Am186™CC/CH/CU Microcontroller Ethernet Driver CodeKit Software

V1.3 - Thursday, February 11, 1999

AMD
Embedded **P**rocessor **D**ivision

Readme File : Version 1.3 – February 11, 1999

1 Overview

This CodeKit software contains an interrupt-driven Ethernet driver for the AMD Am186™CC/CH/CU microcontroller. This driver supports the transmission and reception of standard Ethernet driver packets on the Am186CC/CH/CU microcontroller. This CodeKit software includes a small application that performs an external loopback test on the Am186CC/CH/CU microcontroller customer development platform (CDP) ISDN TA daughter module Ethernet connector.

IMPORTANT: By loading this software or any portion thereof, you agree to all of the terms of the License Agreement in the file LICENSE.PDF. Do not use this software until you have carefully read and agreed to terms and conditions in the Agreement. If you do not agree to the terms of the Agreement, do not use this software or any portion thereof.

For support: email epd.support@amd.com
or call 1-800-222-9323

Changes from previous versions (Versions 1.2 and earlier) of this CodeKit software: Version 1.3 of this CodeKit software is designed for the Ethernet driver to run out of the DRAM on the Am186CC/CH/CU microcontroller CDP, not SRAM like previous versions of this CodeKit software. This version of the CodeKit software uses packet SRAM on the ISDN TA daughter module (Revision 2.0 and later). For revisions prior to Revision 2.0, this CodeKit software uses packet SRAM on the CDP main board.

1.1 Tools & System Requirements

The following target hardware and tools are required to build and execute this CodeKit software.

ITEM	DESCRIPTION
Target Hardware	Am186CC/CH/CU microcontroller CDP, Revision 1.1 and above with the ISDN TA daughter module, Revision 1.1 and above
Monitor	AMD E86MON™ software, Version 3.32 (or newer)
Workstation Operating System	This CodeKit software builds under DOS 6.2, Windows® 3.1, Windows 95, 98 or NT.
Tool Suite	Microsoft® Visual C++ V1.52
Compiler*	Microsoft C/C++ Optimizing Compiler Version 8.00c
Linker*	Microsoft Segmented Executable Linker Version 5.60.339 Dec 5 1994

*Note the compiler and linker are part of the Visual C++ V1.52 tool suite.

1.2 Installing The CodeKit Software

This CodeKit software is delivered as a ZIP file. To unzip using the command line program `PKUNZIP.EXE` perform the following steps:

1. Create an empty directory; a good name is `ETHERDRV`.
2. Copy the ZIP file to the new directory.
3. Change directories (use the `CD` command) to the new directory.
4. Execute the command `PKUNZIP ETHERDRV`.

You do not have to use the '-d' switch as all the files are in one simple directory.

1.3 Files and Directories

The ZIP file contains the following files in a single directory.

FILE	DESCRIPTION
`186INIT.H`	Declarations for communication between the 186-specific initialization routines.
`AM186CC.H`	Contains hardware-specific declarations for the Am186CC/CH/CU microcontroller.
`AMDRIVE.H`	Definitions for PCN driver.
`BUILDIT.BAT`	A batch file that builds the CodeKit software.
`CLEANIT.BAT`	Cleans up intermediate files after a build.
`ENVIRON.BAT`	Sets up the build environment.
`ETHRDEMO.HEX`	Ethernet driver test application.
`LED.H`	Low-level LED driver include file.
`MAIN.C`	Ethernet demonstration code file.
`MAKEHEX.EXE`	Converts an `EXE` program into a `HEX` file for downloading into the target board using E86MON software.
`PCN.H`	Contains various structures for use in PCI/PCnet™ devices.
`PCNDRV.C`	32-bit network interface driver for use with the PCI/PCnet device.
`PCNDRV.H`	This contains all the basic structures used by the `PCNDRV.C` driver.
`PNPAPPN.H`	Contains function prototypes for Plug & Play (PnP) initialization functions.
`README.DOC`	This document in Microsoft Word 97 format
`README.PDF`	This document in PDF format.
`STDINCS.H`	Common typedefs and definitions for AMD x86 systems.
`TIPLEDS.C`	This file contains hardware-dependent LED display code.

