SHDesigns Download Manager Version III
The Resident Download Manager

Date: Thursday, July 24, 2008 S. Henion

Version:

DC-Library 1.7A
ST-Library 1.6B
Update Manager 1.7A
Boot loader: 1.3.4

Table of Contents

1.0 Description

The SHDesigns Resident Download Manager (DLM) operates completely different than any other method available. Rather than trying to maintain a single Flash image of the user program, the DLM treats the Flash as a file system that can store compressed user programs. In most systems, two or more images can be stored with enough room for a configuration file.

The DLM operates as a boot loader. It exists at the beginning of Flash and is permanently resident. User programs exist as compressed files in a Flash file system. These programs are uncompressed to RAM to run. This requires enough memory to run from RAM. Since the boot loader is never overwritten, it will always provide recovery from power loss or reset.

This also allows boards to be built in production with a generic boot loader/DLM. Then customized later with a specific user program. Additional files may be downloaded and used for configuration parameters.

The DLM supports both Dynamic C and Softools C programs.

1.1 Requirements

To use the Resident Download Manager, the following is required:

1. Rabbit board with Ethernet. *
2. At least 256k of RAM on the Rabbit board or module, 512k or more is preferred.
3. Dynamic C version 8.3 or later or Softools C.
4. PC with Ethernet. *
5. For accessing FAT files from a user program, the Z-World FAT library (Dynamic C programs) or the SHDesigns FAT library (Softools programs) is required.

* Ethernet is required for the initial creation of the .bin file that can be programmed into the board. The .bin file can later be programmed into boards with the Z-World RFU utility. See section 1.5 on non-Ethernet support. Version 1.6 of the library and the DLM-Serial.bin can be used in serial-only boards.

1.2 Compression

Files (user .bin files) are stored using LZ77 compression. This is a quick algorithm and requires little code to decode. Compression rates of 55% of original size are typical. The

DLM uses the lower 64k of Flash. With 512k of Flash, this allows about 440k of space for user programs. At 55% compression, two, 370k user programs could be stored. With a 256k flash, A 320k user program can be used (or two 180k programs). If the bin includes compressed files (jpegs, gifs), the compression rate will be less.

LZ77 (Lempel-Ziv) is an open-source algorithm. It is not to be confused with the LZW method that is proprietary and patented.

1.3 Primary, Backup and Initialization Programs

The boot manager will first try to load the primary program. If the CRC fails or the program fails to run, it will load the backup program. If no valid program exists, it will run the download manager code.

A program can reboot into the backup with a single call. A backup program can be used to download the latest .bin code from a web page or just be the previous version of the code.

An initialization program can be specified. The primary program may take several seconds to be decompressed to RAM. The initialization program can be used to notify the user that the board is booting up.

The boot loader can also load files from the FAT file system in serial flash or external DataFlash or MMC.SD cards. This is useful for applications that wish to update their code by themselves. The program can download a new program to the Serial Flash FAT file system and it will be run on the next boot. A backup program can be stored in main flash in case the user program corrupts the downloaded file.

A board is first programmed with the boot loader. Then user programs are added via the Update Manager. The flash can be save to mass-produce boards from one .bin.

The boot loader loads programs into RAM. For boards with fast RAM, it will use the fast RAM, the battery-backed RAM is not touched. The following table describes the memory map at the start of a user program:

	00000-3FFFF	***40000-7FFFF***	***80000-BFFFF***	***C0000-FFFF***
512k RAM	RAM	RAM	RAM	RAM
512k Fast RAM + 256k Slow RAM	Fast RAM	Fast RAM	Slow RAM	Slow RAM (duplicated)
512k Fast RAM + 512k Slow RAM	Fast RAM	Fast RAM	Slow RAM	Slow RAM

A user program would normally remap the memory as needed. However, it must use the same mapping for the first 256k (00000-3FFFF). If the program is >256k the second 256k

must also use the same mapping. In general the top 512k can be remapped as needed.

1.4 Boot loader/DLM and User code

Much of the boot loader/DLM code is duplicated in the library used by user programs. Versions before 7/14/06 would only allow reading of flash files when running a user program. To update files, the board needed to be rebooted to the DLM.

The current library supports reading and writing to Flash files while the user code is running. Since the user program is running in RAM, it can update the flash without disturbing the running code.

However, if the board were to be reset during the process of updating the primary program, the program may not be valid. If you plan on updating the primary program within your program, it is advisable to have a backup program present.

Note: updating serial flash or MMC/SD files with the UpdateMgr application requires rebooting to the DLM. Also, when DLM_Tick() returns 3, the user code should flush the ZW FAT library so the file system is up to date. This is best done by un-mounting the partition.

Updating the FAT files in a running system is planned in a future update.

1.5 Updating the Boot loader

The device must be rebooted to the boot loader/DLM to update the boot loader. This will normally erase all the file entries and the main program will have to be downloaded again.

As of 7/14/2006, the DLM can be updated in a user program and not lose any files. This is only while running the user program; not when the boot loader/DLM is running.

1.6 Non-Ethernet Support

The current library supports any network interface. This can be PPP, GPRS, or WiFi. This was the main reason for adding read/write support while running a user program. The boot loader only supports Ethernet as a network interface.

The Boot loader now also supports updating via a serial port. This can be used to update systems that have no Ethernet. Serial is not supported from the main program at this time.

For boards with no Ethernet, the initial boot loader may need to be modified to use the desired serial port. Program the board with the boot loader. Remove the programming cable and reconnect it with the “Diag connector”. Use the Update Manager to connect via a serial port at 115200 baud. Select the board and use the “DLM Serial” setting in the Advanced tab to set the desired port and baud rate. Then use the “Save Flash” option to create a boot loader with your settings.

