mk3-prog-cli

Generated by Doxygen 1.8.13

Contents

Chapter 1

Mojo-NES MK3 programmer Command Line Interface

Command Line Interface for the Awesome Mojo-NES MKIII programmer.

This utility allows to manage mojo-nes-mk3 cartridges, using a mojo-nes-mk3 programmer. The utility allows to program and read flash and RAM chips. A driver system allows to support several mapper chip implementations.

Building

You will need a working GNU GCC compiler and an Awesome Mojo-NES MKIII programmer to burn the ROM to a MegaWiFi cartridge. You will also need to install `libftdi` and `libmpsse` libraries, including development headers. Once you have your development environment properly installed, the makefile should do all the hard work for you. Just browse the Makefile to suit it to your dev environment and type:

```
$ make
```

The mk3-prog program should be built, sitting in the working directory, ready to use.

Usage

Once you have plugged a Mojo-NES MKIII cartridge into an Awesome Mojo-NES MKIII Programmer, you can use mk3-prog. The command line application invocation must be as follows:

```
$ mk3-prog [option1 [option1_arg]] [...] [optionN [optionN_arg]]
```

The options (option1 $\sim$ optionN) can be any combination of the ones listed below. Options must support short and long formats. Depending on the used option, and option argument (option_arg) must be supplied. Several options can be used on the same command, as long as the combination makes sense (e.g. it does make sense using the flash and verify options together, but using the help option with the flash option doesn't make too much sense).

Option	Description
-f, –firm-ver	Get programmer firmware version
-c, –flash-chr <arg>	Flash file to CHR ROM
-p, –flash-prg <arg>	Flash file to PRG ROM
-C, –read-chr <arg>	Read CHR ROM to file
-P, –read-prg <arg>	Read PRG ROM to file
-e, –erase-chr	Erase CHR Flash
-E, –erase-prg	Erase PRG Flash
-s, –chr-sec-er <arg>	Erase CHR flash sector
-S, –prg-sec-er <arg>	Erase PRG flash sector
-V, –verify	Verify flash after writing file
-i, –flash-id	Obtain flash chips identifiers
-R, –read-ram <arg>	Read data from RAM chip
-W, –write-ram <arg>	Write data to RAM chip
-b, –fpga-flash <arg>	Upload bitfile to FPGA, using .xcf file
-a, –cic-flash <arg>	AVR CIC firmware flash
-F, –firm-flash <arg>	Flash programmer firmware
-m, –mpsse-if <arg>	Set MPSSE interface number
-M, –mapper <arg>	Set mapper: 1-NOROM, 2-MMC3, 3-NFROM
-d, –dry-run	Dry run: don't actually do anything
-r, –version	Show program version
-v, –verbose	Show additional information
-h, –help	Print help screen and exit

The <arg> text indicates that the option takes an input argument. For the options requiring an argument that represents a memory image file, or a memory address, the syntax is as follows:

- Memory image file: file_name[:start_addr[:length]]. Along with the file name, optional address and length fields can be added, separated by the colon (:) character, resulting in the following format:
- Address: Specifies an address related to the command (e.g. the address to which to flash a cartridge ROM or WiFi firmware blob).

Some examples of the command invocation and its arguments are:

- `$ mk3-prog -VeEc chr_rom_file -p prg_rom_file` → Erases entire cartridge (both CHR and PRG flash chips), flashes chr_rom_file to CHR flash, prg_rom_file to PRG flash, and verifies the writes.
- `$ mk3-prog --erase_chr -c chr_rom_file:0x1000` → Erases entire CHR flash chip and flashes contents of chr_rom_file to CHR flash, starting at address 0x1000.
- `$ mk3-prog -S 0x10000` → Erases PRG flash sector containing 0x100000 address.
- `$ mk3-prog -Vp prg_rom_file:0x10000:32768` → Flashes 32 KiB of prg_rom_file to address 0x10000, and verifies the operation.
- `$ mk3-prog --read_chr chr_rom_file::1048576` → Reads 1 MiB of the CHR flash chip, and writes it to chr_rom_file. Note that if you want to specify length but do not want to specify address, you have to use two colon characters before length. This way, missing address argument is interpreted as 0.

