SHDesigns Ethernet Downloader for
Z-World Rabbit Boards

Date: Saturday, November 01, 2003

1.0 INTRODUCTION

The Ethernet Download Utility allows Rabbit-based TCP/IP boards to upgrade the FLASH code in the field without the serial programming cable. This functionality consists of 3 parts:

1. A small library included in a user program: UDPDOWNL.LIB.
2. A small RAM-based program that downloads the new code and programs FLASH.
3. The PC Utility to find and program boards.

Unlike other solutions for network downloading, this implementation requires little changes to a users code. There are no hardware changes or library changes for most Rabbit products..

2.0 HOW IT WORKS

The Z-World solutions impose restrictions on the hardware and software. Even their serial solutions require changes to the libraries and dividing FLASH in half.

The main problem is their network libraries. There is no way to pare them down small enough to fit in RAM in a running system. If a program could be built small enough, it could be downloaded to xmem, copied to root and run.

This solution resolves this problem by writing a stripped-down network stack. UDP sockets are simple to implement. To support UDP only the following is needed:

1. Basic IP support (source and destination headers.).
2. ARP support (reply only).
3. UDP packet support.

The stack also provides ICMP support to allow ‘Pings’ for debugging.

Since the downloader operates as a client and not as a server, ARP is simple. It only needs to respond to a request. It does not need to use ARP to find the PC application MAC address. It just replies back to the MAC address that the packet came from.

Packet buffering is also simplified. The stack never generates a packet by itself. It just receives a packet, processes it, and then optionally modifies it and sends it back.

Source network routing is ignored. The stack assumes that it will only have to reply back to a local LAN IP. Actually, the stack will work through routers and sub nets, but the “find board” function only works on the local segment as broadcasts are not passed through routers.

Given a BIOS of about 11k, the entire program is less than 23k.

Now that we have a small RAM program, the rest is fairly easy to implement. The library routine just has to allocate a buffer in xmem to store the program. After the RAM loader code is downloaded and a 'RUN' command is received, it copies the program to root RAM, re-maps RAM to 0, and reboots to the new code.

The library also has additional functions. One is to preserve the IP address of the board. Since the entire board is reset, the network needs to be reinitialized. The IP is passed in RAM to the new program. An additional function of the library is to respond to "Query" commands. The query command is a broadcast on the LAN for all board to identify themselves. Each board that supports the download function will reply with a user-supplied string. This allows the PC utility to identify boards by name. The user can include the current version in the string to allow the PC user to identify boards that need upgrading.

The library code is quite small, about 1k of code and about a MTU (system maximum packet size) of root RAM. It also requires one UDP socket for communications.

The library uses no extra xmem memory until a request to download is received. It will then allocate enough xmem to store the program. The user program is notified that a request has been received and so it can free up xmem if needed. Only programs that use all of xmem would need to do this.

The library and the download utility report their status to the PC program. They will report that they need the Ram loader or are running it. Also, memory allocation and code blocks are acknowledged.

3.0 IMPLEMENTING THE LIBRARY

3.1 Dynamic C Changes

udpdownl.lib and udpdnld.h needs to be added to the DC compiler LIB.DIR file for the compiler to locate the library. I.e. if the files are in "C:\program files\rabbit\mylibs" add the following lines to the LIB.DIR file in the main compiler directory:

C:\program files\rabbit\mylibs\udpdownl.lib
C:\program files\rabbit\mylibs\udpdnld.h

There are no changes required for the Dynamic C libraries. This library has been compiled with version 7.25 of Dynamic C. It should work with any 7.2x compiler or later.

3.2 User Program Changes

In the user's program the changes are simple. Just add the following code:

You must define at least 1 additional UDP socket and buffer:
#define UDP_SOCKETS 1   // allow enough for downloader and DHCP
#define MAX_UDP_SOCKET_BUFFERS 1

Note: DHCP will need additional UDP sockets and buffers

After the “#use dcrtcp.lib” add the following line:

#use udpdownl.lib

In the main code add the following functions:

After the “sock_init()” call add:

UDPDL_Init(string)

Where “string” is a string to report back to the PC. An example would be:

UDPDL_Init(“XYZ Corp. Controller Version 1.0.3a”);

Then when the “search boards” is done in the PC app. It will show this string on the screen.

If NULL is passed as the string, the board will report: "Rabbit Board UDP Download". UDPDL_Init() will return 0 if success. The only reason it would fail is if there are not enough UDP buffers or sockets.

Note: if the IP address of the board is changed after UDPDL_Init() is called, you will have to call UDPDL_Init() again to update the UDP socket..

In the main loop of the program, the function “UDPDL_Tick()” needs to be called periodically. Normally this is where the tcp_tick() is called.

UDP_Tick() normally returns 0. When the PC requests a buffer for the RAM code, it will return 1. If the user program has allocated all of xmem, it can use this flag to know when it needs to free some xmem for the downloaded program. Normally applications do not use all of xmem and they can ignore this.

Once UDP_Tick() has returned 1, the user program will soon be stopped. The user program may need to shutdown features or notify other apps that it will be shutting down.

If you would like to maximize the speed of the RAM code download, implement UDPDL_Tick() as follows:

(in main() poll loop)
```
        if (UDPDL_Tick())
```

```
{
        // example, shut down interrupts
        // stop my timer interrupt
        //TimerBUninit(); // stop timer ints
        //xrelease(my_big_xmem_buffer,buf_sz);
        while (1)
                UDPDL_Tick();         // do nothing else!
}
```

As soon as UDPDL_Tick() returns 1, this loop will do nothing but process the download commands.

