wflash - WiFi bootloader Command Line Interface

Generated by Doxygen 1.8.13

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

Module Documentation

3.1 main

wflash command line application. Uses wflash protocol to manage the Flash memory of a MegaWiFi cartridge.

Data Structures

- struct MemImage

  Structure containing a memory image (file, address and length)
- struct Flags

  Commandline flags (for arguments without parameters).

Macros

- #define MAX_FILELEN 255

  Maximum length of the file name string.
- #define MAX_WRITELEN 1376

  Maximum length of a write command.
- #define MAX_MEM_RANGE 24

  Maximum length of a memory range.
- #define VERSION_MAJOR 0x00

  Major version of the comman-line application.
- #define VERSION_MINOR 0x01

  Minor version of the comman-line application.
- #define ByteSwapWord(word) do{(word) = ((word)>>8) | ((word)<<8);}while(0)

  16-bit byte swap macro

Functions

- void PrintVersion (char prgName[ ])
- void PrintHelp (char *prgName)
- int ParseMemArgument (MemImage *m)
- int ParseMemRange (char inStr[ ], uint32_t *addr, uint32_t *len)
- void PrintMemImage (MemImage *m)
- void PrintMemError (int code)
- uint16_t * AllocAndFlash (MemImage *fWr, int autoErase, int noPatch, int columns)
- uint16_t * AllocAndRead (MemImage *fRd, int columns)
- int main (int argc, char **argv)

3.1.1 Detailed Description

wflash command line application. Uses wflash protocol to manage the Flash memory of a MegaWiFi cartridge.

Author

Jesús Alonso (doragasu)

Date

2017

3.1.2 Function Documentation

3.1.2.1 AllocAndFlash()

```
uint16_t* AllocAndFlash (
            MemImage * fWr,
            int autoErase,
            int noPatch,
            int columns )
```

Allocs a buffer, reads a file to the buffer, and flashes the file pointed by the file argument. The buffer must be deallocated when not needed, using free() call.

Parameters

in	*fWr*	Memory image to flash.
in	*autoErase*	Set to true to perform an erase operation before flashing the memory image. Only the covered range is erased.
in	*noPatch*	If nonzero, ROM must be written 1:1 (i.e. it will be neither patched nor trimmed).
in	*columns*	Number of columns of the console, used to display the progress bar while flashing.

Returns

Pointer to the allocated and flashed memory if OK, NULL if error.

Note

fWr.len is updated if not specified.

Warning

A successfully allocated buffer must be freed externally to the function, using a free() call.

Definition at line 290 of file main.c.

3.1.2.2 AllocAndRead()

```
uint16_t* AllocAndRead (
            MemImage * fRd,
            int columns )
```

Allocs a buffer and reads from cart. Does NOT save the buffer to a file. Buffer must be deallocated using free() when not needed anymore.

Parameters

in	*fRd*	Memory image to read.
in	*columns*	Number of columns of the console, used to display the progress bar while flashing.

Returns

Pointer to the allocated and readed memory if OK, NULL if error.

Warning

A successfully allocated buffer must be freed externally to the function, using a free() call.

Definition at line 372 of file main.c.

3.1.2.3 main()

```
int main (
            int argc,
            char ** argv )
```

Entry point. Parses command line and executes requested actions.

Parameters

in	*argc*	Number of input parameters.
in	*argv*	List of input parameters.

Returns

0 if completed successfully, non-zero if error.

Definition at line 414 of file main.c.

3.1.2.4 ParseMemArgument()

```
int ParseMemArgument (
            MemImage * m )
```

Receives a MemImage pointer with full info in file name (e.g. m->file = "rom.bin:6000:1"). Removes from m->file information other than the file name, and fills the remaining structure fields if info is provided (e.g. info in previous example would cause m = {"rom.bin", 0x6000, 1}).

Parameters

in,out	*m*	Pointer to the MemImage structure to parse.

Returns

0 if input properly parsed, non-zero if error.

Definition at line 164 of file main.c.

3.1.2.5 ParseMemRange()

```
int ParseMemRange (
            char inStr[],
            uint32_t * addr,
            uint32_t * len )
```

Parses an input string containing a memory range, obtaining the address and length. If any of these parameters is not present, they are set to 0. Parameters are separated by colon character.

Parameters

in	*inStr*	Input string containing the memory range.
out	*addr*	Parsed address.
out	*len*	Parsed length.

