

***ACTF<sup>TM</sup>***  
for  
TERN 16-bit Embedded Microcontrollers

***Technical Manual***

© TERN, Inc. All Rights Reserved.  
Portions © Borland International. All Rights Reserved.  
Portions © Paradigm Systems. All Rights Reserved.



1950 5<sup>th</sup> Street, Davis, CA 95616, USA

Tel: 530-758-0180

Fax: 530-758-0181

Email: [sales@tern.com](mailto:sales@tern.com)

<http://www.tern.com>

## COPYRIGHT

i386-Engine, A-Engine, V25-Engine, and MemCard are trademarks of TERN, Inc.

LOC31, TD31, BC31, TDREM-xx, ACTF are trademarks of TERN, Inc.

Am188ES is a Trademark of Advanced Micro Devices, Inc.

Borland C++ 3.1 and Turbo Debugger are trademarks of Borland International.

Microsoft, MS-DOS Hyper Terminal and Windows are trademarks of Microsoft Corporation.

LOCATE, DEBUG/RT and PDREM are trademarks of Paradigm Systems.

Version 3.00

June 23, 2010

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of TERN, Inc.



© 1997-2010

1950 5<sup>th</sup> Street, Davis, CA 95616, USA

Tel: 530-758-0180 Fax: 530-758-0181

*Email: sales@tern.com*

*http://www.tern.com*

### Important Notice

***TERN*** is developing complex, high technology integration systems. These systems are integrated with software and hardware that are not 100% defect free. ***TERN products are not designed, intended, authorized, or warranted to be suitable for use in life-support applications, devices, or systems, or in other critical applications.*** ***TERN*** and the Buyer agree that ***TERN*** will not be liable for incidental or consequential damages arising from the use of ***TERN*** products. It is the Buyer's responsibility to protect life and property against incidental failure.

***TERN*** reserves the right to make changes and improvements to its products without providing notice.

Temperature readings for controllers are based on the results of limited sample tests; they are provided for design reference use only.

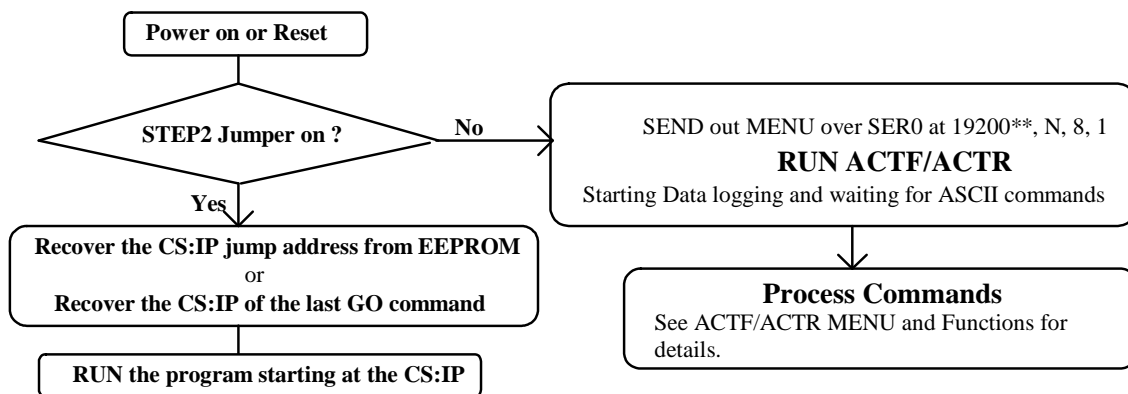
# Chapter 1: Chapter 1: Introduction

## 1.1 What is ACTF™?

ACTF™ is a FLASH version of ACTR™. It is a unique firmware in a Flash EEPROM (AMD29F010/040). ACTF™ can communicate with a PC via a terminal, such as PC windows terminal, setup to 19200\*\*, 8, N, 1, via serial link. By **typing text commands**, you can download your complete application for a finished product.

The ACTF/ACTR™ provides an interactive menu and on-line help for you, so you do not have to dig into manuals. ACTF/ACTR™ not only provides you an easy access to C functions, but also allows you to download a remote debugger kernel for your C/C++ program development, operate the controller and exercise C functions immediately.

A functional flowchart of ACTF™ is shown below:



**\*\*NOTE: BAUD Rate will differ based on the controller. See Appendix E for details.**

Whether downloading a debug kernel for development or for your finished application, an intermediate load utility must be downloaded first (you can not run code out of and program the flash simultaneously). This load utility is downloaded into the SRAM, which then erases the Flash and prepares for downloading into the flash. Chapter 2 of this manual goes into more detail about downloading.

ACTF™ can be used in your final product. After successfully downloading your application into the Flash, setting the power up jump address, and setting a jumper, your application will at every power-on/reset.

## 1.2 Minimum Requirements

### 1.2.1 Minimum Hardware Requirements

- The same minimum hardware requirements for the controller you are using, plus:
- Either ACTF surface-mounted flash or ACTF Flash Chip (DIP package)

### 1.2.2 Minimum Software Requirements

- TERN Paradigm C/C++ Development Kit (DV-P)
  - \* The capabilities of ACTF kit are included with your full DV-P development package.

## 1.3 Hardware and PC-ACTF Communication

The ACTF utility can be installed on your controller in two different ways: either in the on-board flash or with a TERN ACTF Flash chip (DIP package).

If your TERN controller uses a 32-pin ROM socket, it will be necessary to replace the TERN DEBUG ROM with the ACTF chip. As with installing any DIP package into a socket, it is necessary to align pin 1 of the DIP package with pin 1 of the socket. In this case make sure that the notch in the ACTF chip (indicating pin 1) is aligned with the notch in the ROM socket (also indicating pin 1).

See Figure 1.1 for ACTF ROM (DIP) installation.

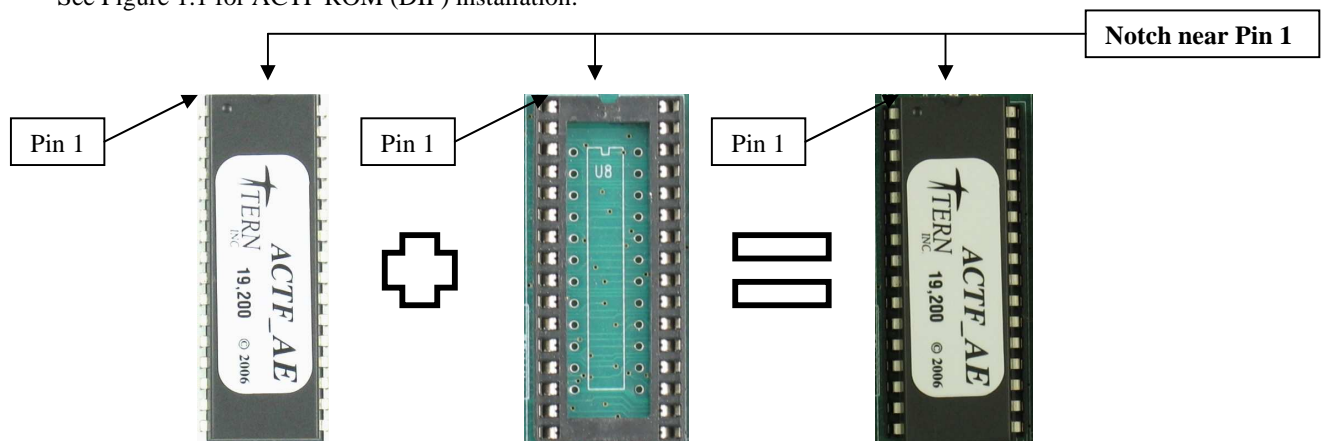


Figure 1.1 ACTF ROM Chip (DIP Package)

### ACTF ROM (DIP) Chips

- 186-based boards (both 128KB and 512KB): ACTF\_AE (except A-Core-88™)
- A-Core-88: ACTF\_AC
- 386-based boards (both 128KB and 512KB): ACTF\_IE
- V25-based boards (128KB only): ACTF\_V25

**These chips can be purchased from TERN with immediate availability.**

If your TERN controller uses on-board surfaced mounted flash, then the ACTF utility already resides in the upper-most sector of the flash. Then to communicate with the ACTF utility, you must setup a hyper terminal on your PC, or use the RTLOAD utility from within the Paradigm C/C++ environment. Figure 1.2 shows a surface-mounted flash chip.