1.4 Setting Up the Build Environment

Before the CodeKit software is built, you need to set up the build environment. Do this by running the ENVIRON.BAT command file. Running the command file sets up the environment variables listed in the table below.

This batch file assumes that the tools have been installed in their default locations on the 'C:' drive.

ENV Variable	Setting
TOOLROOTDIR	C:\MSVC
INCLUDE	C:\MSVC\INCLUDE
LIB	C:\MSVC\LIB

1.5 Setting Up the Target Hardware

Note that there are currently several versions of the Am186CC/CH/CU microcontroller available. The version number is on the microcontroller itself. The A0 version contains a hardware errata that results in bus contention with the Ethernet controller. This often results in the system DRAM not being refreshed frequently enough to retain data. This problem has been resolved by running the program out of SRAM using a modified version of E86MON software that loads executable HEX files in SRAM.

1.5.1 CDP (Am186CC/CH/CU Microcontroller, Version A1 and above) with Revision 1.0 or 1.1 ISDN TA Daughter Module

For the Ethernet driver to work on the CDP, the CDP must be set up for accessing SRAM with memory chip select 0 (MSC0) by jumpering MSC0 on JP10. The demonstration program should be compiled with the DRAM option defined with the /D command line parameter in the BUILDIT.BAT file. This is already set in BUILDIT.BAT unless you have modified it after unzipping the file..

1.5.2 CDP (Am186CC/CH/CU Microcontroller, Version A1 and above) with Revision 2.0 or above ISDN TA Daughter Module

For the Ethernet driver to work on the CDP, there must not be any chip select jumpers for the on-board SRAM to be set on JP10. The packet SRAM is selected by setting the MSC0# jumper on the ISDN TA daughter module. The demonstration program should be compiled with the DRAM option defined with the /D command line parameter in the BUILDIT.BAT file. This is already set in BUILDIT.BAT unless you have modified it after unzipping the file.

1.5.3 CDP (Am186CC/CH/CU Microcontroller, Version A0) (Does Not Work with Revision 2.0 ISDN TA Daughter Module)

The SRAM needs to be controlled by the lower memory chip select (LCS) on CDPs using version A0 of the Am186CC/CH/CU microcontroller. Jumper LCS on JP10 to enable LCS to select SRAM. You also need to remove the /D DRAM option from the BUILDIT.BAT file and rerun the BUILDIT.BAT file to create the version of the Ethernet driver that runs out of SRAM. Also, to run the program out of SRAM, you need to burn a version of E86MON software that is capable of running code out of SRAM into the Flash memory of the CDP. This version of E86MON software is available on the AMD web site:

http://titan.amd.com/codekits/am186cc/EMON/

Use your browser's download command to load the new E86MON software HEX file into Flash memory. The HEX file is configured to write itself to Flash memory, so no special steps are required. The new version of E86MON software resides at location F000 in Flash memory and can be accessed by typing `G F000:0`. Press the 'A' key on your terminal to let the new version of E86MON software detect your baud rate. Now the new version of E86MON software is running and it can load and execute the Ethernet demonstration.

If you want to use this version of E86MON software rather than the version currently installed on your CDP, use the 'Z' upgrade command to make this version your boot version of the E86MON software.

1.6 Building the CodeKit Software

Building this CodeKit software is very simple:

1. Execute the `ENVIRON.BAT` file to setup your environment.
2. Execute the `BUILDIT.BAT` file.

This results in the `ETHRDEMO.HEX` file being built. This can then be downloaded to the target hardware.

NOTE: The `MAKEHEX.EXE` utility converts an EXE file into a HEX file. The version of `MAKEHEX.EXE` you have is matched to a version of E86MON software. If you have problems loading your HEX file using the E86MON software on the target board, please contact our support team for a correct version of E86MON software. Note that the support team will need to know what version of E86MON software is loaded on your board.