Returns

0 if OK, 1 if error.

Definition at line 215 of file main.c.

3.1.2.6 PrintHelp()

```
void PrintHelp (
            char * prgName )
```

Print utility help.

Parameters

in	*prgName*	Utility program name.

Definition at line 130 of file main.c.

3.1.2.7 PrintMemError()

```
void PrintMemError (
             int code )
```

Prints the error message related to an unsuccessful ParseMemRange() call.

Parameters

in	*code*	Error code returned by ParseMemRange().

Definition at line 261 of file main.c.

3.1.2.8 PrintMemImage()

```
void PrintMemImage (
             MemImage * m )
```

Prints a properly parsed memory image.

Parameters

in	*m*	Pointer to the parsed MemImage structure to print.

Definition at line 250 of file main.c.

3.1.2.9 PrintVersion()

```
void PrintVersion (
             char prgName[ ] )
```

Print program version number.

Parameters

in	*prgName*	Current program name.

Definition at line 120 of file main.c.

3.2 progbar

Draw progress bars for command line applications.

Functions

- void ProgBarDraw (unsigned int pos, unsigned int max, unsigned int width, char text[ ])

3.2.1 Detailed Description

Draw progress bars for command line applications.

Drawn progress bar has the following appearance: <Some text>=""> [========>] 50% Initial text is optional. The bar is auto adjusted to the specified line width. This function must be called for each bar iteration.

Note

It is recommended to hide the cursor (e.g. calling curs_set(0) if using ncurses) when using this module.

Author

Jesus Alonso (doragasu)

Date

2015

3.2.2 Function Documentation

3.2.2.1 ProgBarDraw()

```
void ProgBarDraw (
            unsigned int pos,
            unsigned int max,
            unsigned int width,
            char text[ ] )
```

Draws the progress bar.

Parameters

in	*pos*	Position (relative to max).
in	*max*	Maximum position (pos) value.
in	*width*	Line width. Drawn bar will fill a complete line.
in	*text*	Text drawn at the beginning of the line (NULL for none).

ProgBar: Draw progress bars for command line applications.

Drawn progress bar has the following appearance: <Some text>=""> [========>] 50% Initial text is optional. The bar is auto adjusted to the specified line width. This function must be called for each bar iteration.

Note

It is recommended to hide the cursor (e.g. calling curs_set(0) if using ncurses) when using this module.

Author

Jesus Alonso (doragasu)

Date

2015 Draws the progress bar.

Parameters

`in`	*pos*	Position (relative to max).
`in`	*max*	Maximum position (pos) value.
`in`	*width*	Line width. Drawn bar will fill a complete line.
`in`	*text*	Text drawn at the beginning of the line (NULL for none).

Definition at line 28 of file progbar.c.

3.3 rom_head

Megadrive ROM patch module.

Macros

- #define ROM_HEAD_LEN 512

  Lengh of the complete header (including vectors) in bytes.

Functions

- void RomHeadPatch (uint8_t ∗head)

3.3.1 Detailed Description

Megadrive ROM patch module.

Currently the module only allows to patch the ROM header, adding the wflash header information, and changing the entry point, for the bootloader to be launched instead of the flashed ROM.

Author

Jesús Alonso ()

Date

2017

3.3.2 Function Documentation

3.3.2.1 RomHeadPatch()

```
void RomHeadPatch (
            uint8_t * head )
```

Patches the ROM header for the wflash bootloader to be launched instead of the ROM, while trying to still make the ROM "launchable"

Parameters

in,out	*head*	Pointer to the ROM, including the complete header

Definition at line 108 of file rom_head.c.

3.4 util

Utility definitions and macros.

Macros

- #define TRUE 1

  Try to detect if compiler is running under Windows.
- #define FALSE 0

  For evaluation to FALSE of statements.
- #define MAX(a, b) ((a)>(b)?(a):(b))

  Obtains the maximum value between two numbers.
- #define MIN(a, b) ((a)<(b)?(a):(b))

  Obtains the minimum value between two numbers.
- #define PrintErr(...) do{fprintf(stderr, __VA_ARGS__);}while(0)

  printf-like macro that writes on stderr instead of stdout
- #define DelayMs(ms) usleep((ms)∗1000)

  Delay ms function, compatible with both Windows and Unix.

3.4.1 Detailed Description

Utility definitions and macros.