Figure 1.2 ACTF Surface-mount Flash chip.

To communicate with the ACTF utility, you need to set up a serial terminal at your PC. Because the HyperTerminal can be very slow, it is recommended that you use RTLOAD (the terminal communication utility included with Paradigm C/C++).

To access RTLOAD:

**Step 1.3-1:**

Open Paradigm C/C++ and select 'Tool' from the top menu bar. Select 'RTLOAD' from the drop down menu. (Figure 1.3)

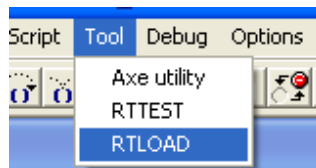


Figure 1.3 Opening RTLOAD

**Step 1.3-2:**

Select the baud rate you require (19200 baud default). Use F5 to cycle through available baud rates. Again, see Appendix E for baud rate details. (Figure 1.4)

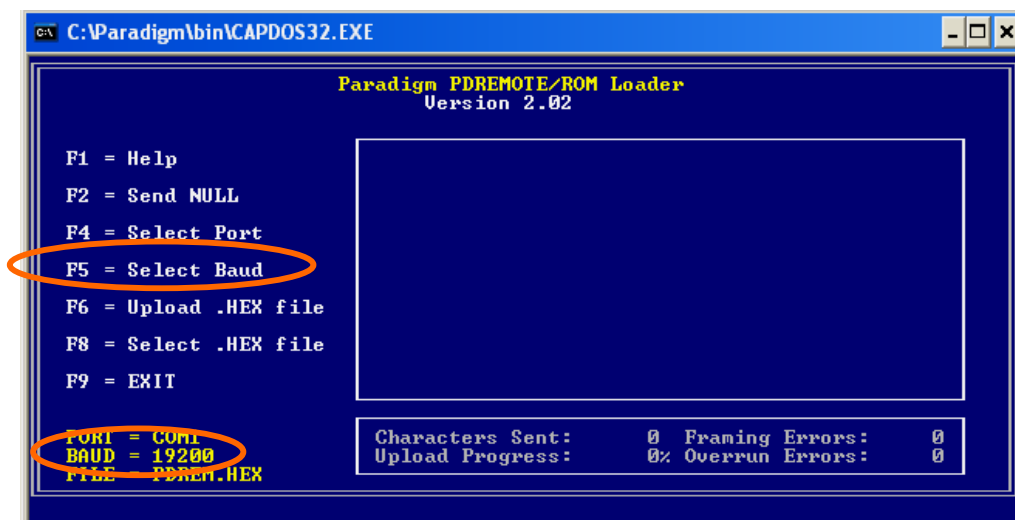


Figure 1.4 RTLOAD Utility; Use F5 for Baud Rate selection.

**Step 1.3-3:**

Select the COM port in which you have your TERN controller connected. Use F4 to cycle through available ports. (Figure 1.5)

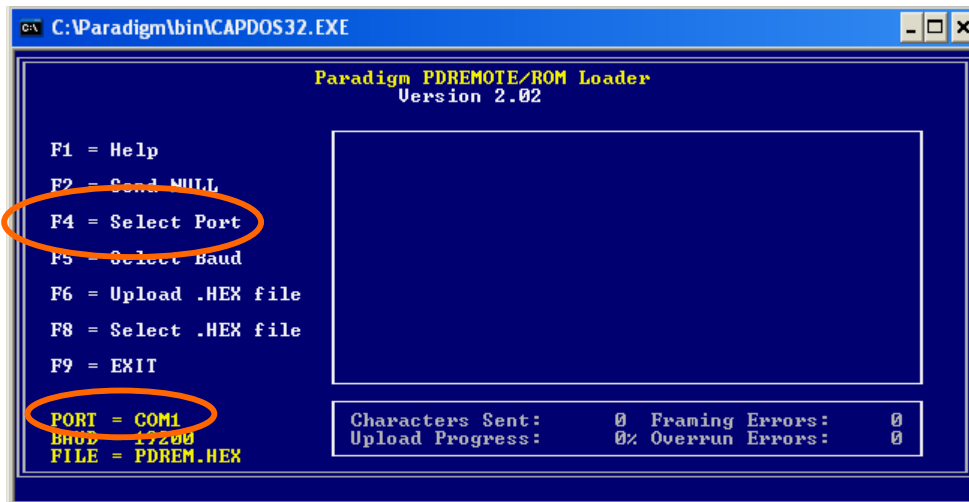


Figure 1.5 Use F4 for COM port selection.

**Step 1.3-4:**

Upon setup of the RTLOAD terminal, power on your TERN controller with the **red Step-2 jumper OFF!** Refer to your controller's technical manual for location of Step-2 jumper. You should see the ACTF menu at the terminal. (Figure 1.6)

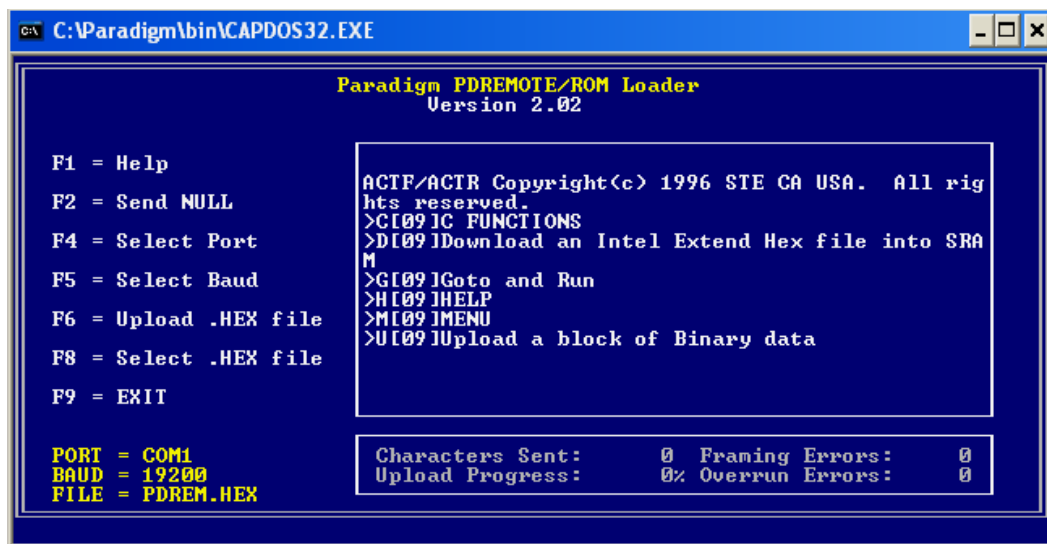


Figure 1.6 ACTF Menu in RTLOAD utility.

You are now setup to run ACTF commands, and/or download debug kernels and application .HEX files.

