

Part I of EX9188 H/W Manual Contents:

1. Introduction.....	2
2. System specification.....	3
2.1 Block Diagram.....	3
2.2 Features & Specification.....	4
3. Pin Assignment.....	6
3.1 Dimension.....	6
3.2 Pin Assignment of 10-pin screw terminal Block.....	6
3.3 Pin Assignment of COM1 converter (DB-9 male).....	7
3.4 Jumper.....	7
4. System Mapping.....	8
4.1 Memory Mapping.....	8
4.2 Interrupt Mapping.....	8
4.3 GPIO Distribution.....	9
5. Operation Principle.....	10
5.1 Use COM4 for Debug system.....	10
5.2 Use COM4 for download program.....	10
5.3 Use COM3 for RS232 port.....	11
5.4 Use COM2 for RS485 port.....	12
5.5 Use COM1 for RS232 port.....	13
5.6 Use COM1 for RS485 port.....	14
5.7 7 Segment & LED display control.....	14
5.8 Use EEPROM.....	15
5.9 Use RTC & NVRAM.....	15
5.10 Use Watchdog Timer(WDT).....	15

1. Introduction

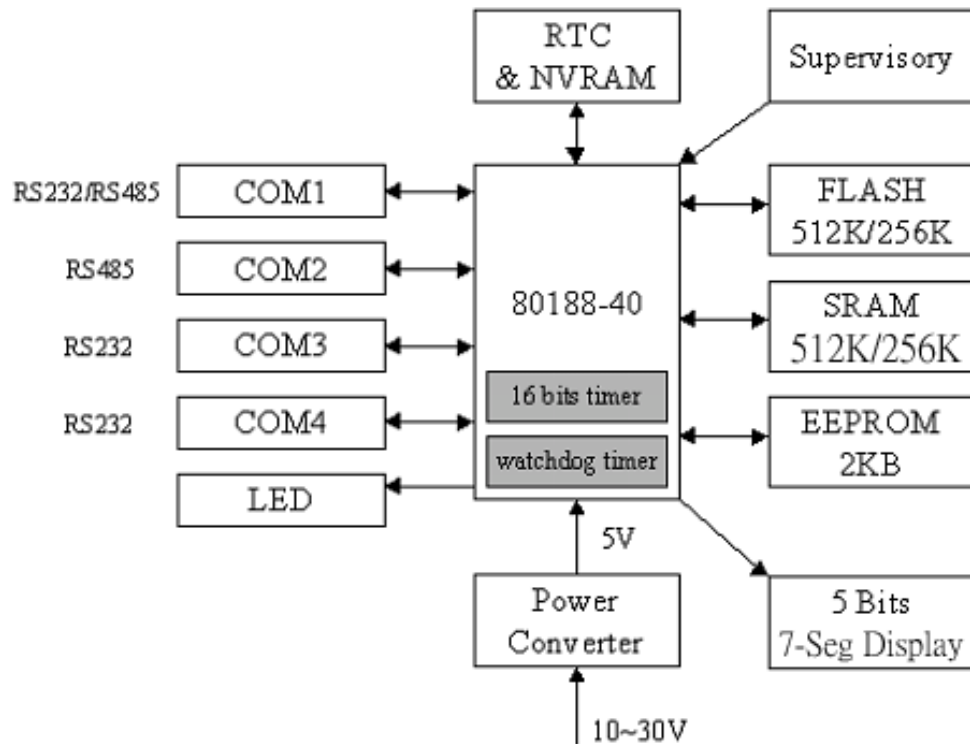
EX9188 module is a multi-purpose embedded controller and provides a low-cost, high –performance solution for various industry applications. The multi serial ports of EX9188 can operate several control interfaces simultaneously so that different devices can connect together. It can also be used to replace the PC or PLC under industrial control.

EX9188 is equipped with AMD 80188 microprocessor, SRAM, flash ROM, EEPROM, 4 communication ports, 5-digit LED display and a real time clock (RTC). There are two RS232 ports (COM3 and COM4), one RS485 port (COM2) and one RS232/RS485 selectable port (COM1).

The operation system (ROM-DOS) is included in EX9188 module. ROM-DOS is functionally equivalent to other brands of DOS. A standard DOS executable program is able to run in ROM-DOS. Users can use control programs to download or upload any data via COM4 port under ROM-DOS.

2. System Specification

2.1 Block Diagram



Default COM port configuration:				
	Baudrate	Dtatbit	Parity	Stopbit
COM1	9600	8	N(none)	1
COM2	9600	8	N(none)	1
COM3	9600	8	N(none)	1
COM4	57600	8	N(none)	1

2.2 Features & Specifications

2.2.1 CPU:

Am188™ ES CPU, 40MHz, 1MB Memory size, 1MB I/O Size, 32 GPIOs, 6 Interrupts, 2 Timers.

2.2.2 SRAM:

Static Memory A617308V-12x2, 256KB, for storing control programs and ROM-DOS can make it to be a RAM-DISK drive.

2.2.3 Flash Memory:



Flash ROM Am29F010-70EC, 512KB, ROM-DOS can make it to be a ROM-DISK drive.

2.2.4 NVSRAM:

DS1302 build-in NVSRAM, 32 bytes for keeping data for 10 years on battery support.

2.2.5 EEPROM:

EEPROM, 2048bytes, for storing parameters or variable data.

2.2.6 Real Time Clock:

DS1302, Y2K-compliant real time clock from year 1980 to 2079.

2.2.7 COM1:

COM1 is selectable for RS232/RS485. Maximum baud rate is 115,200bps.

2.2.8 COM2:

COM2 for RS485 communications. Maximum baud rate is 115,200bps.



2.2.9 COM3

COM3 for RS232 communication. Maximum baud rate is 115,200bps.

2.2.10 COM4:

COM4 for RS232 communication. Maximum baud rate is 115,200bps.

2.2.11 5 units of 7-segment LED display:

7-segment LED display can display 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f and a float point.

2.2.12 Digital Input Initial Pin:

INIT* signal pin is for users to update the disk image.

Note: INIT* must be disconnected from ground immediately before the updating disk image completes, otherwise the system will hang.

2.2.12 One LED:

Users can control LED On or Off.

2.2.14 Power consumption:

Power input :10V~30V DC

EX9188 module static power consumption is 2.0W. Dynamic power consumption is 3.0W.

