 = 0C4H

RESET OPTIONS

DIP SWITCH = 00H -> STANDARD RESET OPTIONS

If the DIP switch is set for 00H (all switches OFF), the INTMDB-64 performs reset operations exactly as described in the MCS BASIC-52 User Manual [Reset Options, p. 145-146] and does not use special protocol when communicating to/from the console. This reset operation is partially summarized below. Note that because the INTMDB-64 does not have separate battery-backed memory, several of these reset options are equivalent.

SUMMARY OF STANDARD RESET OPTIONS DIP SWITCH = 00H)

EPROM COMMAND USED	SOURCE FOR BAUD RATE	EPROM PROGRAMMED	PROGRAM RUN AT RESET	COMMAND MODE & LOG-ON MESSAGE
NONE	SPACE	DON'T CARE	NO	YES
PROG1	EPROM	DON'T CARE	NO	YES
PROG2	EPROM	YES (1)	ROM 1	NO (1)
PROG3	EPROM	DON'T CARE	NO	YES
PROG4	EPROM	YES (1)	ROM 1	NO (1)
PROG5	EPROM	YES (1)	ROM 1	NO (1)
PROG6	USER	USER	USER	USER

NOTE: (1) If user program EPROM has not been programmed, then the standard log-on message is used and the command mode is entered.

DIP SWITCH <> 00H -> CUSTOM RESET OPTIONS

If the DIP switch is set to a value other than 00H, the program performs a customized reset operation [Reset Options, p. 147]. This option allow the user to select a baud rate and BASIC-52 program using the DIP switch. The standard reset options allow the first program in EPROM to be executed at power-on as described above; the customized reset options allow any program (1 through 9) to be executed at power-on. This unique feature allows autoexecution of a variety of application specific programs from a single EPROM. In addition, certain DIP switch settings implement console protocol operation with up to sixteen INTMDB-64s controlable from a single console or computer.

DIP SWITCH = 01H to 0FH -> CUSTOM RESET OPTIONS (NO PROTOCOL)

If the DIP switch is set for reset option 0 and address 1 to 0FH, console protocol is DISABLED. The baud rate, BASIC-52 program, and log-on message is determined by the table that follows.

RESET OPTION 0 - BAUD RATE, PROGRAM and LOG-ON (NO PROTOCOL)

RESET OPTION R3-R0	ADDRESS A3-A0	BAUD RATE	PROGRAM RUN AT RESET	LOG-ON MESSAGE
0	1	300	NO	YES
0	2	300	ROM 1 (1)	YES
0	3	300	ROM 1 (1)	NO
0	4	600	NO	YES
0	5	600	ROM 1 (1)	YES
0	6	600	ROM 1 (1)	NO
0	7	1200	NO	YES
0	8	1200	ROM 1 (1)	YES
0	9	1200	ROM 1 (1)	NO
0	A	2400	NO	YES
0	B	2400	ROM 1 (1)	YES
0	C	2400	ROM 1 (1)	NO
0	D	4800	NO	YES
0	E	4800	ROM 1 (1)	YES
0	F	4800	ROM 1 (1)	NO

NOTE (1) If user program EPROM has not been programmed, then the command mode is entered.

DIP SWITCH = 10H to 0FFH -> CUSTOM RESET OPTIONS (with PROTOCOL)

If the "ADDR RST OPT" DIP switch is configured for reset option 1 thru 0FH for any address, the console protocol is used. The baud rate, BASIC-52 program, and log-on message is determined by the following table. Notice that programs other than ROM 1 may be executaed at power-on reset. Each INTMDB-64 on the master/slave communication link must have a unique address. This address is set using the first four switches (S1-4) of the DIP switch and correspond to address bits A0-A3.

RESET OPTION 1 to 0FH - BAUD RATE, PROGRAM and LOG-ON (with PROTOCOL)

RESET OPTION R3-R0	ADDRESS A3-A0	BAUD RATE	PROGRAM RUN AT RESET	LOG-ON MESSAGE
1	ADDRESS	SPACE	ROM 1 (1)	NO
2	ADDRESS	EPROM (2)	ROM 1 (1)	NO
3	ADDRESS	300	ROM 1 (1)	NO
4	ADDRESS	600	ROM 1 (1)	NO
5	ADDRESS	1200	ROM 1 (1)	NO
6	ADDRESS	2400	ROM 1 (1)	NO
7	ADDRESS	4800	ROM 1 (1)	NO
8	ADDRESS	EPROM (2)	ROM 2 (1)	NO
9	ADDRESS	EPROM (2)	ROM 3 (1)	NO
A	ADDRESS	EPROM (2)	ROM 4 (1)	NO
B	ADDRESS	EPROM (2)	ROM 5 (1)	NO
C	ADDRESS	EPROM (2)	ROM 6 (1)	NO
D	ADDRESS	EPROM (2)	ROM 7 (1)	NO
E	ADDRESS	EPROM (2)	ROM 8 (1)	NO
F	ADDRESS	EPROM (2)	ROM 9 (1)	NO

NOTE

(1) If user program EPROM has not been programmed, the command mode is entered.

(2) If baud rate has not been stored in EPROM by PROG1 to PROG5 command, then baud rate is determined from the first "space" character from the console.

PROTOCOL OPERATION

Protocol operation is used to coordinate communications of multiple INTMDB-64/MetraBus systems. It is, therefore, optional for a single system. When used, all communication to and from the INTMDB-64 is via protocol software allowing protocol decode and INTMDB-64 address validation. As such, it can be thought of as a front-end coordinator for the INTMDB-64/MetraBus system(s). When protocol is off or unused (see reset options) the console terminal is "connected" to the board with all communications (J3, RS-232 or J2, RS-422) passing directly to BASIC-52.

This protocol front-end allows every INTMDB-64 in the communication system to listen to the commands and pass on to the BASIC-52 programs only those messages addressed properly. Likewise, responses from the INTMDB-64 to the console are only gated on the communication link if the console is "connected" to the particular board. Note that the baud rate for the console or computer and each INTMBD-64 must be the same.

The INTMDB-64 uses two flag bytes and one status byte allowing the system controller (a terminal or computer) to determine and effect operation of an INTMDB-64/MetraBus with minimal program (BASIC-52) interaction. This allows simultaneous operation of multiple INTMDB-64/MetraBus systems with status posting. A controller could monitor the status of each system on a polled basis and respond accordingly. The controller could also post instructions in all or several systems for synchronization or other action.

GROUP MESSAGE BYTE (1CH Internal Memory)

The group input message byte or flag is posted in all systems simultaneously by the group message command sequence. This flag is designed for operations such as synchronization (starting operations or setting time). Note that the group message "disconnects" the console. See the modified use of the statement ONEX1 for generating a BASIC-52 interrupt after a group message.

EXAMPLE: **GROUP INPUT MESSAGE BYTE**

```
10  TIME=0 : DBY(1CH)=0 : CLOCK 0
20  IF DBY(1CH)=0 THEN GOTO 20        wait to receive group message
30  CLOCK1
.
.                                     fall thru for non-group message
```

LOCAL INPUT MESSAGE BYTE (1DH Internal Memory)

The local input message byte or flag is posted in the addressed system by the local message command sequence. In multiple INTMDB-64/MetraBus systems, this flag serves to coordinate operations from the controller to the various INTMDB-64s. Note that the local message leaves the console "connected" to the addressed system. See the modified use of the statement ONEX1 for generating a BASIC-52 interrupt after a group message.