The DLM-Serial.bin file can be used on boards with no Ethernet.

Version 1.6 of the library now supports a serial connection while running a user program.

1.7 Automatic Updates From Within a User Program.

The library API allows programs to be updated within a user program. The program can load a file from a web server, FTP or any method. The program would then mark the file as the primary program. On the next reboot, the new program will run. The typical steps are as follows:

1. Check size of the .bin/.blz file to be downloaded.
2. Use ffile_free() to see if there is enough space.
3. If not enough space, either delete the primary program or delete the backup and mark primary as the backup.
4. Use handle=ffile_create() to create a file.
5. Download file and use ffile_write(handle,data,length) to save to flash.
6. Call ffile_close(handle) to close file.
7. Use ffile_setPrimary(handle) to mark file as primary.
8. If old file was not deleted earlier, delete
9. Call dlm_reboot(UDPDNLD_MODE_RUN_PRIMARY) to boot to new program.

The Web2File directory includes code to download a file from a web server and store it in Flash using the API. This is for Softools, but can be ported to DC easily.

The HTTP_UPLOAD directory contains a sample app that allows files to be uploaded via a web browser.

2.0 Software Installation

The installer: SHD_DLM.exe will ask for an install directory. The library can be installed completely outside the compiler directory. If you select a compiler directory, the installer will select subdirectories under the path selected. The directories are:

.\bin	- PC programs, Update Manager and LZ77 Utility
.\bin Files	- Flash Images of resident DLM.
.\DCLib	- Dynamic C library
.\DC-Demo	- Sample application
.\ST-Lib	- Softools Library.
.\ST-Demo	- Sample Application
.\ST-CStart	- Modified Cstart to run in RAM.
.\ST-Init	- Example Softools init application
.\Web2File	- Sample function to copy a web page to a file.

.\http_upload - sample app the allows uploading files through a web browser.

3.0 Initial Install

There are two methods of doing an initial install:

1. Install the Boot loader/DLM then download a user program.
2. Create a combined .bin file with the Boot loader/DLM with the user program.

The combined is best for production. A single .bin file is burned into the Flash with the RFU or FlashIt. To create a combined .bin file, program a board with the boot loader, configure it, add your files, then use the **Save Flash** command to create a .bin for production.

If only the boot loader/DLM is programmed to Flash, then the user program must be downloaded before the system is fully functional. Once a user program is downloaded, a the Update Manager can be used to create an image that can be used by the RFU or FlashIt programs.

A board would initially be programmed with one of the following files:

DLM-III.bin	- Supports serial Flash and FAT.
DLM-III-D.bin	- Includes more verbose output.
DLM-MMC.bin	- Supports MMC cards.
DLM-MMC-D.bin	- MMC with verbose debug output.
DLM-Serial.bin	- Serial-only support

The UpdateMGR program can be used to add a user program. The “Save Flash” command in the advanced tab can then be used to create a combined .bin file.

4.0 Dynamic C User Programs

User programs are compiled to run from RAM. On systems with fast RAM, use the “Compile to Flash, Run in RAM” mode must be used. The user programs are compressed by default by the Update Manager.

4.1 Source changes

The following lines should be added after the #use dcrtcp.lib

```
#define DLM_RW
//#define DLM_VERBOSE // optional
#use SHD_DLM.lib
```

The define DLM_RW will allow the files to be updated while a user program is running. Otherwise, the board must be rebooted to the DLM to update files.

The DLM_VERBOSE define will add debugging printf() output

After the network is up (i.e. after sock_init()) add:

DLM_Init("Board name");

Where: "board name" will be displayed in the PC application.

When your application has started successfully, add the following call:

DLM_Reboot(UDPDNLD_MODE_RUN_OK);

This tells the boot loader that you have successfully run. If this is not called, then the boot loader will try to run the backup program on the next reboot.

The your main loop add:

DLM_Tick();

This will add the communication to the PC application. This should be called at least once a second for proper communication with the Update Manager application.

This will return non-0 when the user accesses the board from the DLM. You can use this to power down devices and close files. If the FAT file system is being used, partitions should be un-mounted to flush any writes. When DLM_Tick() returns 3, the next call will force a reboot.

void DLM_SetPort(unsigned port);

The above function can be called before DLM_Init() to set the port to a different port than the default (2000).

4.2 Compile to RAM mode

Note: this is not supported on the RCM32xx/33xx and RCM38xx boards, use compile to flash, run in RAM.

Compiling to run in RAM with DC 8.x and later is a bit difficult. There is no "Compile to RAM and create .bin file" menu option. In the project options set the Bios Memory Setting to Code and BIOS in RAM". In the "Default Compile" mode set to compile to .bin file (either with defined target or using attached device.) If using defined target, be sure to set up the "Targetless" tab. Then create the .bin file with F5. It will warn about RAM programs not designed to be compiled to a .bin file. If you don't get this warning,

the settings are not correct.

To use the file system, the board must be programmed with the RFU with one of the boot loader .bin files. It should be booted at least once to initialize the file system and back up the ID block.

Note: the user block is not accessible when running RAM programs; this is a limitation of Dynamic C.

If you have a 512K RAM board and get a "Out of xmem space", this is due to a bios option. Look for the following lines in rabbitbios.c. Note Dynamic C version 9.4 and later have this parameter in \lib\bioslib\memconf.lib:

```
#if (_RAM_SIZE_==0x80) && !FAST_RAM_COMPILE
                            // Number of 4k pages of RAM. _RAM_SIZE_
#define RAM_SIZE 0x40       // is defined internally by Dynamic C
#else                       // during the cold boot stage, but can
#define RAM_SIZE _RAM_SIZE_  // changed here. If _RAM_SIZE_ is 0x80
(512K)
#endif                      // we reserve the upper half for xalloc
                            // and map top of DATA to the top of the
                         // lower half of RAM. The xalloc initialization
                            // code will recognize that the space is
                            // available if RAM_SIZE < _RAM_SIZE_.
```

Change the: #define RAM_SIZE 0x40 **to** #define RAM_SIZE 0x80. That will allow all of the RAM to be used.