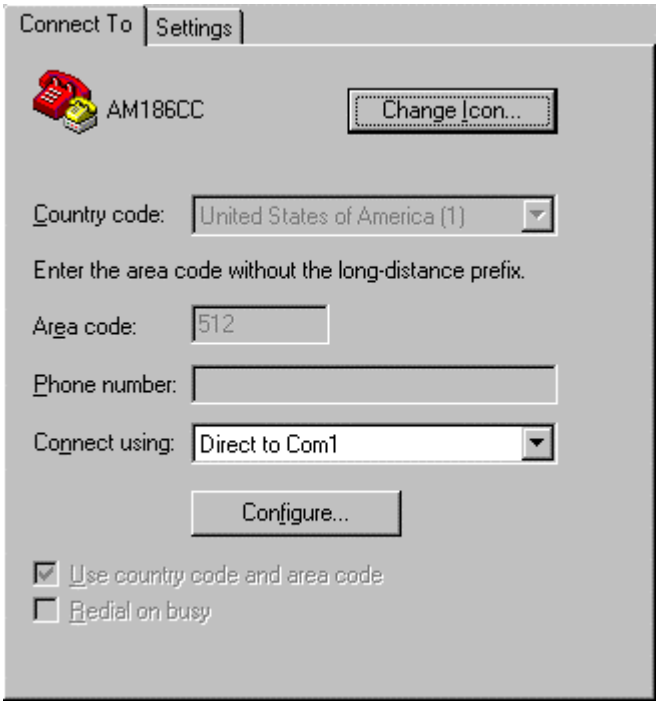
For help with `MAKEHEX.EXE` just run `MAKEHEX` with no arguments.

1.7 Loading the Code

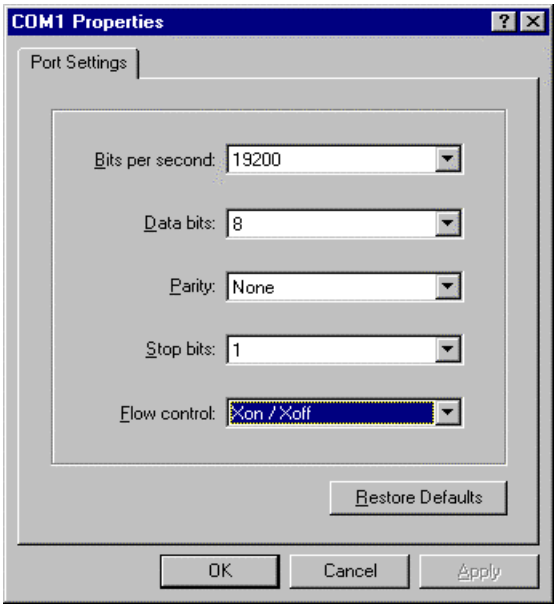
You need some kind of terminal program like HyperTerminal (on Windows 95 or NT) or Terminal on Windows 3.1. The following are the steps to use for connecting to an Am186CC/CH/CU microcontroller CDP using COM1 on your PC with HyperTerminal:

Step 1: Connect a null modem cable from COM1 of your PC to either of the serial ports on your CDP.

Step 2: Run HyperTerminal and create a new connection directly to COM1.



Step 3: Click the Configure… button to set the baud rate and other communications parameters to 19200 Baud, 8 Data Bits, No Parity, 1 Stop Bit and Xon/Xoff Flow Control.



Step 4: Reset the target board, wait a second or two, then press the 'A' key (either upper or lower case). The E86MON software (running on the target board) detects the baud rate and issues you a prompt, like this:

```
Welcome to AMD's EMon 186!      (? <Enter> for help)
cc86mon:
```

Step 5: This step loads the program into RAM. If you reset the board, you will have to load the program again[1]. Select "Send Text File" from the "Transfer" menu. (Note, this is a basic ASCII file transfer. In other words, the file is sent as is with no protocol, just as if you could type it in really fast.) In the resulting dialog box, select the `ETHRDEMO.HEX` file. The file transfer begins. After it is finished, your terminal window looks something like this:

```
cc86mon: :040000003004F001694
Transferring hex file (Press Esc to abort).....................................
File download successful
cc86mon:
```

Step 6: To execute the program, use the 'G' (for GO) command. The program should now run.

