SHDesigns Serial Downloader for
Z-World Rabbit Boards and
the Softools C compiler

Date: Wednesday, May 12, 2004

1.0 INTRODUCTION

The Serial Download Utility allows Rabbit-based 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 serial downloading, this implementation requires little changes to a users code. There are no hardware changes or library changes.

There are three modes the software may be used:
1. Direct serial connection
2. Modem on Rabbit board that answers incoming call from Serial Download Program
3. Modem on Rabbit board that calls into the Serial Download Program

2.0 HOW IT WORKS

The Z-World solutions impose restrictions on the hardware and software. Their serial solutions require changes to the libraries and dividing FLASH in half. Rather than divide the FLASH in half between the downloader and the user program, this library uses a very small stand-alone program to download code to FLASH.

2.1 Implementation

The library uses a small RAM-based loader. This loader is small (16k.) The library routine has to allocate a buffer in xmem to store this 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 RAM loader may be downloaded from the PC or included in FLASH. When included in FLASH, no xmem is needed. An additional 16k of FLASH space is used.

The library also has additional functions. One is to preserve the serial parameters. Since the entire board is reset, the serial port and modem need to be reinitialized. These parameters are passed in RAM to the new program.

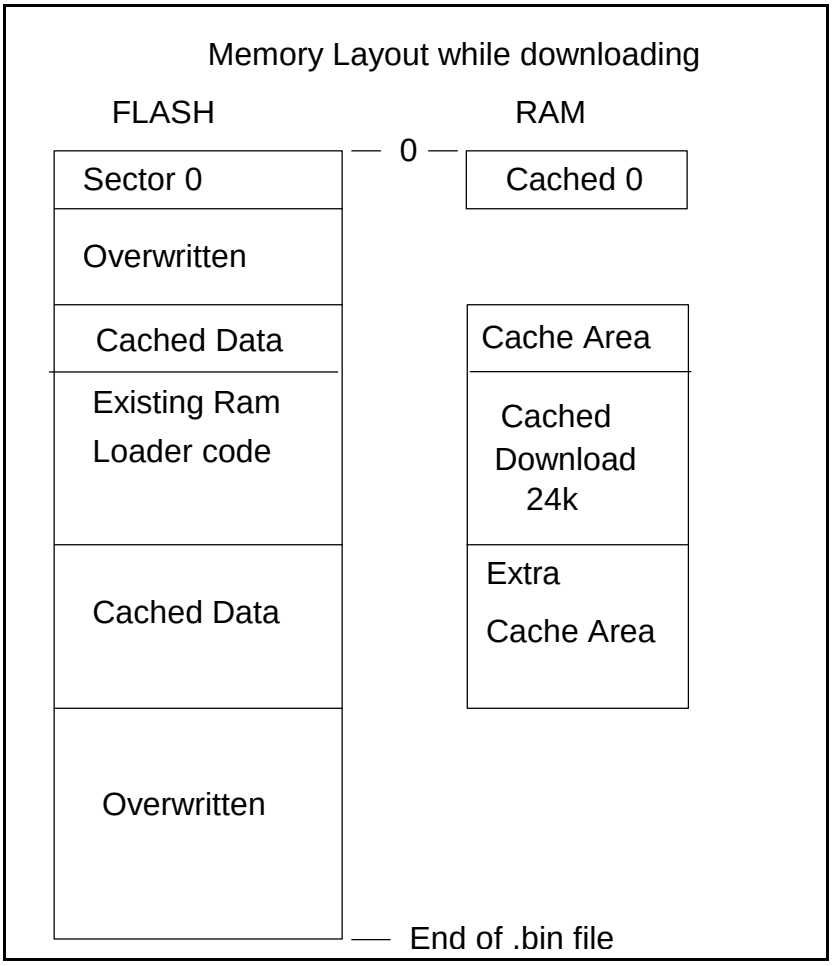
One function of the library is to respond to "Query" commands. The query command is a send by the PC utility to ask the board to identify itself. Each board that supports the download function will reply with a user-supplied string. This allows the PC utility to identify boards by name. The user can include the current version in the string to allow the PC user to identify boards that need upgrading.

The library uses no extra xmem memory until a request to download is received. It will then allocate enough xmem to store the RAM loader 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. If the RAM loader is included in FLASH, no extra xmem is needed.