## Chapter 2: Code/Product Development

### 2.1 Introduction

This chapter will lead you through the many uses of the ACTF utility. The ACTF utility can be used for a number of things. If you are using a AM186/88 or SC520 based controller with surfaced-mounted flash you can use it to first download a debug kernel to allow for code development via Paradigm C/C++. Or, if you are using an ACTF Flash chip, you may be using the ACTF utility solely to download your already developed ACTF downloadable application .HEX file into the flash for a finished product. In either case this chapter will guide you through the steps.

### 2.2 Downloading a Debug Kernel into Flash (ROM & Surface-mount)

All TERN controllers installed with surface-mounted flash, that do not use a ROM socket, require a debug kernel to be downloaded into the flash for communication with the Paradigm C/C++ software for product development (these boards include any 186ES/ER, select 386, and 586 boards).

By default this is already done for you at the factory, but since the sector where the debug kernel resides is not protected, it can be erased easily. This downloading into the surfaced-mounted flash is done using the ACTF utility which resides in the top sector of the surface-mounted flash. The top sector is protected, so you won't be able to inadvertently erase the ACTF utility.

This section will guide you through downloading a debug kernel into the surfaced-mounted flash.

#### DOWNLOAD

Assuming you followed the instructions in **Chapter 1**, you should see the ACTF menu in the **RTLOAD** terminal.

```

C:\Paradigm\bin\CAPDOS32.EXE
Paradigm PDREMOTE/ROM Loader
Version 2.02

F1 = Help
F2 = Send NULL
F4 = Select Port
F5 = Select Baud
F6 = Upload .HEX file
F8 = Select .HEX file
F9 = EXIT

PORT = COM1
BAUD = 19200
FILE = PDREM.HEX

ACTF/ACTR Copyright(c) 1996 STE CA USA. All rights reserved.
>C[09]IC FUNCTIONS
>D[09]I Download Intel Extend Hex file into SRAM
>G[09]I Goto and Run
>H[09]I HELP
>M[09]I MENU
>U[09]I Upload a block of Binary data

Characters Sent: 0 Framing Errors: 0
Upload Progress: 0% Overrun Errors: 0
  
```

Figure 2.1 ACTF Menu

Note that the ACTF utility only recognizes UPPER CASE characters.

To download a debug kernel:

Since the ACTF utility runs out of the Flash and you can't program the flash and run code out of it at the same time, you must download another utility into the SRAM first, which will do the actual programming.

**Step 2.2-1:**

Type upper case 'D' and press ENTER. You will see "Ready to receive Intel Extend HEX file"

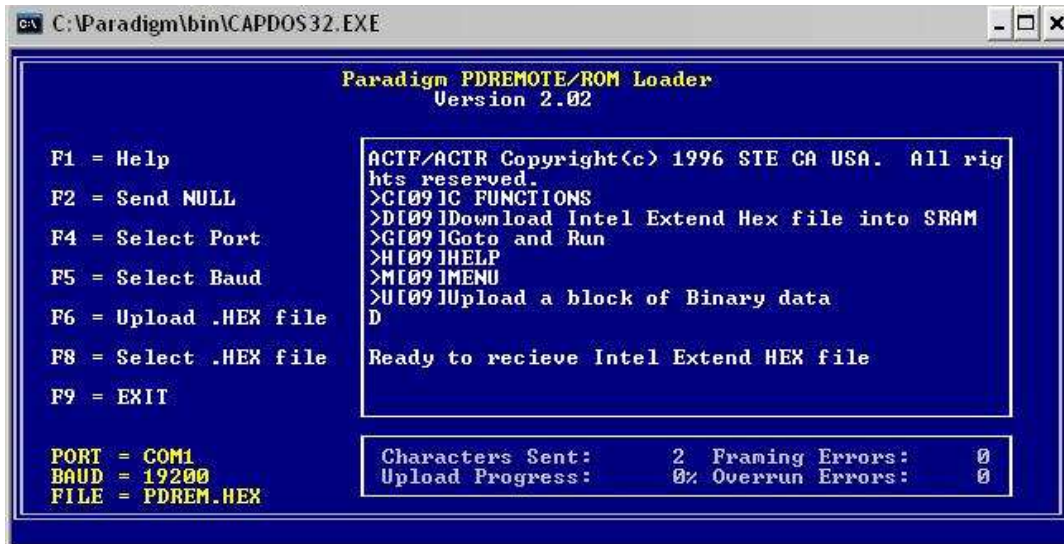


Figure 2.2 Ready to Download File into SRAM

**Step 2.2-2:**

Press **F8** to select the file which will be downloaded into the SRAM. Then type in the file necessary for your controller. A quick discussion of which files are needed is listed below Figure 2.3.

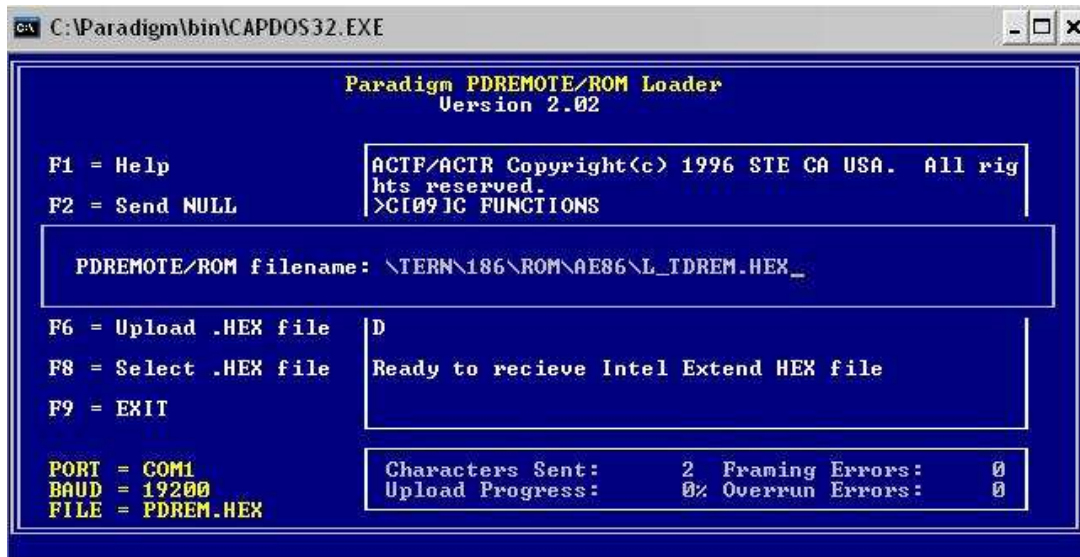


Figure 2.3 Selecting the necessary SRAM file (186ES Example shown).





**Step 2.2-3:**

After SRAM download is complete, the user must execute this code. To do this, we must jump to the starting address where this code resides in the SRAM. This address is designated as 0x04000 on all controllers, with all SRAM loader files.

**NOTE:** Executing the loader file means erasing the sector in which the Debug Kernel will be loaded. On boards using **DIP ACTF Flash** and **586-based boards**, the loader file will erase the ENTIRE flash (except for the ACTF sector of course).

If you are using a DIP ACTF Flash on a **386-Engine** (IE,IEP,etc), the loader will jump to 0x04000 and erase the Flash on its own. You will not need to type 'G04000' in this case. If this is the situation, skip to Step 2.2-4.

Using the ACTF command 'G' (Go To address), type **G04000**, then press **ENTER**.

