ADAM-4500

PC-based Communication Controller

User's Manual

Copyright Notice

Acknowledgments

2nd Edition, July 2004

Table of Contents

Chapter 1 General Information

1.1 Introduction

The ADAM-4500 is a fully functional standalone controller, designed for industrial automation and control, enclosed in a small package. It provides an ideal means of producing an IBM PC compatible hardware platform.

Emulating Open PC environments

The module is much like a compact computer, and includes an 80188 CPU, 256 KB Flash ROM, 256 KB SRAM, COM1, COM2, and a program download port. Its built-in ROM-DOS is an MS-DOS equivalent operating system, which provides all of the basic functions of MS-DOS except for BIOS. A user can run standard PC software and application programs, written in high level languages such as C or C++, within the ROM-DOS environment. Moreover, the module also provides free ROM memory for application downloading and free RAM memory for application operation.

Built-in RS-232/RS-485 Communication Ports

The ADAM-4500 has two communication ports to let the controller easily communicate with the other devices in your application. The COM1 port can be configured as an RS-232 or an RS-485 communication interface via the jumper setting. The COM2 port is dedicated as an RS-485 port. This unique design makes the controller suitable for use in a variety of applications.

Built-in Real-time Clock and Watchdog Timer

The controller also includes a real-time clock and a watchdog timer function. The real-time clock ensures time recording while events occur. The watchdog timer is designed to automatically reset the microprocessor when the system fails. This feature greatly reduces the level of maintenance required and makes the ADAM-4500 ideal for use in applications that require a high level of system stability.

1.2 Features

- Built-in boot ROM-DOS to run PC programs
- Free ROM/RAM memory for user's applications
- 2-wire, multi-drop RS-485 networking
- Communication speed up to 115.2 Kbps
- RS-232/RS-485 modes (jumper selectable)
- Automatic data flow control in RS-485 mode
- Built-in real-time clock and watchdog timer
- Easy mounting on a DIN-rail or panel
- Program download cable and utility included

1.3 Specifications

System

- CPU: 80188-40
- Flash ROM: 256 KB (170 KB free memory for users)
- Operating system: Boot ROM-DOS
- SRAM: 256 KB (234 KB free memory for users)
- Timer BIOS: Yes
- Real-time clock: Yes
- Watchdog timer: Yes
- COM1: RS-232/RS-485
- COM2: RS-485
- Program download port (RS-232): Tx, Rx, GND

RS-232 Interface
- Signals: TxD, RxD, RTS, CTS, DTR, DSR, DCD, RI, GND
- Mode: Asynchronous full duplex, point to point
- Connector: DB-9 pin
- Transmission speed: up to 115.2 Kbps
- Max transmission distance: 50 feet (15.2 m)

RS-485 Interface
- Signals: DATA+, DATA-
- Mode: Half-duplex, multi-drop
- Connector: Plug-in terminal block
- Transmission speed: up to 115.2 Kbps
- Max transmission distance: 1220 m (4000 feet)

Power
- Unregulated +10 to +30 V_{DC}
- Protected against power reversal
- Power consumption: 2.0 W

Mechanical
- Case: ABS with captive mounting hardware
- Plug-in screw terminal block:
  Accepts 0.5 mm^2 to 2.5 mm^2, 1 - #12 or 2 - #14 to #22 AWG

Environment
- Operating temperature: -10 to 70°C (14 to 158°F)
- Storage temperature: -25 to 70°C (-13 to 158°F)
- Humidity: 5 to 95 %, non-condensing

1.4 System Diagram

ADAM-4500

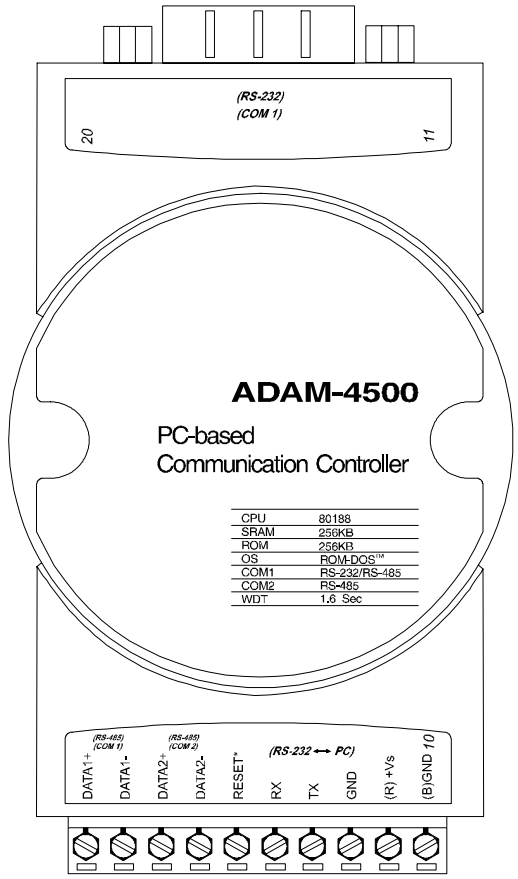


Figure 1-1: ADAM-4500 Diagram

Function Block Diagram

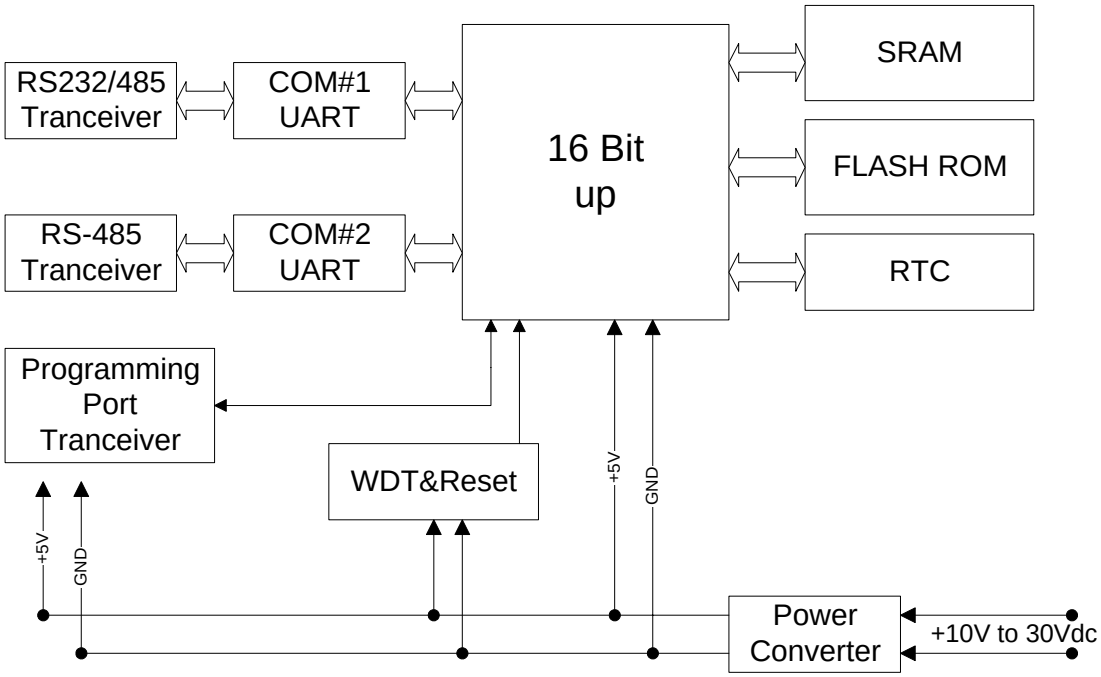


Figure 1-2: ADAM-4500 Function Block Diagram

Chapter 2 Installation Guidelines

This chapter provides guidelines to set up and install the ADAM-4500 communication controller. A simplified hookup scheme is provided that lets you configure a successful system off-line before implementing it in your application.