2.2.15 Dimensions and Environment:

Module Dimension: 117 mm x 71 mm x 26mm

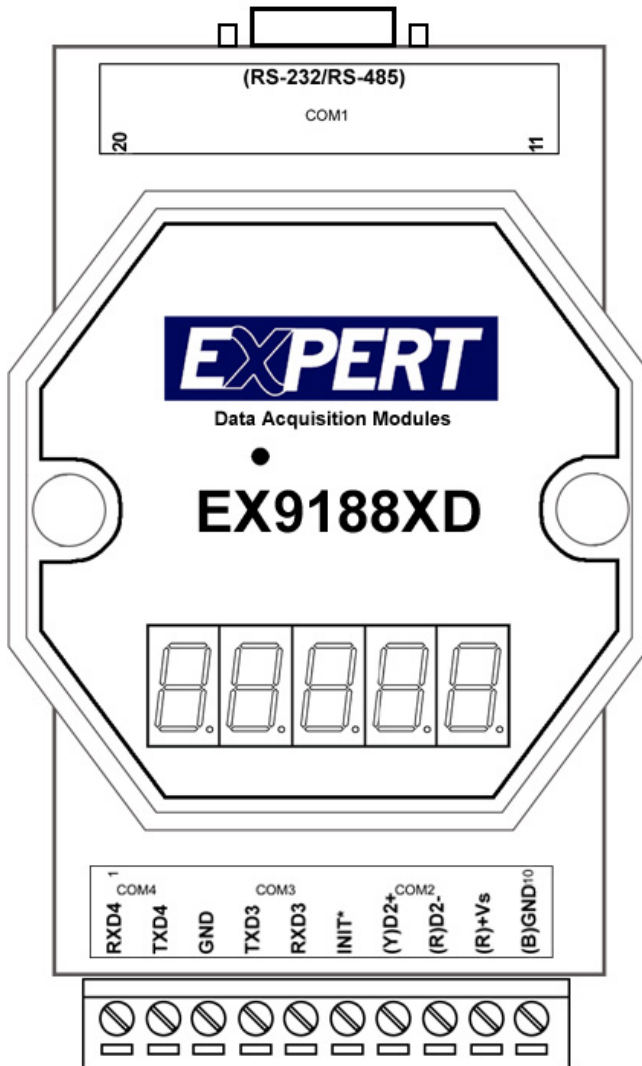
Operating Temperature: -40⁰C~80⁰C

Storage Temperature:-40⁰C~85⁰C

Humidity: 0%~90%

3. PIN Assignment

3.1 Dimension



3.2 PIN assignment of 10-pin screw terminal block

Pin	Name	Description
1	RXD4	RXD pin of COM4 (RS-232)
2	TXD4	TXD pin of COM4 (RS-232)
3	GND	GND pin of COM4 & COM3
4	TXD3	TXD pin of COM3 (RS-232)
5	RXD3	RXD pin of COM3 (RS-232)
6	INIT*	Initial pin for ROM-DISK download

7	D2+	DATA+ pin of COM2 (RS-485)
8	D2-	DATA- pin of COM2 (RS-485)
9	+VS	V+ of power supply (+10 to +30V DC unregulated)
10	GND	GND pin of power supply

Note 1: COM2=(D2+, D2-)

Note 2: COM3=(TXD3, RXD3, GND)

Note 3: COM4=(TXD4, RXD4, GND)

Note 4: COM3&COM4 share the same GND-pin

3.3 Pin assignment of COM1 connector (DB-9 Male):

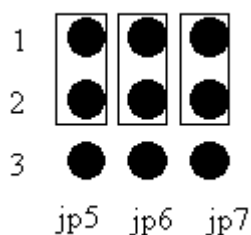
Pin	Name	Description
1	DCD	Data Carrier Detect
2	RXD	Receive Data (JP5,JP6,JP7 select RS-232)
	D1+	DATA+ of RS-485 (JP5,JP6,JP7 select RS-232)
3	TXD	Transmit Data (JP5,JP6,JP7 select RS-232)
	D1-	DATA- of RS-485 (JP5,JP6,JP7 select RS-232)
4	DTR	Data Terminal Ready
5	GND	Signal ground
6	DSR	Data Set Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	RI	Ring Indicator

Note 1: The COM1 can be used as RS-232 or RS-485 port selected by Jumper.

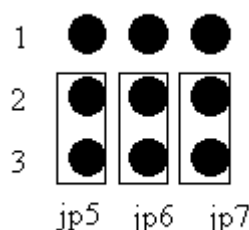
Note 2: The connector of COM1 is the same as the DB-9 RS-232 connector of PC.

3.4 Jumper

COM1 for RS232 port:



COM1 for RS485 port:



4. System Mapping:

4.1 Memory Mapping:

Device	Address Mapping
Flash ROM	From 8000:0000 to F000:FFFF
SRAM	From 0000:0000 to 3000:FFFF
COM1 BASE	0x200
COM2 BASE	0x100
COM3	FF80 to FF88
COM4	FF10 to FF18

4.2 Interrupt Mapping:

Interrupt No.	Interrupt Mapping
0	
1	Trace
2	NMI
3	Break point
4	Detected overflow exception
5	Array bounds exception
6	Unused opcode exception
7	ESC opcode exception
8	Timer 0
9	Reserved
0A	DMA-0
0B	DMA-1
0C	COM 1
0D	COM 2
0E~10	Reserved
11	COM 4
12	Timer 1
13	Timer 2
14	COM 3

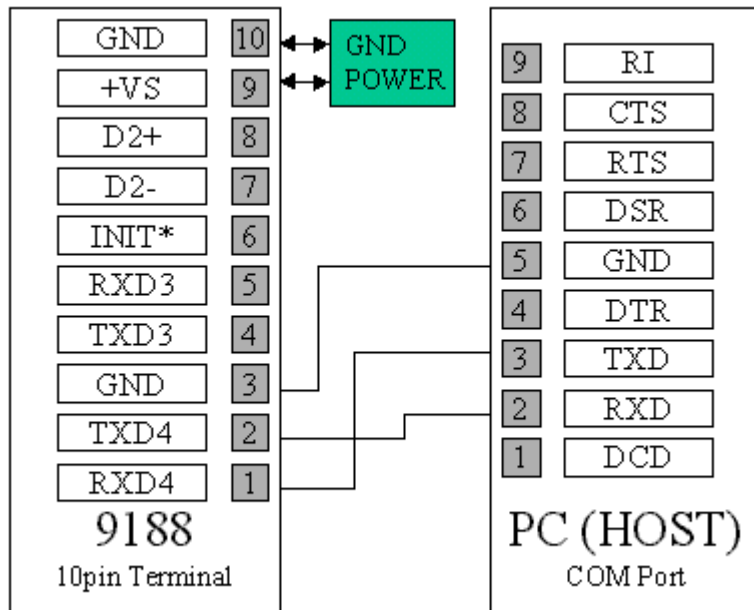
4.3 GPIO distribution

0		×
1		×
2	PCS6	E2-WP
3	PCS5	E2-SDA
4		×
5	DEN	COM1-485DIR
6	SRDY	COM2-485DIR
7		×
8		×
9		×
0A		×
0B		×
0C	DRQ0	RTC-RST
0D	DRQ1	RTC-SDA
0E		×
0F	MCS1	INIT
10		×
11	PCS1	COM1CS
12	PCS2	COM2CS
13	PCS3	LED
14	RTS0	7seg-SDA
15	CTS0	7seg-LOAD
16		×
17		×
18		×
19		×
1A	UZI	SCLK
1B		×
1C		×
1D		×
1E		×
1F		×

5. Operation Principle

5.1 Use COM4 for Debug System

EX9188 module's COM4 is a communication interface between system and host (PC side) Figure:



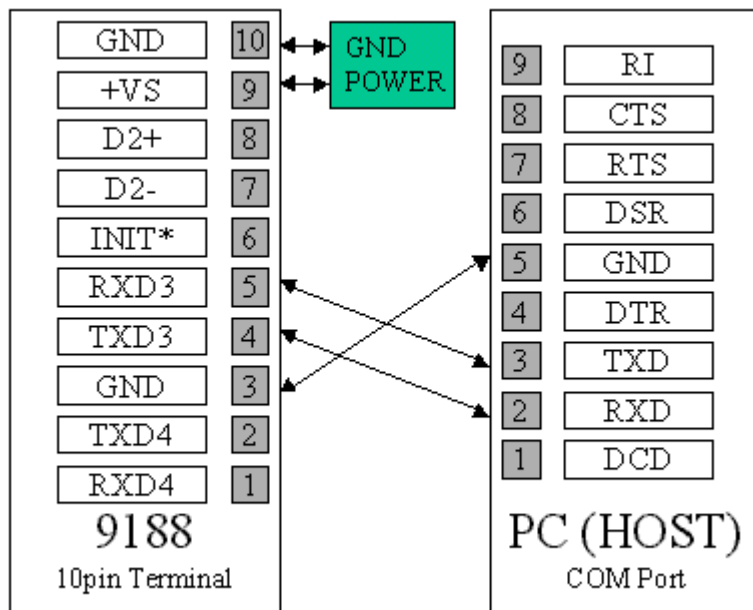
Use Hyper Terminal program under Windows 9X or 2000. After 9188 is powered on, a command prompt will be shown in the window of hyper terminal.