EXAMPLE: **Local Input Message Byte**

```
10  L=0 : DBY(1DH)=0
20  X=DBY(1DH)
30  IF X=L THEN GOTO 20
40  L=X : IF X>1 THEN GOTO 500
50  ON X GOTO 100,200                    Program is streered
.                                        by this statement
.
100 PRINT "MESSAGE WAS 0"
110 GOTO 20
200 PRINT "MESSAGE WAS 1"
210 GOTO 20
500 PRINT "MESSAGE OUT OF RANGE, ",X
510 GOTO 20
```

LOCAL OUTPUT STATUS BYTE (1EH INTERNAL MEMORY)

The local status byte can be read by the controller at any time using the interrogate command sequence (the INTMDB-64 must have enabled protocol using either the PROT1 command or by DIP switch option at reset). This status byte allows the INTMDB-64 to post a message to the controller in a non-interrupt, non-interactive (console disconnected) mode. Normally the status bytes are polled by the controller one at a time from each INTMDB-64/MetraBus system. Note that reading a status byte "disconnects" the console.

EXAMPLE: **SETTING the STATUS BYTE**

```
10  REM  **MAKE SOME MEASUREMENT OF VARIABLE X**
.
.
100 IF X<100 THEN DBY(1EH)=0 ELSE DBY(1EH)=100
```

PROTOCOL COMMAND SEQUENCE

The Protocol Command Sequence begins with an <ESC> character (1BH) located in the expansion EPROM at location 4003H. When the INTMDB-64 sees the <ESC> character, it disconnects BASIC-52 from the console (if connected) and proceeds to decode the command sequence that follows. Following the <ESC> character is a hexadecimal digit (0-9,A-F uppercase only) or a 'G'. This represents the system address or a group message. The exact character sequence is given in the **PROTOCOL CHARACTER KEY** section that follows. When a message is sent, it is echoed back by the addressed system (system address 0 for group message) and stored as pending (location 1BH). The controller examines the echo and indicates whether the message is valid (and, therefore, stored at location 1CH for group or 1DH for local) by acknowledging with a <CR> character or invalid by acknowledging with another character (or <ESC> which restarts the command sequence).

NOTE: The last <CR> causes the message byte to be stored.

PROTOCOL CHARACTER KEYS

<ESC>	Character stored at 2003H, normally ESC (1BH)
<0-F>	Hexadecimal character (0 to 9 or A to F upper case only)
<CR>	Character Carriage Return (0DH)
<LF>	Character Linefeed (0AH)
aa	Two hexadecimal characters representing local status byte at 1EH of internal memory
bb	Two hexadecimal characters representing local input message to be stored at 1DH of internal memory
cc	Two hexadecimal characters representing group input message to be stored at 1CH of internal memory
G	Character G (upper case only)
,	Character comma
0	Character zero

INTERROGATE STATUS

<ESC><0-F><CR>	Sent by Console (Master)
<0-F>,aa<CR><LF>	Response by Addressed Board (Slave)

INTERROGATE STATUS and ESTABLISH CONSOLE CONNECTION

<ESC><0-F><CR>	Sent by Console (Master)
<0-F>,aa<CR><LF>	Response by Addressed Board (Slave)
<CR>	Acknowledge by Console (Master)

SEND LOCAL MESSAGE and ESTABLISH CONSOLE CONNECTION

<ESC><0-F>,bb<CR>	Sent by Console (Master)
<0-F>,bb<CR><LF>	Response by Addressed Board (Slave)
<CR>	Acknowledge by Console

SEND GROUP MESSAGE for BOARD ADDRESS 0

<ESC>G,cc<CR>	Sent by Console (Master)
0,cc<CR><LF>	Response by Addressed Board (Slave)
<CR>	Acknowledge by Console (Master)

SEND GROUP MESSAGE for BOARD ADDRESSES 1 thru F

<ESC>G,cc<CR>	Sent by Console (Master)
<CR>	Sent by Console (Master)

CONNECTOR PIN ASSIGNMENTS

CONNECTOR PINOUT J1 (METRABUS MDB-64)

Mating Type: 50 Pin Ribbon

Pin	Signal	Direction
1	D0/	Bi-directional
3	D1/	Bi-Directional
5	D2/	Bi-directional
7	D3/	Bi-Directional
9	D4/	Bi-directional
11	D5/	Bi-Directional
13	D6/	Bi-directional
15	D7/	Bi-Directional
17	CLEAR/	Output
19	WSTRB/	Output
21	R/W/	Output
23	A0/	Output
25	A1/	Output
27	A2/	Output
29	A3/	Output
31	A4/	Output
33	A5/	Output
35	BUSY/	Input
2, 4, 6 38,40	GND	Ground
37	+15V	Power (Note 1)
39	-15V	Power (Note 2)
41,42...49,50	+5V	Power (Note 3)

Note 1: +15 Volts is not sourced on the INTMDB-64 but level tested only. It may optionally be used as back-up to battery supply.

Note 2: -15 Volts is not sourced, tested or used by INTMDB-64 board.

Note 3: +5 Volt is not sourced on the INTMDB-64 but level tested. Power for the INTMDB-64 may be from connector J1 or from the optional battery and DC-DC converter as set by jumper J6.

CONNECTOR PINOUT J2 (RS-422 TO/FROM MASTER)

Mating Type: DB9P

Pin	Signal	Direction
1	Signal Ground	
4 *	Receive (non-inverting)	To Board
5 *	Receive (inverting)	To Board
7	Signal Ground	
8 *	Send (non-inverting)	To Console or Previous Board
9 *	Send (inverting)	To Console or Previous Board

* NOTE: On-board termination of 220 ohms, inverting to non-inverting. Signals active only if RTS not active (or open) on RS-232 connector J3.

CONNECTOR PINOUT J3 (RS-232 TO/FOM MASTER)

Mating Type: DB25S

Pin	Signal	Direction
2	Received Data	To Board
3	Transmitted Data	To Console
4	Request to Send	To Board
5	Clear to Send	To Console
6	Data Set Ready	To Console
7	Signal Ground	
8	Carrier Detect	To Console

CONNECTOR PINOUT J4 (RS-422 TO/FROM SLAVE)

Mating Type: DB9P

Pin	Signal	Direction
1	Signal Ground	
4 *	Send (non-inverting)	To Next Board
5 *	Send (inverting)	To Next Board
7	Signal Ground	
8 *	Receive (non-inverting)	From Next Board
9 *	Receive (inverting)	From Next Board

* NOTE: On-board termination of 220 ohms, inverting to non-inverting.

CONNECTOR PINOUT J5 (RS-232 to LINE PRINTER)

Mating Type: DB25S

Pin	Signal	Direction
3	Transmitted Data	To Printer
6	Data Set Ready	To Printer
7	Signal Ground	
8	Carrier Detect	To Printer

CONNECTOR PINOUT J8 (OPTIONAL SPARE I/0)

Mating Type: 10 Pin Ribbon (See Note 1) 3M # 3446-6302

Pin	Signal	Direction
1	MetraBus CLEAR	Output
2	MetraBus CLEAR/	Output
4	P1.1/	Output
6	P1.1	Output
8	P3.5/	Input
10	P3.3/	Input
3,5,7,9	GND	Ground

Note 1: Requires user installation of 10 pin ribbon header, and 74LS04 integrated circuit at U35.