**IMPORTANT NOTE:** An error may occur with some loader files after using G04000. This is an unintentional software bug. To bypass this error, simply type G04000 again after the error message. (Figure 2.5)

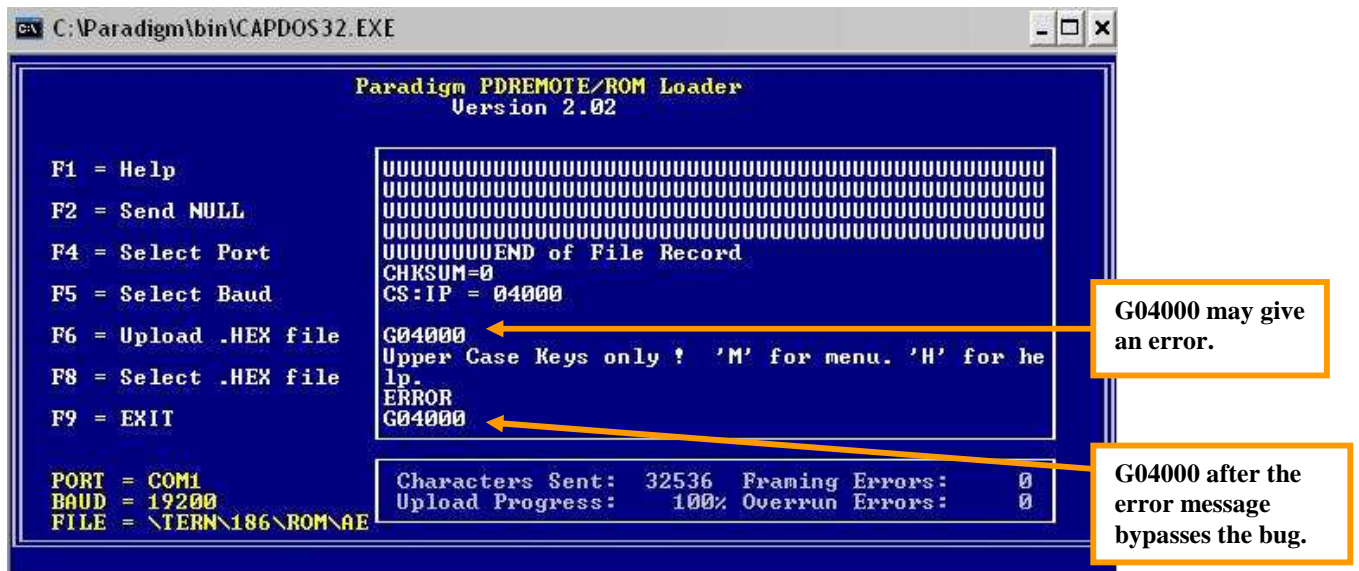


Figure 2.5 Download into SRAM complete (186ES Example shown).

**Step 2.2-4:**

After 'G04000' has been typed and the user has pressed ENTER, the loader will go through and erase (prepare) the necessary sector where the Debug Kernel will be loaded. Note that this will take longer on 586-based controllers and DIP ACTF Flash boards, since the entire flash will be erased.

Once the erase procedure is complete, the user should see a ready statement such as 'Ready to receive TREM???.HEX file at 19200 baud'.

Press **F8** to choose the necessary file.

This time, we will be choosing the Debug Kernel itself. This file name will differ based on which controller is in question. A quick discussion of the different Debug Kernels is listed below Figure 2.6.

**Step 2.2-4 (continued):**

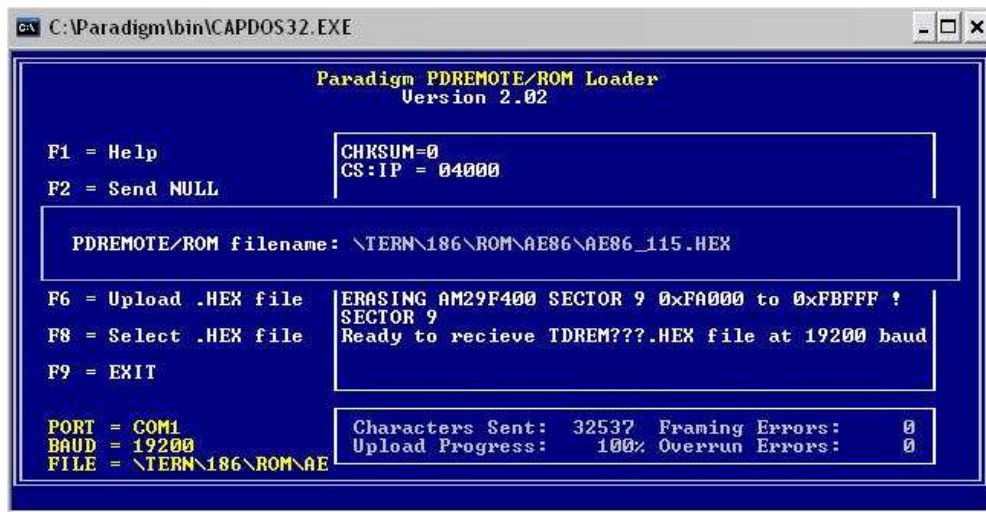


Figure 2.6 Selecting the Debug Kernel (186ES Example shown).

*NOTE:* Below each listed Debug Kernel is the **Jump Address** (execution point of the code) for that particular kernel.

### **DEBUG KERNELS**

#### **188ES-based Controllers**

- \tern\186\rom\af\_0\_115 (for DIP 128KB, 512KB Flash, and **Flashcore-B™** controller)
  - Address: 0xE0000

#### **186ES-based Controllers**

- \tern\186\rom\ae86\ae86\_115.hex (for 20, 40 MHz - 186ES/R8820 CPU's)
  - Address: 0xFA000
- \tern\186\rom\ae86\ee80\_115.hex (for 80 MHz - R1120 CPU)
  - Address: 0xFA000

#### **186ER-based Controllers**

- \tern\186\rom\re\re40\_115.hex (for 40 MHz)
  - Address: 0xFA000
- \tern\186\rom\re\re80\_115.hex (for 80 MHz)
  - Address: 0xFA000

#### **386-based Controllers**

- \tern\386\rom\3860\_115.HEX
  - Address: 0xFA000

#### **586-based Controllers**

- \tern\586\rom\5860\_115.HEX
  - Address: 0x80000

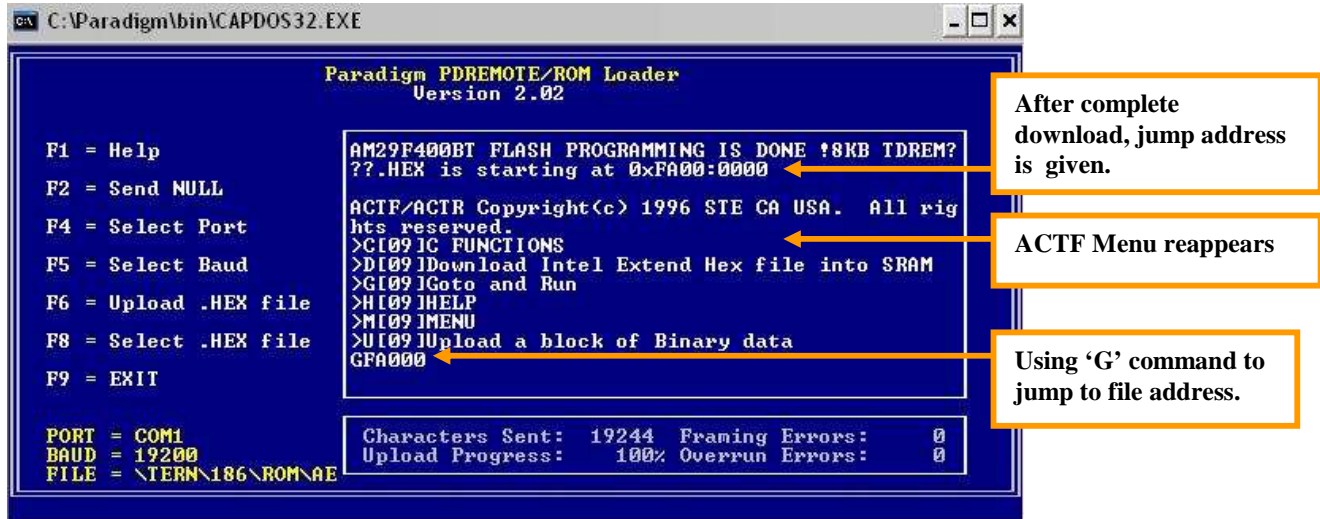
When the proper file is typed in, press **ENTER**.