Authors

This program has been written by doragasu.

Contributions

Contributions are welcome. If you find a bug please open an issue, and if you have implemented a cool feature/improvement, please send a pull request.

License

This program is provided with NO WARRANTY, under the GPLv3 license.

Chapter 2

Todo List

Module main

- Currently programmer configuration file and programmer type are harcoded inside the binary. It would be advisable to read them as a command line switch, or from a configuration file, or from an environment variable.

Global SC_SPI_CLK

Test if CLK can be increased by changing crystal oscillator to a 16 MHz one.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Module Documentation

6.1 avrflash

Flashes elf files to programmer MCU or cart CIC.

Functions

- int AvrFlash (const char path[ ], const char cfg[ ], const char mcu[ ], const char file[ ], const char prog[ ])

6.1.1 Detailed Description

Flashes elf files to programmer MCU or cart CIC.

Author

Jesus Alonso (doragasu)

Date

2016

6.1.2 Function Documentation

6.1.2.1 AvrFlash()

```
int AvrFlash (
            const char path[],
            const char cfg[],
            const char mcu[],
            const char file[],
            const char prog[] )
```

Uses avrdude to flash specified firmware file.

Parameters

in	*path*	Path of the avrdude binary.
in	*cfg*	avrdude configuration file (-C avrdude switch).
in	*mcu*	Microcontroller (-p avrdude switch argument).
in	*file*	Firmware to flash, elf format needed for the fuses to work.
in	*prog*	Programmer (-c avrdude switch).

Returns

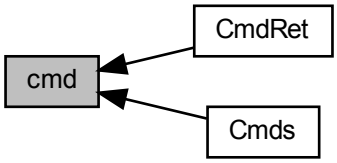
-1 if spawning avrdude failed, -2 if avrdude didn't exit properly, -3 if malloc failed or avrdude return code if otherwise.

Definition at line 55 of file avrflash.c.

6.2 cmd

Allows sending commands to the programmer, and receiving results.

Collaboration diagram for cmd:



Modules

- CmdRet

  Return values for functions in this module and error codes.
- Cmds

  Supported system commands. Also includes OK and ERROR codes.

Data Structures

- struct CmdRdWrHdr

  Command header for memory read and write commands.
- struct CmdErase

  Command header for erase commands.
- union Cmd

  Generic command request.
- struct CmdFlashId

  Flash chip identification information.
- struct CmdRepEmpty

  Empty command response.
- struct CmdRepFwVer

  Firmware version command response.
- struct CmdRepFlashId

  Flash ID command response.
- union CmdRep

  Generic reply to a command request.

Macros

- #define CMD_MAXLEN 32

  Maximum length of a command.
- #define CMD_SRAM_MAXLEN 8∗1024

  Maximum SRAM length is 8 KiB.
- #define CmdRepFree(pRep) free(pRep)

Enumerations

- enum CmdMapper { CMD_MAPPER_MMC3X = 0, CMD_MAPPER_TKROM }

  Supported mappers.

Functions

- int CmdInit (unsigned int channel)
- int CmdSend (const Cmd *cmd, uint8_t cmdLen, CmdRep **rep)
- int CmdSendLongCmd (const Cmd *cmd, uint8_t cmdLen, const uint8_t *data, int dataLen, CmdRep **rep)
- int CmdSendLongRep (const Cmd *cmd, uint8_t cmdLen, CmdRep **rep, uint8_t *data, int recvLen)

6.2.1 Detailed Description

Allows sending commands to the programmer, and receiving results.

Author

Jesus Alonso (doragasu)

Date

2016

6.2.2 Macro Definition Documentation

6.2.2.1 CmdRepFree

```
#define CmdRepFree(
             pRep ) free(pRep)
```

It looks like libmpsse does NOT free returned responses (it is at least NOT documented), so we must free the memory on our own. As this can change in the future, we must be extra careful with this "feature".