2.1 System Requirements to Setup ADAM-4500

The following list gives an overview of what is needed to setup, install and configure an ADAM-4500 system.

- ADAM-4500 module
- An IBM PC/AT compatible computer that can download programs with an RS-232 port.
- Power supply for the ADAM-4500 (+10 to + 30 V_{DC})
- ADAM-4500 download utility software
- Download cable (RS-232 interface)

Host computer

Any computer that is IBM PC/AT compatible which can run and write programs in an MS-DOS environment, and provides an RS-232 communication port to allow users to download programs into the ADAM-4500, can function as the host computer.

Power supply

For ease of use in industrial environments, the ADAM-4500 will accept industry standard +24 V_{DC} unregulated power. The ADAM-4500 was designed to operate using any power supply voltage between +10 and +30 V_{DC}. Power ripples must be limited to 5 V peak to peak while the voltage in all cases must be maintained between +10 and +30 V_{DC}.

Utility software

A Driver CD-ROM containing menu-driven utility software is provided with ADAM-4500 to help users to download programs to the ADAM-4500. The ADAM-4500 utility software is executed in an MS-DOS or compatible environment.

Download cable

A customized download cable is included. A user can connect it between the COM port of a host computer and the download port of the ADAM-4500 for downloading programs.

2.2 Steps to Successfully Setup the System

Step 1: Review the requirements

Before you power on the ADAM-4500, make sure you have a host computer, a power supply, and the utility software and download cable.

Step 2: Wiring the power cables and download cable

Connect the power cable between the power supply and the ADAM-4500. Make sure the power source is between +10 and +30 V_{DC}.

We advise use of the following standard colors (as indicated on the ADAM-4500) for the power cables:

+Vs	(R)	Red
GND	(B)	Black

Connect the download cable between the host computer and the ADAM-4500. A customized

download cable is provided with the ADAM-4500, which includes a DB-9 pin connector for the connection to the host computer, and three color-coded wires for connection to the ADAM-4500 terminal block. The following figure shows how to connect the ADAM-4500 to the host computer.

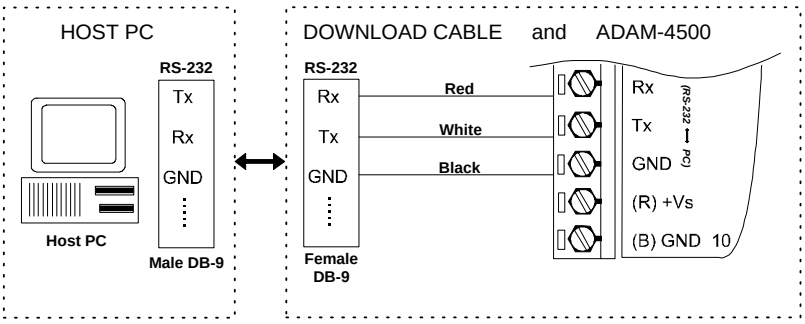


Figure 2-1: Download Cable Connections

Step 3: Run utility software in host computer

Together with the ADAM-4500 you will find a Driver CD-ROM containing a utility ADAM4500.EXE file. This file is a menu-driven software utility provided for downloading user's programs. It uses a screen simulating operation of the ADAM-4500 communication controller. When the file is executed, the main screen appears, as shown in figure 2.2.

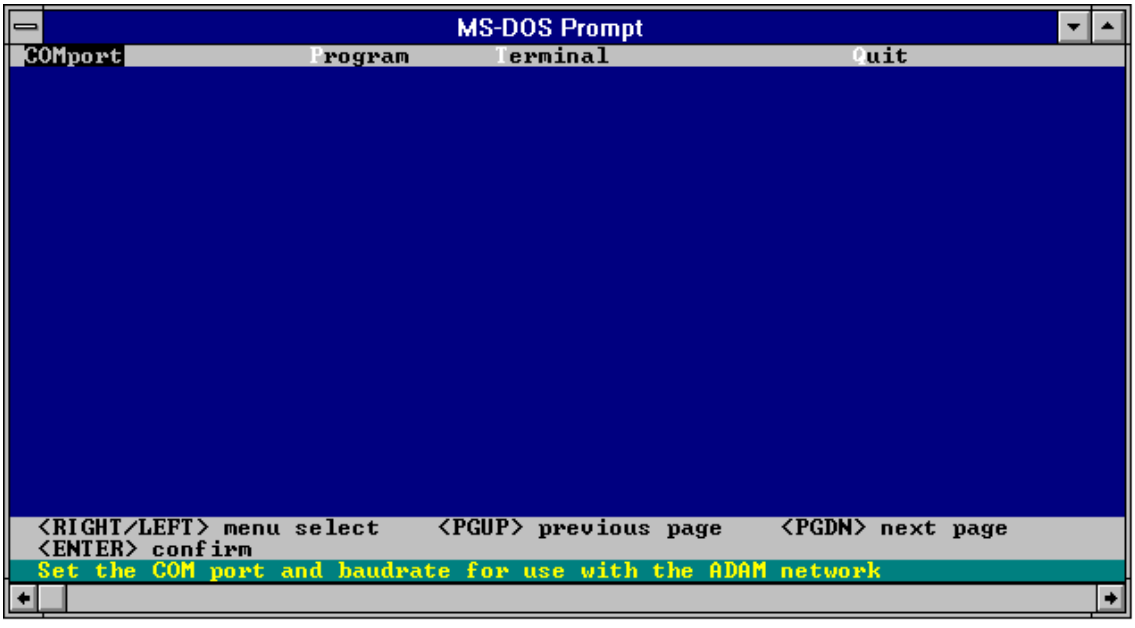


Figure 2-2: Main Screen

Setup COM port

First, highlight the "COMport" option on the top bar and press <enter>. The status field (shown below) will appear. Second, highlight the COM port you used to connect the ADAM-4500 to the PC, then press <enter>. The Baudrate is set to a default value of 57600 bps and cannot be

changed. The screen is as shown in figure 2.3.

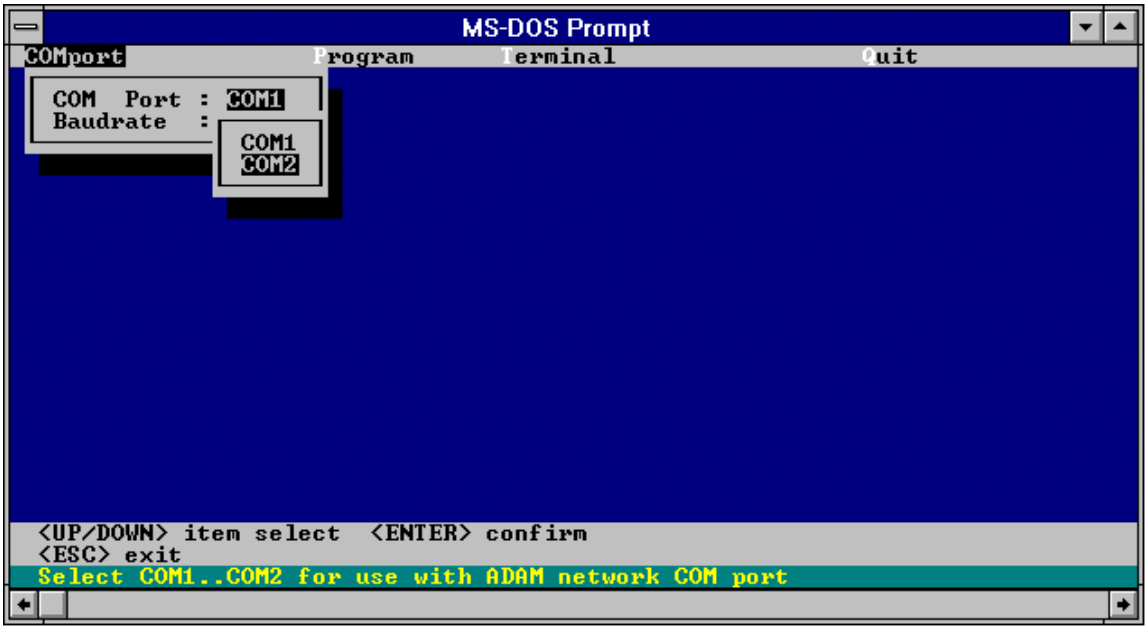


Figure 2-3: Select Communication Port

Step 4: Power on ADAM-4500

Highlight the "Terminal" option, then press <enter>. Power on the ADAM-4500. After 5 seconds, the screen shown in figure 2.4 will appear. The ADAM-4500 system is successfully started up.