5.2 Use COM4 for downloading program

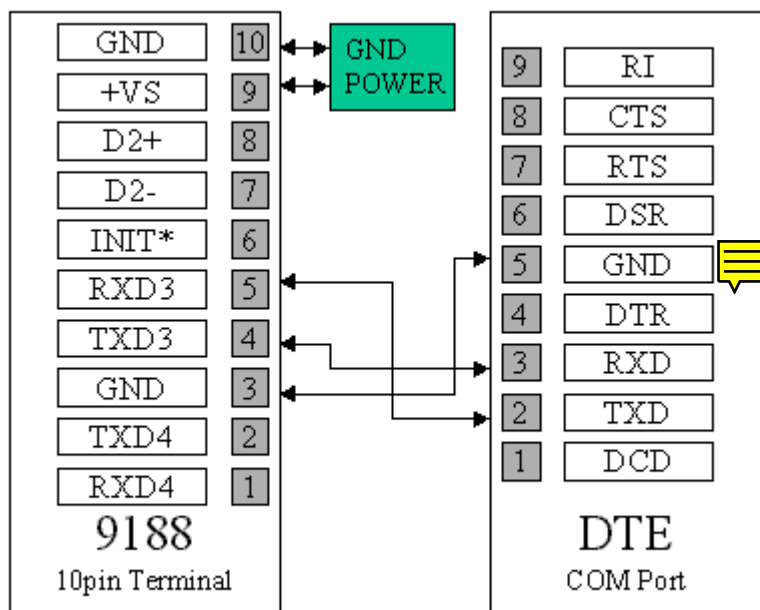
Power on the system when EX9188 module's INIT* pin is wired to ground and COM4 is connected to PC. The disk image can be downloaded from PC to flash ROM of 9188 module under Hyper-Terminal by clicking "transfer", "receive file", then choose Xmodem as the protocol and key in the file name and path. If the update is not successful, then repeat the process. If users want to debug the system from COM4, just power on the system with INIT* floating. **Note: INIT* must be disconnected from ground immediately before the updating disk image completes, otherwise the system will hang.**

5.3 Use COM 3 for RS232 port.

Connect COM3 to DCE. Wiring Diagram:



Connect COM3 to DTE. Wiring diagram:



The 9188 library functions can be used to control COM3.

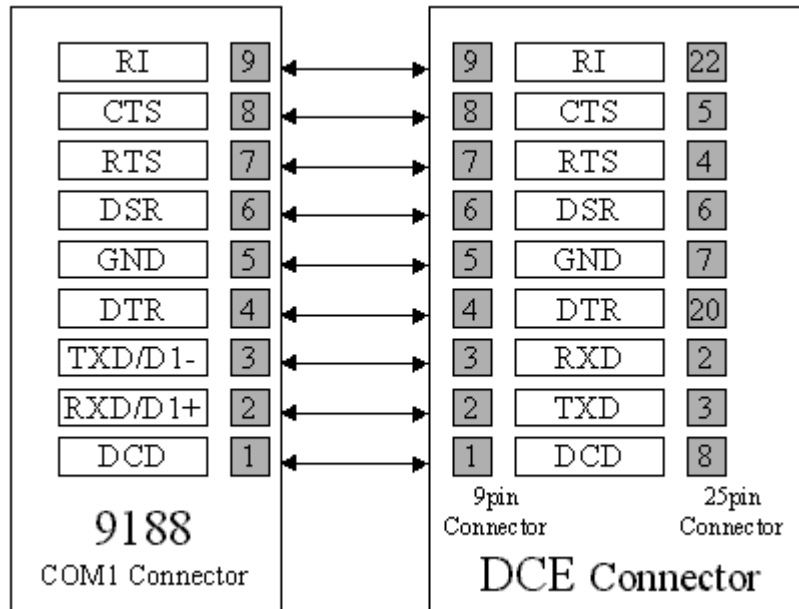
“InstallCom3{ }” is used to initialize the baud rate of the format of COM3. An interrupt driven driver will be loaded and 1KB of queue buffer is given. “ToCom(3,data)” can be used to send data to COM port. Use “IsCom(3)” to know if there is any data received and “ReadCom(3)” can get the first data from queue.

5.4 Use COM2 for RS485 port.

COM2 is a 2-wire RS-485 port. D+ and D- should connect to Data+ and Data- of RS485 network. “InstallCom2{ }” is used to initialize the baud rate of the format of COM2. An interrupt driven driver will be loaded and 1KB of queue buffer is given. In order to transmit data, “Set485DirToTransmit(2)” must be called to set the proper direction of data flow. Then “ToCom(2,data)” is used to send data. After transmission is finished, COM2 should be set to receive mode by calling “WaitTransmitOver(2)” and “Set485DirToReceive(2)”. Use “IsCom(2)” to know if there is any data received and “ReadCom(2)” can get the first data from queue.

5.5 Use COM1 for RS232 port

Refer to Sec. 3.4. connect COM1(9 pin male connector)to DCE..o



Wiring diagram:

If COM1(9pin male connector) is to connect to DTE, just swap pin 2&3.

The 9188 library functions can be used to control COM1.

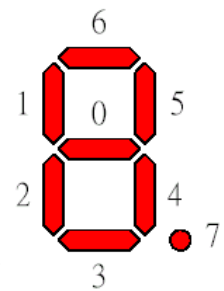
“InstallCom1 { }” is used to initialize the baud rate of the format of COM1. An interrupt driven driver will be loaded and 1KB of queue buffer is given. “ToCom(1,data)” can be used to send data to COM port. Use “IsCom(1)” to know if there is any data received and “ReadCom(1)” can get the first data from queue.

5.6 Use COM1 for RS485 port

Refer to Sec. 3.4 pin2 and 3 of jp5, 6 and 7 must be connected to have RS485 interface. D+ and D- of COM1 should connect to Data+ and Data- of RS485 network. “InstallCom1{ }” is used to initialize the baud rate of the format of COM1. An interrupt driven driver will be loaded and 1KB of queue buffer is given. “Set485DirToTransmit(1)” must be called to set the proper direction of data flow in order to transmit data. Then “ToCom(1,data)” is used to send data. After transmission is finished, COM1 should be set to receive mode by calling “WaitTransmitOver(1)” and “Set485DirToReceive(1)”. Use “IsCom(1)” to know if there is any data received and “ReadCom(1)” can get the first data from queue.

5.7 7-Segment LED & LED display control

There are 5 display units (7-segment LED) and 1 LED in 9188 module. The single LED is controlled by “LedOn()” and “LedOff()”. The 5 display units (7-segment LED) can also be controlled by software.



“Init5DigitLed()” must be called first

to do the initialization. “Show5DigitLed(pos,data)” is used to display the digits on the 7-segment LED. The argument “pos” represents the position of the 7-segment LED and “data” must be an integer between 0 to 18 and the numbers

represent ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘-’, ‘.’ respectively.

“Show5DigitLedWithDot(num,data)” is used to show a number with a floating point. And “Show5DigitLedSeg(num,data)” is used to control every segment of 7-segment LED. The argument “data” must be an integer between 0 to 7 and each number represent each segment as show in the diagram on the right side.

5.8 Use EEPROM

The EEPROM is designed to save parameters or variable data. The following data may be stored in the EEPROM:

- Module ID, configuration
- COM port configuration
- Small database

“EnableEEP()”, “DisableEEP()”, “ReadEEP()”, “ProtectEEP()” and “WriteEEP()” are used for read/write EEPROM.

5.9 Use RTC & NVRAM

DS1302 build-in RTC&NVRAM, there is a Li-battery to backup the RTC&NVRAM for 10 years. The features of RTC&NVRAM are listed as the followings:

- ROM DOS supports RTC time & date
- Mini BIOS supports RTC time & date.
- Leap year valid up to 2079, data valid 10 years
- NVRAM: 31 bytes
- Read/Write cycles without limit

“ReadNVRAM()” and “WriteNVRAM()” are used to read/write NVRAM.

5.10 Use watchdog Timer(WDT)

A watchdog timer can be activated and “inactivated” by calling “EnableWDT()” and “DisableWDT()”. After WDT is activated, the timer starts to count up to 1.6 seconds and then reset the system. “RefreshWDT()” can be used to reset the counter of the WDT.