Definition at line 164 of file cmd.h.

6.2.3 Enumeration Type Documentation

6.2.3.1 CmdMapper

```
enum CmdMapper
```

Supported mappers.

Enumerator

CMD_MAPPER_MMC3X	MMC3X mapper.
CMD_MAPPER_TKROM	TKROM-like mapper.

Definition at line 49 of file cmd.h.

6.2.4 Function Documentation

6.2.4.1 CmdInit()

```
int CmdInit (
            unsigned int channel )
```

Module initialization. Call before using any other function.

Parameters

in	*channel*	Channel number of the FTX232H device to use for communications.

Returns

CMD_OK if the command completed successfully. CMD_ERROR otherwise.

Definition at line 26 of file cmd.c.

6.2.4.2 CmdSend()

```
int CmdSend (
            const Cmd * cmd,
            uint8_t cmdLen,
            CmdRep ** rep )
```

Sends a command, and obtains the command response.

Parameters

in	*cmd*	Command to send.
in	*cmdLen*	Command length.
out	*rep*	Response to the sent command.

Returns

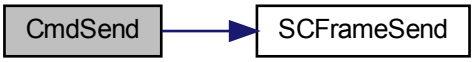
CMD_OK if the command completed successfully. CMD_ERROR otherwise.

Note

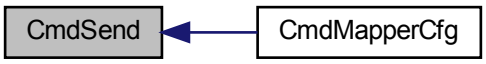
This function does not allow sending or receiving commands with long payloads. Use CmdSendLongCmd() or CmdSendLongRep() for long payloads.

Definition at line 43 of file cmd.c.

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.4.3 CmdSendLongCmd()

```
int CmdSendLongCmd (
            const Cmd * cmd,
            uint8_t cmdLen,
            const uint8_t * data,
            int dataLen,
            CmdRep ** rep )
```

Sends a command with a long data payload, and obtains the command response.

Parameters

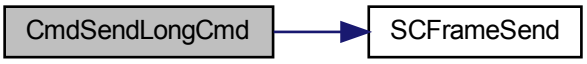
in	*cmd*	Command to send.
in	*cmdLen*	Command length.
in	*data*	Long data payload to send.
in	*dataLen*	Length of the data payload.
out	*rep*	Response to the sent command.

Returns

CMD_OK if the command completed successfully. CMD_ERROR otherwise.

Definition at line 63 of file cmd.c.

Here is the call graph for this function:



Here is the caller graph for this function:

6.2.4.4 CmdSendLongRep()

```
int CmdSendLongRep (
            const Cmd * cmd,
            uint8_t cmdLen,
            CmdRep ** rep,
            uint8_t * data,
            int recvLen )
```

Sends a command requiring a long response payload.

Parameters

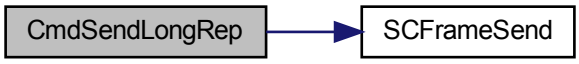
in	*cmd*	Command to send.
in	*cmdLen*	Command length.
out	*rep*	Response to the sent command.
in	*data*	Long data payload to receive.
in	*recvLen*	Length of payload to receive.

Returns

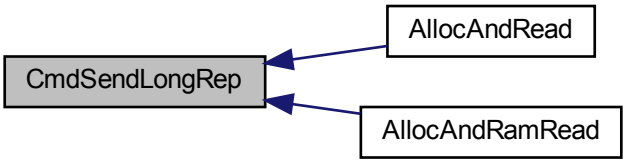
Length of the received payload if OK, CMD_ERROR otherwise.

Definition at line 96 of file cmd.c.

Here is the call graph for this function:



Here is the caller graph for this function:



6.3 latticeflash

Flash specified xcf file to FPGA.

Functions

- int LatticeFlash (const char prgcmd[ ], const char xcf[ ])

6.3.1 Detailed Description

Flash specified xcf file to FPGA.