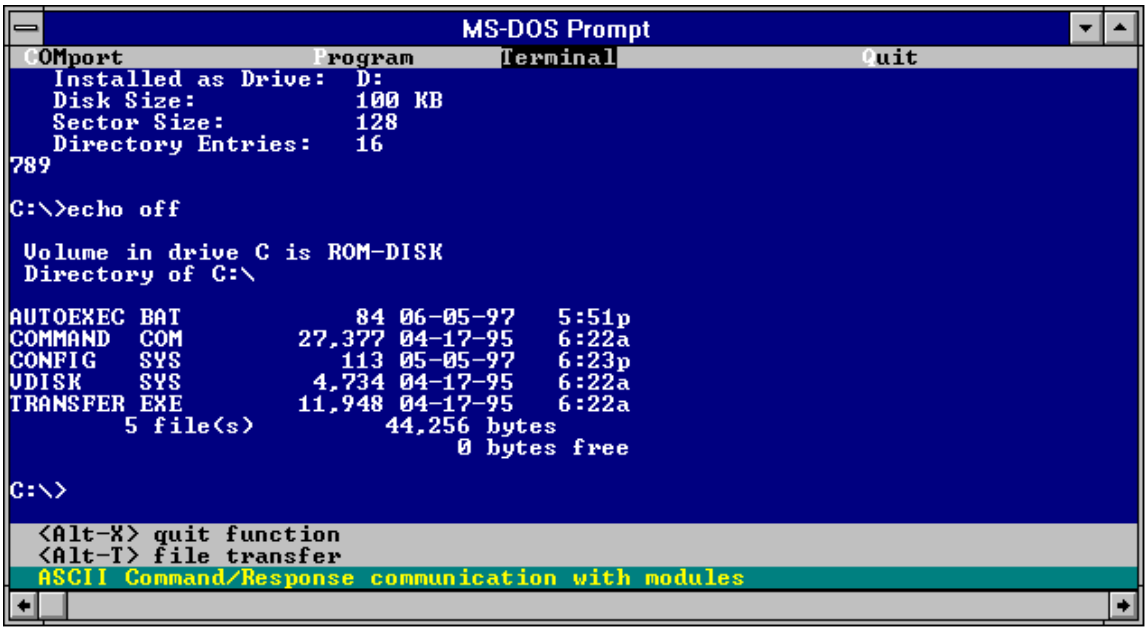


Figure 2-4: Emulating Screen of ADAM-4500

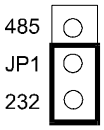
2.3 Jumper Setting

We designed the ADAM-4500 with ease-of-use in mind. It has three jumper settings. The

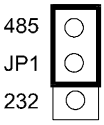
following sections explain how to configure the module. You may want to refer to the figure below for help in identifying card components.

COM1 Port Setting (JP1)

Jumper JP1 lets you configure the COM1(3F8) port as an RS-232 or as an RS-485 interface for different applications. Jumper settings are shown below:



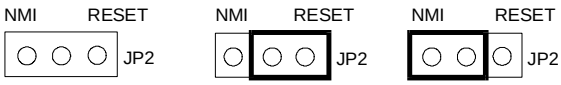
RS-232 Mode
(Default)



RS-485 Mode

Watchdog Timer Setting (JP2)

Jumper JP2 lets you configure the watchdog timer in disable mode, reset mode or NMI (Non-maskable interrupt) mode.



Disable Watchdog Timer

Reset Mode

(Default)

Non-maskable Interrupt (NMI)

Reset Function Setting (JP3)

Jumper JP3 enables or disables the function of the reset pin, as shown below:

2.4 Communication Wiring

The ADAM-4500 offers two serial ports: COM1(3F8) in RS-232 or RS-485, and COM2 (2F8) in RS-485. COM1 can be configured as an RS-232 or as an RS-485 interface via the jumper setting. This provides flexibility for communications between the controller and other devices in your application.

RS-232 Connection

The ADAM-4500 has a DB-9 pin connector as its RS-232 port connector. Since the connection for an RS-232 interface is not standardized, different devices implement the RS-232 connection in different ways. If you are having problems with a serial device, be sure to check the pin assignments for the connector. The following table shows the pin assignments for the ADAM-4500's RS-232 port.

Pin No.	Description
1	DCD
2	RxD
3	TxD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

Table 2-1: Pin Assignments of RS-232 Port

RS-485 Connection

The RS-485 standard supports half-duplex communication. This means that just two wires are needed to both transmit and receive data. Handshaking signals (such as RTS, Request To Send) are normally used to control the direction of the data flow. A special I/O circuit in the ADAM-4500 automatically senses the direction of the data flow and switches the transmission direction. No handshaking signals are necessary. This RS-485 control is completely transparent to the user.

We recommend that shielded twisted-pair cables complying with the EIA RS-485 standard be used in the network to reduce interference. Only one set of twisted-pair cables is required to transmit both Data and RTS signals. We advise use of the following standard colors for the communication cables.

DATA +	Yellow
DATA -	Green

Chapter 3 Programming and Downloading

This chapter explains how to program applications and download programs into the ADAM-4500 controller. Additionally, it points out limitations and concerns of which you should be aware.

3.1 Programming

The operating system of ADAM-4500 is ROM-DOS, an MS-DOS equivalent system. It allows users to run application programs written in assembly language as well as high level languages such as C or C++. However, there are limitations when running application programs in the ADAM-4500. In order to build successful applications, you should keep the following limitations and concerns in mind.

Mini BIOS Functions

The ADAM-4500 provides only two serial communication ports for connecting peripherals, so the mini BIOS of ADAM-4500 only provides 10 function calls. Since the user's program can not use other BIOS function calls, the ADAM-4500 may not work as intended. Additionally, certain language compilers such as QBASIC directly call BIOS functions that are not executable in ADAM-4500. The ADAM-4500 mini BIOS function calls are listed in the following table.

Function	Subfunction	Task
07h		186 or greater co-processor esc instruct
10h	0eh	TTY Clear output
11h		Get equipment
12h		Get memory size
15h	87h	Extended memory read
	88h	Extended memory size
	c0h	PS/2 or AT style A20 Gate table
16h	0	Read TTY char
	1	Get TTY status
	2	Get TTY flags
18h		Print "Failed to BOOT ROM-DOS" message
19h		Reboot system
1ah	0	Get tick count
	1	Set tick count
	2	Get real time clock
	3	Set real time clock
	4	Get date
	5	Set date
1ch		Timer tick

Converting Program Codes

The ADAM-4500 has an 80188 CPU. Therefore, programs downloaded into its flash ROM must first be converted into 80186 or 80188 compatible code, and the floating point operation must be set to emulation mode. For example, if you were to develop your application program in Borland C, you would compile the program as indicated in the screen below.