SOFTWARE

BASIC-52 GENERAL DESCRIPTION

MetraByte's INTMDB-64 is designed around the Intel 8052AH-BASIC microcontroller featuring a rich implementation of "standard" BASIC and includes a ROM resident (8K) BASIC interpreter, the MCS BASIC-52. The INTMDB-64 and associated firmware was specifically tailored to address the needs of process control, measurement, and instrumentation applications. MCS BASIC-52 is generally considered to be unmatched for these types of applications due to its ease of program development and straight forward, free form language format.

In addition to its rich complement of "standard" BASIC commands and functions, such as floating point arithmetic and transcendental operations, MCS BASIC-52 contains a multitude of unique enhancements. These functions add a good deal of flexibility and sophistication to the basic language while providing several tasks that would be otherwise unavailable without long, complicated assembly language development. Included are Bit-wise logical operators, such as AND, OR, and EXCLUSIVE OR are supported as well as hexadecimal arithmetic.

MCS BASIC-52 is an interpreted language allowing programs to be developed and executed interactively. This avoids the tedious process of editing, assembling, compiling, loading, running, etc. associated with most other computer languages. MCS BASIC-52 was designed for resident high level program development using the high performance 8052AH.

MCS BASIC-52 and MetraBus ENHANCEMENTS

This chapter covers several important details of the MCS BASIC-52 User's Manual, discusses the INTMDB-64 implementation of BASIC-52 and fully explains the BASIC-52 enhancements supporting the MetraBus. Source code for the various EPROM resident BASIC-52 enhancements and protocol support is available on disk from MetraByte for a nominal charge (this diskette also contains the terminal emulation package listed in APPENDIX D). MCS BASIC-52 source code is available from Intel Corporation through their user's library.

DEFINITION of TERMS

MCS BASIC-52 operates in two modes, a COMMAND or direct mode and the interpreter or RUN mode. COMMANDS will be executed only if the processor is in the COMMAND or direct mode. COMMANDS are acted upon immediately. As many as sixteen INTMDB-64s may be controlled from a single console operation. Certain MCS BASIC-52 functions are executed only PROTOCOL is OFF or connection has been established for a particular board (see chapter on PROTOCOL OPERATION).

A BASIC-52 program is composed of one or more statements [Defintion of Terms, p. 4]. These statements begin with a line number, followed by the statement body and terminated with a carriage return or colon (in the case of multiple statements). Certain statements can be executed in the COMMAND mode, others cannot. An arithmetic operator performs operations on variables and/or constants.

DATA FORMAT

MCS BASIC-52 supports real numbers from:

+/-1E-127 thru +/-.99999999E+127, inclusive.

Integers ranging from 0 thru 65536 (0FFFFH) may be entered as either decimal or hexadecimal. Hex numbers must begin with a valid digit. Variables are always stored as real numbers.

STACK STRUCTURE

MCS BASIC-52 reserves the first 512 bytes of external data memory for two "software" stacks [Stack Structure, p. 8]. These stacks are used for MetraBus I/O instructions as well as for data exchange (argument stack) using the PORT3 instruction. The control stack is described in the MCS BASIC-52 User's Manual and is used for loop control. The argument stack occupies locations 301 through 510 in external RAM memory. This stack stores all BASIC-52 constants currently in use. The argument stack holds data during all MetraBus Read/Write cycles. Data is either retrieved from the Metrabus (via the argument stack) by POPing it off the stack or written to the Metrabus (via the argument stack) by PUSHing it onto the stack. The PORT3 instruction also leaves the data in the argument stack.

COMMANDS, STRUCTURES, and OPERATORS

MCS BASIC-52 contains the following commands, statements, and operators.

COMMANDS	STATEMENTS	OPERATORS
RUN	BAUD	ADD (+)
LIST	CALL	DIVIDE (/)
LIST#	CLEAR	EXPONENTIATION (**)
NEW	CLEARS	MULTIPLY (*)
NULL	CLEARI	SUBTRACT (-)
RAM	CLOCK0	LOGICAL AND (.AND.)
ROM	CLOCK1	LOGICAL OR (.OR.)
XFER	DATA	LOGICAL X-OR (.XOR.)
PROG	READ	LOGICAL NOT
PROG1	RESTORE	ABS( )
PROG2	DIM	INT( )
FPROG	DO-WHILE	SGN( )
FPROG1	DO-UNTIL	SQR( )
FPROG2	END	RND
	FOR-TO-STEP	LOG( )
	NEXT	EXP( )
	GOSUB	SIN( )
	RETURN	COS( )
	GOTO	TAN( )
	ON-GOTO	ATN( )
	ON-GOSUB	=,>,>=,<,<=,<>
	IF-THEN-ELSE	ASC( )
	INPUT	CHR( )
	LET	CBY( )
	ONERR	DBY( )
	ONEXT1	XBY( )
	ONTIME	GET
	PRINT	IE
	PRINT#	IP
	PH0.	PORT1
	PH0.#	PCON
	PH1.	RCAP2
	PH1.#	T2CON
	PUSH	TCON
	POP	TMOD
	PWM	TIME
	REM	TIMER0
	RETI	TIMER1
	STOP	TIMER2
	STRING	TIME
	UI0	XTAL
	UI1	MTOP
	UO0	LEN
	UO1	FREE
		PI

INTMDB-64 IMPLEMENTATION EXCEPTIONS TO BASIC-52

PROG, PROG1, PROG2, PROG3, PROG4, PROG5, PROG6

"Program EPROMs" [EPROM File Commands, p. 23-27] is fully implemented but **requires that protocol operation be disabled prior to EPROM programming** (see BASIC-52 enhancement commands PROT0 and PROT1).

Protocol service routines reside in external EPROM. As a result, any serial port (console) activity while programming EPROMs will cause invalid program vectoring thus "hanging" the system. **Program voltage must be ON (SW3) when programming.**

FPROG, FPROG1, FPROG2, FPROG3, FPROG4, FPROG5, FPROG6

"Fast algorithm for programming EPROMs" [EPROM File Commands, p. 25-27] is **not** implemented. VCC on EPROM cannot be increased to 6 volts.

ONEX1 [ln num]

"On external interrupt" [ONEX1, p. 51] can be used by installing and wiring to optional connector J8 and installing optional inverter U35.

or

If bit 20.7H has been set by the user, any protocol message received will generate an external interrupt via software. (see example).

```
10  ONEX1 200
20  DBY(20H)=DBY(20H).OR.80H
.
.
200 PRINT #"MESSAGE RECEIVED", DBY(1CH), DBY(1DH)
210 RETI
```

PRINT@, PH0.@, PH1.@, LIST@

"Print user defined output routine" [LIST@, p. 17] [PRINT @, p. 59] is not implemented.

PUSH [expr]. POP [expr]

"Push data on argument stack" and "pop data off argument stack" [PUSH, POP, p. 60-61] are used with the MetraByte write and read statements and the read port 3 statement (see examples).

```
100  PUSH X : MBOUT 32   (write variable X to MetraBus address 32)
200  MBIN I : POP Y      (read MetraBus address I. Put data into variable Y)
300  PORT3 : POP Z       (reads Port 3 and puts data into variable Z)
```

UI1, UI0, UO1, UO0

"User console input/output enable/disable" [UI,UO, p. 67-68] should **not** be used because they are used by the protocol software. Use the enhancement commands of PROT0 and PROT1 for changing the user console input/output routines.