Author

Jesus Alonso (doragasu)

Date

2016

6.3.2 Function Documentation

6.3.2.1 LatticeFlash()

```
int LatticeFlash (
            const char prgcmd[],
            const char xcf[] )
```

Uses lattice prgcmd to burn xcf file to FPGA

Parameters

in	*prgcmd*	Complete path to lattice programmer.
in	*xcf*	XCF input file to burn to FPGA.

Returns

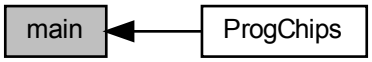
-1 if spawning avrdude failed, -2 if avrdude didn't exit properly, -3 if any other error occurred, or avrdude return code if otherwise.

Definition at line 34 of file latticeflash.c.

6.4 main

Command Line Interface for the mojo-nes-mk3 programmer.

Collaboration diagram for main:



Modules

- ProgChips

  Programmable flash chips.

Data Structures

- struct MemImage

  Definition of a file representing a memory image.
- union Flags

  Supported option flags.

Macros

- #define VERSION_MAJOR 0x00

  Major version of the program.
- #define VERSION_MINOR 0x04

  Minor version of the program.
- #define MAX_FILELEN 255

  Maximum file length.
- #define PROG_ERASE_FULL 0xFFFFFF

  Return value for Erase operation error.
- #define PROG_SRAM_BASE 0x6000

  SRAM base address.
- #define PROG_SRAM_LEN (8 ∗ 1024)

  SRAM length.
- #define AVR_CHIP_MCU "m8515"

  Definition of the chip of the programmer (ATMEGA8515)
- #define AVR_CHIP_CIC "t13"

  Definition of the CIC chip (ATTINY13)
- #define AVR_PATH "/usr/bin/avrdude"

  avrdude binary path
- #define AVR_PROG_CFG "/usr/share/mk3-prog/mk3prog.conf"

Configuration file of the programmer to use.

- #define AVR_PROG_MCU "mk3prog-mcu"

  MCU programmer defined in mk3prog.conf.
- #define AVR_PROG_CIC "mk3prog-cic"

  CIC programmer defined in mk3prog.conf.
- #define LATT_PROG_PATH "/usr/local/diamond/3.7_x64/bin/lin64/pgrcmd"

  Path to the FPGA programmer program/script.
- #define CMD_SET_ADDR(field, addr)

  Copies the specified address to a byte array field.
- #define CMD_SET_LEN(field, len)

  Copies the command length to the specified byte array field.
- #define CondPrintf(cond, ...) do{if(cond) printf(__VA_ARGS__);}while(0)

  Printf-like macro that prints only if condition is TRUE.
- #define PrintVerb(...) do{if(f.verbose) printf(__VA_ARGS__);fflush(stdout);}while(0)

  Printf-like macro that prints only if f.verbose is TRUE.
- #define try(code, error)

Functions

- uint8_t * AllocAndRead (uint8_t chip, MemImage *f, unsigned int cols)
- uint8_t * AllocAndRamWrite (MemImage *f)
- uint8_t * AllocAndRamRead (MemImage *f)
- int CmdMapperCfg (CmdMapper mapper)
- int main (int argc, char **argv)

6.4.1 Detailed Description

Command Line Interface for the mojo-nes-mk3 programmer.

This utility allows to manage mojo-nes-mk3 cartridges, usina a mojo-nes-mk3 programmer. The utility allows to program and read flash and RAM chips. A driver system allows to support several mapper chip implementations.

Todo

- Currently programmer configuration file and programmer type are harcoded inside the binary. It would be advisable to read them as a command line switch, or from a configuration file, or from an environment variable.

Author

Jesus Alonso (doragasu)

Date

2016

6.4.2 Macro Definition Documentation

6.4.2.1 CMD_SET_ADDR

```
#define CMD_SET_ADDR(
                field,
                addr )
```

Value:

```
do{ \
    (field)[0] = (addr)>>16;            \
    (field)[1] = (addr)>>8;             \
    (field)[2] = (addr);                \
}while(0)
```

Copies the specified address to a byte array field.