**Step 2.2-5:**

Press **F6** to commence download of the .HEX file into the Flash. A stream of 'V's will indicate the progressing download of the file.

Once the download is complete, then most\*\* boards will reset, and the ACTF menu will reappear.

Now you may jump to the .HEX file's address (listed above) using the 'G' command.



**Figure 2.7 Jumping to the Debug Kernel (186ES Example shown).**

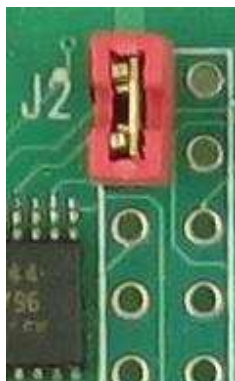
\*\*Some boards like the **SerialDrive™** will not reset, but will instruct the user to power the board off, then power it on again, effectively doing a manual reset and bringing up the ACTF menu.

**Step 2.2-6:**

After the user has jumped to the file's starting address, install the Step-2 jumper. The controller will then jump to and execute the file on power-up!

**Debug Kernel:** The LED should double-blink on power-up, indicating the Debug Kernel has been successfully executed from Flash.

*NOTE: Step-2 jumpers differ by controller. Refer to the controller-in-question's PDF manual for details.*



**Figure 2.8 Step-2 (Red) Jumper (CEye™ Example shown).**

## 2.3 Generating an ACTF Downloadable .HEX file for User Application

This section assumes that you have successfully developed and debugged your application, and it is ready to be programmed into the ACTF Flash using the Paradigm C/C++ TERN Edition with **Development Kit**. It is important to note that the ACTF downloadable .HEX file is different than a non-ACTF .HEX file which is used to burn into an EPROM .

To **generate** an ACTF downloadable .HEX file:

### **Step 2.3-1:**

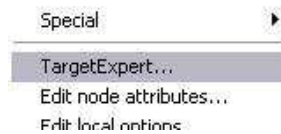
Launch Paradigm C/C++ TERN Edition

### **Step 2.3-2:**

Open the correct project, and select the target you wish to create a .HEX file for.

### **Step 2.3-3:**

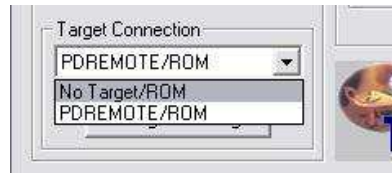
Right-mouse click on the “.axe” node and select ‘Target Expert’.



**Figure 2.9 “Target Expert”**

### **Step 2.3-4:**

In the ‘Target Expert’ window, change the ‘target connection’ from *PDREMOTE/ROM* to *No Target/ROM*.



**Figure 2.10 Setting the target.**

### **Step 2.3-5:**

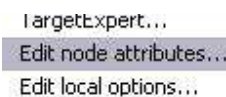
Right-mouse click on the config node of your target and select ‘Edit node attributes’. You want to change this config file to the **actf** config file. You want to change the config file used in your project to a copy of one of these standard configuration files.

**186/188 CPU Boards(No onboard i2chip Ethernet module):** \tern\186\config\actf186.cfg

**186 CPU Boards(w/ onboard i2chip Ethernet module;SLC™/ST™/GE™):** \tern\186\config\slc186.cfg

**386 CPU Boards:** \tern\386\config\actf386.cfg

**586 CPU Boards:** \tern\586\config\586.cfg



**Figure 2.11 Setting the ACTF configuration.**

### **Step 2.3-6:**

After you have selected the new config file, open it for editing (double-click on it).

**Step 2.3-7:**

Among the first statements of the config file are define statements which select the value **FLASH**.

**NOTE:** ACTF386.CFG for 386-based controllers will contain a line for board selection. User must select correct controller number before proceeding.

```
#define BOARD 4 // 0-IE/ID, 1-IEP, 2-ID16, 3-IEM/IEP16, 4-IEL/SD
```

**Figure 2.12 ACTF386.CFG Controller Selection.**

Un-comment the correct statement to match the size of your flash (all surface-mounted flash are 512KB, while the ACTF Flash chips can be 128KB or 512KB).

**Addresses**

- 128KB Flash -> 0xE0000
- 256KB Flash -> 0xC0000
- 512KB Flash -> 0x80000

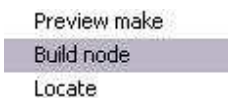
```
// #define FLASH 128 // Use 128KB ROM chip 29F010-70
#define FLASH 256 // Code starts at 0xC0000 for SLC o
// #define FLASH 512 // Use 512KB ROM chip 29F040-70
// or 256KW on-board 16-bit Flash 29F400
// #define FLASH 0 // Put Code & Data in Ram
```

**Figure 2.13 Setting Flash size.**

**NOTE:** If “slc186.cfg” is required, the lowest starting address for the code MUST be 0xC0000, with FLASH 256 defined.

**Step 2.3-8:**

Re-build your target by right-mouse clicking on the ‘.axe’ node and selecting *Build Node*. This will generate a .HEX file with the same name as your target.



```
Preview make
Build node
Locate
```

**Figure 2.14 Building the .HEX file.**

**Step 2.3-9:**

You now have an ACTF downloadable .HEX file ready to program into the ACTF flash. You will find this file in the folder where your project is contained (working directory).

Confirm that you have a current .hex file by verifying modification date and time.

## 2.4 Downloading ACTF .HEX files for User Application

You now have successfully written, debugged, and generated your ACTF downloadable *.HEX* file (section 2.3), You are ready to download it into the ACTF flash. Recall that the ACTF flash comes in two forms: the surface-mounted flash and the DIP ACTF Flash chip, which installs into the controller's ROM socket. Protocol for downloading .HEX files into each is discussed below.

### FLASH-BASED PERFORMANCE

Please note that on some platforms, code will execute slower once it's programmed into Flash ROM. This is typically the case when your board clock rate is 80 MHz or faster (including all 586-based systems). When debugging, code runs out of SRAM, which due to architectural reasons can run quite a bit faster with higher clock rates.

If this is unacceptable, TERN does offer some custom startup options that allow your code to be burnt into Flash ROM, but continue to run out of SRAM at powerup. If you're interested, please contact TERN technical support for assistance.

### 2.4.1 Downloading into Surface-mounted or ROM Flash

This section shows how to program surface-mounted flash. This is EXACTLY the same procedure that was used in downloading the Debug Kernel (Section 2.2).

#### Step 2.4-1:

Once again, open RTLOAD application inside Paradigm C/C++ environment. Power on your TERN controller **without** the Step-2 jumper installed.