3.3 Including the RAM loader in FLASH

As of 3/24/2003, the library can include the RAM loader in FLASH. This requires about 22k more of FLASH memory. Note: Version 1.1a of the Download program is required. Earlier versions will download the RAM loader anyway.

To include the RAM loader add the following define before the “#use UDPDOWNL.LIB line:

```
#define UDPDL_LOADER "x:\\path\to\\file\\UDPDL2200.bin"
```

The UDPDL_LOADER define is used in the library as follows:

```
#ximport UDPDL_LOADER _udp_dl_loader
```

The UDPDL_LOADER define is the file name to ximport. Dynamic C has problems with paths for ximport. Sometimes it will work with just the file name in quotes. But it often does not use the right directory. It is usually better to use the full path to the loader.

This file must not be encrypted. You can use the EncryptBin.exe utility to create an decrypted version.

3.4 The Sample program

In the Sample directory is a file called DLTest.c. This is basically a program that does nothing except allow a user program to be loaded. It is a good idea to test with the sample first to make sure the environment is configured properly.

Compile the program to FLASH and run it. Note: do not try to run the downloader under the debugger. The DC IDE will reset the board when the RAM loader tries to run and report a communications error.

Download to flash and reset the board then run the PC utility. You should be able to download one of your regular .bin files to the board. This will then run your FLASH bin file. If your .bin file does not have the library functions included, you will not be able to download again.

Make the changes to your source as shown in the previous sections. Then once you program it to flash, any new version can be downloaded via the PC utility.

4.0 RAM DOWNLOAD PROGRAM

The RAM program is supplied pre-compiled. Versions are available for most Rabbit boards. The 2200 module will work with Rabbit 2k-based boards. This will work on all of the 22xx modules that use either the first 256k flash or both 256k flash chips for code or data.

SHDesigns will provide the binary for any of the Rabbit-based boards. Source is not provided. This is mainly to prevent supporting many different loaders. If a custom downloader is required, contact SHDesigns at: rabbit@shdesigns.org.

Ram Loaders are named as follows:

DL-board#.bin
Where 'board#' is the Z-World board number. I.e DL-RCM2200.bin for the RCM2200.

The complete list of loaders is found in the loaders directory. An "encrypted" and "Unencrypted" directories have the latest loaders.

As of 3/23/2003, online versions of these files are also encrypted.

The "protected" directory has new RAM loaders that will survive a reboot or power loss. The protected loaders have a file name of PDL-board#.bin

5.0 ENCRYPTION

The download utility version 1.1 and higher supports encrypted .bin files. This prevents users from using the bin files with any other downloader.

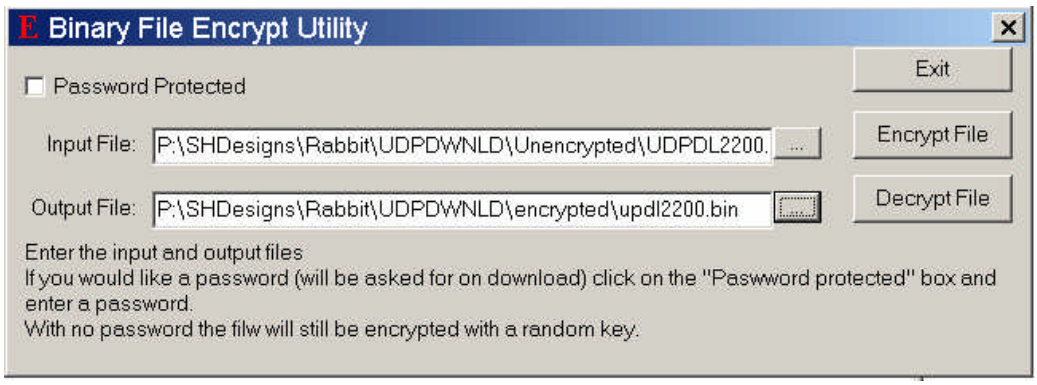
Encryption is done as follows:

1. A small header is added to the file
2. If a password is used, a encryption key is generated. If no password is used, a pre-defined key is used.
3. A second random key is generated.
4. This second key is encrypted with the password key.
5. The .bin file is appended and encrypted with the second key.

The password is not saved in the header. It is used to generate a unique key. There is no way to recover the password from the key. Thus, there is no information in the header on the size of the password. Unlike other encryption methods, the strength of the encryption does not depend on the length of the password.

The encryption keys are 96-bits long. The header starts with the string "Encrypted program file." This identifies the file as encrypted. If a user types the file from a command prompt, they will see only this string.

A utility called EncryptBin.exe can be used to encrypt user files.



If the "Password Protected" check box, a password can be entered in the field to the right.

The input and output file fields can be entered or the "..." buttons on each can be used to browse for the files.

Pressing "Encrypt File" will encrypt the file. The status area in the bottom of the dialog will indicate a successful conversion.

Note: There is a difference between having no password and an empty password. If the "Password Protected" check box is not checked, the user will not be prompted for a password. If the box is checked and no password is entered, the user will still be prompted for a password and the loader will accept an empty password.

The "Decrypt file" button will decrypt a file, no password is needed.

Since this utility can decrypt a file with no password, it should not be distributed to end users.

6.0 Password Protecting the PC Utility

The 1.4e and later versions of the program allows it to be password protected. This was asked for by one of my clients to prevent end users from running the program.

The only way to set the password is to edit one of the resources in the file. I use “resource hacker” that can be found at: http://www.users.on.net/johnson/resourcehacker/
In the resources edit the following string entry:

String Table --> 7 --> 1033 --> 102

It should have a string of “-None-”. Change this to the required password and the program will not run until this password is entered