Definition at line 89 of file main.c.

6.4.2.2 CMD_SET_LEN

```
#define CMD_SET_LEN(
                field,
                len )
```

Value:

```
do{ \
    (field)[0] = (len)>>8;          \
    (field)[1] = (len);             \
}while(0)
```

Copies the command length to the specified byte array field.

Definition at line 96 of file main.c.

6.4.2.3 try

```
#define try(
                code,
                error )
```

Value:

```
do{if ((code) < 0) {PrintErr(error);errCode = -1; \
        goto dealloc_exit;}}while(0)
```

Evaluate a value. If less than 0, print the specified error. Else do nothing.

Parameters

in	*code*	Code to evaluate.
in	*error*	Error message to print if code < 0.

Definition at line 669 of file main.c.

6.4.3 Function Documentation

6.4.3.1 AllocAndRamRead()

```
uint8_t* AllocAndRamRead (
            MemImage * f )
```

Allocates a RAM buffer, and reads range specified in MemImage input from the in-cart RAM chip.

Parameters

in	*f*	Memory image with the range to read.

Returns

Pointer to the raw data of the allocated and read image file, or NULL if error occurred.

Warning

Buffer must be externally deallocated when no longer needed, using free().

Definition at line 608 of file main.c.

Here is the call graph for this function:

6.4.3.2 AllocAndRamWrite()

```
uint8_t* AllocAndRamWrite (
            MemImage * f )
```

Allocates a RAM buffer, reads the specified MemImage file, and writes it to the in-cart RAM chip.

Parameters

in	*f*	Memory image with the range to write and file to read.

Returns

Pointer to the raw data of the allocated and read image file, or NULL if error occurred.

Warning

Buffer must be externally deallocated when no longer needed, using free().

Definition at line 539 of file main.c.

Here is the call graph for this function:

6.4.3.3 AllocAndRead()

```
uint8_t* AllocAndRead (
            uint8_t chip,
            MemImage * f,
            unsigned int cols )
```

Allocates a RAM buffer, and reads range specified in MemImage input from the specified Flash chip to the allocated buffer.

Parameters

in	*chip*	Flash chip to read.
in	*f*	Memory image with the range to read.
in	*cols*	Number of columns of the terminal, used to draw the status bar.

Returns

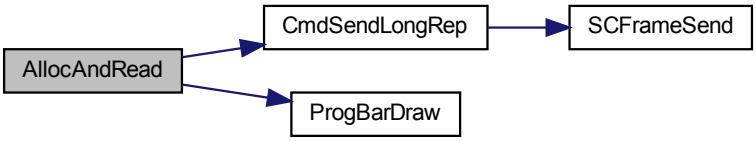
Pointer to the raw data of the allocated and read image file, or NULL if error occurred.

Warning

Buffer must be externally deallocated when no longer needed, using free().

Definition at line 482 of file main.c.

Here is the call graph for this function:



6.4.3.4 CmdMapperCfg()

```
int CmdMapperCfg (
            CmdMapper mapper )
```

Send mapper configuration command.

Parameters

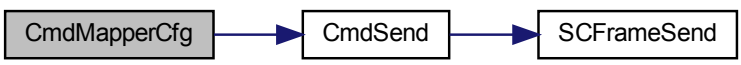
in	*mapper*	Mapper to set.

Returns

0 if OK, -1 if error.

Definition at line 651 of file main.c.

Here is the call graph for this function:



6.4.3.5 main()

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

Program entry point. Parses input parameters and perform requested actions.

Parameters

in	*argc*	Number of input parameters.
in	*argv*	Array of input parameters strings to be parsed.

Returns

0 if OK, less than 0 if error.

Option index, for command line options parsing

Character returned by getopt_long()

Definition at line 682 of file main.c.

6.5 progbar

Draw progress bars for command line applications.

Functions

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

6.5.1 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

6.5.2 Function Documentation

6.5.2.1 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).

Definition at line 31 of file progbar.c.