Part II of EX9188 S/W Manual Contents:

1. Quick Start.....	17
2. System Mapping.....	17
3. MiniBios.....	18
3.1 What is BIOS:.....	18
3.2 What Feature Does the miniBios Support.....	19
4. ROM-DOS.....	20
5. Hyper Terminal.....	21
5.1 Comm.exe.....	21
5.2 Hyper Terminal.....	23
6. File Transfer.....	24
6.1 Transfer.exe.....	24
6.2 Xm.exe.....	25
7. ROMDISK.EXE.....	26
8. TOROM.EXE.....	27
9. VDISK.....	28
10. Appendix.....	29
10.1 Boot Sequence.....	29
11. 9188 Library	
11.1 Using TC or BC	32
11.2 Function Calls	35
11.3 Restrictions of Using C	42
11.4 Tips of Using 9188 Library Functions	43
11.5 Constants Defined in Library Header File.....	

1. Quick Start

1. Please refer EX9188 H/W manual for wire connection
2. Connect COM4 of EX9188 to com1(2) of PC.
3. Under Hyperterminal of PC
4. Power On EX9188.
5. HyperTerminal of PC will show the boot message of EX9188 and DOS prompt (a:\>)
6. DOS command can be executed at command prompt as dir a:\; dir b: .
7. For file transfer or downloading file, please refer to P.22.
8. For updating the contents of ROM DISK, please refer to P.23&P.24(ROMDISK.EXE & TOROM.EXE).
9. Please refer to the S/W manual to get others function & operation as RAMDISK, VDISK

2. System Mapping

EX9188 module is a multi-purpose embedded controller and operated under ROM DOS. When power-on, miniBIOS will initialize the system then ROM DOS will take over the control.

System memory mapping :

Memory Address	Memory status	H/W component
0x00000~0x3FFFF	SRAM(256K)	SRAM SPACE
0x40000~0x7FFFF	Reserve(256K)	
0x80000~0xEFFFF	Flash Memory(448K) , ROM DISK initial at 0x80000	FLASH SPACE
0xF0000~0xFBFFF	ROM DOS(48K)	
0xFC000~0xFDFFF	Reserve(8K)	
0xFE000~0xFFFFF	MiniBIOS(8K)	

3. miniBIOS

The Datalight's miniBIOS, as the name implies, is the minimum BIOS needed to run Datalight's ROM-DOS. It is not intended to replace full BIOS but to serve those embedded situations that do not require full BIOS support.

3.1 What is BIOS?

The BIOS (Basic input/output System) is the first program to gain control when power is applied to any computer system. The job of the BIOS is to bring the system up, initialize relevant hardware and RAM, as well as provide a software layer between DOS and various hardware devices. ROM-DOS requires BIOS to run.

The Datalight miniBIOS is a BIOS subset that provides support for only the BIOS features absolutely essential to the operation of ROM-DOS. The miniBIOS includes BIOS support for a remote console, timer, BIOS extensions, and hardware equipment identification. The miniBIOS does not provide support for floppy or hard disks, printers, the standard PC keyboard, or monitors.

3.2 What Features Does the miniBIOS Support

The following table shows the calls made by the ROM-DOS kernel to the BIOS. ROM-DOS makes no hardware assumptions, but instead, performs all interactions with the hardware through BIOS call

Minimum BIOS Calls used in a Diskless System

Name	Interrupt	Sub-function
Coprocessor Esc instruction	07H	reserved
Timer 0 tick	08H	Reserved (hardware)
Keyboard Input	09H	(stubbed for APM)
Serial receive char	0BH	Reserved (hardware)
Video TTY Output	10H	0EH
Get equipment list	11H	
Get memory size in K	12H	
Disk I/O	13H	reserved
Extended Memory Support	15H	reserved
Keyboard Support	16H	00H,01H,02H
Keyboard, push Scan code into Buffer	16H	05H
Boot failure message	18H	
System DOS-boot	19H	
Time of Day	1AH	00H,01H
Timer Tick	1CH	

4. ROM-DOS

EX9188 Module is equipped with DataLight ROM-DOS. ROM-DOS is a disk operating system that can be loaded in Read Only Memory (ROM) and can run entirely from within ROM and also from a hard or floppy disk, such as in a desktop system. ROM-DOS is functionally equivalent to other brands of DOS and can run programs that are executable under a standard DOS (which executes from RAM). With ROM-DOS, the executable program resides on a disk or is placed in ROM along with ROM-DOS.

5. Hyper Terminal

After H/W installation (refer to H/W manual), Hyper Terminal can be used to control the EX9188 module. COMM.exe can be executed under DOS

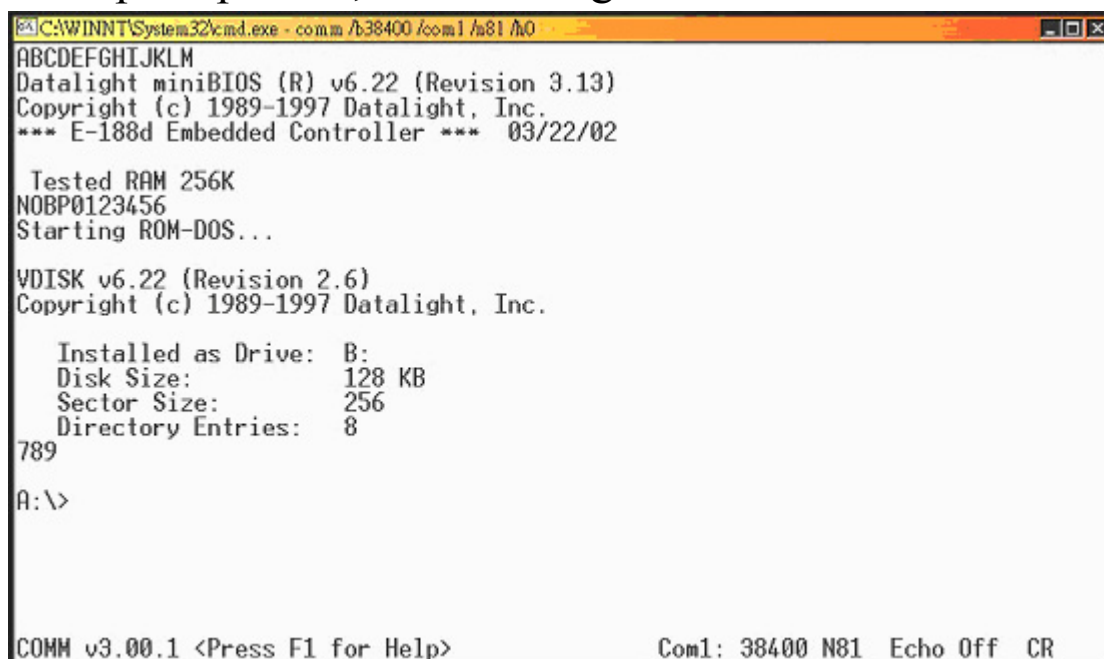
5.1 Comm.exe

COMM.EXE can be executed under DOS prompt, the parameter as follows:

/B# Initial Baud (300,1200..9600..115200)
/Com# COM port (1..4)
/Irq# IRQ number (3-15)
/N81, /E71 Initial port parameters
/H# handshaking 0=none, 1=XON, 2=RTS

<e.g> EX9188 is connected to PC COM1, baud rate=57600, 8 data bit, without parity, with 1 stop bit :
comm.exe /com1 /b57600 /n81 /h0

Power on the EX9188 module, HyperTerminal shows the DOS prompt A:\>, as following:



```
C:\WINNT\System32\cmd.exe - comm /b38400 /com1 /n81 /h0
ABCDEFGHJKLM
Datalight miniBIOS (R) v6.22 (Revision 3.13)
Copyright (c) 1989-1997 Datalight, Inc.
*** E-188d Embedded Controller *** 03/22/02

Tested RAM 256K
NOBP0123456
Starting ROM-DOS...

VDISK v6.22 (Revision 2.6)
Copyright (c) 1989-1997 Datalight, Inc.

Installed as Drive: B:
Disk Size: 128 KB
Sector Size: 256
Directory Entries: 8
789
A:\>

COMM v3.00.1 <Press F1 for Help> Com1: 38400 N81 Echo Off CR
```

Other hotkey as follows :

F1	Help screen
Alt-P	Toggle N81/E71
Alt-B	Toggle Baud Rate
Alt-K	Send break signal
Alt-C	Clear Screen
Alt-T	Toggle CR/LF Translation
Alt-D	Dial Modem
Alt-X	Exit COMM
Alt-E	Toggle Echo (Duplex)
PgUp	Upload (Ascii/Xmodem/Zmodem)
Alt-H	Hangup Modem
PgDn	Download (Ascii/Xmodem/Zmodem)

Easy Way (creating a shortcut):

Under Windows, user can create short-cut on desk top:
Press the right button of mouse at the blank space of desktop

Desktop→New→Shortcut→Cmdline:COMM.EXE→

Next→Select a name for the shortcut→select an icon for the shortcut→finish.