4.3 Compile To Flash Run in RAM mode (RCM3200/3300/3800)

This requires a bios change. The boot loader does the copy from Flash to RAM, so a minor change in the bios is needed to skip this step.

In DC 9.25 and earlier the bios file is BIOS\rabbitbios.c in the DC main directory. For 9.52 and later the file is stdbios.c in the lib\bioslib directory.

At the top of the bios add:

```
#ifdef SHD_DLMIII
// comment out this line if you don't want this warning
#warns "Code is being compiled for the SHDesigns Resident DLM"
#undef SHD_DLMIII
#define SHD_DLMIII 1
#else
#define SHD_DLMIII 0
#endif
```

Look for:

```
.swapProgToRAM::
;; Swap the compiled program over to RAM from flash. Maps flash to
;; the first 2 physical memory banks (MB0CR, MB1CR), and RAM to
```

```
;; the top banks (MB2CR, MB3CR)

      ;; Map flash to bottom 2 memory banks
      ld          a, FLASH_WSTATES | CS_FLASH
ioi   ld          (MB0CR), a                    ; Map memory bank 0
ioi   ld          (MB1CR), a                    ; Map memory bank 1
```

Add a #if to comment out the RAM chip select setup.

```
.swapProgToRAM::
;; Swap the compiled program over to RAM from flash. Maps flash to
;; the first 2 physical memory banks (MB0CR, MB1CR), and RAM to
;; the top banks (MB2CR, MB3CR)

#if SHD_DLMIII == 0
      ;; Map flash to bottom 2 memory banks
      ld          a, FLASH_WSTATES | CS_FLASH
ioi   ld          (MB0CR), a                    ; Map memory bank 0
ioi   ld          (MB1CR), a                    ; Map memory bank 1
#endif
```

Then programs you want to compile to a .bin to run from the DLM will need a define SHD_DLMIII added to the project. This must be project setting, not defined in your .c files because it is a bios setting. An alternative would be to copy the bios and add the define to the bios. Then in the project settings select the modified bios.

It would be best to have two projects defined for the board. One for debugging and another for generating .bin files. You will not be able to run programs compiled with the SHD_DLMIII setting in the debugger. Also, the file system will not be available. A work around is to compile to RAM. This will allow the file system to be used while in the debugger.

> Note, the battery-backed RAM is not used at all in the boot process. So, bbram variables will be maintained.

4.4 RCM4200 and DC 10.xx

Use the DLM-IIIRCM4200.bin file for the initial boot loader. Make the changes as described in section 4.3.

Dynamic C has a bug when creating .bin files. When using “Compile to .bin file using defined target” it will generate a bad file. Select the compiler option to 'Compile to .bin using attached target”. The use “Compile to .bin, Compile to flash, run in RAM.” This will create a valid .bin file.

4.5 DC Sample

The DC-Demo directory has a sample application that will perform the following operations:

1. Displays debug information.
2. Read and display a file called “testfile.txt” if it exists in flash.
3. Create a file called “Logfile.txt” and write 1000 lines of text to it.
4. Enter the main loop and wait for download requests.

There is also a sample file called “Web2file.c” in the web2file directory. This is a function to open a web page and save the file to a flash file. This can be used as a sample for writing your own DLM functions. This file is for Softools, but it can easily be ported to DC.

4.6 Init Program

A user program can be defined as an “Init program”. An Init Program is run when the boot loader starts up. The purpose of an init program is to give the user some feedback during the time it takes to decompress and load large user programs.

The Init program can only be stored in the main flash.

One the hardware is initialized, it needs to return to the boot manager. The following code will reboot without resetting the hardware:

```
root void JumpRAM(void)
{
#asm
     ipset 3
     ld        a,0x85
     ioi  ld (MB3CR),a
     ld   iy,0xff00
     ld   ix,_J2Ram
     ld   b,20
     ld   de,2
     ld   c,0xf
ldirfix:
     xor  a
     ldp hl,(ix)
     ld      a,c
     ldp (iy),hl
     add ix,de
     add iy,de
     djnz    ldirfix
     xor       a
     ioi  ld (GCDR),a    ; clock doubler off
     db      0xc7,0x00,0xff,0xf0 // ljump to fff00
_J2Ram:
     // map lower 256k to FLASH
     ld        a,0x88    // flash cs
     ioi     ld (MB0CR),a
     ioi     ld (MB1CR),a
```

```
      jp      0
_J2RamEnd:
#endasm
}
```

The code remaps Flash to 0 then jumps to 0.

4.7 User Block Access

When compiling to RAM, Dynamic C does not allow access to the user block. The flash file system can be used instead.

4.8 Update Flash Without Rebooting to DLM

Add a #define:

#define DLM_RW

That will enable read/write support while in a user program.

For devices with two 256k flash chips, you will need to make a change to xmem.lib.