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

2.2 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 parameters are 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.).

When the first block of the user program is downloaded, the Downloader writes 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. The RAM loader in FLASH is also patched with the current parameters. This allows full recovery.

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 or saved in the cache.

The "Cached download" area starts just 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 typical 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. The RAM loader in FLASH is patched with the current parameters.
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 data 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 NO_DL_CACHE option is used, 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 forced to a 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 help prevent failure.

Chance of failure is very small. This has been tested repeatedly by manually hitting reset during the download. 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. 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 IMPLEMENTING THE LIBRARY

3.1 User Program Changes

In the user's program the changes depend on how the serial loader is to be used. Each of the following sections describe the changes.

Note: The download process can not be run under the WinIDE debugger. Since the RAM loader gets copied over the debug area, WinIDE will reset the board and report a loss of communications.

3.1.1 Dedicated direct connection

In this case, the serial port is dedicated for serial download only.

Include the “serdownl.h” file in your source.

In your main() function add the following:

A call to SerDL_Init() as follows:

SerDL_Init(int port,long baud,“string”);

where “string” is the ID string to report to the PC. I.e. “ZYX Corp controller 1.0”. This string will appear in the board list in the PC utility.

The port is the serial port to use A=0, B=1, C=2 etc. There are two flags that can be or’ed with the port:
USE_SERIAL_D - use I/O port D pins
NO_DL_CACHE - Do not cache downloaded code (will try to preserve xmem.)

In your main loop of your program, call SerDL_Tick(). This function handles the download function. This should be called a few times a second for the PC utility to communicate properly.

Note: SerDL_Tick() will return non-0 when the PC utility starts a download. The user code may increase the polling to improve the response of the PC utility.

3.1.2 Shared serial connection

In this case, the serial port is used for some other function until the user selects download. This requires that the loader be included in flash (see section 3.2.)

Include the library include file in your source:

#include “serdownl.h”

When your user interface selects a download, call the following function:

SerLoad(int use_modem,int port,long int baud_rate,char * id_string,NULL,NULL);

Where the parameters are:

use_modem - first parameter of 0 specifies direct connection

The port is the serial port to use A=0, B=1, C=2 etc. There are two flags that can be or’ed with the port:

USE_SERIAL_D - use I/O port D pins
NO_DL_CACHE - Do not cache downloaded code (will try to preserve xmem.)

baud - long integer specifying the baud rate.
string - An ascii string. This string will be reported to the PC utility.

The SerLoad() function will not return.

3.1.3 Modem serial loader

The modem mode can either dial out to a PC running the download utility or answer a call from one. In this mode, the RAM loader must be included in FLASH (see section 3.2).

Include the library include in your source:

#include “serdownl.h”

Note: the serdlinc.h parameters are not used.

When your user interface selects a download, call the following function:

SerLoad(use_modem,port,baud_rate, id_string, modem_init,phone_num);

Where:
mode = 1 modem answer mode, 2= modem dial out

baud = baud rate

The port is the serial port to use A=0, B=1, C=2 etc. There are two flags that can be or’ed with the port:
USE_SERIAL_D - use I/O port D pins
NO_DL_CACHE - Do not cache downloaded code (will try to preserve xmem.)

modem_init = string of commands to send to modem. Separate commands by \r
each command will be sent separately and wait for OK.
phone_number = string to dial, this is added to the ATDT command to the modem.
(not used for answer mode, use NULL);

3.2 Internal I/O control

Since the RAM loaders a pre-compiled, the user can not modify them if their specific hardware need some initialization to work properly.

The library contains a global array defined as follows:

char SdlIOInit[43];

This array of chars is used by the RAM loader to initialize internal I/O. The data consists of pairs of an I/O address followed by the data. An address of 0 ends the list. Up to 21 address/data pairs can be entered. The serdownl.h defines this array as an extern.

The following example configures port E pins as outputs and sets their values:

```
char io_data[]={PEFR,~((1<<7)|(1<<1)),PEDDR,(1<<7)|(1<<1),PEB7R,0,PEB1R,0xff,0};

    // before call to SerLoad() set up I/O init
    memcpy(SdlIOInit,io_data,sizeof(io_data));
```

3.3 Link the appropriate Library

The library can include the RAM loader in FLASH. This requires about 16k more of FLASH memory.