PGW

"Program EPROM in BASIC program" [PGW, p. 72] should **not** be used because the protocol flags will be destroyed. Also see caution for PROG statement.

Editing BASIC-52 Programs on a Personal Computer

MCS BASIC-52 contains a minimal line editor [Line editor, p.8]. A BASIC-52 program line cannot be changed once it has been entered. If corrections are necessary, the line must be re-typed. It is, however, possible to delete characters within a program line prior to the <cr>. This is done using the RUBOUT or DELETE character (7FH). RUBOUT will cause the last character entered to be erased from the text input buffer. Additionally, a control-D will cause the entire line to be erased.

If you are familiar with the IBM BASICA editor, it's use in generating MCS BASIC-52 is highly recommended since it is a relatively flexible line editor allowing several valuable options (MERGE, RENUMBER, etc. as well as allowing program storage in ASCII format). BASIC-52 programs will, obviously, not run on the IBM PC/XT/AT or compatibles, but program development is quite straight forward.

If you prefer, there are several very good page editors and word processing packages that may be used for program development. However, many of the packages append a preamble to text files automatically. Since the INTMDB-64 will not recognize this format preamble, it must be stripped out prior to downloading executable BASIC-52 code.

NOTE: Programs developed for the INTMDB-64 must be stored in ASCII format prior to downloading the INTMDB-64. This is easily done using the IBM BASCIA option ",A" as follows:

SAVE "file.xxx",A

The optional program diskette (containing source code for the BASIC-52 language enhancements also contains this program) is available, for a nominal charge, from MetraByte. This program (written in BASIC) is "EMUL.BAS and is reproduced in APPENDIX D.

MetraBus Language Enhancements

The following BASIC-52 language enhancements have been added to support the MetraBus functions. These are discussed in detail in the paragraphs that follow.

COMMANDS	STATEMENTS
DISP	RESET
PROT0	ENABLE
PROT1	DISABLE
EXAMPLE	MBIN
	MBOUT
	PORT3

NOTE: The language enhamcements are enabled after BASIC-52 enters the command mode by setting bit 45 [Command/Statement Extensions, pg 153]. However, the language enhancements may be enabled in a program from power-up (without setting bit 45) via the following line of code:

```
10        DBY(25H) = DBY(25H).OR.32
```

Failure to follow this procedure will result in a BAD SYNTAX ERROR when running a program under the above conditions (from power-up with language enhancements). Programs may, of course, be run from the

INTMDB-64 Language Enhancments

COMMAND **DISP [expr1],[expr2]**

ACTION TAKEN:

The DISP command displays on the console the hexadecimal contents of internal memory from [expr1] thru [expr2].

EXAMPLE:

>DISP 100,200 <cr>

```
0064 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 FF ED 89 00 45 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

COMMAND: **PROT0, PROT1**

ACTION TAKEN:

The PROT0 command disables console protocol operation and leaves the INTMDB-64 connected to the console. EPROM programming can proceed after this command is executed. The PROT1 command enables console protocol operation (the INTMDB-64 is "connected" to the console until protocol message is received). Note that these routines set/clear BIT 28 and 30 [UI1, UI0, UO1, UO0, p. 67-68] and redirect console input and output data through the protocol software.

NOTE: IF PROTOCOL IS IN EFFECT, PROT0 MUST BE ISSUED PRIOR TO EPROM PROGRAMMING.

EXAMPLE:

>PROT0 <cr>

>PROT1 <cr>

COMMAND: **EXAMPLE [expr]**

ACTION TAKEN:

The EXAMPLE [expr] command first executes a NEW and RAM command and then moves programming examples and test routines from the expansion EPROM to RAM where they can be executed using the RUN statement. The [expr] is the example number from 1 to the last program installed in the expansion EPROM. The following is an example usage as well as a list of the EPROM resident programs with [expr] number.

EXAMPLE:

>EXAMPLE2 <cr>

EPROM Resident Example and Test Routines

EXAMPLE	DESCRIPTION
1	Timeout of watch-dog timer
2	MetraBus test using MDG-8
3	Read PORT 3 bits
4	Monitor protocol to line printer
5	Check-out DIP switch

STATEMENTS: **RESET or RESET A**

MODE: **COMMAND and/or RUN**

ACTION TAKEN:

The RESET statement causes a 100 mSec CLEAR/ pulse to be clocked onto the MetraBus only. The RESET A (reset all) causes a power-on reset for both the MetraBus and the microcontroller (analogous to a system-wide CLEAR).

EXAMPLE:

>RESET **<cr>**

or

>RESET A **<cr>**

or

```
10  IF I>0 THEN RESET
```

STATEMENTS: **ENABLE, DISABLE**

MODE: **COMMAND and/or RUN**

ACTION TAKEN:

The ENABLE statement starts the watch-dog timer. The Watch-Dog timer performs an automatic INTMDB-64/MetraBus CLEAR & RESET in response to MetraBus non-activity within the specified time period (1 or 10 Seconds). This eliminates the possibility of a system wide "hang" (and subsequent manual RESET) in the event of an infinite loop or other atypical behavior. Note that the watch-dog timer is not disabled by program errors or program termination via <CTRL-C>. The DISABLE command must be explicitly given either as part of an ONERR statement or from the console. The watch-dog timer should be in the disabled state for INPUT statements. Also note that the watch-dog timer is disabled upon power-on or reset and must be explicitly enabled, if desired.

EXAMPLE:

```
10 ENABLE
50 DISABLE
100 IF I>10 THEN DISABLE
```

STATEMENTS: **MBIN [expr]**

MODE: **COMMAND and/or RUN**

ACTION TAKEN:

The MBIN [expr] statement reads the MetraBus and places the specified data onto the argument stack. Any integer from 0 thru 65535 inclusive may be spscified. Values outside these limits will be trapped by BASIC-52 with a subsequent "BAD ARGUMENT ERROR" message being transmitted to the controller. Values represented by 16 bits will be truncated to the lower six bits (0 to 63 inclusive) with this value being used as the MetraBus address.

The action of the MBIN [expr] can be described in the following steps:

[expr] is evaluated for address
MetraBus R/W line is evaluated for Write Complete
address placed on the MetraBus address lines
delay for address to settle
data read from MetraBus data bus
data placed on argument stack

Note that the MetraBus can be read using the XBY operator but R/W status checking is not done nor are the address lines allowed to settle.

EXAMPLE:

```
10 MBIN 32 : POP X

or

50 IF PORT1.AND.4 <> 0 THEN GOTO 50
60 X=XBY(6000H+32)
70 X=XBY(6000H+32)
```

Both examples read address 32 and place the data into variable X. Line 50 is simply checking for a completed R/W cycle. It is normally unnecessary, but may be used for succesive, high speed writes to the MetraBus.
Line 60 sets up the MetraBus address and line 70 reads the data after the address bus has settled.

STATEMENT: **MBOUT [expr]**

MODE: **COMMAND and/or RUN**

ACTION TAKEN:

The MBOUT [expr] statement is used to transfer data from the argument stack to the the MetraBus. Any integer from 0 thru 65535 inclusive may be specified in [expr]. Values outside these limits will be trapped by BASIC-52 with a subsequent "BAD ARGUMENT ERROR" message being transmitted to the controller. Values represented by 16 bits will be truncated to the lower six bits (0 to 63 inclusive) with this value being used as the MetraBus address.

MBOUT [expr] action can be described in the following steps:

[expr] is evaluated for address
MetraBus R/W line is evaluated for write complete
address placed on the MetraBus address lines
data from the argument stack placed on MetraBus data lines
and the R/W and WSTRB/ lines strobed

Note that the MetraBus can be written using the XBY operator but this does not check the R/W line or allow the address lines to settle.

EXAMPLE:

```
10 PUSH X : MBOUT 32

or

50 IF PORT1.AND.4 <> 0 THEN GOTO 50
60 XBY(6000H+32)=X
```

Both examples write to address 32 from variable X.
Line 50 is simply checking for a completed R/W cycle. It is normally unnecessary, but may be used for multiple, high speed data writes.

STATEMENTS: **PORT3**

MODE: **COMMAND and/or RUN**

ACTION TAKEN:

The PORT3 statement is used to interrogate (Read only) the voltage status bits and spare input lines of the 8052's port 3. The bits are mapped as follows:

BIT	USAGE
P3.0 *	RXD (serial received data)
P3.1 *	TXD (serial transmit data)
P3.2	+5 Volt status (0=low voltage, 1=OK)
P3.3	Spare input (J8)
P3.4	+15 Volt status (0=low voltage, 1=OK)
P3.5	Spare input (J8)
P3.6 *	WR/ signal
P3.7 *	RD/ signal

* Bit set to ZERO in assembly code for PORT3.

EXAMPLE:

```
10   PORT3 : POP X
20   IF X.AND.14H = 14H THEN GOTO 100
30   PRINT "LOW VOLTAGE DETECTED"
.
.
100
```

PORT USAGE

The folowing list specifies the port bit usage by the INTMDB-64 besides those standard to BASIC-52 (Special Function Operators, p. 86-89).

I/O PORT BIT ALLOCATION

BIT	USAGE	READ/WRITE	MSC BASIC-52 & EXTENTIONS
P1.6	MetraBus BUSY/ signal 0 = BUSY, 1 = normal	R	PORT1
P1.2	MetraBus R/W signal 0 = READ, 1 = WRITE	R	PORT1
P1.1	Spare (J8)	W	PORT1
P1.0	MetraBus CLEAR/ signal 0 = reset, 1 = normal	W	PORT1
P3.5	Spare (J8)	R	PORT3 : POP x (NOTE 1)
P3.4	+15 volt Status Signal 0 = low voltage, 1 = OK	R	PORT3 : POP x (NOTE 1)
P3.3	Spare (J8)	R	PORT3 : POP x (NOTE 1)
P3.5	+5 volt Status Signal 0 = low voltage, 1 = OK	R	PORT3 : POP x (NOTE 1)

NOTE 1: PORT3 is a BASIC-52 statement extension that leaves the byte value of PORT 3 on the argument stack. This value is accessed by POPing the value into a variable.

MEMORY USAGE

The folowing list specifies the locations in internal and external memory used by the INTMDB-64 besides those standard to BASIC-52 [Memory Usage, p. 166-169].

INTERNAL MEMORY ALLOCATION

Location	Usage
18H THRU 1AH	Temporary storage for Command Extensions
1BH	Pending message temporary storage
1CH	Group input message byte
1DH	Local input message byte
1EH	Local output status byte
1FH	Protocol status decode state
20H	Bits used specifically as follows
BIT 20.0H	Set when Protocol used
BIT 20.1H	Set when OK to send/receive console
BIT 20.2H	Set when serial bit TI received if OK to send/receive console.
BIT 20.3H	Set when serial bit RI received if OK to send/receive console.
BIT 20.4H	Set when receive not allowed
BIT 20.5H	Set when local message pending
BIT 20.6H	Set when group message pending
BIT 20.7H	Set by user if IE1 (TCON.3) to be set after valid message
21H	Used for SBUF when protocol used
BIT 23.4H	Console output control, Set during reset for protocol handling
BIT 23.6H	Console input control, Set during reset for protocol handling

EXTERNAL MEMORY ALLOCATION

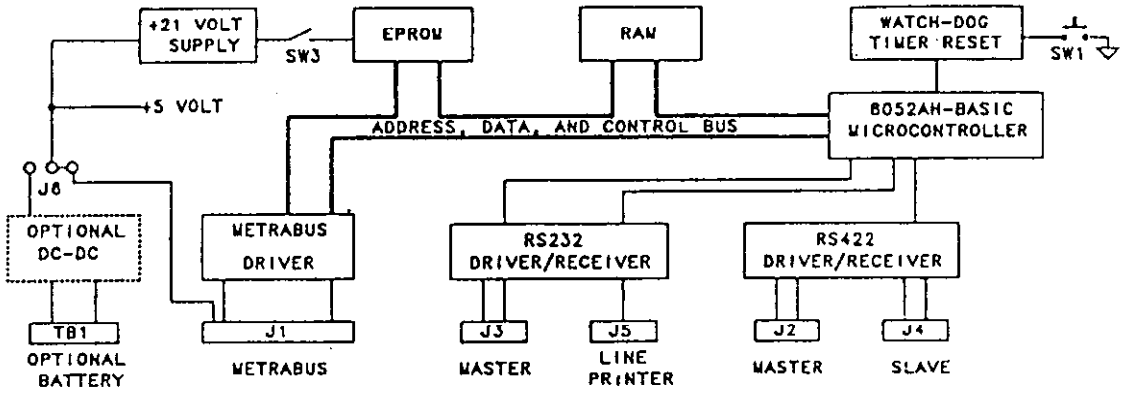
Location	MCS BASIC Usage
0000H THRU 1FFFH	Standard option memory (U18)
2000H THRU 3FFFH	User installable expanded memory (U19)
6000H	MetraBus read/write register
6040H	Read DIP switch, write complete RESET
6080H	Write enables watchdog timer
60C0H	Write disables watchdog timer

EPROM MEMORY ALLOCATION

Location	BASIC-52 Usage
2000H THRU 5FFFH	Command extention, reset, and protocol routines (U9)
8000H THRU BFFFH	Standard option EPROM user programs (U7)
C000H THRU FFFFH	User installable expanded EPROM user programs (U8)

APPENDIX A

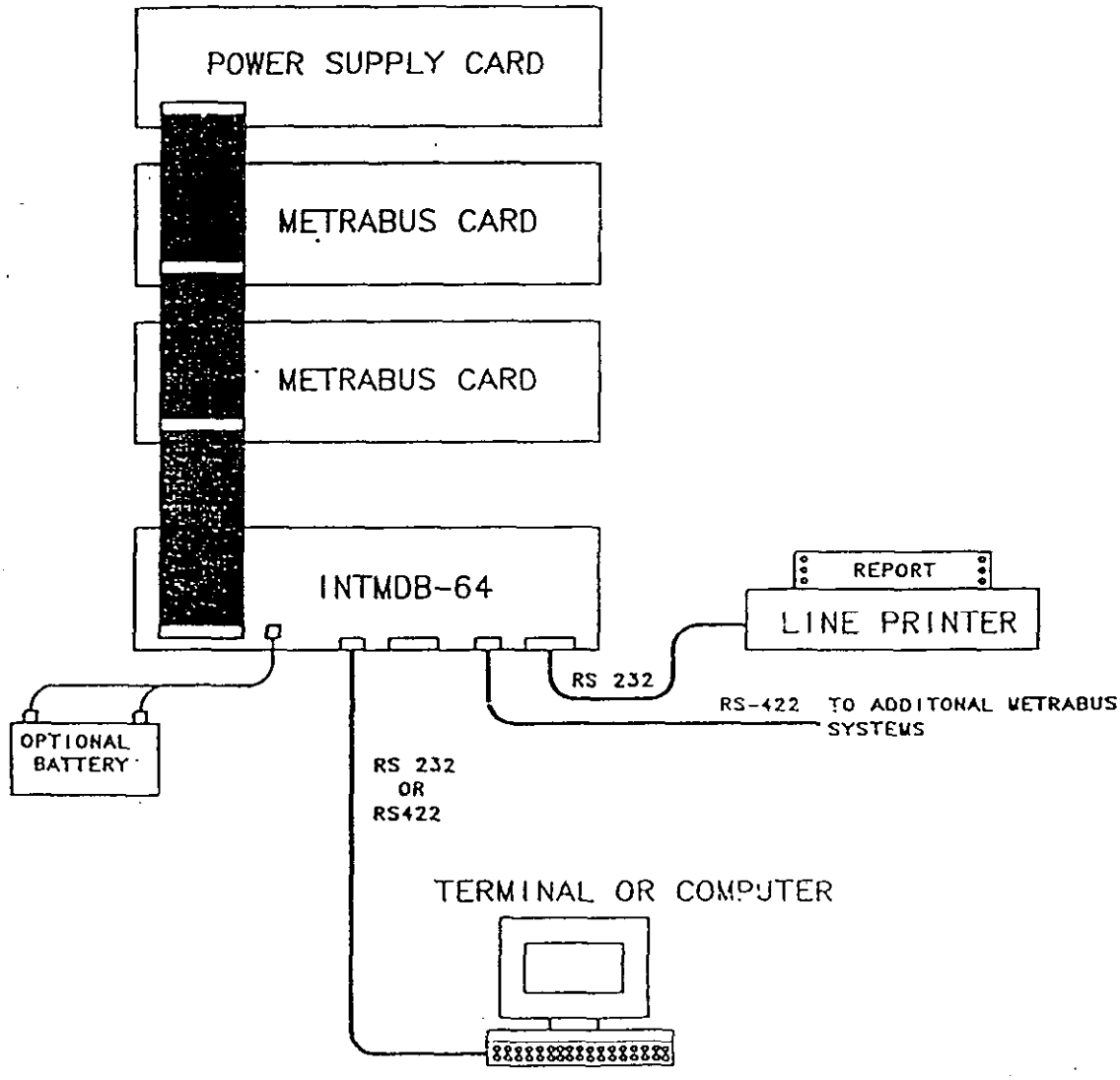
INTMDB-64 BLOCK DIAGRAM



INTMDB-64 BOARD BLOCK DIAGRAM

APPENDIX B

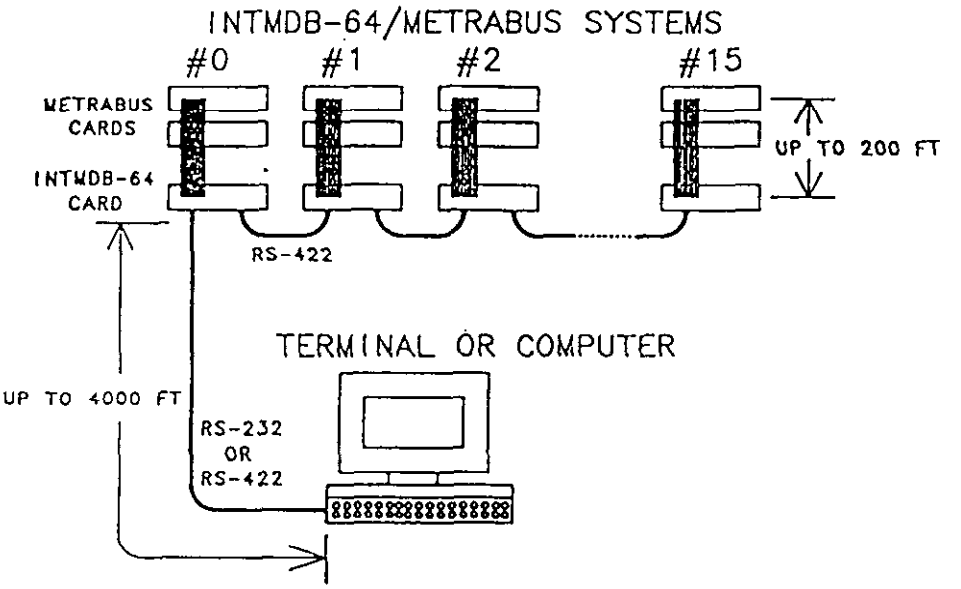
INTMDB-64 MetraBus System



INTMDB-64 METRABUS SYSTEM

APPENDIX C

INTMDB-64 Satellite System



COMPUTER WITH SATELLITE INTMDB-64 SYSTEMS

APPENDIX D

Terminal Emulation Program Listing