In the function WriteFLashArray() at about line 813 look for the comment:

> // Check if we are outside the flash(es) area

Then at about line 847 look for a comment:

> // check for unintentional attempt to write into ID or User Blocks

Add an #if 0 to disable all the code between the comments. It should be similar to the following:

```
#if 0
      // Check if we are outside the flash(es) area
#ifndef CS_FLASH
      #ifndef CS_FLASH2
      retval = -1;            // no flash at all!
      #endif      //CS_FLASH2
#endif      // CS_FLASH
#ifdef CS_FLASH
      if (flashnum == 1) {
            if (((mbXcrBegin & MBXCR_CS_MASK) != CS_FLASH) ||
                ((mbXcrEnd & MBXCR_CS_MASK) != CS_FLASH)) {
                  retval = -1;      // flash 1 is not mapped here!
            }
      }
#else //ifndef CS_FLASH
      if (flashnum == 1) {
            retval = -1;            // no "primary" flash!
      }
#endif      //CS_FLASH
#ifdef CS_FLASH2
      if (flashnum == 2) {
            #if (RUN_IN_RAM_CS == 2)
            retval = -1;            // CS2 is RAM, not flash!
            #else
            if (((mbXcrBegin & MBXCR_CS_MASK) != CS_FLASH2) ||
                ((mbXcrEnd & MBXCR_CS_MASK) != CS_FLASH2)) {
```

```
                retval = -1;      // flash 2 is not mapped here!
            }
            #endif
      }
#else //ifndef CS_FLASH2
      if (flashnum == 2) {
            retval = -1;            // no "secondary" flash!
      }
#endif      //CS_FLASH2
#endif // 0
// check for unintentional attempt to write into ID or User Blocks
```

The above is needed as the lib remaps the flash as needed to allow access while running in RAM. The DC libs force their own mapping that does not work with some boards.

4.9 Serial Support

To add user-program serial support add the following defines before the #use SHD_DLM.lib:

#define DLM_SERIAL X

Where 'X' is 1 through 6 for serial ports A-F respectively

For ports that use parallel port D add:

#define SERX_USEPORTD

'X' would be A-D

Lastly add DLM_Ser.lib to the project or global LIB.DIR.

The Lib will automatically open the port and process packets.

If you wish to share the serial port with the DLM, there is global variable: DLM_SerEnable. Set this to 0 to disable processing within DLM_Tick(). Then you can use the serial read/write functions on the already open serial port (opened automatically on DLM_Init().)

Note: the lib will use the baud rate configured by the UpdateMgr under the DLM_Serial options.

5.0 SofTools Programs

5.1 Project Changes

Softools programs will need to use a modified cstart.asm to be able to run in RAM. The cstart_ram.asm provided does this. Two defines need to be added to your project setting. Either as part of the project ASM defines, or as a define for cstart_ram.asm:

RAM_COMPILE - Define as 1 to enable compiling to run in RAM.
FAST_RAM - Define this for boards with FAST RAM. (3200/3300/3800)

They can also be defined in the cstart_ram.asm file. Adding the defines to the project settings allows the same source file to be used in multiple projects.

Note, this cstart is designed for Softools WinIDE versions 1.68 and later. Contact SHDesigns for instructions on how to modify an earlier cstart.asm.

The lib: SHD_DLM.lib needs to be added to the project “Lib and Obj” list. The SHD_DLM-D.lib can also be used as it will output debugging information on stdio.

For serial support, Use SHD_DLM-SerX.lib. Where 'X' is A-F for ports A-F. A “-D” version is also provided that will include verbose messages on DLM progress.

Add an include for SHD_DLM.h to your source.

After the network is up (sock_init()) add:

```
DLM_Init(“Board name”);
```

Where: “board name” will be displayed in the PC application list of boards.

When your application has started successfully, add the following call:

```
DLM_Reboot(UDPDNLD_MODE_RUN_OK);
```

This tells the boot loader that you have successfully run. If this is not called, then the boot loader will try to run the backup program on the next reboot.

The your main loop add:

```
DLM_Tick()
```

This will add the communication to the PC application. This should be called at least once a second for proper communication with the Update Manager application.

This will return 1 when the user requests to run the DLM. You can use this to power down devices and close files. The next call will force a reboot.

void DLM_SetPort(unsigned port);

The above function can be called before DLM_Init() to set the port to a different port than the default (2000).

5.2 ST Sample

The ST-Demo directory has a sample application that will perform the following operations:

1. Read and display a file called “test.html” if it exists in flash.

2. Create a file called “Logfile.txt” and write 1000 lines of text to it.
3. Open the web page: http://shdesigns.org/test/test.html and save it to flash as test.html.
4. Create a test file and perform seek tests.
5. Enter the main loop and wait for download requests.

The function to save an http file to flash is called “Web2file.c” in the web2file directory. This is a function to open a web page and save the file to a flash file. This can be used as a sample for writing your own DLM functions.

5.3 Init Program

A user program can be defined as an “Init program”. An Init Program is run when the boot loader starts up. The purpose of an init program is to give the user some feedback during the time it takes to decompress and load large user programs.

The Init program can only be stored in the main flash.

One the hardware is initialized, it needs to return to the boot manager. The following function will reboot without resetting the hardware:

```
void JumpRAM(void);
```

This function remaps Flash to 0 then jumps to 0. The directory .\ST-init has this function in a library called ST_Init.lib. There is also a project to rebuild the lib.

5.4 Debugging under WinIDE

Use FlashIt to program the appropriate boot loader file. If you do not have FlashIt, you can use the RFU from Dynamic C.

Remove the programming cable and let the board boot up. It will create the default configuration table for the board (memory map, file system.)

Then you build your program under WinIDE and run it under the debugger. It should read the flash and allow access to the files.

5.5 Serial Support

As described in section 5.1, serial support is added by linking in the appropriate library. The library will use the baud rate and port D settings used by the UpdateMgr DLM_Serial options (Advanced tab).

On DLM_Init() the lib will open the serial port automatically. If you wish to share the serial port with another device you can disable the DLM_Tick() processing by setting a global int, DLM_SerEnable to 0.

Then your program can use the serial read write functions. They can also use the stdio functions (fwrite(), fread() etc) by using the FILE * DLM_serio.

Add a #define DLM_SERIAL to you programs before including SHD_DLM.h; that will allow access to the serial variables.