There are 4 versions of the library:

SDL-NoLoader.lib	- library, RAM loader must be downloaded from PC utility.
SDL-NoLoader-D.lib	- library, RAM loader must be downloaded from PC utility. Includes debug info on _stdio.
SDL-Flash.lib	- library includes the RAM loader. No debug info.
SDL-Flash-D.lib	- library includes the RAM loader. Includes debug info on _stdio. Ram loader will output debug info on serial port A at 19200 baud.

When using the flash versions, a new segment will be created called RAM_LOADER. This can be located anywhere in FLASH. For best chance of error recovery, either fix the segment at a specific address, or use a segment alignment.

3.4 The Sample program

In the SerDLTest directory is a file called SerDLTest.c. This is basically a program that does nothing except allow a user program to be loaded. It will flash DS2 on the RCM22xx demo board.

Compile the program to FLASH and run it. 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.

If DEDICATED is defined, a dedicated connection is used. Otherwise it will default to using a modem. Edit the parameters to SerLoad() and SerDL_Init() as needed. See the “serdownl.h” file for parameter information.

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.

A sample output is shown below:

```
Ready to download over me!
Port=B, BAUD=38400
Tons of useless debug info:
SEGSIZE=c0      MMIDR= 0
MECR = 0        MTCR =74
MB0CR=c8        MB1CR=c8
MB2CR=c5        MB3CR=c5
GCSR = 8        STACKSEG=74
DATASEG=0
Run loader in 14 seconds.
Run loader in 13 seconds.
Run loader in 12 seconds.
Run loader in 11 seconds.
Run loader in 10 seconds.
Run loader in 9 seconds.
Run loader in 8 seconds.
Run loader in 7 seconds.
Run loader in 6 seconds.
Run loader in 5 seconds.
Run loader in 4 seconds.
Run loader in 3 seconds.
Run loader in 2 seconds.
Run loader in 1 seconds.
Run RAM Downloader - 38400 baud, Addr=0x5000, 19656 bytes
Ram Loader Startup.
Out 0x75 to I/O port 0x7d.
Out 0x77 to I/O port 0x82.
Out 0x7f to I/O port 0x0.
Out 0x79 to I/O port 0xff.
Opened port  at 38400 baud
RX:         +++
RX: ATH0
RX: OK
Old loader found at:5000.
Cache is 458752 bytes, 200 to 70200.
Startup Complete :-).
TX: AT
RX: AT
RX: OK
TX: AT&F
RX: AT&F
RX: OK
TX: ATS0=0
RX: ATS0=0
RX: OK
Modem init ok
RX: RING
RX: ATA
RX: CONNECT 38400
Modem CONNECT :)
Query command, reply: Serial Modem update.
Download command. Address=0x       0, len=1024
Write block: Address = 0x       0, len=512.
        Block 80000 write OK
Write block: Address = 0x   86E00, len=512.
        Block 86e00 write OK
Write block: Address = 0x       0, len=512..Save to Cache.
Write block: Address = 0x     200, len=512..Save to Cache.
```

```
Download command. Address=0x     400, len=1024
Write block: Address = 0x     400, len=512..Save to Cache.
Write block: Address = 0x     600, len=512..Save to Cache.
Download command. Address=0x     800, len=1024
Write block: Address = 0x     800, len=512..Save to Cache.
Write block: Address = 0x     A00, len=512..Save to Cache.
Download command. Address=0x     C00, len=1024
Write block: Address = 0x     C00, len=512..Save to Cache.
Write block: Address = 0x     E00, len=512..Save to Cache.
Download command. Address=0x    1000, len=1024
Write block: Address = 0x    1000, len=512..Save to Cache.
(output for blocks 1200-1600 not shown)
Download command. Address=0x    9800, len=1024
Write block: Address = 0x    9800, len=512..Save to Cache.
Write block: Address = 0x    9A00, len=512..Save to Cache.
Download command. Address=0x    9C00, len=202
Write last block. Add=0x    9C00, len=512
Write block: Address = 0x    9C00, len=512..Save to Cache.
Write cache to flash.
Write block: Address = 0x     200, len=19968.
        Block 80200 New=old, no write.
        Block 80400 New=old, no write.
        Block 80600 New=old, no write.
        Block 80800 New=old, no write.
        Block 80a00 write OK
        Block 80c00 write OK
        Block 80e00 write OK
        Block 81000 write OK
        Block 81200 write OK
        Block 81400 write OK
        Block 81600 write OK
        Block 81800 New=old, no write.
        Block 81a00 New=old, no write.
        Block 81c00 New=old, no write.
        Block 81e00 New=old, no write.
        Block 82000 write OK
(write output for 82200-89900 not shown.)
        Block 89a00 write OK
        Block 89c00 write OK
Write block: Address = 0x       0, len=512.
        Block 80000 write OK
Reboot command
RX: ~
RX: NO CARRIER
RX: ATH0
RX: OK
Ready to download over me!
```