#### Step 2.4-2: (Start Download)

*See Step 2.2-1*

#### Step 2.4-3: (Downloading RAM loader utility)

*See Step 2.2-2*

**Note:** Use the following RAM loading files instead of the ones mentioned in *Step 2.2-2*. For a list of exactly which loader file is required for your controller, refer to Appendix E.

#### RAM LOADER FILES (for user .HEX file loading)

##### **V25-based Controllers**

- \tern\v25\rom\lo\_ee128.hex

##### **188ES-based Controllers**

- \tern\186\rom\lo\_ee128.hex (for DIP 128KB Flash)
- \tern\186\rom\lo\_ee512.hex (for DIP 512KB Flash, or **Flashcore-B™** controller)

##### **186ES-based Controllers**

- \tern\186\rom\ae86\l\_29f400.hex (for 20 / 40 / 80 MHz - 186ES/R8820/R1120 CPU's)

##### **186ER-based Controllers**

- 
- \tern\186\rom\re\l\_29f40r.hex

**386-based Controllers (See Appendix D for optional Flash)**

- \tern\386\rom\l\_29f400.hex (for surface-mount flash on **SerialDrive™**)
- \tern\386\rom\lo\_ee128.hex (for DIP 128KB Flash)
- \tern\386\rom\lo\_ee512.hex (for DIP 512KB Flash)

**586-based Controllers**

- \tern\586\rom\l\_29f400.hex

**Step 2.4-4: (Erasing Flash Sectors)**

*See Step 2.2-3*

**Step 2.4-5: (Selecting user .HEX file)**

After 'G04000' has been typed and the user has pressed ENTER, the loader will go through and erase (prepare) all the sectors that are not protected in the Flash.

Once the erase procedure is complete, the user should be a ready statement such as '*Ready to receive TREM???.HEX file at 19200 baud*'.

Press **F8** to choose the necessary file.

It is important to note that RTLOAD is a DOS-based program, and only accepts files/paths in **8.3 naming format**. You may need to rename your application file.

The user must type in the file name (including its path). For example: C:\Tern\186\Samples\AE\LED.HEX

Once the file is typed in, press ENTER.

**Step 2.4-6: (Loading user .HEX file and executing code)**

Recall that depending on how the user defined FLASH in the configuration file, the start address of the application will differ:

- #define FLASH 128 -> 0xE0000
- #define FLASH 256 -> 0xC0000
- #define FLASH 512 -> 0x80000

*See Step 2.2-5 > Download file and jump to address (mentioned above) based on user-configuration file.*