```
10    'EMUL.BAS
20    '*************************************************************************
30    '
40    '          CONSOLE  EMULATOR  FOR  INTMDB-64
50    '
60    '          VERSION:  1.0              JULY 20, 1987
70    '
80    '*************************************************************************
90    CLS : LOCATE ,,1 : DIM E$(100)
100   ON KEY (3) GOSUB 530 : KEY 3, "DWNLD"
110   ON KEY (4) GOSUB 710 : KEY 4, "UP LD"
120   ON KEY (5) GOSUB 950 : KEY 5, "RESET"
130   KEY (3) ON
140   KEY (4) ON
150   KEY (5) ON
160   KEY ON
170   GOSUB 980
180   C$="" : X=0 : ECHO=1
190   '************************** CONSOLE OPERATIONS **********************
200   ON X GOSUB 550 730, 970
210   A$=INKEY$ : IF A$<>"" GOTO 270
220   IF LOC(1)=1 THEN GOTO 220
230   B$=INPUT$(1,#1)
240   IF B$=CHR$(10) THEN GOTO 220
250   PRINT B$;
260   GOTO 220
270   IF A$=CHR$(27) THEN ECHO=0 : GOTO 340                    'Set No Echo
280   IF A$=CHR$(3) THEN PRINT #1,A$; : GOTO 200               'Control C
290   IF A$=CHR$(4) THEN GOTO 480                              'Control D
300   IF A$=CHR$(8) THEN GOTO 440                              'Backspace
310   IF A$=CHR$(17) THEN PRINT #1,A$; : GOTO 200              'Control Q
320   IF A$=CHR$(19) THEN PRINT #1,A$; : GOTO 200              'Control S
330   PRINT A$
340   C$=C$+A$
350   IF A$<>CHR$(13) THEN GOTO 200
360   PRINT #1,C$;
370   IF ECHO=1 THEN GOTO 400
380   IF C$=CHR$(13) THEN ECHO=1
390   C$="" : GOTO 200
400   IF X=3 THEN GOSUB 970 : GOTO 180
410   IF LOC(1)<LEN(C$) THEN GOTO 400                          'Flush ECHO
420   C$=INPUT$(LEN(C$),#1) : C$=""
430   GOTO 200
440   IF LEN(C$)=0 THEN GOTO 200
450   C$=LEFT$(C$,LEN(C$)-1)
460   PRINT CHR$(29);CHR$(255);CHR$(29);
470   GOTO 200
480   IF LEN(C$)=0 THEN GOTO 200
490   FOR I= 1 TO LEN(C$)
500   PRINT CHR$(29);CHR$(255);CHR$(29);
510   NEXT I
520   C$="" : GOTO 200
```