Select the shortcut and press right button of mouse→

Properties→Program→Cmdline:COMM.EXE/com1/b57600/n81→Misc→Background:always suspend(delete√)→OK

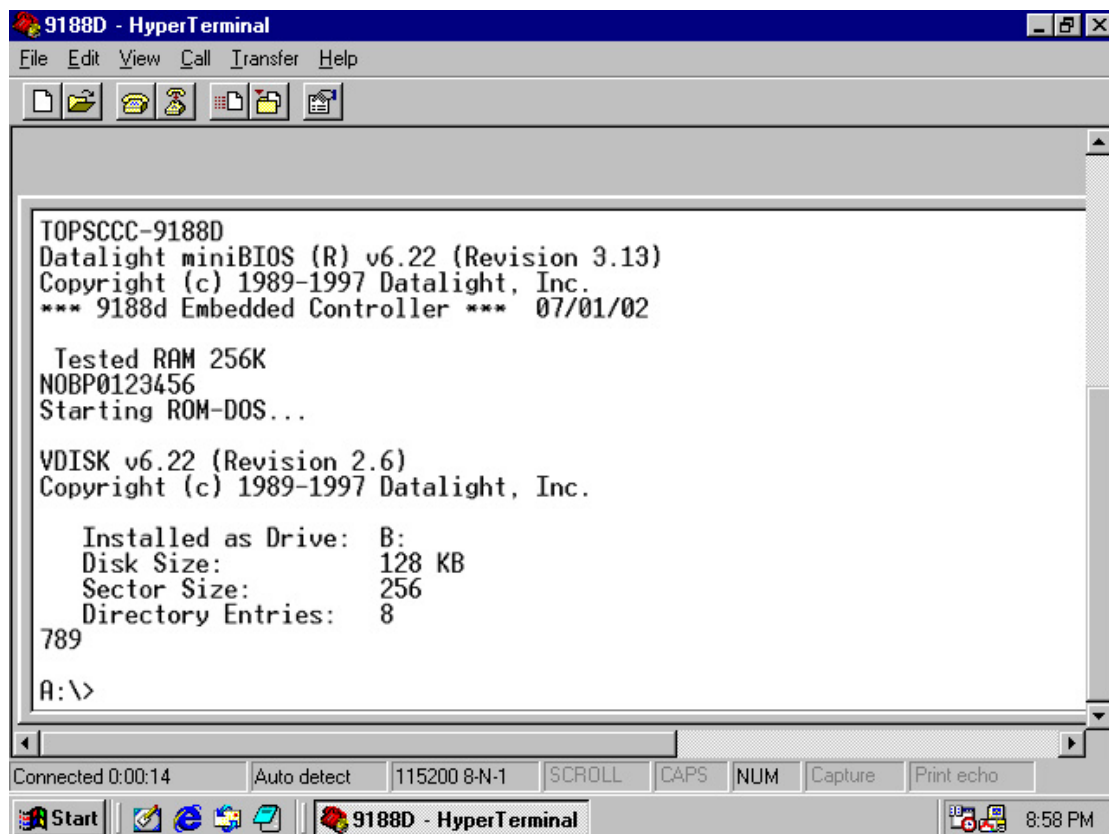
Just click on the shortcut next time if COMM.exe is required.

5.2 HyperTerminal

Start→Program→Accessories→Communication→HyperTerminal

Select the correct COM port, baud rate(57600),Databit(8) parity(none),stopbit(1) and choose “none” for flow control. Then select File→Properties(modify if needed)→OK.

After power-on the system, HyperTerminal will show the DOS prompt as the following figure:



6. File Transfer

Please use “transfer.exe” when using “comm.exe” to communicate with 9188.

Please use “xm.exe” when using HyperTerminal to communicate with 9188.

comm.exe ↔ transfer.exe

HyperTerminal ↔ xm.exe

6.1 Transfer.exe

1. Download a file from 9188:
 - a. Type “transfer.exe /s /bc <filename>” and <enter> at DOS prompt
--- <filename> file which you want to send.
 - b. Press PageDown
 - c. Enter X --- Xmodem
 - d. Enter Filename (including path) --- File name in PC
 - e. “RRRRRRRR...” will prompt on terminal –It means download is in process.
2. Upload a file to 9188:
 - a. Type “transfer.exe /r /bc <filename>” and <enter> at DOS prompt
 - b. Press PageUp
 - c. Enter X --- Xmodem
 - d. Enter Filename (including path) --- File for uploading
 - e. “TTTTTTTT...” will prompt on terminal –It means upload is in process.

6.2 Xm.exe

1. Download a file from 9188:
 - a. Type “xm.exe /s <filename>” and <enter> at DOS prompt -- <filename> is the file you want

to send.

- b. Under HyperTerminal Select: Transfer
→Receive file→Use Xmodem as receiving
protocol→Place receive file in chosen folder
(point to the file which is going to be
downloaded)→OK
- c. Wait until download file transmission
completed.

2. Upload a file to 9188:

- a. Type “xm.exe /r <filename>” and <enter> at
DOS prompt.
- b. Under Hyper Terminal select: Transfer→Send
file→Protocol (Xmodem)→file name(the file
which will be sent)→OK. (within 60Sec)
- c. Wait until upload file transmission completed

7. ROMDISK.EXE

ROMDISK.EXE is a utility of ROM-DOS. It is for making a ROM-DOS image file. The steps to create a ROM-DISK image file are listed below:

1. Create a sub-directory called ROM.
2. Copy all the files needed for Romdisk.
3. run “ROMDISK.EXE” (romdisk.exe will put all files in the directory ROM into a file named Rom.img)
4. Download file to EX9188 via TOROM.EXE (see “Using torom.exe”)
5. Reboot EX9188, you will see disk A in EX9188 contains all the files of directory ROM.

8. TOROM.EXE

TOROM.EXE is used to transfer the ROMDISK image file to Flash ROM of EX9188 module and the contents of Flash ROM of ROM disk will then be updated. The steps are listed below:

1. Connect EX9188 module to PC. Use HyperTerminal as terminal.
2. run TOROM.EXE
3. Under HyperTerminal, select: Transfer→Send file→Protocal(Xmodem)→File name→OK.(Use Xmodem protocal & key in Rom image file name then execute file transfer function).
4. When ROM image file transmission is completed, reboot EX9188 module.

Note: When execute the TOROM.EXE on EX9188 module, the process of step3 must be finished under 60 seconds otherwise timeout will occur. If timeout is the case, INIT* pin will have to be used to download the program to Flash ROM. Please refer to sec. 5.2 of H/W manual.

5. You will see a updated ROM disk.

9. VDISK

Vdisk is a device driver which can create a virtual disk or RAM DISK from system memory. The contents of virtual disk will be lost when power-off. Vdisk.sys is a system file which can be configured in config.sys, the format is shown below:

Device=vdisk.sys [size[secs[dirs]]] /E

Size is for RAM DISK size, default is 64KB; Secs is for sector size, default is 512 bytes, valid values are 128, 256, 512.; Dirs is the number of sub directories or files in root of RAM DISK, default is 64.