Author

Jesús Alonso (doragasu)

Date

2017

3.4.2 Macro Definition Documentation

3.4.2.1 TRUE

```
#define TRUE 1
```

Try to detect if compiler is running under Windows.

For evaluation to TRUE of statements

Definition at line 23 of file util.h.

3.5 wflash

Allows remotely programming ROMs to MegaWiFi cartridges, as well as performing other support functions like erasing, reading and getting Flash chip identifiers.

Data Structures

- struct WfMemRange

  Memory range definition.
- struct WfCmd

  Command definition.
- union WfBuf

  Command buffer. Allows accessing the buffer as a command or as raw data.
- struct WfData

  Local module data structure.

Macros

- #define WF_VERSION_MAJOR 0x00

  Major number of the commands implementation.
- #define WF_VERSION_MINOR 0x01

  Minor number of the commands implementation.
- #define WF_MAX_DATALEN 1440

  Maximum payload length.
- #define WF_HEADLEN 4

  Header lenght.
- #define WF_CHANNEL 1

  MegaWiFi channel.
- #define WF_CMD_OK 0

  OK reply code to a command.
- #define WF_CMD_ERROR 1

  ERROR reply code to a command.
- #define WF_OK 0

  Function completed successfully.
- #define WF_ERROR -1

  Function completed with error.

Enumerations

- enum {
  WF_CMD_VERSION_GET = 0, WF_CMD_ECHO, WF_CMD_ID_GET, WF_CMD_ERASE,
  WF_CMD_PROGRAM, WF_CMD_READ, WF_CMD_RUN, WF_CMD_AUTORUN,
  WF_CMD_MAX }

  Available commands.

Functions

- void WfInit (void)
- int WfConnect (char host[ ], uint16_t port)
- void WfClose (void)
- uint8_t ∗ WfBootVerGet (void)
- uint8_t ∗ WfFlashIdsGet (void)
- int WfFlashErase (uint32_t addr, uint32_t len)
- int WfFlash (uint32_t addr, uint32_t len, uint8_t data[ ])
- uint32_t WfRead (uint32_t addr, uint32_t len, uint8_t buf[ ])
- int WfBoot (uint32_t addr)
- int WfAutoRun (void)

3.5.1 Detailed Description

Allows remotely programming ROMs to MegaWiFi cartridges, as well as performing other support functions like erasing, reading and getting Flash chip identifiers.

WFlash command definitions.

Author

Jesús Alonso (doragasu)

Date

2017

Author

Jesús Alonso (doragasu)

3.5.2 Enumeration Type Documentation

3.5.2.1 anonymous enum

```
anonymous enum
```

Available commands.

Enumerator

WF_CMD_VERSION_GET	Get bootloader version.
WF_CMD_ECHO	Echo data.
WF_CMD_ID_GET	Get flash chip ID.
WF_CMD_ERASE	Erase range.
WF_CMD_PROGRAM	Program data.
WF_CMD_READ	Read data.
WF_CMD_RUN	Run from address.
WF_CMD_AUTORUN	Run from entry point in cart header.
WF_CMD_MAX	Maximum command value delimiter.

Definition at line 27 of file cmds.h.

3.5.3 Function Documentation

3.5.3.1 WfAutoRun()

```
int WfAutoRun (
            void  )
```

Boots the ROM automatically.

Returns

WF_OK if boot command completed, or WF_ERROR if command failed.

Note

Because the bootloader will only wait a small fraction of time before shutting down the WiFi module, it is possible that WF_ERROR is returned on a boot command completed successfully.
The address used to boot is read from the beginning of the NOTES field of the ROM header. The boot address is located there by the ROM upload utility.

Definition at line 309 of file wflash.c.

3.5.3.2 WfBoot()

```
int WfBoot (
            uint32_t addr )
```

Reads a data block from the specified Flash address

Parameters

in	*addr*	Address from where to boot the ROM.

Returns

WF_OK if boot command completed, or WF_ERROR if command failed.

Note

Because the bootloader will only wait a small fraction of time before shutting down the WiFi module, it is possible that WF_ERROR is returned on a boot command completed successfully.

Boots the ROM from the specified address.

Parameters

in	*addr*	Address from where to boot the ROM.

Returns

WF_OK if boot command completed, or WF_ERROR if command failed.

Note

Because the bootloader will only wait a small fraction of time before shutting down the WiFi module, it is possible that WF_ERROR is returned on a boot command completed successfully.