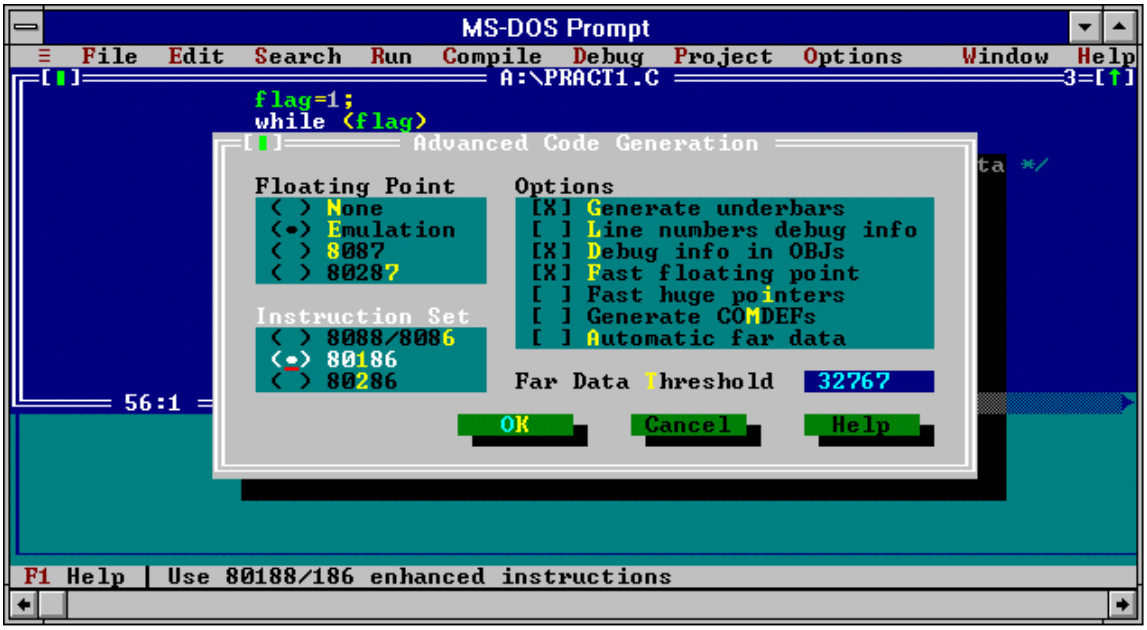


Figure 3-1: Converting Program Codes

Other Limitations

1. The ADAM-4500 does not support the standard PC function "8253". Therefore, the C language function call "delay ( )" cannot be used in ADAM-4500 applications.

2. Certain critical files are always kept in flash ROM, such as the operating system files, BIOS, and monitoring files. Aside from the storage space needed for these critical files, the ADAM-4500 has an additional 170 KB of free ROM space for downloading user applications. An additional free 234 KB of SRAM is provided for operation of applications.

Programming the Watchdog Timer

The ADAM-4500 is equipped with a watchdog timer function that resets the CPU or generates an interrupt if processing comes to a standstill for any reason. This feature increases system reliability in industrial standalone and unmanned environments.

If you decide to use the watchdog timer, you must write a function call to enable it. When the watchdog timer is enabled, it must be cleared by the application program at intervals of less than 1.6 seconds. If it is not cleared at the required time intervals, it will activate and reset the CPU, or generate an NMI (Non-maskable interrupt). You can use a function call in your application program to clear the watchdog timer. At the end of your program, you need an additional function call to disable the watchdog timer.

The following program shows how you might program the watchdog timer in C programs:

Example:

```
main ()
{
      wdt_enable();             /* enable ADAM-4500 WDT function */
      {                         /* user's function block */
            .
            .
            wdt_clear();        /* clear WDT timer */
            .
            .
```

```
        }
        wdt_disable();           /* disable ADAM-4500 WDT function */
}
```

Interrupt Types

Three types of interrupts may occur in the ADAM-4500. The following table shows the types of interrupts.

Interrupt Name	Interrupt Type
Non-Maskable Interrupt (NMI)	02h
COM1 Interrupt	0Ch
COM2 Interrupt	0Eh

Memory Mapping

The following table shows the memory mapping of the ADAM-4500 controller.

0xF8000 -- 0xFFFFF	Monitor program
0xF6C00 -- 0xF7FFF	Mini BIOS
0xCC000	Start of Application ROM Disk (about 171 K)
0xC0000	Start of ROM-DOS (about 48 K)
0x40000 -- 0xBFFFF	No Use
0x00400 -- 0x3FFFF	SRAM area
0x00000 -- 0x003FF	System area
0x003F8 -- 0x003FF	COM1
0x002F8 -- 0x002FF	COM2
0x00070 -- 0x00071	Real time clock

3.2 Downloading and Transferring

This section explains how to download application programs from a PC into the ADAM-4500 flash ROM and how to transfer files from a PC into ADAM-4500's SRAM.

Install Utility Software on Host PC

A Driver CD-ROM containing the following files and directories is included with each ADAM-4500.

. ALLFILE <DIR>
. EXAMPLE <DIR>
. LIBRARY <DIR>
. MANUAL <DIR>
. A1-DIS.HEX
. ADAM_A1.HEX
. ADAM_DEM.HEX
. ADAM4500.EXE
. ADAMMINI.BAT
. ADAMMINI.HEX
. DEMO-DIS.HEX
. HEXCAT.EXE
. ROMDISK.EXE
. ROM-DOS.HEX

Copy all the files and directories on the Driver CD-ROM to the host computer hard drive.

Preparing the ALLFILE directory

Applications programs are downloaded from a host-PC to the flash ROM of the ADAM-4500 using the ADAM-4500 utility software. The ADAM-4500 utility software is first installed on a host-PC. The directory ALLFILE will be included among the contents copied from the Driver CD-ROM to the host-PC hard drive. The user must then load into ALLFILE the following required files: The application program intended for installation in ADAM-4500; COMMAND.COM; AUTOEXEC.BAT; and CONFIG.SYS. The user should make certain that AUTOEXEC.BAT contains the name of the user's application program so that the application will automatically begin executing whenever the ADAM-4500 is powered on. When downloading to the ADAM-4500's flash ROM, the utility software first clears all non-permanent files from the flash ROM, then installs all the files contained in directory ALLFILE into the flash ROM. It is therefore critical that all the required files be available in directory ALLFILE when the utility software tries to install ALLFILE's contents on the ADAM-4500's flash ROM.

Downloading into flash ROM (ADAM-4500's C-drive)

With the ADAM-4500 utility software and the directory ALLFILE, loaded with its proper contents, installed on the host-PC, you can execute the utility software. After the utility software has begun executing, select the COMport of the host PC that has been connected to ADAM-4500. Then select "Program" from the bar menu and press <enter> to begin downloading. The screen shown in figure 3-2 will appear.

