SHDesigns Ethernet Downloader for Z-World Rabbit Boards and the Softools Compiler

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.
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.

Note: program files compiled with Dynamic C are not supported. The downloader will try to preserve the Softools ID block that is not compatible with the DC-generated code.

License: This library is licensed for a single user. Support is provided for at least one year after purchase. Upgrades will be available online and are free.

Support:
There is a message board for the SHDesigns libraries at: http://rabbit.shdesigns.org
Email support is provided at rabbit@shdesigns.org

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 on single-flash boards.

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.

2.1 The Custom UDP/IP Network Stack

This solution resolves the size 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. (source, dest port and CRC.)

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 or router gateways. 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 modifies it and sends it back.

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. When routers are used, the stack will automatically send the packet back to the router it came from; no additional code is needed.

The entire RAM program is usually less than 16k. Compare that to a minimal TCP/IP program with the Z-World stack that requires 60k or so.

2.2 The Library

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. If the RAM loader is included in flash, no xmem is needed. After the RAM loader code is downloaded and a ‘RUN’ command is received, it copies the program to the start of RAM, re-maps RAM to 0, and reboots to the new code. The library “patches” the RAM loader image with the current location of the RAM loader, the current IP address and the existing ID block (MAC address.) This allows the RAM loader to maintain the current IP settings of the running code.

An additional function of the library is to respond to “Query” commands. The query command is a UDP broadcast on the LAN for all board to identify themselves. Each board that supports the download function will reply with a user-supplied ID 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 PC utility can also automatically update boards based on their ID strings. Since UDP broadcasts are not forwarded through routers, only boards on the local LAN segment will respond to the query. The utility has the feature to query (ping) boards on other segments or across the Internet by a specific IP address or host name.

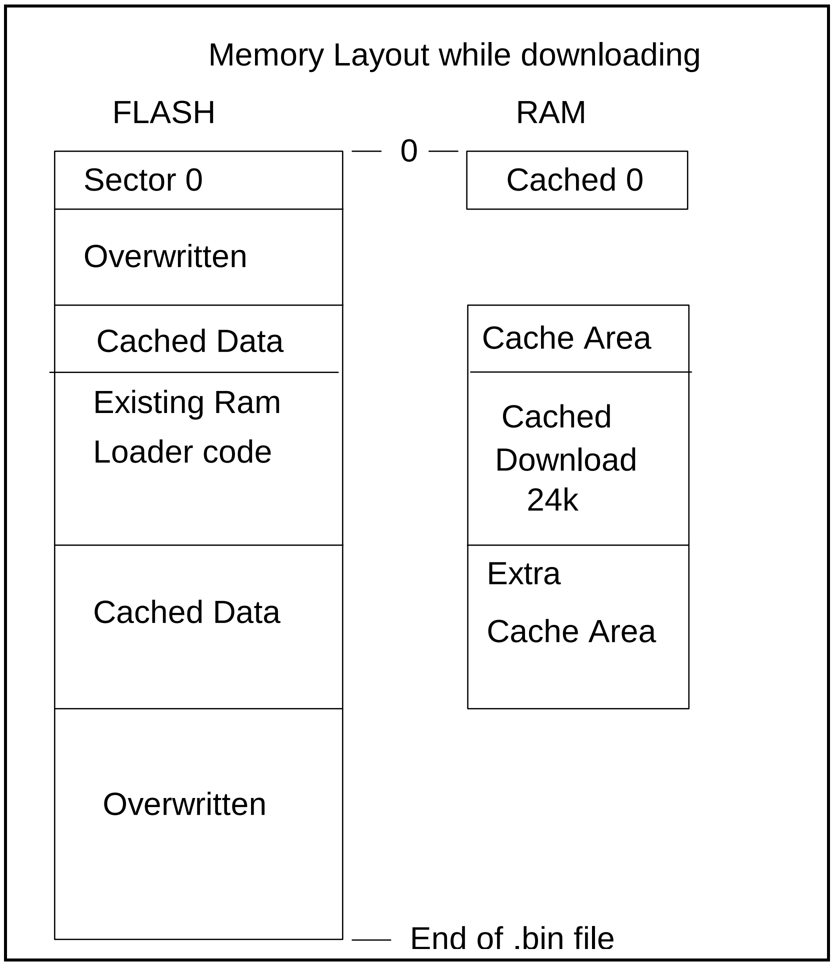
The library code is quite small, about 1k of far code and about 600 bytes of RAM. It also requires one UDP socket for communications.