5.6 RCM4200 support

The RCM4200 will work fine under WinIDE. The ris one small change needed in the ethernet library to add the board type for the RCM4200.

Add the ethinitall.c to your project (located in the ST-cstart directory), You may need to add the include/stcpip directory to the compile include path.

In the linker settings, make sure ethinitall.obj is linked before the TCPIP library.

Edit the Softools SoftoolsIDBlock.h file and add the following define:

```
#define RCM4200 0x2900 // RCM4200, ASIX, 58.98 MHz, 512K flash, 512K
SRAM, 512K FSRAM, sflash
```

(one line)

Then build your program as described in section 5.1.

Note: with the above changes, the 4200 can be used even without the DLM

5.7 Using WinIDE to program the boot loader.

If you do not have the DC RFU program or FlashIt, you can use WinIDE to program to load the boot loader.

1. Create a project with the same name as the boot loader. I.e DLM-III-D
2. Create a .c file with an empty main()
3. Build the project.
4. Copy the boot loader to the main directory of the project over the .bin file created.
5. Connect to the board and use the program to flash menu or icon.

6.0 Library API

The library allows user programs to read and write files in flash. Note, writing to flash will interfere with interrupts.

A user program can add file, delete files, read files and set a file as a primary or backup program. This can be used to allow programs to automatically update their code via any means: web download, serial update, PPP or copy from a removable device.

The file system supports up to 12 files. The file names can be up to 128 characters long.

Up to 4 files may be open at one time. Files are allocated in sector-size blocks. This is usually 4096 bytes although some older boards have 512-byte sectors.

In addition to the file system calls, there are four other calls:

int DLM_Reboot(int mode);

Mode controls rebooting. The values are:

UDPDNLD_MODE_RUN_DLM = Reboot to the DLM
UDPDNLD_MODE_RESTART = Reboot
UDPDNLD_MODE_RUN_BACKUP = Reboot to Backup program
UDPDNLD_MODE_RUN_PRIMARY = Reboot to Primary program
UDPDNLD_MODE_RUN_OK = Acknowledge successful start

The UDPDNLD_MODE_RUN_OK is the only call that will return. The return value will be UDPDNLD_MODE_RUN_PRIMARY or UDPDNLD_MODE_RUN_BACKUP to indicate what the current program is running. Calling with the OK clears the flag, so if the return value is needed later, it should be saved.

int DLM_Init(char * s");

This call sets the string display in the PC application board list and also initializes the network socket. The string should be static as the function only save a pointer to the string. It is read later when the DLM runs.

The init function will normally return 0. A return of 1 indicates that the UDP socket can not be opened. This is usually only happen if not enough UDP buffers have been reserved.

void DLM_SetPort(unsigned port);

This can be used to change the UDP port used to communicate. This is normally set in the PC application Advanced settings page.

int DLM_Tick(void);

Should be called at leas once a second to handle DLM requests.

The return values are:

0 – No packet or read
1 – Write packet received
2 – Reboot command received (next call will reboot)
3 – RTC clock changed

6.1 Flash File API calls

The file API can be used like the user block. It has the advantage of being treated as a file. These files can also be saved or loaded from the Download Manager. These files could be used for web pages, config data, log files; whatever is needed.

Note: as of 05-07-2006, the library supports read and write on files. A seek has also been implemented.

int ffile_open(char * name);

Opens an existing file for reading or writing. The return value is a handle to use for reading from or writing to the file. The handle will be >=0. A return value <0 indicates an error (see section 6.2)

Note: the handle can be used for the ffile_setPrimary() and ffile_setBackup() functions.

unsigned ffile_read(int handle,char far * buff,unsigned len);

Using a handle from ffile_open() or ffile_create(), read up to 'len' bytes into buff. The function returns the number of bytes read. At end of file, it will return 0.

For Softools the data can be far. For Dynamic C, the buffer must be in the root or stack areas.

int ffile_seek(int handle,long pos,int mode);

Seek to a position in a file. The parameters are:

handle	- file handle from open or create call
pos	- the position relative to the point specified my mode
mode	One of:
	FSEEK_SET - pos is from start of file
	FSEEK_CUR - pos is relative to current position

FSEEK_END - pos is relative to end of file

long ffile_tell(int handle)

Returns current position from beginning of file.

void ffile_close(int handle);

Close a file opened for reading or writing.

int ffile_create(char * name);

Create a new file, it a file of that name exists, it will return an error. To empty a file, use ffile_delete() then use ffile_create() to recreate it.

unsigned ffile_write(int handle,char far * buff,unsigned len);

Write data for a file handle returned by ffile_create() or ffile_open(). Will write 'len' bytes starting a *buff. It returns the number of bytes written.

Writing to flash will disable interrupts. This may interfere with PPP connections or serial ports.

For Softools the data can be far. For Dynamic C, the buffer must be in the root or stack areas.

int ffile_delete_index(int index);
int ffile_delete(char * name);

Delete a file by its index # or by name. Note deleting a file that is marked as Primary or Backup will remove the setting. The ffile_setPrimary() or ffile_setBackup() must be used to restore the setting.

int ffile_index(char * name);

Get the entry # for a file. This is used for setting the file as primary or backup.

int ffile_setPrimary(int index);
int ffile_setBackup(int index);

Set a file as the primary or backup boot image. Index is the value returned by ffile_index().

long int ffile_free(void);

Return the number of free bytes in Flash.

int ffile_FATLink(char * fname);

Create a link to a FAT file. This will create file that points to a FAT file with the same name. The entry number is returned. If this value is used on a ffile_setPrimary() or ffile_setBackup(), the boot loader will load the image from a file in the FAT file system. Not only root directory files are supported with the DLM version 1.1 or earlier..

int ffile_getPrimary(void);
int ffile_getBackup(void);

These are used to get the current file # assigned to the primary and backup programs.