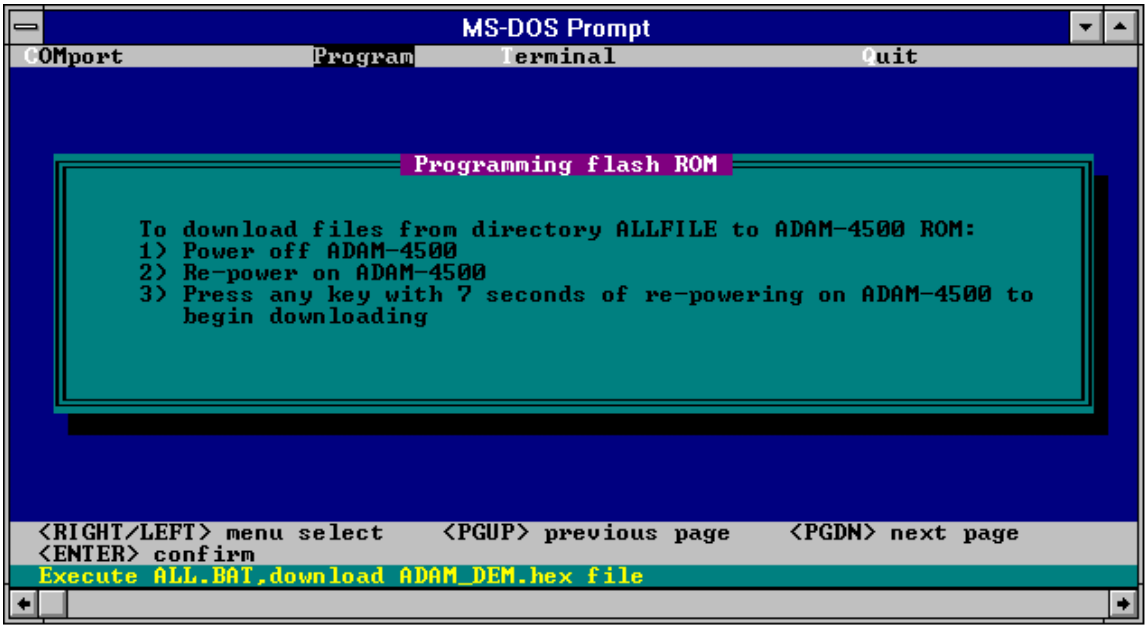


Figure 3-2: Program downloading

Follow the instructions shown on the screen. Power off the ADAM-4500 and then re-power on. Then press any key within 7 seconds to burn the files contained in ALLFILE into the ADAM-4500's flash ROM. After the files are successfully burned into the flash ROM, the screen in Figure 3-3 will appear. Power off and power on the ADAM-4500 once again. The ADAM-4500 controller will automatically execute the application program.

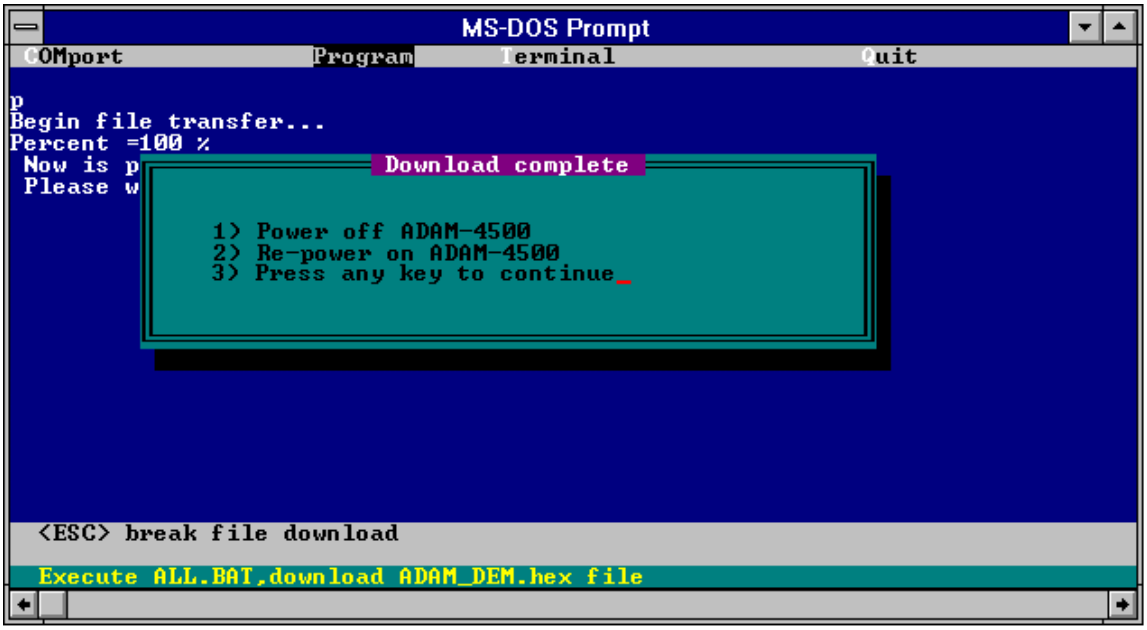


Figure 3-3: Download Complete

Transferring files to SRAM (ADAM-4500's D-drive)

The ADAM-4500 provides 234 KB of free SRAM for use in program operation, and as working memory space in the event you want to test your system control logic or simulate system performance before downloading the execution code to the flash ROM. You can transfer files from a host-PC to the ADAM-4500's SRAM (D drive). Execute the utility software, select terminal mode, and press Alt-T. File transfer will begin and the screen shown in figure 3-4 will appear.

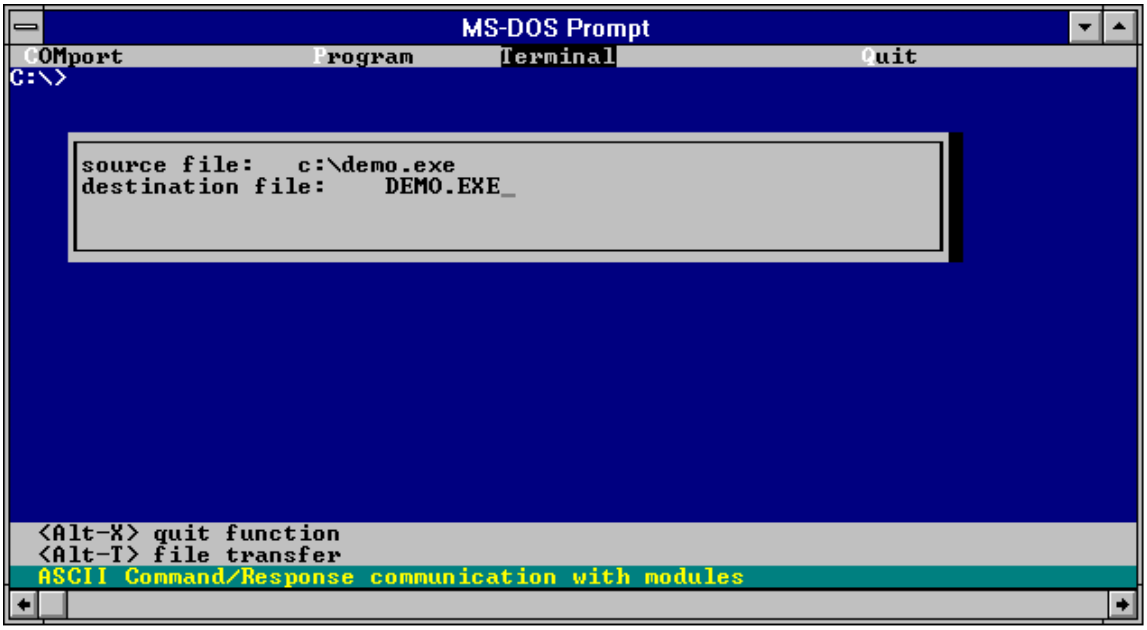


Figure 3-4: File Transferring

Key in the specific directory and file names you want to transfer. Press <enter> to complete the file transfer. You can check the files in D drive.
We recommend that users test new applications on their host PC before installing them in the

ADAM-4500. However, the performance of the host PC may differ greatly from that of the ADAM-4500. Therefore, users may find it desirable to install an application that has been tested on the host PC into ADAM-4500's D drive for testing prior to burning it into the flash ROM. It takes less time to alter and reinstall a program into the SRAM than into the flash ROM, so using the SRAM during debugging on the ADAM-4500 may be more efficient.

3.3 Steps to Building a Successful Application

Step 1: Write and simulate control logic on a PC

Connect a host-PC to the I/O device network you want to control. Write the control logic for your application on the host-PC and execute the application on the PC to verify that it works properly.