NOTE: If your receive an INT21H error when you run the program, this means INT21H support was not loaded. Use the 'LL' (Load Library) command before loading the program to load INT21H support.

[1] See the *E86MON™ Software User's Manual* for instructions on how to load programs into Flash memory.

2 About the Demonstration Program

The demonstration program transmits an ICMP echo request packet that is returned to the receiver by the loopback plug. The program repeats this process until you terminate the program.

The program prompts you to press a key to start. When this is done, the program performs the PnP initialization of the PCnet-ISA Ethernet controller. A menu provides you with several options, listed in the table below (not all of the options below are shown in the menu, but they are available).

Command	Action
G	Transmit echo request packets and wait for the packet to be received.
U	Unacknowledged transmit. Repeatedly transmit packets, even if no packets are received after the first transmission.
S	Stop transmitting packets / Pause.
M	Transmit max packet size – 1500 bytes.
1	Minimum packet size – 64 bytes.
2	200 byte packet size.
5	500 byte packet size.
Q	Terminate program.

A typical test would consist of starting the program, pressing a key when prompted, and using the 'G' (for GO) command to start a loopback test that should continue until you terminate the program. The loopback test shows that packets are being transferred on your terminal program and shows the contents of the received packets.

2.1 Troubleshooting the Demonstration Program

The program prompts you that the Ethernet controller failed to initialize.

The most common cause of this is simply the CDP's ISDN TA daughter module not being plugged in securely.

Another possible cause is that the jumpers on the CDP are incorrectly configured for the demonstration program.

The program transmits one packet, then stalls.

This is usually a result of the Ethernet driver loopback plug not being securely plugged into the CDP's ISDN TA daughter module when the program is executed.

Another possible cause is that the CDP isn't configured correctly.

The program reports that the transmit FIFO is full.

The CDP is probably configured for the incorrect SRAM configuration; check the CDP setup.

The program locks up for no apparent reason.

This is often the result of running the program in DRAM with the A0 version of the Am186CC/CH/CU microcontroller. A bus mastering errata prevents the Am186CC/CH/CU microcontroller from regaining control of the address bus. This prevents the microcontroller from properly refreshing the system DRAM, and eventually

leads to the program's crashing. You can avoid this problem by loading a version E86MON software that has been modified to run the program out of SRAM. This procedure is described in Section 1.6 in this document.

Note that if your CDP uses the A1 version of the Am186CC/CH/CU microcontroller, this should not be a problem because the bus mastering errata is corrected in the A1 version.

To exit the demonstration, press the escape key from either terminal. Control is returned to the E86MON software on the serial port that it was controlling before the demonstration program was started.

3 Ethernet Driver Code Functions

The following table lists the important functions in `MAIN.C`.

Function	Action
`MAIN`	Calls Ethernet driver and CDP initialization functions. Runs the demonstration program.
`Init_Packet`	Sets up packet contents address/data.
`Init_Board`	Sets up board memory configuration.

The following table lists the important functions in `PCNDRV.C`.

Function	Action
`IntHandler`	Handles packet receive and transmit interrupts.
`PcnInit`	Initializes Ethernet controller.
`pcnShutDown`	Terminates the driver.
`pcnOutPut`	Transmits a single packet.
`pcnRProcess`	Called to check for receive packet.

The following table lists the important functions in PNPAPPN.C.

Function	Action
ip_wake1	Sends PnP initialization key to Ethernet controller.
ip_wake2	Sends special AMD PnP initialization key to Ethernet controller.
ip_cfg_init	PnP initialization.
ip_cfg_w	Writes to PnP register.
ip_bcr_w	Writes to bus configuration register.
ip_bcr_r	Reads from Ethernet controller bus configuration register.