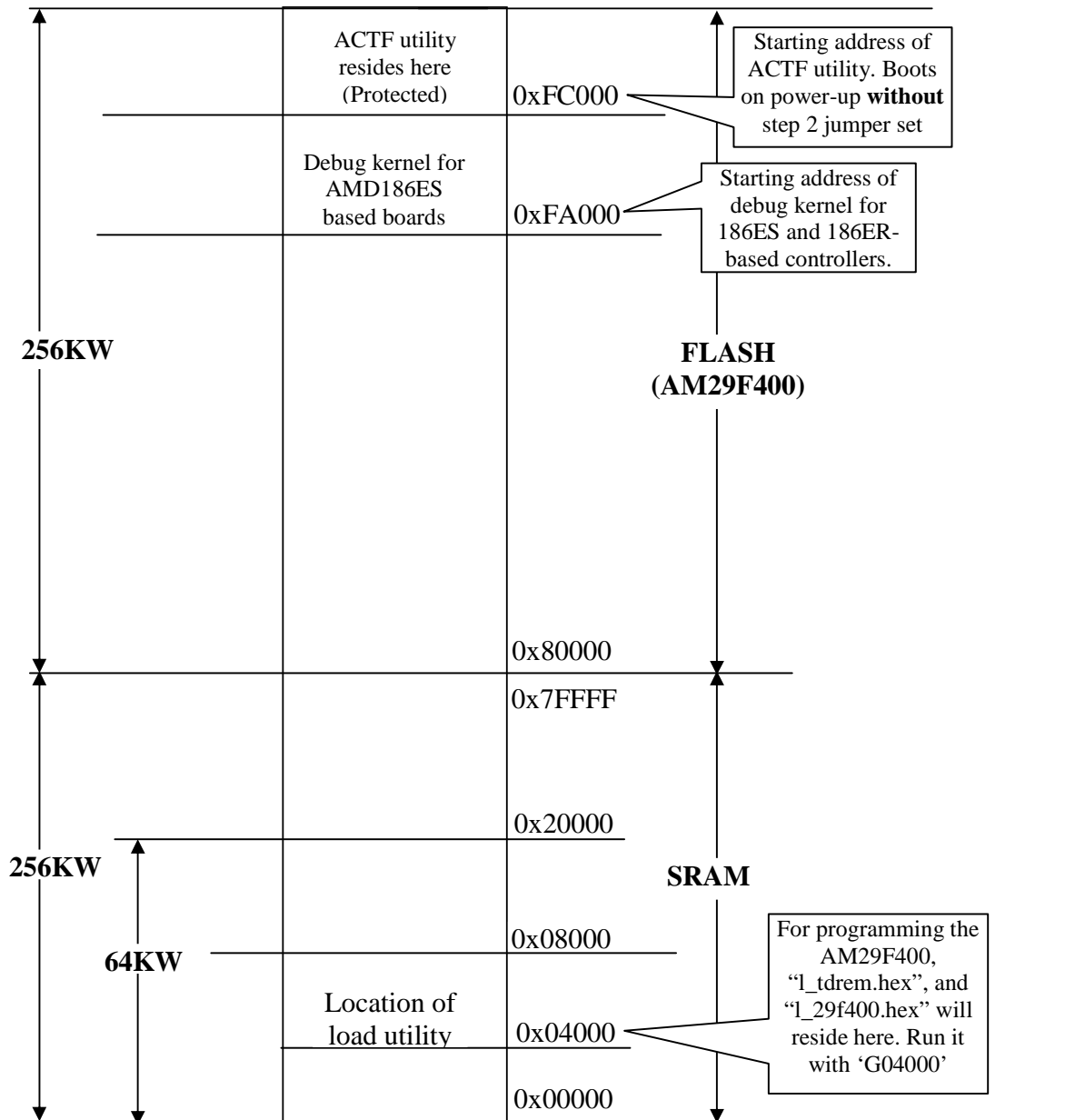
*and then*

*See Step 2.2-6 > Set step-2 jumper.*



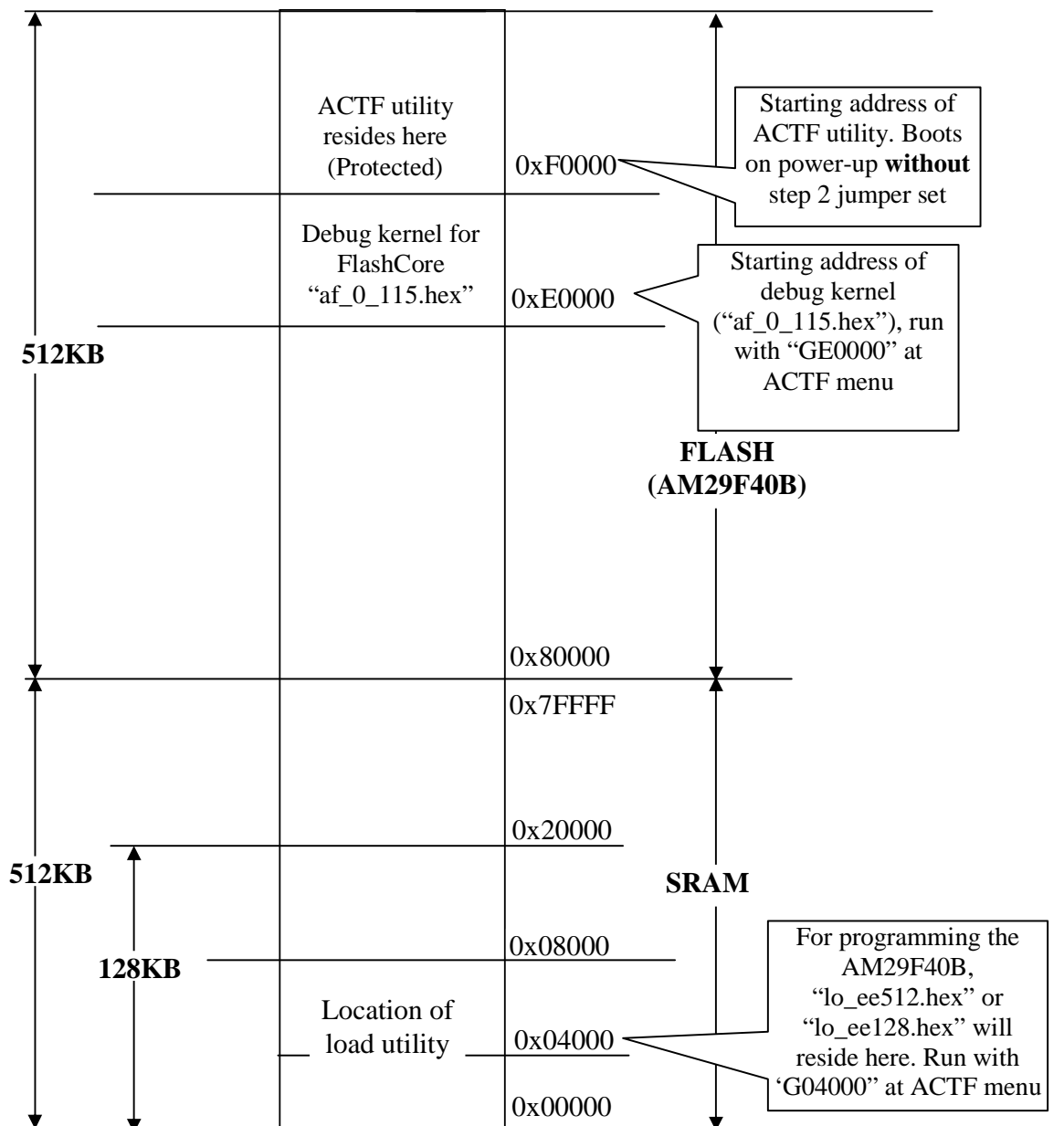
## Appendix A: 16-bit Surfaced-Mounted ACTF Flash

The following memory-mapping layout corresponds to users of the 186ES and 186ER-based controllers. With the step 2 jumper **off**, the ACTF utility will boot at power-up. With the step 2 jumper **set**, the CPU will read the CS:IP jump address from the on-board EEPROM and jump to that address for execution. The jump address can be set by you to point at the debug kernel, for code development, or to the start of your user application in the Flash/SRAM. The jump address can be set at the ACTF menu or by writing to the EEPROM in software (example: c:\tern\186\samples\ae\step2.c).



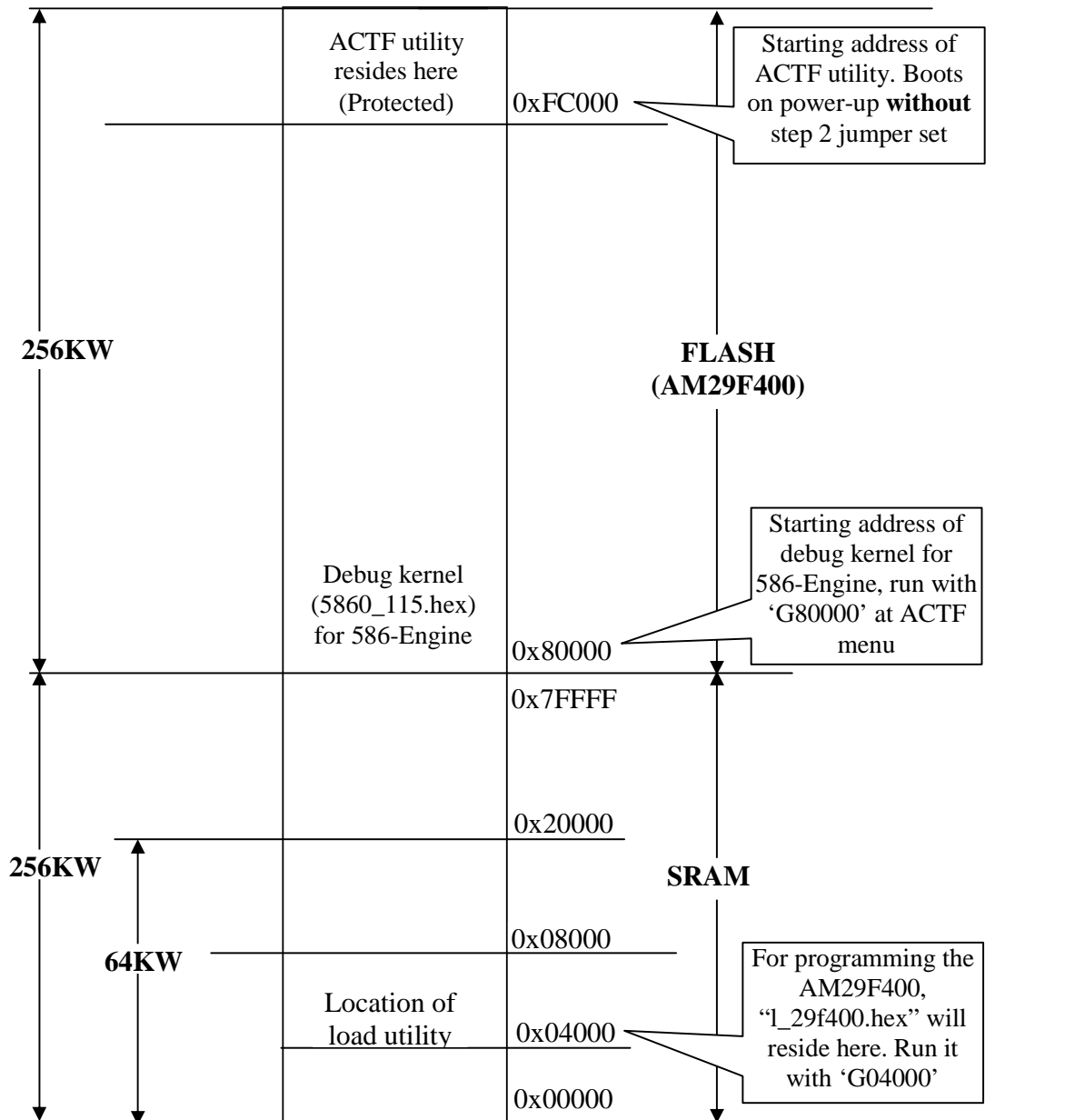
## Appendix B: ROM Mapping (DIP Package)

The following layout corresponds to the memory mapping of **188ES**, **Flashcore-B™**, or **386**-based controllers (using DIP Flash). With the step 2 jumper **off**, the ACTF utility will boot from top protected sector of Flash (AM29F040B). With the step 2 jumper **set**, CPU will read CS:IP jump address from on-board EEPROM and jump to that location for execution. The jump address can be set to 0xE0000 for debugging, or to anywhere in the Flash/SRAM that holds your user application for stand alone operation. Note that the 128KB Flash will start from 0xE0000, not 0x80000.