Step 2: Connect the cables

Replace the host-PC in the control network with the ADAM-4500 and reconnect the proper power and communications cable in the network to the ADAM-4500. Connect the download cable between the host-PC and the ADAM-4500.

Step 3: Convert and download codes to flash ROM

Within the host computer, convert the application program into 80186 or 80188 compatible code. Create an AUTOEXEC.BAT file for the application program and write the application program's name in the AUTOEXEC.BAT file. Also create COMMAND.COM and CONFIG.SYS files for the application program. Load the ADAM-4500 utility software into the host-PC. Load the converted application program and the files AUTOEXEC.BAT, COMMAND.COM, and CONFIG.SYS into directory ALLFILE. Execute the utility software to download the contents of ALLFILE into the ADAM-4500 flash ROM.

Step 4: Power on ADAM-4500 to complete the application

After all the files in directory ALLFILE have been completely transferred to the flash ROM, re-power on the ADAM-4500.

Chapter 4 Function Library

The ADAM-4500 is packaged with a Driver CD-ROM that contains a directory called LIBRARY. This directory contains a number of function calls that enable a user to efficiently write applications for ADAM-4500. The library supports both Turbo C 2.0 and Microsoft C 6.0 version programming languages.

The "LIBRARY" directory contains the following two sub-directories:

. MSC60	Library for Microsoft C 6.0 version
. TC20	Library for Turbo C 2.0 version

Function Library for Microsoft C
The MSC60 directory contains the following files:

ADAM4500.H	Declaration file
4500L.LIB	Library file for compiler large mode
4500S.LIB	Library file for compiler small mode
ADAM5510.H	Declaration file
5510L.LIB	Library file for compiler large mode
5510S.LIB	Library file for compiler small mode

Function Library for Turbo C
The TC20 directory contains the following files:

ADAM4500.H	Declaration file
4500L.LIB	Library file for compiling large mode
4500S.LIB	Library file for compiling small mode
5510DRV.H	Declaration file for RTC, Backup RAM and Timer functions
UTILITYL.LIB	Library file for compiling large mode
UTILITYM.LIB	Library file for compiling medium mode
UTILITYC.LIB	Library file for compiling compact mode
UTILITYS.LIB	Library file for compiling small mode

The function calls included in the directory LIBRARY are described in the following pages.

<ADAM4500.H>

comm_init

Syntax:
int comm_init(int buf_size)

Function description:
Initializes the communication port and interrupt routine before other function calls use the communication port.

Parameter	**Description**
int buf_size	Sets the buffer size of every communication port for storage of received data. The unit of size is bytes.

Return:
Returns a 1 if command succeeds. Returns a 0 if it fails.

comm_exit

Syntax:
int comm_exit()

Function Description:
If a user calls the **comm_init** function, the user must call this function to release the communication port before the user's program terminates.

Return:
Returns a 1 if command succeeds. Return a 0 if it fails.

comm_open

Syntax:
int comm_open(unsigned char port, unsigned long baud, int parity, int data, int stop, int cmd_type)

Function Description:
To open a communication port for user to send and receive data.

Parameter	**Description**
unsigned char port	Specifies communication port. 1: COM1 port 2: COM2 port
unsigned long baud	Baud rate setting. The eight allowable baud rates are: 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200. Sized in bps (bits per second).
int parity	Parity setting. 0: no parity, 1: odd, 2: even
int data	Data bits setting. There are four allowable data lengths: 8, 7,

6, and 5. Sized in bit units.

int stop	Stop bits setting 1, 2 (stop bit = 1 is for data bits = 5,6,7, or 8) (stop bit = 2 is for data bits = 6,7, or 8) For details, refer to RS-232 chip set data book
int cmd_type	Received data format setting. 0: single byte mode. The data format is a single byte character. User can use **comm_get_rec_datas** to receive single byte data. 1: command mode. The data format is a string. User can use **comm_get_rec_str** to receive string data.

Return:
Returns a 1 if command succeeds. Returns a 0 if it fails.

comm_send

Syntax:
int comm_send(unsigned char c, unsigned char port)

Function Description:
To send a character to a specified communication port.

Parameter	**Description**
unsigned char c	Represents character to be sent.
unsigned char port	Specifies communication port to which character is sent (COM1 or COM2) 1: COM1 port 2: COM2 port

Return:
Returns a 1 if command succeeds. Returns a 0 if it fails.

comm_get_rec_datas

Syntax:
int comm_get_rec_datas(unsigned *length, unsigned char **data, unsigned char port)

Function Description :
The function is called to receive a single byte of data. The function call returns a success flag if the communication port receives any data. The function call returns a failed flag if the buffer of the communication port is empty. The length of the data packet received (in bytes) is stored in parameter "length". The contents of the data packet received are stored in parameter "data".

Parameter	**Description**
unsigned *length	Returns the length of the received data
unsigned char **data	Returns the received data contents. User must first allocate a data buffer (Use **comm_int** function call). Before exiting program, user must free this buffer (Use **comm_exit** function

call).

unsigned char port	Specifies communication port (COM1 or COM2) 1: COM1 port 2: COM2 port

Return:
Returns a 1 if command succeeds. Returns a 0 if it fails.

comm_get_rec_str

Syntax:
int comm_get_rec_str(unsigned char **data, unsigned char port)

Function Description:
This function call is employed to received string data. The function call returns a success flag if the communication port receives string data (terminal character is 0x0d) . The function call returns a fail flag if the buffer of the communication port is empty.

Parameter	**Description**
unsigned char **data	Returns the received string contents. User must first allocate a data buffer (Use **comm_int** function call). Before exiting program, user must free this buffer (Use **comm_exit** function call).
unsigned char port	Specifies communication port (COM1 or COM2) 1: COM1 port 2: COM2 port

Return:
Returns a 1 if command succeeds. Returns a 0 if it fails.

led_init

Syntax:
void led_init()

Function Description:
This function must be called to initialize the ADAM-4500's LED before a user's program can control the LED.

led

Syntax:
void led(int type)

Function Description:
LED ON/OFF control

Parameter	Description
int type	0: LED OFF 1: LED ON

wdt_enable

Syntax:
wdt_enable()

Function Description:
This function enables the watchdog timer function. After a user calls this function, the user must call the wdt_clear() function to refresh the watchdog timer; otherwise the CPU resets, or a non-maskable interrupt is generated.

wdt_clear

Syntax:
wdt_clear()

Function Description:
This function refreshes the watchdog timer, thereby avoiding the resetting of the CPU or the generation of a non-maskable interrupt.

wdt_disable

Syntax:
wdt_disable()

Function Description:
This function disables the watchdog timer function.

<5510DRV.H>

ADAMdelay

Syntax:
void ADAMdelay(unsigned short msec)

Description:
Delays program operation by a specified number of milliseconds.

Parameter	Description
msec	From 0 to 65535.

Return value:
None.

Example:
```
void main(void)
{
/* codes placed here by user */
ADAMdelay(1000); /* delay 1 sec. */
/* codes placed here by user */
}
```

Remarks:
None.

Get_NVRAM_Size

Syntax:
unsigned char Get_NVRAM_Size(void)

Description:
Gets the battery backup RAM size. The unit is sectors, each sector is 4 KB in size. Maximum size is 60 KB theoretically.

Parameter	**Description**
None.	

Return value:

sector	Number of sectors NV RAM size is set to, from 1 to 15.

Example:
```
void main()
{
unsigned char sector;
sector = Get_NVRAM_Size();
}
```

Remarks:
None.

GetRTCtime

Syntax:
unsigned char GetRTCtime(unsigned char Time)

Description:
Reads Real-Time Clock chip timer. A user can activate a program on the date desired.