Definition at line 283 of file wflash.c.

3.5.3.3 WfBootVerGet()

```
uint8_t * WfBootVerGet (
             void  )
```

Obtains the version numbers of the bootloader.

Returns

A two byte array with the bootloader version numbers (the first is the major number and the second is the minor number), or NULL if the version numbers could not be obtained.

Definition at line 134 of file wflash.c.

3.5.3.4 WfClose()

```
void WfClose (
             void  )
```

Closes a previously established connection with a MegaWiFi host.

Definition at line 94 of file wflash.c.

3.5.3.5 WfConnect()

```
int WfConnect (
             char host[],
             uint16_t port )
```

Connects to specified MegaWiFi host (address/IP and port).

Parameters

in	*host*	Host name of the MegaWiFi node. Address name and IP are supported.
in	*port*	TCP port number of the MegaWiFi host.

Returns

WF_OK if connection is successful, WF_ERROR otherwise.

Definition at line 47 of file wflash.c.

3.5.3.6 WfFlash()

```
int WfFlash (
            uint32_t addr,
            uint32_t len,
            uint8_t data[] )
```

Programs a data block to the specified Flash address

Parameters

in	*addr*	Address to which the block will be written.
in	*len*	Length of the data block.
in	*data*	Data block to program to the Flash.

Returns

The number of bytes programmed to the Flash chip.

Definition at line 202 of file wflash.c.

3.5.3.7 WfFlashErase()

```
int WfFlashErase (
            uint32_t addr,
            uint32_t len )
```

Erases an address range of the flash chip.

Parameters

in	*addr*	Start address of the range to erase.
in	*len*	Length of the address to range.

Returns

WF_OK if connection is successful, WF_ERROR otherwise.

Warning

Definition at line 178 of file wflash.c.

3.5.3.8 WfFlashIdsGet()

```
uint8_t * WfFlashIdsGet (
             void  )
```

Obtains the Flash chip identifiers.

Returns

A four byte array with the Flash chip identifiers or NULL if the Flash chip identifiers could not be obtained. The returned numbers are:

1. The manufacturer ID
2. The chip ID
3. The chip ID (second byte)
4. The chip ID (third byte)

Definition at line 157 of file wflash.c.

3.5.3.9 WfInit()

```
void WfInit (
             void  )
```

Module initialization. Must be called once before using the module.

Definition at line 43 of file wflash.c.

3.5.3.10 WfRead()

```
uint32_t WfRead (
             uint32_t addr,
             uint32_t len,
             uint8_t buf[] )
```

Reads a data block from the specified Flash address

Parameters

`in`	*addr*	Address from which to start reading.
`in`	*len*	Length of the data block to read.
`in`	*buf*	Buffer where the readed data will be placed.

Returns

The number of bytes programmed to the Flash chip.

Definition at line 245 of file wflash.c.

Chapter 4

Data Structure Documentation

4.1 Flags Struct Reference

Commandline flags (for arguments without parameters).

Data Fields

-
```
union {
  uint16_t all
    Simultaneous access to all fields.
  struct {
    uint8_t verify:1
      Verify flash write if TRUE.
    uint8_t verbose:1
      Print extra information on screen.
    uint8_t flashId:1
      Show flash chip id.
    uint8_t erase:1
      Erase flash.
    uint8_t pushbutton:1
      Read pushbutton status.
    uint8_t dry:1
      Dry run.
    uint8_t boot:1
      Enter bootloader.
    uint8_t noPatch:1
      Do not patch the source ROM.
    uint8_t autoRun:1
      Run from entry point in cart header.
    uint8_t unused:7
  }
};
```
- int cols

  Number of columns of the terminal.

4.1.1 Detailed Description

Commandline flags (for arguments without parameters).

Definition at line 45 of file main.c.

The documentation for this struct was generated from the following file:

- main.c

4.2 MemImage Struct Reference

Structure containing a memory image (file, address and length)

Data Fields

- char ∗ file

  File name.
- uint32_t addr

  Memory address.
- uint32_t len

  Block length.

4.2.1 Detailed Description

Structure containing a memory image (file, address and length)

Definition at line 38 of file main.c.

The documentation for this struct was generated from the following file:

- main.c

4.3 RomHead Struct Reference

ROM header information structure, including interrupt vectors.