```
530     '*************************************************************************
540     X=1 : RETURN
550     INPUT "FILE TO DOWNLOAD";F$
560     ON ERROR GOTO 690
570     OPEN F$ FOR INPUT AS #2
580     ON ERROR GOTO 0
590     E$=""
600     IF EOF(2) THEN GOTO 670
610     LINE INPUT#2,D$
620     PRINT #1,D$
630     C$=E$
640     IF LOC(1)=0 THEN GOTO 640
650     E$=INPUT$(1,#1) : IF E$<>CHR$(10) THEN PRINT E$
660     IF (C$=CHR$(10) AND E$=">") THEN GOTO 600 ELSE GOTO 630
670     CLOSE #2
680     X=0 : LOCATE,,1 : RETURN
690     PRINT : PRINT "*****FILE ERROR*****" : PRINT
700     RESUME 550
710     '******************************** FILE UPLOAD ************************
720     X=2 : RETURN
730     INPUT "FILE TO UPLOAD TO";F$
740     ON ERROR GOTO 930
750     OPEN F$ FOR OUTPUT AS #2
760     ON ERROR GOTO 0
770     X=0
780     PRINT #1,"LIST"
790     IF LOC(1)=0 THEN GOTO 790
800     E$=INPUT$(1,#1)
810     IF E$<>CHR$(10) THEN GOTO 790
820     LINE INPUT #1, E$
830     IF LEFT$(E$,1)=CHR$(10) THEN E$=RIGHT$(E$,LEN(E$)-1) : GOTO 830
840     PRINT E$
850     IF MID$(E$,3,5)="READY" THEN GOTO 880
860     PRINT #2,E$
870     GOTO 820
880     IF LOC(1)=0 THEN GOTO 880
890     E$=INPUT$(1,#1) : IF E$<>CHR$(10) THEN PRINT E$;
900     IF E$<>">" THEN GOTO 880
910     CLOSE #2
920     LOCATE,,1 : RETURN
930     PRINT : PRINT "*****FILE ERROR*****" : PRINT
940     RESUME 730
950     '******************************** RESET COMM PORT ********************
960     X=3 : RETURN
970     CLOSE #1
980     OPEN "COM1:2400" AS #1
990     X=0
1000    RETURN
```

APPENDIX E

EPROM Installation & Erasure; Installing RAM

The INTMDB-64 may be configured with an additional 16Kbytes of EPROM for program storage as well as an additional 8KBytes of RAM for data storage. The optional EPROM is type 27128 (300 nSec) and may be cleaned (erased) with a strong ultraviolet light source. The EPROM must be removed from the socket (this is a zero force insertion socket) by simply lifting up on the metal wire bar that runs beneath the EPROM. Once removed, the EPROM should be exposed to an UV light source with safety interlock designed for EPROM erasure. These devices may be purchased from a variety of computer and electronic supply houses. Nominal erasure time is about 20 minutes using one of these devices.

Installing EPROM, whether new or ERASED, is simply a matter of lowering the metal bar, positioning the EPROM over the empty socket and gently pressing it in until seated. Avoid bending the legs on the chip. These legs are fragile and can only be straightened once or twice before breaking.

An additional 8KBytes of RAM may also be added to the INTMDB-64. The optional RAM is type HM-3-2064-5 (8K X 8, 150 nSec). It is inserted as above, using the same gentle pressure.

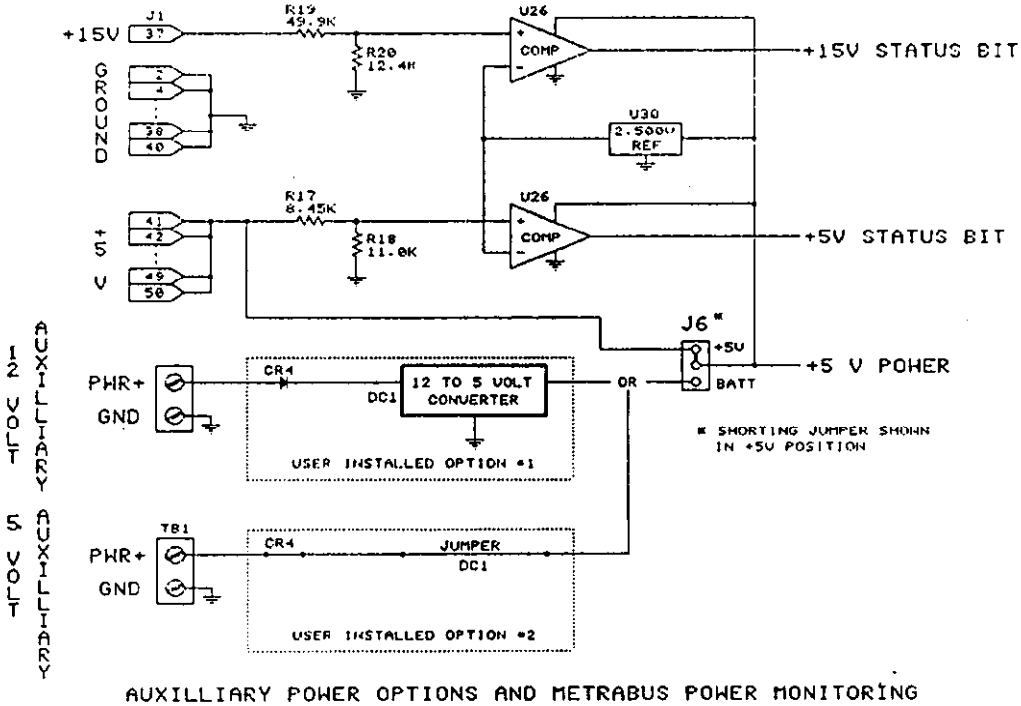
APPENDIX F

Installing the DC:DC Option

The INTMDB-64 requires an external source of power (+5 Vdc). This external power may be supplied by a MetraBus power supply board via the MetraBus cable (Jumper J6 should be in the +5V position) or by an external +5 or +12 Vdc supply wired directly to TB1.