6.2 Error values

FERR_NOTFOUND	- File not found
FERR_TOOMANYFILES	- Too many files opened or no free file entries
FERR_NOXMEM	- Insufficient xmem to buffer file.
FERR_FILE_EXISTS	- Can not create file as it already exists.
FERR_SEEK	- seek past beginning or end of file.

6.3 Compression

The boot loader will load compressed files. The Update manager will by default compress files (if "Compress File" is checked in the Add File dialog.) If you download programs via your own method (i.e a FTP or Web download), they should be compressed before being used. The program LZ77.exe can be used to compress a .bin file to a compressed image. The check box "Include header and CRC" **must** be selected. There is no support to compress the files within the rabbit.

In Softools, the LZ77.lib can be used by user programs to decompress files if needed. There is no support for decompressing files in DC. As an option, you can use the zcompress program from DC and use the DC decompression lib.

The LZ77 source is included in the LZ77 directory. There is a project to build the SofTools libraries for decompression. The LZSS.c could be ported to DC if needed as it is a small amount of code. The lzssOriginal.c is the original full source. This could be used to port the compressor. It does use a considerable amount of RAM and code to do the compression. The Readme.txt file has details on the implementation.

Compressed files will include the LZ_Header structure before the compressed data. This will have a key of "LZ77", the CRC of the data and the length of the data.

Uncompressed user programs are supported. However, they will not be checked if they are valid. The compressed .bin files have a header that includes a CRC of the file. The Boot Loader will abort loading a compressed file if the CRC fails.

7.0 Update Manager

The Update Manager is used to download new firmware and control various options like setting backup files, configuring Serial Flash/MMC, IP address and Port. It can be used to generate images for use in production.

Note: Windows XP with SP2 may block responses from the boards with the default settings. This can usually be seen that the board can not be searched for, but can be found by pinging the IP address.

To resolve this, Microsoft recommends opening the port:

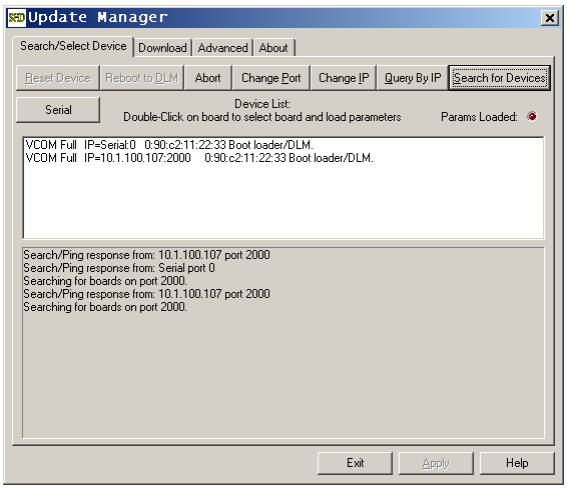
Adding the port exception:

1. Click **Start**, click **Run**, type Wscui.cpl, and then click **OK** to open Windows Firewall.
2. Click the **Exceptions** tab, and then click **Add Port** to display the **Add a Port** dialog box.
3. Enter the port number that your program uses. (DownloadMgr uses port 2000 by default.)
4. Select the TCP or UDP protocol, depending upon what your program uses. (select UDP.)
5. In the **Name** field, type a descriptive name for the port.
6. Click **Change Scope** to view or to set the scope for the port exception, and then click **OK**.
7. Click **OK** to close the **Add a Port** dialog box.
8. To verify that the port settings are correct for your program, test the program.

Full details can be found in MS Knowledge Base article: ID:875357 Troubleshooting Windows Firewall settings in Windows XP Service Pack 2.

There are four tabs in the top of the application as shown in the following sections:

7.1 Search/Select device Tab.



The first is the "Search/Select Device" tab. This will list the devices running the DLM. Select the device to update and it will retrieve the file list. Then you can proceed to the "Download" tab.

Note: Version 1.2 has changed the selection process. You must now double-click on a board to select it. It will then load the configuration and file list. This is to allow using the set IP and reset device commands without the application trying to load the parameters first.

A LED labeled "Params Loaded" will display green when a board has been selected and the parameters have been loaded. It will be red when there is no board selected and loaded.

When running a user application, the library dated before 7/14/2006 will require the board to be rebooted to the DLM to update files. The current lib will allow flash files to be updated in a running system with no need to reboot.

Each entry will show the board name, IP, MAC address and status. The status values are:

Boot loader/DLM - Running from boot loader
User Program. - Running user program, read only access to files.
User R/W Program. - Running user program with library that supports read/write to flash.

The "Change IP" can be used to change the network IP address of the device. This is only temporary. When you select a device, the file list and configuration information is read from the device. If the IP address is not valid for the current LAN, the logs will show "No response from device." Use the "Change IP" button to set the IP to one compatible with the current LAN. If the board does not respond to a a request for configuration data, it will automatically ask if you want to run the set IP function.

The Advanced Tab can be used to change the DLM IP permanently. Note a user program will override the IP settings when it reboots to run the DLM. This IP is used when the boot loader can not load a user program.

Once files are downloaded and all changes are made, the “Reset Device” button can be used to restart the device and run the latest software.