Example: In config.sys:

device=vidsk.sys

device=vdisk.sys 32 128 16

Set 64kb RAM DISK B: ; 512 byte SECTOR (when ROM DISK A: exist)

Set 32kB RAM DISK C: ; 128byte SECTOR, 16 files in root directory.

10. Appendix:

10.1 Boot Sequence

After power on reset, mini BIOS will initialize the H/W and ROM-DOS will enable the operation system. In Boot Sequence will show out the message as below table: (If you can not boot success then reboot again, if still can not reboot that please contact with your supplier)

Boot Sequence

Type	Boot Diagnostic	Description
MiniBios	A	UMCS and PIO set ok.
MiniBios	B	LCS set ok.
MiniBios	C	Disable DMA0.
MiniBios	D	Disable DMA1.
MiniBios	E	Disable hardware interrupt.
MiniBios	F	Disable Timer and WDT.
MiniBios	G	Do Nothing.
MiniBios	H	Set interrupt mask register.
MiniBios	I	Initialize BIOS stack, stack RAM has not been tested yet.
MiniBios	J	Power On Ram Test.
MiniBios	K	Setup local environment for c.
MiniBios	L	Interrupt vector table setup.
MiniBios	M	Initialize the supported BIOS interrupts.
MiniBios	CopyRight Message	Show CopyRight Message
MiniBios	N	RAM size test.
MiniBios	O	Init System Environment, and then scan for bios extentions, and jump to.

ROMDOS	B	BIOS extension has gained the control.
MiniBios	P	Int 19, jump to ROM-DOS(tm) entry point.
ROMDOS	0	Interrupts are enabled, ROM-DOS has control, and the first instructions have been executed.
ROMDOS	1	Startup code (decompress) has completed. The startup code copies the DOS data into RAM. The DOS code is also copied for a disk boot of ROM-DOS or a ROM boot with the copy to RAM feature enabled. To make room for data compression, the startup code relocates the ROM-DOS code to the top of memory. The data is decompressed to its full size in lower memory. The stack is also set up and uninitialized data is zeroed. Boot failures at this point are typically due to insufficient RAM to accommodate the code and full data size, or an incomplete ROM-DOS image in ROM.
ROMDOS	2	Minimum DOS structures allocated. The memory pool is set up and the default structures are at the top of RAM. The DOS interrupts are also set up.
ROMDOS	3	Interrupts have been initialized. Boot failures at this point may be due to another process using an interrupt that ROM-DOS has set up for its own use. An example of this is a watchdog timer that traps Int 21h.
ROMDOS	4	Built-in devices have been initialized. BIOS interrupt calls are made during the initialization (Int 13h for disk drive support, Int 10h for video). Failures

		may be due to incomplete BIOS interrupt support or a failure to find a disk of any type in the system.
ROMDOS	5	Root PSP is now in existence.
ROMDOS	6	Default drive has been determined.
ROMDOS	7	The first pass of CONFIG.SYS processing is complete. All CONFIG.SYS statements except the INSTALL= are processed (device drivers are loaded). Standard handles such as PRN, AUX, and CON are opened.
ROMDOS	8	All internal structures allocated. TSR programs listed in CONFIG.SYS INSTALL= statements are loaded. Failure at this point may indicate a faulty TSR program.
ROMDOS	9	ROM-DOS has been loaded high (if DOS=HIGH). The DOS buffers have been created and copied to the HMA area if sufficient space.
ROMDOS	Dos prompt or application start	The standard handles have been re-opened and the final program has been called via Int 21h. This program is typically COMMAMD.COM or an application program. Failure to reach the DOS prompt after boot diagnostic 9 is usually caused by not finding the program (not on the disk), a corrupted file, a command interpreter from a different operating system, or a faulty application program. Failure may also be due to insufficient RAM to run the command

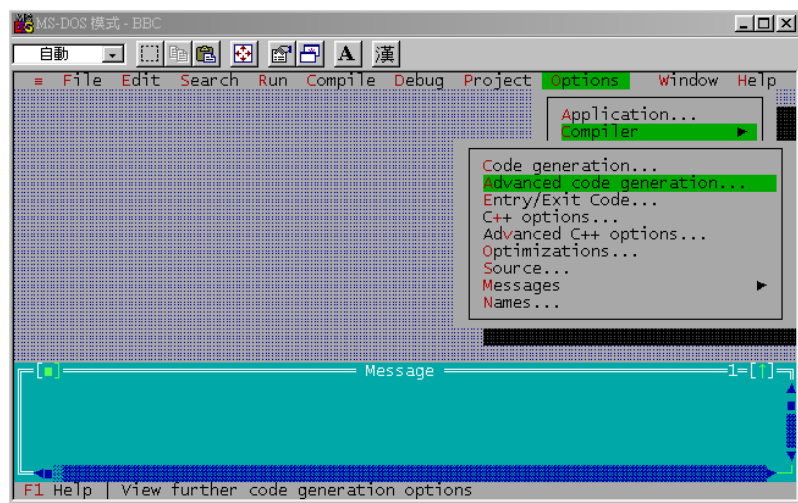
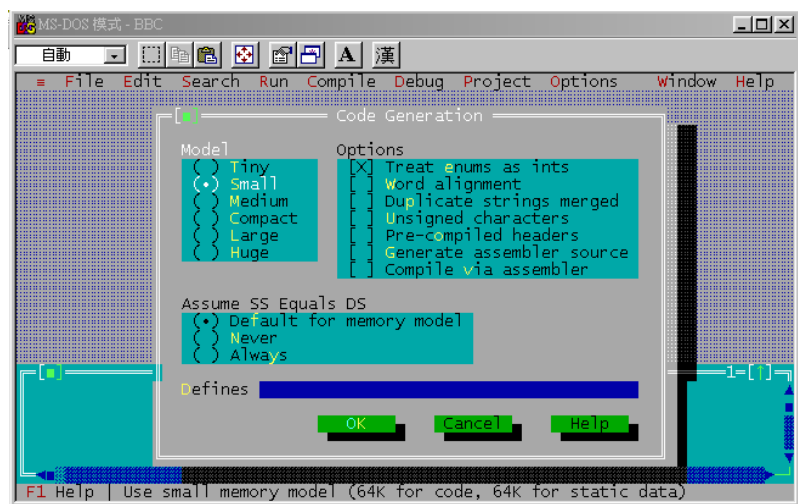
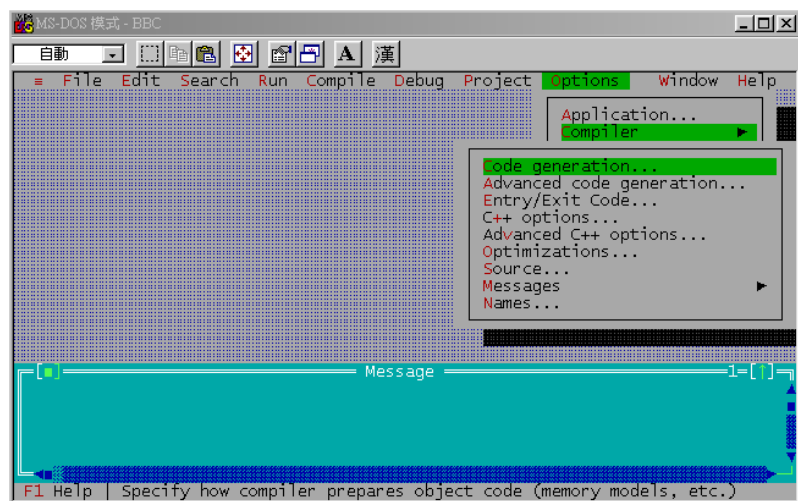
11. 9188 Library