Here is the caller graph for this function:

6.6 pspawn

Spawns a child process using a pseudo terminal.

Functions

- int pspawn (const char *file, char *const arg[ ])

6.6.1 Detailed Description

Spawns a child process using a pseudo terminal.

Spawns a child process using a pseudo-terminal to avoid input/output buffering. The standard output of the child process is redirected to the standard output of the parent process.

Author

Jesus Alonso (doragasu)

Date

2016

6.6.2 Function Documentation

6.6.2.1 pspawn()

```
int pspawn (
            const char * file,
            char *const arg[] )
```

Spawns a child process using a pseudo-terminal.

Parameters

in	*file*	File name of the process to spawn.
in	*arg*	Array of the arguments passed to the new process. By convention, argument 0 is the process name.

Returns

0 if OK, -1 if there was an error when trying to spawn the new process.

Spawns a child process using a pseudo-terminal.

Parameters

in	*file*	File name of the process to spawn.
in	*arg*	Array of the arguments passed to the new process. By convention, argument 0 is the process name.

Returns

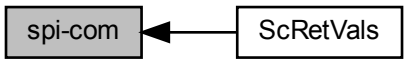
-1 if the process spawn failed, -2 if child didn't exit properly, or the process return code otherwise.

Definition at line 31 of file pspawn.c.

6.7 spi-com

Handles SPI and framing for communications.

Collaboration diagram for spi-com:



Modules

- ScRetVals

  Return values for this module.

Macros

- #define SC_SPI_MODE SPI0

  SPI mode for using with MPSSE library.
- #define SC_SPI_CLK 100000
- #define SC_SOF 0x7E

  Start of frame marker.
- #define SC_EOF 0x7D

  En of frame marker.
- #define SC_VID 0x0403

  USB Vendor ID of the programmer board.
- #define SC_PID 0x6010

  USB Device ID of the programmer board.
- #define SC_IFACE IFACE_B

  FT2232 interface used to communicate with the microcontroller.
- #define SC_MAX_DATALEN 32

  Maximum data payload is 32 bytes long.

Functions

- struct mpsse_context ∗ SCInit (unsigned int channel)
- int SCFrameSend (struct mpsse_context ∗mpsse, char ∗data, uint16_t datalen)
- char ∗ SCFrameRecv (struct mpsse_context ∗mpsse, uint8_t ∗maxlen)

6.7.1 Detailed Description

Handles SPI and framing for communications.

Frames are delimited by the SOF/EOF characters. Following SOF, payload length in sent using 1 byte. Then data follows, and finally EOF character ends transmission.

Author

Jesus Alonso (doragasu)

Date

2016

Note

Uses open source mpsse library to interface FT2232 in MSPSSE mode.

6.7.2 Macro Definition Documentation

6.7.2.1 SC_SPI_CLK

```
#define SC_SPI_CLK 100000
```

Maximum CLK for atmega8515 as SPI slave, is FOSC/4=12MHz/4

Todo Test if CLK can be increased by changing crystal oscillator to a 16 MHz one.

Definition at line 29 of file spi-com.h.

6.7.3 Function Documentation

6.7.3.1 SCFrameRecv()

```
char* SCFrameRecv (
            struct mpsse_context * mpsse,
            uint8_t * maxlen )
```

Receives data through the MPSSE interface, using a tiny framing protocol.

Parameters

in	*mpsse*	Handler of the previously opened MPSSE interface.
in,out	*maxlen*	Maximum length of the data payload to receive. On function return, holds the number of bytes received.

Returns

Pointer to the received data, or NULL if reception failed.

Warning

It looks like libmpsse does NOT free the received data buffer, so we have to make sure to free it ourselves when not needed! Other option is maybe using Fast functions inside fast.c

Definition at line 95 of file spi-com.c.

6.7.3.2 SCFrameSend()

```
int SCFrameSend (
            struct mpsse_context * mpsse,
            char * data,
            uint16_t datalen )
```

Sends data through the MPSSE interface, using a tiny framing protocol.