## Appendix C: 586-Engine Mapping

The following layout corresponds to the memory-mapping of the 586-Engine. With the step 2 jumper off, the ACTF utility will boot from the top protected sector of the Flash (AM29F400). With the step 2 jumper on, the CPU will check for a valid battery back-up. If the battery back-up is validated, CPU will read CS:IP from CMOS SRAM and jump to that location for execution. If no battery back-up, CPU will write 0x80000 to CMOS SRAM and then jump to that location for execution. The CS:IP jump address can be set at the ACTF menu with a ‘G’ command. (See 586-Engine technical manual)



## Appendix D: 16-bit Flash Programming for 386-Based Boards

### Overview

On the TERN i386-Engine-P (IE-P), i386-Engine-M (IE-M), and i386-Drive (ID), an optional 16-bit Flash can be installed for non-volatile storage of completed applications. In past documentation, this was referred to as ‘Step Three’ of the TERN development process.

This Guide explains the process of compiling a completed application into a .HEX file, and then downloading this .HEX file into the on-board Flash for automatic power-up execution. The flash chip (U15) is a surface mounted, 16-bit 256KW blank flash (Am29F400BT).

Before you start following this guide, you should already be able to compile and download your application into the battery-backed SRAM from within the debugger. You should also be able to successfully run your application from the battery backed SRAM in stand-alone mode (Step Two of the development process).

### Minimum Requirements:

TERN Paradigm C++ Development Kit (DV-P Kit)

i386-Engine-P, i386-Engine-M, i386-Drive with the 16-bit Flash (U15, Am29F400).

A **Debug ROM** (IE16\_115, or IE8\_115) should be installed in the 32-pin DIP socket.

### Memory Mapping:

Memory for the 16-bit Flash configuration is shown in figure 1. The Debug ROM is located at the top of the memory map and is the first block to execute after power-on/reset.

Flash memory is mapped starting at address 0x80000.

### Generating a HEX File (Section 2.3 for more details)

For this procedure, refer to the sample project `\tern\386\rom\flash_ie16.ide` as guide for the correct final configuration.

- 1) For your target application (*led\_iep* in the sample), you must first change the configuration file from the one used during debugging. This allows you to generate a .HEX file as output, as well as relocating the file to the appropriate memory addresses for Flash.

Change the configuration node from 386.cfg to actf386.cfg (you can right-click on the .cfg node, and then choose ‘Edit Node Attributes’) to point to a copy of:

<b>IE DEBUG 32K</b>	0xFFFFF
	0xF8000
<b>16-bit Flash 256K</b>	0xBFFFF
	0x80000
<b>SRAM 512K</b>	0x7FFFF
	0x00000

**Figure 1: Memory mapping configuration**

`\tern\386\config\actf386.cfg`.

- 2) In this new configuration file, make sure the correct options are selected for your board. Double-check the BOARD type, as well as the Flash size (should be 512).
- 3) Right-click on the `.axe` node (`led_iep.axe`) and select 'Target Expert'. Change the 'Target Connection' option to: '**No Target/ROM**'.
- 4) Right-click on the `.axe` node again, and choose 'Build Node'. A .HEX file named after your target will be created in your working (or output) directory (`led_iep.hex`, for this sample).

### **Downloading a HEX file into the 16-bit Flash**

*NOTE: Be sure that the 'Step 2' address is setup correctly to 0x08000. If you are not sure, run `step2.c` in the debugger for your controller. A `step2` target is made available for you in `flash_ie16.ide`; just download and run it.*

The downloading process requires an intermediate loading program, `l_f16.c`, to prepare the 16-bit Flash, and to receive the final HEX file. This file is located in `C:\TERN\386\ROM\`.

Download the `l_f16.axe` application into your controller using the debugger. After the debugger has downloaded the program, terminate the debug session (*Debug->Terminate Debug Session*) immediately **without running**; `l_f16` tries to use the serial port 0, and will crash your debugger.

Start a terminal program (either *Hyperterminal*, or *Tools->RTLOAD* within the Paradigm environment), and configure it for 19200 baud, no parity, 8 bit, 1 stop bit operation. See **chapters 1 & 2** for RTLOAD usage.

Place the red "Step 2 jumper" on the board (refer to your controller manual if you're not sure where this is), and then reset the controller with the STEP2 jumper installed.



## Appendix E: BAUD Rates/Loader Files

Table E.1 lists the BAUD rates that Tern controllers use to communicate, based on system frequency. This frequency is dependant on the Processor used as well as the crystal. Default rates are in **bold**. The third column indicates the loader file (into SRAM) needed prior to downloading the user application (.HEX) into Flash (Section 2.4, step 2.4-3).

Controller	Serial Port 0 BAUD Rate	Loader File
586D	<b>19200 (133MHz)</b>	\\tern\586\rom\l_29f400.hex
586E	<b>19200 (133MHz)</b>	\\tern\586\rom\l_29f400.hex
586P	<b>19200 (133MHz)</b>	\\tern\586\rom\l_29f400.hex
A104*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
A104S*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
AC*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
AC86	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
AE*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
AE86	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
AE86D	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
AE86P	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
AEP*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
BBA*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
CEye	<b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
EE	9600 (80MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
EL	9600 (80MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
FB**	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
FN	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
GE	<b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
ID*	<b>19200 (33MHz)</b>	\\tern\386\rom\lo_ee128.hex OR lo_ee512.hex
IE*	<b>19200 (33MHz)</b>	\\tern\386\rom\lo_ee128.hex OR lo_ee512.hex
IEL	<b>9600 (32MHz)</b> or 19200 (64MHz)	\\tern\386\rom\lo_ee128.hex OR lo_ee512.hex
IEM*	<b>19200 (33MHz)</b>	\\tern\386\rom\lo_ee128.hex OR lo_ee512.hex
IEP*	<b>19200 (33MHz)</b>	\\tern\386\rom\lo_ee128.hex OR lo_ee512.hex
MD88*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
RA	<b>19200 (80MHz)</b>	\\tern\186\rom\re\l_29f40r.hex
RB	<b>19200 (80MHz)</b>	\\tern\186\rom\re\l_29f40r.hex
RD	<b>19200 (80MHz)</b>	\\tern\186\rom\re\l_29f40r.hex
RE	<b>19200 (80MHz)</b>	\\tern\186\rom\re\l_29f40r.hex
RM	<b>19200 (80MHz)</b>	\\tern\186\rom\re\l_29f40r.hex
RL	<b>19200 (80MHz)</b>	\\tern\186\rom\re\l_29f40r.hex
SC	<b>19200 (80MHz)</b>	\\tern\186\rom\re\l_29f40r.hex
SCA	<b>9600 (80MHz)</b> or 19200 (40MHz)	\\tern\186\rom\ae86\l_29f400.hex
SD	<b>9600 (32MHz)</b> or 19200 (64MHz)	\\tern\386\rom\l_29f400.hex
SL*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
ST	<b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
ST1	<b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex
TD20/40*	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\lo_ee128.hex OR lo_ee512.hex
TD86	9600 (20MHz) or <b>19200 (40MHz)</b>	\\tern\186\rom\ae86\l_29f400.hex

**Table E.1 BAUD Rate and Loader File by Controller**

\* Boards using ACTF ROM Flash (DIP Package).

\*\* FB contains surface-mount flash, but uses same loader files as 188ES ROM Flash boards.