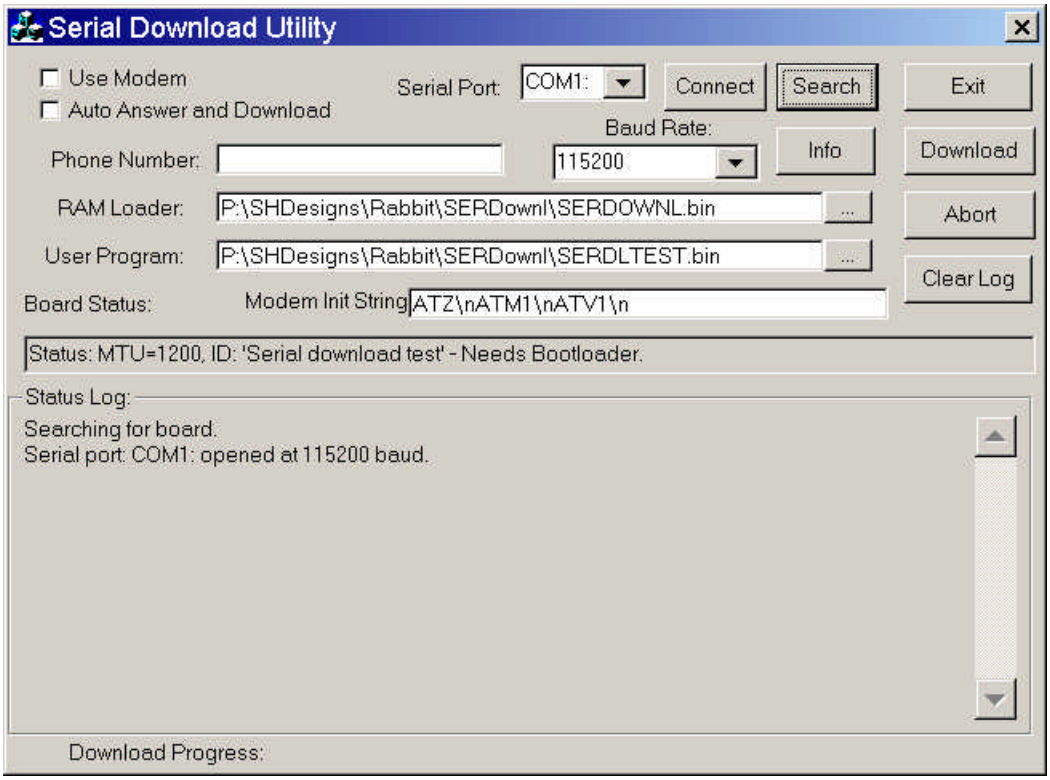
4.0 RAM DOWNLOAD PROGRAM

The RAM program is supplied pre-compiled. The pre-supplied loaders are compatible with most Rabbit boards.

There are two versions of the loaders:

SerLoader.bin	- Normal loader, 16k
SerLoader-D.bin	- Includes debug info on serial port A at 19200 baud, 22k.

5.0 PC DOWNLOAD UTILITY



5.1 Installation

The PC utility is called SerDownload.exe. There is no installation needed. It can run from anywhere.

The only registry settings it uses is to store the last RAM loader file and user binary file locations and the port, modem settings. The next time the program is run, the file settings will default to the last used.

5.2 Requirements

Any PC can be used that runs windows. The program has also been run under Wine in Linux.

The PC Download utility requires either a properly-wired serial cable connected to the target or an internal or external modem.