If a +5 Vdc supply is used, the user must install a jumper (DC1) and a diode as shown below.

If a +12 Vdc supply is used, a 12V to 5V DC to DC converter is installed at DC1. The suggested part is International Power Devices available from MetraByte Corp. (P/N DUS1205).

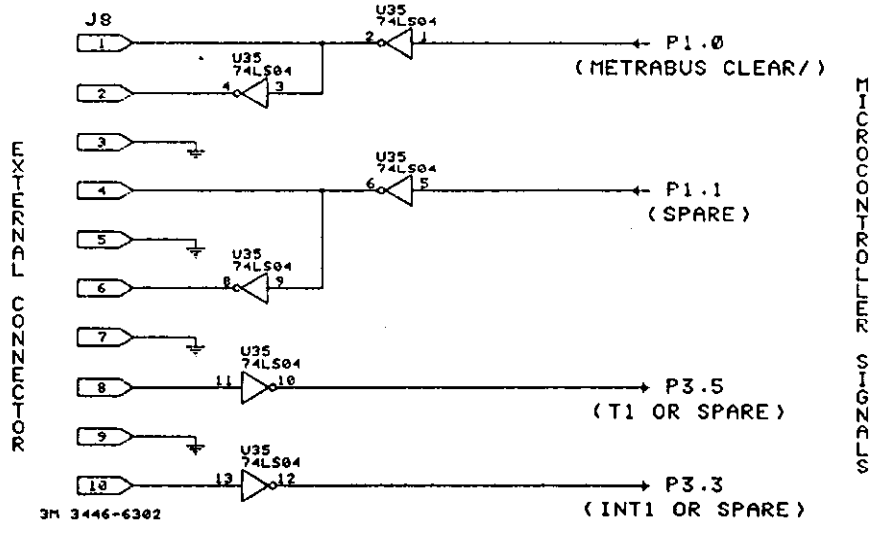


AUXILLIARY POWER OPTIONS AND METRABUS POWER MONITORING

APPENDIX G

Installing the Optional I/O Connector

An optional 10 pin header and ribbon cable allow output to other serial devices such as printers. This option requires a 10 pin header connector (3M #3446-6302) and an 74LS04 (U35) providing the required drive current for the printer. The connector pinout is shown below:

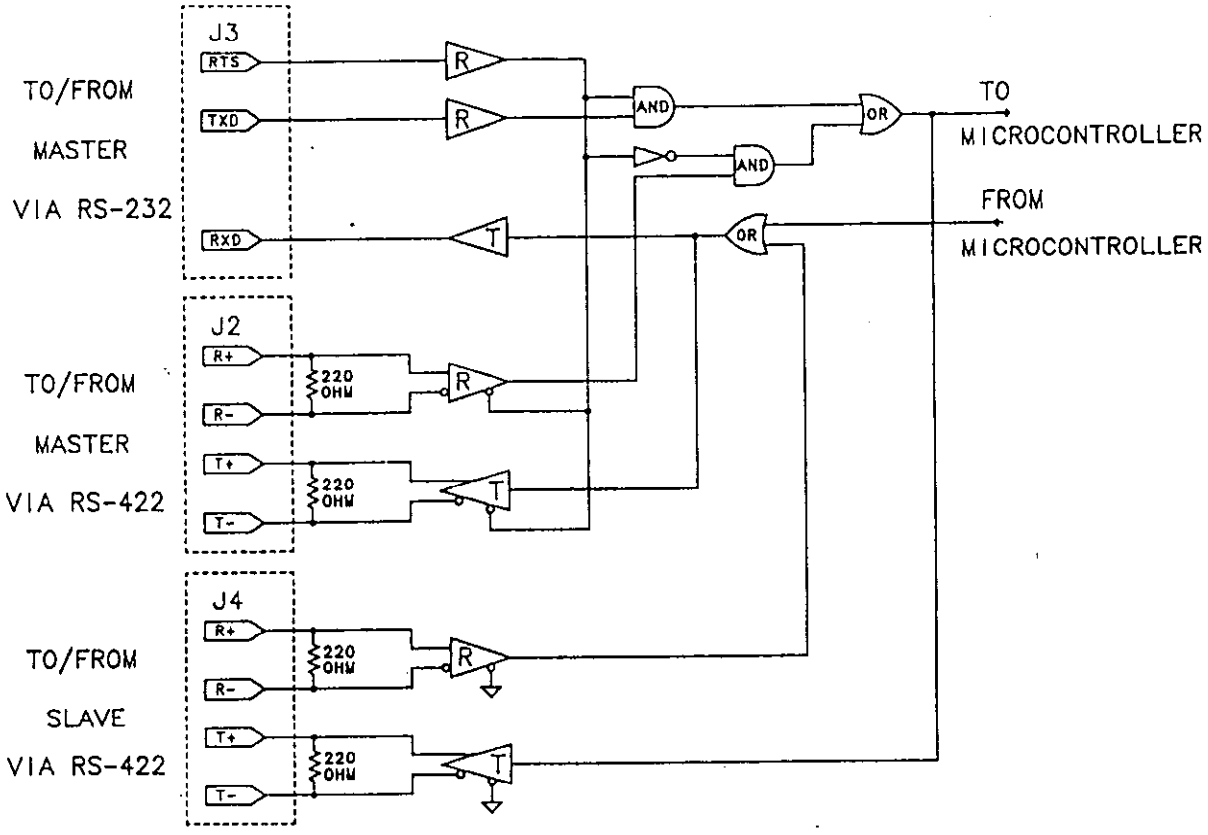


USER INSTALLED OPTIONAL EXTERNAL CONNECTOR

APPENDIX H

INTMDB-64 Communication Port Connections

The following diagram illustrates communication flow to and from the INTMDB-64:



SIMPLIFIED CONTROLLER COMMUNICATION PORT CONNECTIONS
(INTMDB-64)

APPENDIX I

SPECIFICATIONS

MetraBus Interface

I/O Ports:	64 Addresses
Data Bus:	8 Bits
Data Transfer Rate:	10,000 x 8-Bit Transfers/Sec (max)
Recommended Cable Length:	100 Feet (Full Speed) 200 Feet (Reduced Data Rate)
Power Level Monitoring: (STATUS BITS)	+5 Vdc Below 4.4 V +15 Vdc Below 12.6 V

Microcontroller

Programming Language:	INTEL BASIC-52
Program Enhancements:	ENABLE/DISABLE WATCH-DOG TIMER RESET MetraBus OUTPUT/INPUT MetraBus PROTOCOL ON/OFF READ PORT3 DISPLAY MEMORY LOAD EXAMPLE/TEST ROUTINES
EPROM Size:	16 KBYTES (Enhancements) 16 KBYTES (User Programs) OPTIONAL 16 KBYTES (User Programs)
EPROM Type:	27128 300 NSec(User Programs) 27128 200 NSec (Enhancements)
EPROM Erasure:	UV LIGHT (Remove EPROM from socket)
RAM Size:	8 KBytes RAM 8 KBytes RAM (Optional)
RAM Type:	HM-3-2064-5 (8K X 8, 150 NSec)
Programming:	EPROM, +21 Vdc Supply W/ DISABLE
Crystal Frequency:	11.05920 MHz

TIMING (MetraBus Control Signals and System Reset)

CLEAR:	100 mSec (min)
R/W:	10 uSec
WSTRB:	5 uSec
WATCH-DOG TIMER:	1 or 10 Sec RESET (via Jumper) If ENABLED

COMMUNICATION PARAMETERS

Master Control:	RS-232 or RS-422 Ports
SLAVE Pass-through:	RS-422 (Only)
Line Printer:	RS-232
Optional I/O:	Via 10 Pin Header (User Installed)
BAUD Rate:	All Standard Rates Thru 19.2K BAUD
Recommended BAUD rate:	2400 or 4800
Protocol (Actual):	8 DATA, NO PARITY, 2 STOP
Protocol (Equivalent):	7 DATA, PARITY, 1 STOP PARITY Equals SPACE (ZERO)

POWER CONSUMPTION

+5 Vdc:	800 mA (max) (Full Memory, +21V ON and EPROM Programming) 600 mA (+21V OFF)

BATTERY OPERATION (User Installed)

DC-DC Converter:	INTERNATIONAL POWER DEVICES P/N DUS1205 (or Equivalent)
Required Power:	12V (Nominal, +/- 10% at 500 mA) via Screw Terminal TB1
Optional External Connector:	10-pin Ribbon Header (3M #3446-6302)
Circuitry:	74LS04 (or Equivalent)

PHYSICAL and ENVIRONMENTAL

Size:	16 X 4.75 Inches (40.63 X 12.06 cm)
Operating Temp:	0 to 70 Deg C
Storage Temp:	-40 to 100 Deg C