Parameter	**Description**	
Time	RTC_sec	the second
	RTC_min	the minute
	RTC_hour	the hour
	RTC_day	the day
	RTC_week	day of the week
	RTC_month	the month
	RTC_year	the year
	RTC_century	the century

Return value:

The value requested by the user.

Example:
```
void main(void)
{
printf("\n Century = %d",
GetRTCtime(RTC_century) );
printf("\n Year = %d", GetRTCtime(RTC_year) );
printf("\n month = %d", GetRTCtime(RTC_month) );
printf("\n weekday = %d", GetRTCtime(RTC_week) );
printf("\n day = %d", GetRTCtime(RTC_day) );
printf("\n hour = %d", GetRTCtime(RTC_hour) );
printf("\n min = %d", GetRTCtime(RTC_min) );
printf("\n sec = %d", GetRTCtime(RTC_sec) );
}
```

Remarks:
None.

Get_SysMem

Syntax:
unsigned char Get_SysMem(unsigned char which_byte)

Description:
Reads a byte from security SRAM.

Parameter	**Description**
which_byte	From 0 to 112, user-determined.

Return value:
The value in a byte of security SRAM.

Example:
```
unsigned char SlotValue[4];
void main(void)
{
int I;
/* recover last value */
for(I=0;I < 4;I++)
SlotValue[I] = Get_SysMem(I);
}
```

Remarks:
None.

read_backup_ram

Syntax:
unsigned char read_backup_ram(unsigned int index)

Description:
Reads the value in backup RAM at index address, 60 KB total backup RAM, index = 0 – 61439; absolute addresses from 0x30000 – 0x3EFFF.

Parameter	**Description**
index	From 0 to 61439, 60 KB in total.

Return value:
The single-byte value in backup RAM at address index.

Example:
```
void main(void)
{
unsigned char data;
data = read_backup_ram(500);
/* put your codes here */
}
```

Remarks:
None.

read_mem

Syntax:
unsigned char read_mem (int memory_segment , unsigned int i)

Description:
Reads far memory data, 256 KB Flash memory, from 0x80000L to 0xBFFFFL, where (the Absolute Address) = (SEG*16 + OFFSET). For example, (0x800FFL) = (0x8000*16 + 0x00FF).

Parameter	Description
memory_segment	User-determined address taken from the range 0x8000 to 0xBFFF.
i	Offset for use in location of memory taken from the range 0x0000 to 0xFFFF.

Return value:
The value in memory storage at the indicated address.

Example:
```
void main(void)
{
unsigned char data;
data = read_mem(0x8000, 0x0000);
/* put your codes here */
}
```

Remarks:
None.

Release_All

Syntax:
void Release_All()

Description:
Releases all timer resources of the ADAM-5510 system.

Parameter	Description
None.	

Return value:
None.

Remarks:
None.

Example:
```
void main()
{
int idx;
/*— Initializes the timer built into the 80188
microprocessor —*/
Timer_Init();
/*— Sets time interval of the timer to
1 second. —*/
idx=Timer_Set(1000);
/*— Checks whether the timer has timed out —*/
while(tmArriveCnt[idx]==0)
{
/*— user can attend to other tasks...—*/
}
/*— Resets the current timer to its initial
state. —*/
Timer_Reset(idx);
/*— Releases all timer resources —*/
Release_All()
}
```

Set_NVRAM_Size

Syntax:
void Set_NVRAM_Size(unsigned char sector)

Description:
Sets the battery backup RAM size. The unit is sectors, each sector is 4 Kbytes in size. Maximum size is 60 KB theoretically.

Parameter	**Description**
sector	NV RAM size in 4 KB sectors, from 1 to 15 sectors.

Return value:
None.

Example:
```
void main()
{
Set_NVRAM_Size(31); /* sets NVRAM size to 15
KB*/
}
```

Remarks:
Maximum size is 60 KB theoretically. Actual size available depends on the user's programming.

SetRTCtime

Syntax:
void SetRTCtime(unsigned char Time, unsigned char data)

Description:
Sets date and time of the real-time clock.

Parameter	Description	
Time	RTC_sec	the second
	RTC_min	the minute
	RTC_hour	the hour
	RTC_day	the day
	RTC_week	day of the week
	RTC_month	the month
	RTC_year	the year
	RTC_century	the century
data	New contents.	

Return value:
None.

Example:
```
void main()
{
unsigned char sec=0, min=0, hour=12;
/* set current time 12:00:00. */
SetRTCtime(RTC_sec,sec);
SetRTCtime(RTC_min,min);
SetRTCtime(RTC_hour,hour);
}
```

Remarks:
None.

Set_SysMem

Syntax:
void Set_SysMem(unsigned char which_byte, unsigned char data)

Description:
Writes a byte to security SRAM. Security SRAM supports 113 bytes for user storage of important information.

Parameter	Description
which_byte	From 0 to 112, user determined.
data	Value to be saved.

Return value:
None.

Example:
```
unsigned char data[4] = {1,2,3,4};
void main(void)
{
int I;
/* save current value */
for(I=10;I < 14;I++)
Set_SysMem(I, data[I-10]);
}
```

Remarks:
None.

Timer_Init()

Syntax:
int Timer_Init()

Description:
Initializes the timer built into the 80188 microprocessor. The return value "0" means the initialization of the time was successful. The return value "1" means the timer had already been initialized.

Parameter	**Description**
None.	

Return value:
0: Initialization was successful.
1: The timer had already been initialized.

Remarks:
None.

Timer_Reset

Syntax:
void Timer_Reset(int idx)

Description:
Resets the timer identified by the integer idx to its initial state.

Parameter	**Description**
idx	Timer index.

Return value:
None.

Remarks:
None.

Timer_Set

Syntax:
int Timer_Set(unsigned int msec)

Description:
Requests a timer function from the microprocessor and then sets the time interval of the function. Timer intervals are set in 5 millisecond increments. The function return value is an integer representing the ID of the timer function when it is successful. A return value "-1" means the request failed. Programmers should consider whether an assigned timer has timed-out when programming for timer functions. The value of the variable tmArriveCnt[idx] can be checked to verify timer status. A value of 0 indicates that the timer is still counting. Values other than 0 mean the timer has timed-out.

Parameter	**Description**
msec	Time interval set, max. value is 65536.

Return value:

Integer	Function success, value represents function timer ID. Max. value of 100.
-1	Function failure.

Remarks:
Timer function calls in the ADAM-5510 are emulated as timer functions in a PLC. Applications using timer functions will run less efficiently the more timer functions are running simultaneously in a program. Please refer to Example 9 on the utility diskettes for details.

WDT_clear, WDT_disable, WDT_enable

Syntax:
void WDT_clear(void)
void WDT_disable(void)
void WDT_enable(void)

Description:
Clear watchdog timer.
Disable watchdog timer.
Enable watchdog timer.

When the watchdog timer is enabled, it will have to be cleared at least once every 1.5 seconds. The watchdog timer default value is "disable".

Parameter **Description**
None.

Return value:
None.

Example:
```
void main(void)
{
int I;
WDT_enable();
For(I=0;I < 10;I++)
{
ADAMdelay(1000);
WDT_clear();
}
WDT_disable();
}
```

Remarks:
None.

write_backup_RAM

Syntax:
void write_backup_RAM(unsigned int index, BYTE data)

Description:
Writes a byte to battery backup memory.

Parameter	**Description**
index	An index for data in the battery backup RAM,

	from 0 to 61439; 60 KB battery backup SRAM in total.
data	A byte of data that the programmer wants to write to battery-protected SRAM.

Return value:
None.

Example:
```
void main()
{
unsigned char data=0x55;
/* Writes the data 0x55 into battery backup memory,
index 10 */
write_backup_RAM(10,data);
}
```

Remarks:
None.

Appendix A Register Structure

This appendix gives a short description of each of the ADAM-4500's registers. For more information please refer to the data book for the STARTECH 16C550 UART chip.