5.3 Encrypted .bin files

The loader supports encrypted RAM and user program files. See the later section for details on how to encrypt the files. The encryption may include a password. When a password-protected file is downloaded, the user will be asked for the password. If the password matches, the download will take place. If an incorrect password is entered, the download is aborted.

An encrypted file may have no password. In this case, the user is not prompted for a password.

5.4 Programming the Boards

5.4.1 Direct connection

Select the serial port and baud rate. Then press on the "Connect" button. Make sure "Use modem" is not selected.

Click on search and the utility will poll the serial port for a board running the RAM loader or library. It should find the boards.

The board list should show the board found (there will only be one board in the list.) If none are found or the list needs to be updated, use the "Search" button.

At this point, the list should show the boards by name. They will have the "Needs Bootloader" status appended to the name. If the board includes the RAM loader in FLASH, the status will show "Has bootloader in XMEM FLASH."

Make sure to select the proper RAM loader (if needed by the board) and the user program bin file.

To download, press the download button.

The download will perform the following steps:
1. Send a request to the board for xmem to store the RAM loader.
2. Download the RAM loader (if needed.)
3. Send a "Run" command to start the RAM loader code.
4. Re-search for the board to find response from RAM loader.
5. Download the user code.
6. Send a "Reboot" command to run the user code.

Each step is acknowledged by the Rabbit board. A progress bar on the bottom of the window shows download completion.

Once the download is complete and the board has restarted, you should be able to perform a new search and see the board report the new status string (if new code has different ID string.)

5.4.2 Modem dial out

Enter the following settings:
1. Select "Use Modem"
2. Select the port to use
3. Set the Baud Rate
4. Enter the phone number to dial.

5. Enter the program .bin file
6. If needed change the modem init string.

Click on the "Connect" button. The program should dial out and connect to the Rabbit board. Once it is connected it will search for a response for the Ram loader.

Press "Download". The board should be programmed.

5.4.3 Modem dial in

Enter the following settings:
1. Select "Use Modem"
2. Select the port to use
3. Set the Baud Rate
4. Select "Auto Answer and Download"
5. Enter the program .bin file.
6. If needed change the modem init string.

Click on the "Connect" button. The program will initialize the modem and wait for a connection from the rabbit board. When the Rabbit connects, it will automatically download the program.

5.5 Errors

The utility verifies each operation. It will try to recover from errors. The status log window will show these errors. The download can be retried if it fails.

Some notes:

1. If the user flash file is not found, the download will abort. At this point the RAM loader is running and the FLASH remains untouched. To recover, select the correct user program file and re-run the download. The utility will see that the board already has the RAM loader and skip right to the FLASH download.
2. If the download does not work, use the debug version and monitor the the output on serial port A using the "Diag" connector on the Rabbit programming cable. Use a terminal program like HyperTerm.

6.0 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.

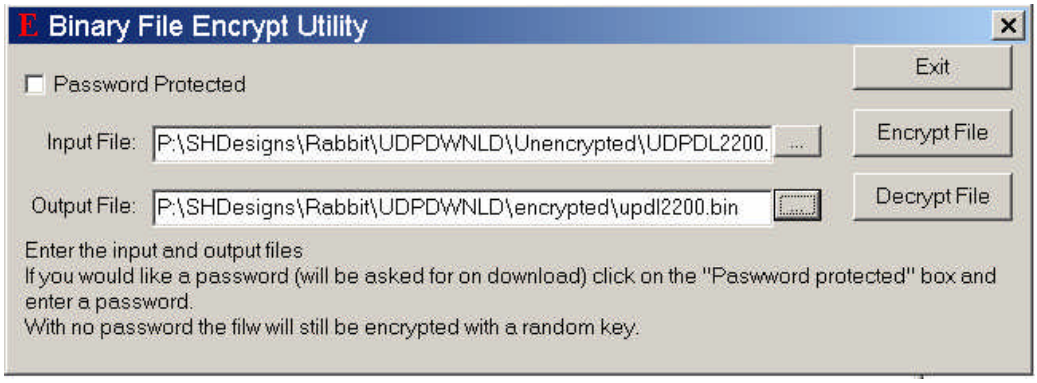
Encryption is done as follows:

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.