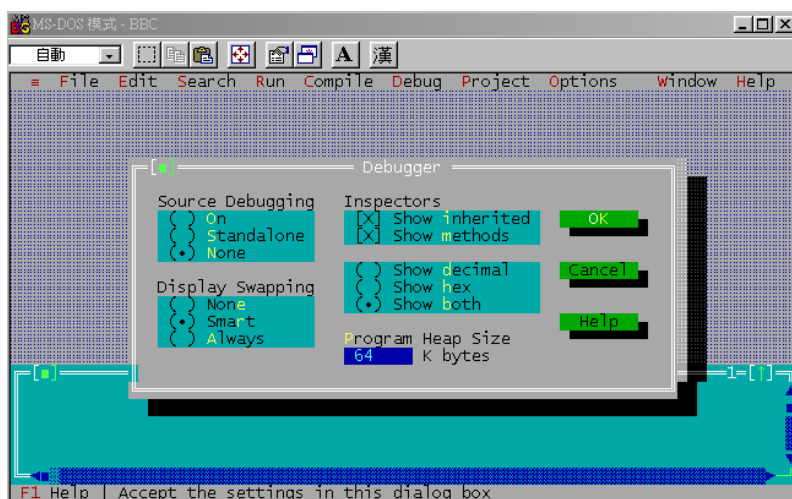
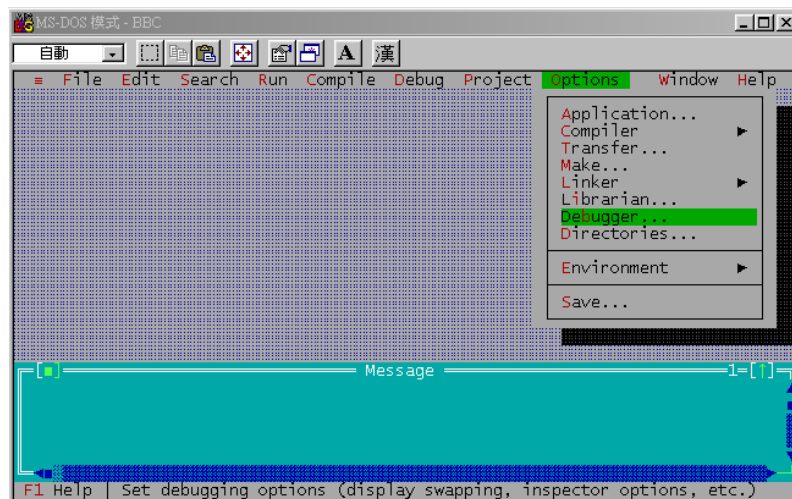
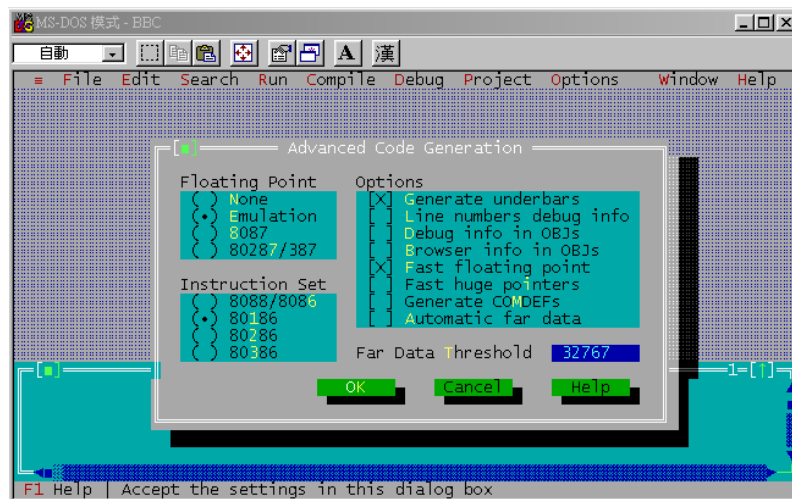
9188 library provides all the necessary functions for controlling COM ports, LED, data storage, watchdog timer, timers and etc. By applying the 9188 library functions, the programmers are able to have easy access to EX9188 and develop all kinds of applications for controlling EX9188.

11.1 Using TC or BC

When using TC or BC on the integrated development environment (IDE), the user should create a new project and add the necessary files (e.g. xxx.c and 9188d.lib) to it.

Be sure to choose the correct compiler model and CPU type on the IDE. The program should be developed in small model and 80186 is the right choice for EX9188 module. When using floating point, the user should choose “emulation” and turn off the debug information. The diagrams below show the proper settings for Borland C++ Ver.3.1.





11.2 Function Calls

Install COM port driver

Function Name	Description
InstallCom	Install driver for one of 4 COM ports
InstallCom1	Install driver for COM1
InstallCom2	Install driver for COM2
InstallCom3	Install driver for COM3
InstallCom4	Install driver for COM4

Uninstall COM port driver

Function Name	Description
RestoreCom	Uninstall driver for one of 4 COM ports
RestoreCom1	Uninstall driver for COM1
RestoreCom2	Uninstall driver for COM2
RestoreCom3	Uninstall driver for COM3
RestoreCom4	Uninstall driver for COM4

Check if there is data in the input buffer of COM port

Function Name	Description
IsCom	Check for one of 4 COM ports
IsCom1	Check for COM1
IsCom2	Check for COM2
IsCom3	Check for COM3
IsCom4	Check for COM4

Read one byte of data from input buffer of COM port

Function Name	Description
ReadCom	Read from one of 4 COM ports
ReadCom1	Read from COM1
ReadCom2	Read from COM2
ReadCom3	Read from COM3
ReadCom4	Read from COM4

Send one byte of data to COM port

Function Name	Description
ToCom	Send data to one of 4 COM ports
ToCom1	Send data to COM1
ToCom2	Send data to COM2
ToCom3	Send data to COM3
ToCom4	Send data to COM4

Clear input buffer in COM ports

Function Name	Description
ClearCom	Clear input buffer In one of 4 COM ports
ClearCom1	Clear input buffer in COM1
ClearCom2	Clear input buffer in COM2
ClearCom3	Clear input buffer in COM3
ClearCom4	Clear input buffer in COM4

Check if the transmission is finished

Function Name	Description
WaitTransmitOver	Check for one of 4 COM ports
WaitTransmitOver1	Check for COM1
WaitTransmitOver2	Check for COM2
WaitTransmitOver3	Check for COM3
WaitTransmitOver4	Check for COM4

Set the direction of RS485

Function Name	Description
Set485DirToTransmit	Set to transmission mode
Set485DirToReceive	Set to receive mode

Check if output buffer is empty

Fncion Name	Description
IsCom3OutBufEmpt	Check for COM3
IsCom4OutBufEmpty	Check for COM4

Get the number of data in input buffer

Function Name	Description
DataSizeInCom	For one of 4 COM ports
DataSizeInCom1	For COM1
DataSizeInCom2	For COM2
DataSizeInCom3	For COM3
DataSizeInCom4	For COM4

Switch on/off LED

Function Name	Description
LedOff	Led Off
LedOn	Led On

For 5-digit LED

Function Name	Description
Init5DigitLed	Initialize 5-digit LED
Show5DigitLed	Show a number in one of 5-digit LED
Show5DigitLedWithDot	Show a number with a dot
Show5DigitLedSeg	Show a segment in one of 5-digit LED
Set5DigitLedTestMode	Set to test mode (all segments are on)
Set5DigitLedIntensity	Set intensity
Disable5DigitLed	Disable 5-digit LED
Enable5DigitLed	Enable 5-digit LED

For NVRAM

Function Name	Description
ReadNVRAM	Read one byte of data from NVRAM
WriteNVRAM	Write one byte of data to NVRAM

For EEPROM

Fncion Name	Description
WriteEEP	Write one byte of data to EEPROM
ReadEEP	Read one byte of data from EEPROM
EnableEEP	Enable EEPROM for writing
ProtectEEP	Set to write-protect

For flash memory

Function Name	Description
---------------	-------------

FlashReadId	Read the flash memory type
FlashWrite	Write one byte of data to flash
FlashErase	Erase one block of data
FlashRead	Read one byte of data