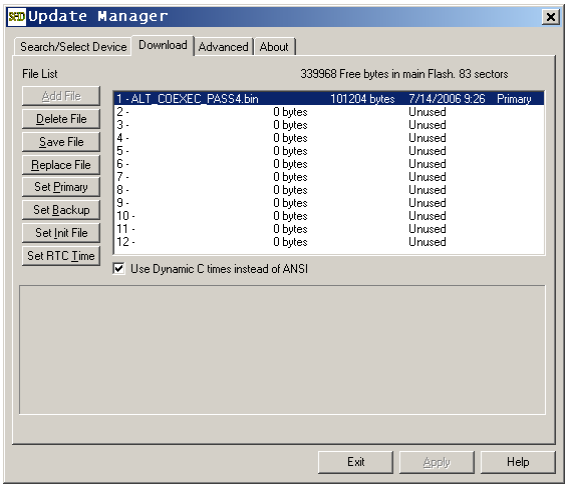
The “Reboot to DLM” button will reset the board and run the DLM.

The “Serial” button allows access via serial RS-232 port. This is only available when running the boot loader/DLM. It will bring up a selection dialog with the port number (COMn:). The advanced tab has a “DLM Serial” option that selects the port to be used. If it is not set, it will use Port A. Port A can be used with the programming cable and the “Diag” connector.

Version 1.6 and later of the libs support updating flash while running user programs.

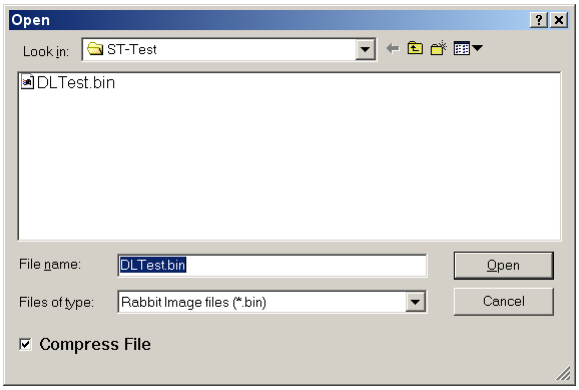
There is an “Abort” button. This may be used to abort a download if the board become unresponsive. It should not be needed.

7.2 Download Tab



The download tab displays the files for the selected board. There are 12 file slots. Files can be marked as “Primary”, “Backup”, “Init” or “Unused”. When the DLM boots, it will try to run the Init file, then primary file. If that fails or is corrupted, it will run the backup.

To add a file, select an empty file slot and click on “Add file”. This will open up a file open dialog box.



Select either a DC or SofTools .bin file. The "Compress File" option should be checked. This will compress the file and then transfer it to the board. If no file is marked "Primary" this file will automatically be marked primary.

Note: when downloading additional files that are not programs (i.e. A configuration file read by a user program), make sure the compress option is off.

The "Replace File" can be used to delete then add a file in one click.

Any selected file can be marked as the primary or backup with the "Set Primary" and "Set Backup" buttons.

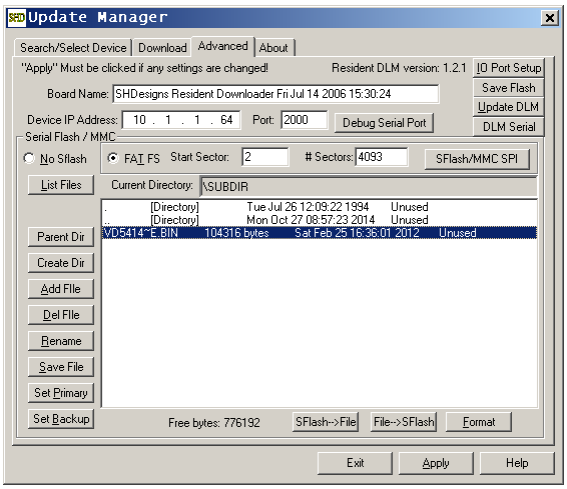
The "Save File" button can be used to save a file to disk. Note, if saved file is in compressed format, the LZ77.exe program can be used to uncompress the file.

Once download is complete, return to the "Search" tab and click on "Reset Device". That will reboot the device and run the primary program.

The "Set Init" button will set a program as the init program.

The "Set RTC" button will set the real time clock of the board to that of the PC. Version 1.4 has an option to use ANSI or Dynamic C time formats. If your programs are written in Softools, disable the DC time stamp option.

7.3 Advanced Tab



The advanced tab allows the following parameters to be set:

Board name: this will be the name displayed in the device list. It will most often be the ID string provided by the main program.

Device IP Address: This will be the default IP address when the boot loader starts (will be overridden by the IP address assigned to a user program when rebooted to the DLM.)

Port: UDP Port to use (2000 is default.)

Note: when these settings are changed, the “Apply” button will be enabled. This must be clicked to commit the changes.

Save Flash: This button will save the flash including the stored files. This file can be used with the RFU or FlashIt programs to program additional boards with a single download.

Update DLM: This button can be use to load a new flash image. This is used to update the boot loader. The Board name area will switch to a progress bar during the download. The Status area in the select tab will show details of the download progress. This will clear the file list and options if the board is running the boot loader. From a user program, the values are maintained.

IO Port Setup: This button will bring up a dialog that allows the I/O ports to be initialized on boot up. The boot loader will clear all I/O ports and set them as inputs by default. This can be used to turn LED's on or off to power down devices while the boot loader is uncompressing the main program. The dialog box has a description on how to enter the port values.

Serial Flash: section is used to maintain the FAT file system. The boot loader uses a simplified FAT file system. It will only support short file names. Note: when the user

program is running, the serial Flash functions will not work. Use the "Reboot to DLM" command from the search tab to reboot to the DLM.

The Serial Flash area has commands similar to that of the Download tab. It has the **Add**, **Delete** and **Save** commands to manage files. The "**Rename**" button can be used to rename a file.

Note: The save file will save the file unchanged. Any user programs downloaded to the board with this utility will be compressed by default. When saved to disk, they will still be compressed. This is intentional to prevent end users form being able to save a .bin file they could use on their own boards.