4 Source Code Functions

4.1 MAIN.C

This is the code that implements the demonstration program. The program begins by initializing the CDP and the Ethernet controller on the ISDN TA daughter module. The program also assigns an arbitrary Ethernet address to the controller which is 00-01-02-03-04-05 (this is assigned to ppcnInit.eAddr in the main function). The address does not really matter for the demonstration because the Ethernet controller is put in promiscuous mode where it accepts all packets. The address may be of concern for more selective applications. The rest of the code in this file deals with handling ICMP echo request packets and generating responses to the requests. This produces an infinite loop of requests and replies when the demonstration program is run with the ‘g’ or ‘u’ options selected at the command prompt.

4.1.1 PCNDRV.C

This is the most critical file because it contains the actual PCN driver code. This file contains the receive/transmit interrupt that controls flags used by the software to indicate successful transmits and receives. This file also contains the functions for transmitting and receiving Ethernet packets, as well as definitions pertaining to the address of packet SRAM.

4.1.2 PNPAPPN.C

This source file contains code that performs the PnP initialization of the Ethernet controller. Upon CDP power-up, the Am79C961 Ethernet controller starts up with its pins in a tri-state mode. A special sequence of data must be sent to the controller to bring it out of tri-state mode for initialization. This is described in detail in the head of the PNPAPPN.C file.

This function also contains important configuration data that determines in which mode the Ethernet controller operates, such as loopback or non-loopback modes (see the ip_cfg_init() function).

4.1.3 TIPLEDS.C

This file contains source that allows the Ethernet driver to send some of its output through the test interface port (TIP) of the CDP. This enables you to see that the Ethernet driver is actually functioning, without having to depend on a terminal program.

More information about the code is supplied in the inline documentation.

5 Code Configuration

5.1 Memory

Setting the `/D DRAM` option in the `BUILDIT.BAT` file sets up the code to access the packet SRAM using MSC0#. If the `/D SMALLSRAM` option is set, the code accesses the packet SRAM using #LCS.

5.2 Loopback vs. Non-Loopback

The default configuration of the Ethernet driver supplied in this CodeKit software is External Loopback mode. In this mode, the transmitter sends a packet out of the Ethernet controller such that external transformers and terminators can be tested. For the packet to be returned to the Ethernet controller, a loopback plug must be put in the CDP ISDN TA daughter module Ethernet connector. The code that dictates whether the Ethernet controller is in loopback mode or normal operation is shown below.

From `PCNDRV.C`

The function `PcnInit` contains the following line:

```
initBlkp->tmode = 0xb084; // transmit descriptor1 see PcNet manual
```

Change the line to the following:

```
initBlkp->tmode = 0x0080;
```

From `PNPAPPN.C`

The function `ip_cfg_init` contains the following line:

```
ip_bcr_w(2,0x4011); // set ISACSR2. ISAINACT set for ISA timing & 10BASE-T loopback
```

Change the line to the following:

```
ip_bcr_w(2,0x0011);    // set ISACSR2. ISAINACT set for ISA timing
```

The above change takes the Ethernet controller out of 10BaseT mode

Making the changes listed above enables the Ethernet controller to be connected to a live network and run without promiscuous mode. This allows the internal hardware of the Ethernet controller to decode packet addresses to select receive packets with only its Ethernet address.

Check the *Am79C961A PCnet™-ISA II* (Full Duplex) Data Sheet, order #19364, for more information about device configuration.

5.3 Packet Data Structures

The type EPACKET located in `MAIN.C` contains the packet structure/protocol used for data transfers. The EPACKET type consists of three subtypes that are described below.

5.3.1 ICMP_HEADER

This structure holds information pertaining to the ICMP protocol such as packet type, length, and checksum. This part of the EPACKET structure is presently set up in `MAIN.C` to generate echo request packets by setting the type field of ICMP_HEAD to decimal 8. This value can be changed to generate different types of packets. The switch statement at the end of `MAIN()` in `MAIN.C` shows what different values of this field can do.