For watchdog timer

Function Name	Description
EnableWDT	Enable watchdog timer
RefreshWDT	Refresh watchdog timer
DisableWDT	Disable watchdog timer
IsResetByWatchDogTimer	Check if 9188 is reset by WDT

For timers

Function Name	Description
TimerOpen	Install timer driver
TimerClose	Uninstall timer driver
TimerResetValue	Reset the value of timer ticks
TimerReadValue	Read the value of timer ticks
StopWatchReset	Reset stopwatch
StopWatchStart	Start stopwatch
StopWatchStop	Stop stopwatch
StopWatchPause	Pause stopwatch
StopWatchContinue	Continue stopwatch
StopWatchReadValue	Read the value of stopwatch timer
CountDownTimerStart	Start countdown timer
CountDownTimerReadValue	Read the value of timer
InstallUserTimer	Install the user's timer function
InstallUserTimer1C	Install the user's timer function on INT1C
DelayTimeMs	Software delay in unit of 1ms
DelayMs	Hardware delay in unit of 1ms
Delay	Hardware delay in unit of 1ms

Delay_1	Hardware delay in unit of 0.1ms
---------	---------------------------------

Standard IO

Function Name	Description
getch4	Instead of getch
kbhit4	Instead pf kbhit
ungetch4	Instead of ungetch
putch4	Instead of putchar

MISC functions

Function Name	Description
GetLibVersion	Get the version of library
ReadInitPin	Get the status of INIT pin
_MK_FP	Make a far pointer

11.3 Restrictions of using C

Due to the hardware structure of 9188, there are some limitations of using some C library functions. The following lists are the functions which are not allowed to use in 9188.

1. Graphic mode functions.
2. Text mode functions which write data directly to screen memory, such as cprintf.
3. Protected mode functions, such as EMS or XMS.
4. LPT port functions, such as biosprint or _bios_printer.
5. COM port functions, such as bioscom or _bios_serialcom.
6. 8087 (math coprocessor) related functions. If floating

point functions are required, please use emulation mode.

7. Win32 functions.

11.4 Tips of using 9188 library functions

NVRAM

There is a NVRAM containing 31 bytes in 9188 module. “ReadNVRAM()” is used for reading one byte of data from NVRAM and “WriteNVRAM()” is used for writing one byte of data to NVRAM. The program below illustrates the use of these functions.

```
Int data=0x55,data2;  
WriteNVRAM(0,data);  
Data2=ReadNVRAM(0); //data2=data=0x55
```

If 2 bytes of data needs to be written to NVRAM, the program is shown below:

```
Int data=0xAA55;  
Char *dataptr=(char*)&data;  
  
WriteNVRAM(0,*dataptr);  
WriteNVRAM(1,*dataptr+1);  
  
Dataptr=(char*)&data2;  
*dataptr=ReadNVRAM(0);  
*(dataptr+1)=ReadNVRAM(1); //data2=0xAA55
```

EEPROM

EEPROM containing 2048 KB is included in 9188 module. There are 8 blocks and each block has 256 bytes. Before writing any data to EEPROM, “EnableEEP()” must be called first to unprotect EEPROM. The program below illustrates the use of these functions.

```
Int data=0x55, data2;  
EnableEEP();  
WriteEEP(1,10,data);  
ProtectEEP();  
Data2=ReadEEP(1,10); //data2=data=0x55
```

Flash Memory

512KB of flash memory is included in 9188 module. MiniBIOS, ROM-DOS and ROM-DISK (drive A:) are stored in flash memory. MiniBIOS and ROM-DOS are resided at segment 0xf000. ROM-DISK occupies the segment 0x8000. The rest of the space should be safe to use. The characteristics of flash memory is each bit of data can be written from 1 to 0, but can not be written for 0 to 1. If the original data is 0xff, any value which writes to the same location is valid. If the original data is 0x00, any value which writes to the same location does not change any thing. The only way to change data for 0 to 1 is to erase it. "EraseFlash()" can be called to erase a block of flash memory.

```
Int data=0xAA55, data2;
```

```
char *dataptr;
```

```
int *dataptr2;
```

```
dataptr=(char*)&data;
```

```
FlashWrite(0xd000,0x1234,*dataptr++);
```

```
FlashWrite(0xd000,0x1235,*dataptr);
```

```
// read data from flash memory
```

```
dataptr=(char*)&data2;
```

```
*dataptr=FlashRead(0xd000,0x1234);
```

```
*(dataptr+1)=FlashRead(0xd000,0x1235);
```

```
//another way of read data
```

```
dataptr2=(int far*)_MK_FP(0xd000,0x1234);
```

```
data2=*dataptr2;
```

5-digit LED display

A 5-digit 7-segment LED display is included in 9188 module. The most left one is the first one and the most right one is the fifth one. “Init5DigitLed” must be called first before using other 5-digit LED related functions.

```
Init5DigitLed();  
Show5DigitLed(1,9);  
Show5DigitLed(2,1);  
Show5DigitLed(3,8);  
Show5DigitLed(4,8);  
Show5DigitLed(5,13); // ‘d’ is 13
```

It will show “9188d” on it.

Watchdog timer

The default value of watchdog timer is 1.6 seconds. After “EnableWDT” is called, the timer will start to count up. When the timer reaches 1.6 seconds, 9188 module will be reset by WDT. “RefreshWDT” is used to reset the timer. If “RefreshWDT” is called before the timer counts up to 1.6 seconds, the 9188 will not be reset. “DisableWDT” is used to disable the WDT.

```
EnableWDT();  
While(!quit){  
    RefreshWDT();  
    //do something  
}  
DisableWDT();
```

COM ports

There are 4 communication ports.

COM1 can be selected as RS232 or RS485 port.

COM2 is a RS485 port

COM3 and COM4 are RS232 ports

“InstallCom” or “InstallCom1/2/3/4” must be called before other COM port related functions are used.

“RestoreCom” or “RestoreCom1/2/3/4” must be called to uninstall the driver before exiting the program. “IsCOM” is used to check if any data coming from COM port and then use “ReadCom” to get the data from input buffer or use “ToCOM” to send the data to COM port.

```
Int port=4;
```

```
Int quit=0,data;
```

```
InstallCom(port,57600L,8,0,1);
```

```
While(!quit){
```

```
    if(IsCom(port)){
```

```
        Data=ReadCom(port);
```

```
        ToCom(data);
```

```
        If(data=='q') quit=1;
```

```
        //do something
```

```
    }
```

```
}
```

```
RestoreCom(port);
```

When using RS485 (COM1 or 2), the direction of data flow of RS485 bus must be controlled by software. The default direction of RS485 is in receive mode. When 9188 wants to transmit data, transmission mode must be set (call “Set485DirToTransmit”). After transmitting data, “Set485DirToReceive” must be called to be set back to receive mode. “WaitTransmitOver” has to be used to make sure all the data has been sent out before setting the system to receive mode.

```
int i,port=2;  
char data[5]=”$01M\r”;  
InstallCom(port,9600,8,0,1);  
Set485DirToTransmit(port);  
for(i=0;i<5;i++) ToCom(port,data[i]);  
while(!WaitTransmitOver(port));  
Set485DirToReceive(port);  
RestoreCom(port);
```


11.5 Constants Defined in Library Header File

```
#define    IN_BUF_SIZE        1024

#define    NoError            0

#define    InitPinIsOpen      0

#define    InitPinIsNotOpen  1

#define    QueueIsEmpty       0

#define    QueueIsNotEmpty    1

#define    PortError          -1

#define    DataError          -2

#define    ParityError        -3

#define    StopError          -4

#define    TimeOut            -5

#define    QueueEmpty         -6

#define    QueueOverflow      -7

#define    PosError           -8

#define    AddrError          -9

#define    BlockError         -10

#define    WriteError         -11

#define    SegmentError       -12
```

```
#define BaudRateError -13
#define CheckSumError -14
#define ChannelError -15
#define TimeIsUp 1
```