All registers are one byte in length. Bit 0 is the least significant bit, and bit 7 is the most significant bit. The address of each register is specified as an offset from the port base address (BASE), COM1 is 3F8h and COM2 is 2F8h.

DLAB is the "Divisor Latch Access Bit", bit 7 of BASE+3.

BASE+0	Receiver buffer register when DLAB=0 and the operation is a read.
BASE+0	Transmitter holding register when DLAB=0 and the operation is a write.
BASE+0	Divisor latch bits 0 - 7 when DLAB=1
BASE+1	Divisor latch bits 8-15 when DLAB=1.

The two bytes BASE+0 and BASE+1 together form a 16-bit number, the divisor, which determines the baud rate. Set the divisor as follows:

Baud rate	**Divisor**	**Baud rate**	**Divisor**
50	2304	2400	48
75	1536	3600	32
110	1047	4800	24
133.5	857	7200	16
150	768	9600	12
300	384	19200	6
600	192	38400	3
1200	96	56000	2
1800	64	115200	1
2000	58		

BASE+1 Interrupt Status Register (ISR) when DLAB=0

bit 0	Enable received-data-available interrupt
bit 1	Enable transmitter-holding-register-empty interrupt
bit 2	Enable receiver-line-status interrupt
bit 3	Enable modem-status interrupt

BASE+2 FIFO Control Register (FCR)

bit 0	Enable transmit and receive FIFOs
bit 1	Clear contents of receive FIFO
bit 2	Clear contents of transmit FIFO
bits 6-7	Set trigger level for receiver FIFO interrupt

Bit 7	**Bit 6**	**FIFO trigger level**
0	0	01
0	1	04
1	0	08
1	1	14

BASE+3 Line Control Register (LCR)

bit 0	Word length select bit 0
bit 1	Word length select bit 1

Bit 1	**Bit 0**	**Word length (bits)**
0	0	5
0	1	6
1	0	7
1	1	8

	bit 2	Number of stop bits
	bit 3	Parity enable
	bit 4	Even parity select
	bit 5	Stick parity
	bit 6	Set break
	bit 7	Divisor Latch Access Bit (DLAB)
BASE+4	Modem Control Register (MCR)	
	bit 0	DTR
	bit 1	RTS
BASE+5	Line	Status Register (LSR)
	bit 0	Receiver data ready
	bit 1	Overrun error
	bit 2	Parity error
	bit 3	Framing error
	bit 4	Break interrupt
	bit 5	Transmitter holding register empty
	bit 6	Transmitter shift register empty
	bit 7	At least one parity error, framing error or break indication in the FIFO
BASE+6	Modem Status Register (MSR)	
	bit 0	Delta CTS
	bit 1	Delta DSR
	bit 2	Trailing edge ring indicator
	bit 3	Delta received line signal detect
	bit 4	CTS
	bit 5	DSR
	bit 6	RI
	bit 7	Received line signal detect
BASE+7	Temporary data register	

Appendix B Safety Instructions

1.Please read these safety instructions carefully.
2.Please keep this User's Manual for later reference.
3.Please disconnect this equipment from AC outlet before cleaning. Don't use liquid or sprayed detergent for cleaning. Use moisture sheet or cloth for cleaning.
4.For pluggable equipment, the socket-outlet shall be installed near the equipment and shall be easily accessible.
5.Please keep this equipment from humidity.
6.Lay this equipment on a reliable surface when install. A drop or fall could cause injury.
7.The openings on the enclosure are for air convection hence protect the equipment from overheating. DO NOT COVER THE OPENINGS.
8.Make sure the voltage of the power source when connect the equipment to the power outlet.
9.Place the power cord such a way that people can not step on it. Do not place anything over the power cord.
10.All cautions and warnings on the equipment should be noted.
11.If the equipment is not used for a long time, disconnect the equipment from mains to avoid being damaged by transient overvoltage.
12.Never pour any liquid into opening, this could cause fire or electrical shock.
13.Never open the equipment. For safety reason, the equipment should only be opened by qualified service personnel.
14.If one of the following situations arises, get the equipment checked by a service personnel:
a: The power cord or plug is damaged.
b. Liquid has penetrated into the equipment.
c. The equipment has been exposed to moisture.
d. The equipment does not work well or you can not get it to work according to user's manual.
e. The equipment has dropped and damaged.
f. If the equipment has obvious sign of breakage.
15.DO NOT LEAVE THIS EQUIPMENT IN AN ENVIRONMENT UNCONDITIONED, STORAGE TEMPERATURE BELOW -20° C (-4° F) OR ABOVE 60°C (140°F), IT MAY DAMAGE THE EQUIPMENT.
The sound pressure level at the operators position according to IEC 704-1:1982 is equal to or less than 70dB(A).

DISCLAIMER: This set of instructions is provided according to IEC 704-1. Advantech disclaims all responsibility for the accuracy of any statements contained therein.

Wichtige Sicherheishinweise

1. Bitte lesen sie Sich diese Hinweise sorgfältig durch.
2. Heben Sie diese Anleitung für den späteren Gebrauch auf.
3. Vor jedem Reinigen ist das Gerät vom Stromnetz zu trennen. Verwenden Sie Keine Flüssig-oder Aerosolreiniger. Am besten dient ein angefeuchtetes Tuch zur Reinigung.
4. Die NetzanschluBsteckdose soll nahe dem Gerät angebracht und leicht zugänglich sein.
5. Das Gerät ist vor Feuchtigkeit zu schützen.
6. Bei der Aufstellung des Gerätes ist auf sicheren Stand zu achten. Ein Kippen oder Fallen könnte Verletzungen hervorrufen.
7. Die Belüftungsöffnungen dienen zur Luftzirkulation die das Gerät vor überhitzung schützt. Sorgen Sie dafür, daB diese Öffnungen nicht abgedeckt werden.
8. Beachten Sie beim AnschluB an das Stromnetz die AnschluBwerte.
9. Verlegen Sie die NetzanschluBleitung so, daB niemand darüber fallen kann. Es sollte auch nichts auf der Leitung abgestellt werden.
10. Alle Hinweise und Warnungen die sich am Geräten befinden sind zu beachten.
11. Wird das Gerät über einen längeren Zeitraum nicht benutzt, sollten Sie es vom Stromnetz trennen. Somit wird im Falle einer Überspannung eine Beschädigung vermieden.
12. Durch die Lüftungsöffnungen dürfen niemals Gegenstände oder Flüssigkeiten in das Gerät gelangen. Dies könnte einen Brand bzw. elektrischen Schlag auslösen.
13. Öffnen Sie niemals das Gerät. Das Gerät darf aus Gründen der elektrischen Sicherheit nur von authorisiertem Servicepersonal geöffnet werden.
14. Wenn folgende Situationen auftreten ist das Gerät vom Stromnetz zu trennen und von einer qualifizierten Servicestelle zu überprüfen:
    a - Netzkabel oder Netzstecker sind beschädigt.
    b - Flüssigkeit ist in das Gerät eingedrungen.
    c - Das Gerät war Feuchtigkeit ausgesetzt.
    d - Wenn das Gerät nicht der Bedienungsanleitung entsprechend funktioni ert oder Sie mit Hilfe dieser Anleitung keine Verbesserung erzielen.
    e - Das Gerät ist gefallen und/oder das Gehäuse ist beschädigt.
    f - Wenn das Gerät deutliche Anzeichen eines Defektes aufweist.

    Der arbeitsplatzbezogene Schalldruckpegel nach DIN 45 635 Teil 1000 beträgt 70dB(A) oder weiger.

DISCLAIMER: This set of instructions is provided according to IEC704-1.
Advantech disclaims all responsibility for the accuracy of any statements contained therein.