The save command can also be used to copy a file generated by a user program. An good example is a log file.

The "**Set Primary**" and "**Set Backup**" buttons will assign files to be run by the boot loader. These commands will create links in the Download tab to these files in slots 10 and 11. This link is by name only. If a user program replaces a file with a new .bin file, it will automatically be run on the next reboot.

The "**SFflash-->File**" and "**File-->SFlash**" buttons can be used to store a complete image of the serial flash. Note: only data from page 0 through the end of the FAT partition are saved.

The "**Format**" button will create a new FAT file system. On an initial boot of the boot loader the Start and Size fields will be set to the size of the serial Flash. Note: these fields set the start and end of the FAT partition. The first sector is reserved for the partition table. When using Dynamic C, it is best to use its format/partition code. Depending on your compile settings, the partition created by the Resident DLM, may not be compatible.

Sflash/MMC SPI: This button configures the interface to the FAT device. See section 8.0 for a description.

DLM Serial: This button configures the serial port for downloading when the DLM/Boot loader is running. Select the serial port and baud rate. For ports A-D, the serial ports default to parallel port C. The "Use Port D" option should be selected if the alternate ports on parallel port D are used.

8.0 Serial Flash/MMC Support

The boot loader can read/write the FAT file system in Atmel serial Flash chips. It will automatically detect if the file system uses the byte reverse setting.

Atmel Serial Flash Chips Supported:
AT45DB041B

AT45DB081B
AT45DB321
AT45DB642
AT45DB1282

Atmel Serial DataFlash Cards Supported:
AT45DCB008
AT45DCB004
AT45DCB002

The Softools serial Flash library will support these devices. The Dynamic C sflash.lib will support these also but the DataFlash cards may have to have their ID's added to the device table in the library.

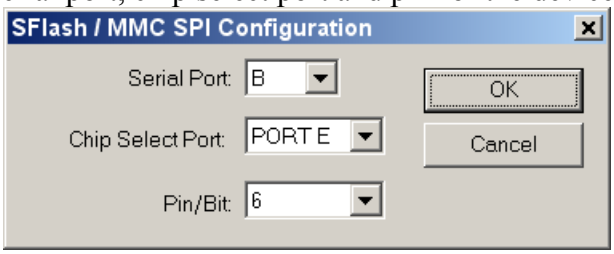
The boot loader supports The Serial Flash devices of the RCM3700, RCM3800 and RCM3300. It will also support Serial Flash chips or DataFlash cards connected to any board that mimics the connections of the RCM3700. The signals are:

Signal	**Rabbit pin**
/CS	PE6
SCK	PB0
SI	PC4
SO	PC5

The /CS pin can also be used to drive a LED. This will show when the card is being accessed.

Note: the 3300/3800 will require configuring the pins to match that of the devices. Refer to the schematics of these modules for the pin assignments.

In the "Advanced" Tab of the UpdateMGR application, use the "**SFlash/MMC SPI**" button to assign the serial port, chip select port and pin for the device if it is different.



The above dialog shows the default settings. The boot loader will automatically set up the port direction and function registers for the ports.

MMC is supported in the same way. The pins have similar names:

Signal	Rabbit pin
/CS	PE6
SCK	PB0
SDI	PC4
SDO	PC5

SHDesigns has MMC drivers for both Dynamic C and Softools.

9.0 Debugging

The Boot Loader will output debug information on serial port A at 115,200 baud. This can be used to verify operation and debug problems. A program like HyperTerm can be used with the ZWorld programming cable connected via the “Diag” connector. TeraTerm pro has been found to operate better as HyperTerm has problems with other programs.

During the startup process a message is displayed:

Press ESC to abort load:

Pressing the ESC key will prevent any programs from loading. This is useful if a main program runs but does not allow the Update Manager to access the board.

The '-D' versions of the Resident DLM images are more verbose. This slows the download speed slightly but has no effect on user programs.

The Advanced Tab allows the debug port to be changed from the default.

10 Planned Enhancements

Serial support has been added in the latest boot loader. Support for serial while running a user program is added in lib version 1.6.

There are no plans to support the WiFi adapter in the boot loader. This adds too much code.

PPP may be added in the future. This is being looked into but requires finishing the PPP libraries for the Softools C Compiler. PPP with devices like GPRS will take many parameters to be passed to the DLM. Support for PPP is not expected in the near future.

Source for the boot loader/DLM may be available at some time to allow custom boot loaders (i.e PPP/GPRS.) The boot loader is written in Softools C; there is no way to port it to Dynamic C due to the complexity and memory map. Also, networking and FAT requires about 130k of code. The Softools version fits easily in 56k.

A “Wizard Mode” for the Update Manager is planned. This would basically be a stripped down version. It would include a user program in the .exe file itself. Then allow a board to be updated with a simpler interface.

Support for the NAND flash and xD cards on the RCM33xx boards is not planned. These devices require a large amount of code, large RAM buffers and have complicated CRC and recovery code.

11. Support

SHDesigns takes support seriously. From our perspective, the libraries and applications are free for users to use. Support is what costs us time. After releasing several free libraries, the time to answer user questions has taken time away from other projects. So, the licensing was added for the DLMs and advanced libraries to cover the costs.

There are two methods for support:

SHDesigns Rabbit discussion board: http://rabbit.shdesigns.org/

Email: Rabbit@shdesigns.org (include DLMII in the subject line, it will help on sorting emails.)

The discussion board is preferred as it provides other users access to answers to questions. It is best for general questions and comments.

Email support can also be used for specific problems.

Please include as much information as possible:

Board type

Compiler version

Debug output from the programming port using the “Diag” connector if possible (the -D versions of the boot loader and libs will add more information.)

For DC, the compile mode and options (Separate I&D on/off).