The library uses no extra xmem memory until a request to download the RAM loader 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 and do not include the RAM loader in FLASH would need to have free xmem (about 16-18k depending on loader.)

The library reports its status to the PC program. It will report that it needs the Ram loader or has it included in FLASH. All memory allocation and download blocks are acknowledged.

2.3 Failure Protection

The RAM loader supports protection from power loss or reset while downloading. This is done as follows:



When the downloader is run, the Ram loader is copied from FLASH to RAM and then run. The existing IP and location in FLASH is passed to this program. Note: The FLASH is not erased and only areas specified as part of the bin file are overwritten. This assures that any user FLASH areas are preserved (i.e. a file system.).

The Downloader starts by writing a small stub (150 bytes) to the first sector. This is the fatal error recovery. This code will reload the existing Ram loader from FLASH if the board is reset. The existing RAM loader in FLASH and sector 0 are then only rewritten after all of the new bin file is downloaded and all other areas of FLASH are written.

As download starts, the new data for sector 0 is saved in RAM to protect the recovery boot code. Additional sectors are written to FLASH.

The “Cached download” area starts below the existing Ram loader in FLASH. This area needs to be protected as long as possible. This cache is larger than the Ram loader code (80k for 128k RAM.) This extra cache area speeds up downloading as FLASH is not being written. Areas shown as “Overwritten” are programmed as they are downloaded.

The cached area will normally start just past the boot sector. It is moved upward until it includes the RAM loader. It will cache as much memory as possible. On 512k RAM systems, it will likely cache all of the .bin file.

The download process is as follows:

1. The boot sector is overwritten with the recovery code pointing to the existing RAM loader image in FLASH.
2. Data between the boot sector and the start of the cache is written as it is downloaded.
3. Downloaded data starting at the cache area is stored in xmem. As much data is cached in memory as possible (approx 80k for 128k boards, 208k for 256k RAM, 464k for 512k RAM.) *
4. Downloaded past the cache is written directly to FLASH.
5. The PC Utility sends a “reboot” command indicating download complete.
6. The cached data excluding the RAM loader area is written to flash. This leaves only the boot sector and the RAM loader area to be updated.
7. The Existing Ram loader is then overwritten in a single write. If this is the same, then no write occurs.
8. The sector 0 is then written.
9. The loader then reports back to the PC utility “Board is rebooting.”

* if the Protect RAM option, then only 24k for the RAM loader image is cached at the top of xmem.

There are a few points that are not fully recoverable from a power loss or reset:

1. At step 1. This is a single sector write. This should only take a few milliseconds so the chance of failure is small.
2. At step 7. This may not be recoverable if the new code has the RAM loader at a different location. The RAM loader image will likely be in the same location and can be force toa specific address.
3. At step 8. This single-sector write is also only a few milliseconds.
4. Any flash that is the same as the new .bin file is not rewritten to save flash rewrites and also to prevent failure.

Chance of failure is very small. This has been tested repeatedly by manually hitting reset during the download. A failure was not seen. The reset would have to occur at the exact moment to cause problems. Everything happens so fast, the windows are small.

The longest single step is 6. This may be a few seconds on a large .bin file. Failure here is recoverable. Step 7 takes less than a second (exact times depend on FLASH type and age.) Steps 1 and 8 are usually less than 100ms.

During steps 2 through 5, a reset or power loss is always recoverable. If this happens, PC utility will report “Block not acknowledged” and after a few retries abort the download. Once the board is rebooted, it will run the RAM loader. In this case the board ID will be “Softools Ram Loader.” The PC Utility should find it in a the search. The download can be restarted. In this case, the process restarts at step 2.

The RAM loader is stored in its own segment called RAM_LOADER. If this is linked to a specific address, then the chance of recovery is even better. Then the only chance of failure will be a power loss during the overwrite of sector 0.