Data Fields

- uint32_t **stackPtr**
- uint32_t **entryPoint**
- uint32_t **busErrEx**
- uint32_t **addrErrEx**
- uint32_t **illegalInstrEx**
- uint32_t **zeroDivEx**
- uint32_t **chkInstr**
- uint32_t **trapvInstr**
- uint32_t **privViol**
- uint32_t **trace**
- uint32_t **line1010Emu**
- uint32_t **line1111Emu**
- uint32_t **errEx** [13]
- uint32_t **int0**
- uint32_t **extInt**
- uint32_t **int1**
- uint32_t **hInt**
- uint32_t **int2**
- uint32_t **vInt**
- uint32_t **int3** [33]
- char console [16]

  Console Name (16) ∗/.
- char copyright [16]

  Copyright Information (16) ∗/.
- char title_local [48]

  Domestic Name (48) ∗/.
- char title_int [48]

  Overseas Name (48) ∗/.
- char serial [14]

  Serial Number (2, 12) ∗/.
- uint16_t checksum

  Checksum (2) ∗/.
- char IOSupport [16]

  I/O Support (16) ∗/.
- uint32_t rom_start

  ROM Start Address (4) ∗/.
- uint32_t rom_end

  ROM End Address (4) ∗/.
- uint32_t ram_start

  Start of Backup RAM (4) ∗/.
- uint32_t ram_end

  End of Backup RAM (4) ∗/.
- char sram_sig [2]

  "RA" for save ram (2) ∗/
- uint16_t sram_type

  0xF820 for save ram on odd bytes (2) ∗/
- uint32_t sram_start

  SRAM start address - normally 0x200001 (4) ∗/.
- uint32_t sram_end

  SRAM end address - start + 2∗sram_size (4) ∗/.

- char modem_support [12]

  Modem Support (24) ∗/.
- char notes [40]

  Memo (40) ∗/.
- char region [16]

  Country Support (16) ∗/.

4.3.1 Detailed Description

ROM header information structure, including interrupt vectors.

rom_head: Megadrive ROM patch module.

Currently the module only allows to patch the ROM header, adding the wflash header information, and changing the entry point, for the bootloader to be launched instead of the flashed ROM.

Definition at line 12 of file rom_head.c.

The documentation for this struct was generated from the following file:

- rom_head.c

4.4 WfBuf Union Reference

Command buffer. Allows accessing the buffer as a command or as raw data.

```
#include <cmds.h>
```

Data Fields

- uint8_t data [WF_MAX_DATALEN]

  8-bit data
- uint16_t wdata [WF_MAX_DATALEN/2]

  16-bit data
- uint32_t dwdata [WF_MAX_DATALEN/4]

  32-bit data
- WfCmd cmd

  Command.

4.4.1 Detailed Description

Command buffer. Allows accessing the buffer as a command or as raw data.

Definition at line 67 of file cmds.h.

The documentation for this union was generated from the following file:

- cmds.h

4.5 WfCmd Struct Reference

Command definition.

```
#include <cmds.h>
```

Data Fields

- uint16_t cmd

    Command code.
- uint16_t len

    Command length.
-

```
union {
  uint8_t data [WF_MAX_DATALEN - 4]
    8-bit data
  uint16_t wdata [(WF_MAX_DATALEN - 4)/2]
    16-bit data
  uint32_t dwdata [(WF_MAX_DATALEN - 4)/4]
    32-bit data
  WfMemRange mem
    Memory range.
};
```

4.5.1 Detailed Description

Command definition.

Definition at line 51 of file cmds.h.

The documentation for this struct was generated from the following file:

- cmds.h

4.6 WfData Struct Reference

Local module data structure.

Data Fields

- WfBuf buf

  Data buffer.
- int sock

  Client socket.
- struct in_addr ∗ srvAddr

  Server address.
-
```
union {
   uint16_t flags
     Various flags.
   struct {
      uint16_t connected:1
        Connected to server if TRUE.
      uint16_t reserved:15
        Unused flags.
   }
};
```

4.6.1 Detailed Description

Local module data structure.

Definition at line 24 of file wflash.c.

The documentation for this struct was generated from the following file:

- wflash.c

4.7 WfMemRange Struct Reference

Memory range definition.

`#include <cmds.h>`

Data Fields

- uint32_t addr

  Start address of the range.
- uint32_t len

  Length of the memory range.

4.7.1 Detailed Description

Memory range definition.

Definition at line 45 of file cmds.h.

The documentation for this struct was generated from the following file:

- cmds.h

Index