If possible, the power supply should have a few seconds of reserve power if power is lost. This reduces the chance of failure significantly (already a low probability.)

Full protection:

A fully protected downloader is possible, but requires more changes to the user code. This will requires a custom cstart.asm and the RAM loader in a specific area of flash. This may be added in the future. However the existing protection is very good.

3.0 Using the Library

3.1 User Program Source Changes

In the user’s program the changes are simple.

1. Include the header file:

```
#include "UDPDOWNL.h"
```

2. Make sure there are enough UDP sockets and buffers.

You must define at least 1 additional UDP socket and buffer. This is not clear how to set the Ethernet library settings in the Softools documentation. Below are the settings used in the demo:

```
const char MAX_UDP_SOCKET_BUFFERS = 4; // default is 4,  1 for DHCP, 1 for DLM
const char MAX_TCP_SOCKET_BUFFERS = 0; // default is 4, demo program uses no tcp
sockets
const unsigned TCP_BUF_SIZE= 0;          // demo uses no tcp sockets, default is 4096
const unsigned UDP_BUF_SIZE= 2048;       // default is 4096
const char MAX_SOCKETS = 3; // demo only uses 1, DHCP uses one
```

The Ethernet library default settings are fine. The above reduces the required xmem needed at run time. These setting are for the demo, adjust them for your application.

3. Add init and tick calls:

In the main code add the following function calls:

After “sock_init() and ipconfig()” calls add:

UDPDL_Init(string,int preserve)

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

UDPDL_Init(“XYZ Corp. Controller Version 1.0.3a”,0);

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

The “preserve” parameter should be 1 to try to preserve xmem contents. Use 0 if you don’t care if all memory is modified. When set to 1, 24k of memory at the end of memory is used (xalloc() allocates from the beginning of xmem.) The first 40 k of RAM (0 to 0x9000) will always be overwritten.

Note: The string must be static. The library only stores the address of the string. If this string is changed, the ID string may be invalid.

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.

UDPDL_Init() can be called multiple times. This allows the program to change the ID string dynamically.

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. You should reinitialize the lib after any sethostid() or ifconfig() is done on the ethernet interface.

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. It should not be called from an interrupt routine.

UDP_Tick() normally returns 0. When the PC requests a buffer to download 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. Note, UDPDL_Tick() may return 1 more than once. If the RAM loader is included in flash, then no xmem is used unless the user forces a download of the RAM loader.

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.

Note: when UDPDL_Tick() returns 1, the user program should call it again within a second. Otherwise the PC Utility will time out as it will not see the acknowledge.

If the RAM loader is included in FLASH, this call will normally not return 1 as there is no request to allocate xmem.

UDPDL_Tick() will return 2 when the PC Utility sends a "Run RAM loader command". On the next call, the library will run the RAM loader and not return.

3.2 Linking the Library

There are several libraries for the downloader. The main difference is the RAM loader:

DL-NoLoader.lib	- Library with no loader (must be downloaded from PC utility.)
DL-Realtek.lib	- Library with Realtek loader in FLASH
DL-Asix.lib	- Library with Asix loader in FLASH
DL-DevKit.lib	- Library with TCP/IP Development kit loader in FLASH
DL-SMSC.lib	- Library with SMSC loader in FLASH
DL-Realtek-D.lib	- Library with Realtek loader in FLASH, debug option *
DL-usingAll.lib	- Library with RAM loader that supports all interfaces in FLASH

* This version will output debug information on serial port A at 115200 baud.

These files are in the lib directory. Just add the desired library to your projects.

The "NoLoader" lib has no RAM loader included. This is for systems with limited FLASH space. This will save from 16 to 22k, depending on the Ethernet driver used. The RAM loader will be need to be downloaded from the PC Utility. Note: when the RAM loader is downloaded from the PC Utility, protection is disabled.

Specify RAM loader location in FLASH:

If the RAM loader is included in the lib, a new segment is created called RAM_LOADER. This is usually located to follow the FARCODE segment. If you specify it follows the FARCODE segment and uses a 4k alignment (0x1000), it will move as needed, but most new versions of the code will have it in the same place. If desired, force it to a specific physical address.

3.3 The Sample program

In the DLMTest directory is a file called DLTest.c. There is a project file also.. 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.

The sample program defaults to using DHCP. Link with the regular library if you need static IP settings. For static or fallback settings, edit the MY_xxx defines in the top of the file.

Compile the program to FLASH and run it. Note: do not try to run the downloader under the WinIDE or the program cable connected. The WinIDE 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 for your application as shown in the previous sections. Then once you program it to flash, any new version can be downloaded via the PC utility.

The demo will output data out port A when it is run. Use the diag connector of the programming cable and a terminal program set to 115200 baud, 8-bit, no parity, no handshaking.

```
Ready to download over me!      - Startup message
EGSIZE=a0                       - Various debug messages
MMIDR=0
MECR=0
MTCR=76
MB0CR=c8
MB1CR=c8
MB2CR=c5
MB3CR=c5
GCSRIO=9
STACKSEG=76
DATASEG=0
xmem=7688a                      - free xmem in hex
Download request pending!       - PC has requested to download RAM loader
Run Ram Loader!                 - Last message before rebooting to RAM loader.
```

4.0 RAM DOWNLOAD PROGRAM

The RAM program is supplied pre-compiled. Versions are available for most Rabbit boards. SHDesigns will provide binary libraries and RAM loaders for any of the Rabbit-based boards. Source is not included. This is mainly to prevent supporting many different loaders. If a custom downloader is required, contact SHDesigns at: rabbit@shdesigns.org.

Ram Loaders are included in the loaders directory and are named as follows:

DL-ASIX.bin	- RAM loader for the ASIX chip set
DL-DEVKIT.bin	- RAM loader for the TCP/IP development board
DL-Realtek.bin	- RAM loader for the Realtek chip set
DL-SMSC.bin	- RAM loader for the SMSC chip set
DL-Realtek-D.bin	- RAM loader for the Realtek chip set with debug output
DL-usingAll.bin	- RAM loader for the All chip sets

Note: These file are encrypted and can not be programmed using FlashIt (they are only designed to run in RAM.) Most users will not need the RAM loader files themselves as they are usually included as part of the library.

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. Users with off-the-shelf boards should encrypt their bin files. This prevents users from buying the boards direct and copying them.

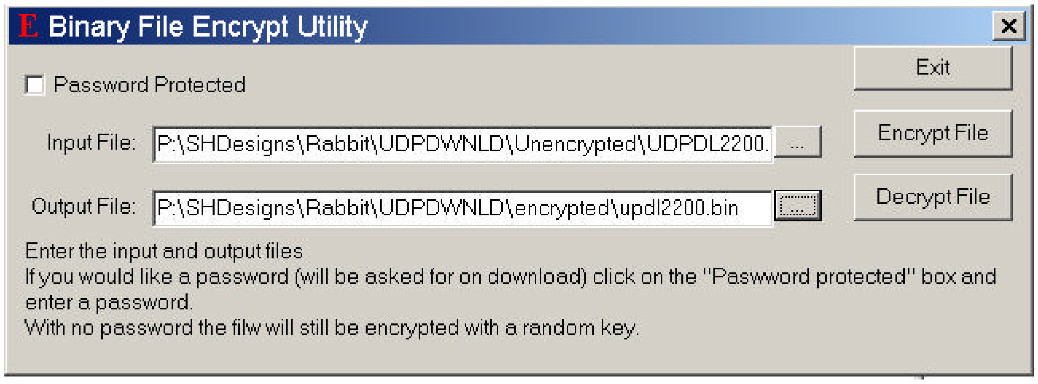
Encryption is done as follows:

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. Since two keys are used, the encryption is even longer than 96 bits.

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



P:\SHDesigns\Rabbit\UDPDWNLD\Unencrypted\UDPDL2200.

P:\SHDesigns\Rabbit\UDPDWNLD\encrypted\updl2200.bin

Enter the input and output files
If you would like a password (will be asked for on download) click on the "Password protected" box and enter a password.
With no password the filw will still be encrypted with a random key.

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.

The utility supports use in batch files with command-line parameters. The format is:

```
C:\path>encryptbin infile outfile
```

This will incrypt infile and save it as outfile.

6.0 Password Protecting the PC Utility

The program allows password protection. 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