Parameters

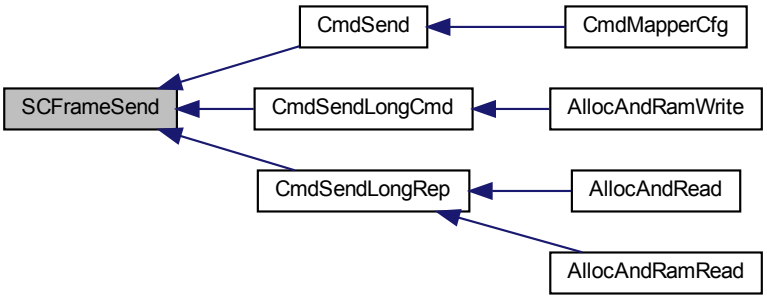
in	*mpsse*	Handler of the previously opened MPSSE interface.
in	*data*	Data payload to send.
in	*datalen*	Length of the data payload to send in bytes.

Returns

SC_OK on success, SC_ERROR on fail.

Definition at line 59 of file spi-com.c.

Here is the caller graph for this function:



6.7.3.3 SCInit()

```
struct mpsse_context* SCInit (
            unsigned int channel )
```

Module initialization. Call this function to obtain the handler needed to call any other function in this module.

Parameters

in	*channel*	Channel number of the FT2232 device to open.

Returns

The handler of the opened FT2232 MPSSE interface, or NULL if opening the interface failed.

Definition at line 34 of file spi-com.c.

6.8 util

General purpose utilities.

Macros

- #define TRUE 1

  TRUE value definition for logic comparisons.
- #define FALSE 0

  FALSE value definition for logic comparisons.
- #define MAX(a, b) ((a)>(b)?(a):(b))

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

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

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

  Delay the specified amount of milliseconds.

6.8.1 Detailed Description

General purpose utilities.

Author

Jesus Alonso (doragasu)

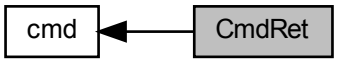
Date

2015

6.9 CmdRet

Return values for functions in this module and error codes.

Collaboration diagram for CmdRet:



Macros

- #define CMD_OK 0

  Function completed successfully.
- #define CMD_ERROR -1

  Function completed with error.

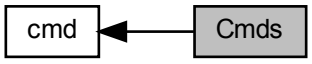
6.9.1 Detailed Description

Return values for functions in this module and error codes.

6.10 Cmds

Supported system commands. Also includes OK and ERROR codes.

Collaboration diagram for Cmds:



Macros

- #define CMD_REP_OK 0
  OK reply code.
- #define CMD_FW_VER 1
  Get programmer firmware version.
- #define CMD_CHR_WRITE 2
  Write to CHR flash.
- #define CMD_PRG_WRITE 3
  Write to PRG flash.
- #define CMD_CHR_READ 4
  Read from CHR flash.
- #define CMD_PRG_READ 5
  Read from PRG flash.
- #define CMD_CHR_ERASE 6
  Erase CHR flash (entire or sectors)
- #define CMD_PRG_ERASE 7
  Erase PRG flash (entire or sectors)
- #define CMD_FLASH_ID 8
  Get flash chips identifiers.
- #define CMD_RAM_WRITE 9
  Write data to cartridge SRAM.
- #define CMD_RAM_READ 10
  Read data from cartridge SRAM.
- #define CMD_MAPPER_SET 11
  Configure cartridge mapper.
- #define CMD_REP_ERROR 255
  Error reply code.

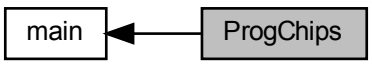
6.10.1 Detailed Description

Supported system commands. Also includes OK and ERROR codes.

6.11 ProgChips

Programmable flash chips.

Collaboration diagram for ProgChips:



Macros

- #define PROG_CHIP_CHR 0

  Character ROM chip (CHR ROM)
- #define PROG_CHIP_PRG 1

  Program ROM chip (PRG ROM)
- #define PROG_CHIP_MAX PROG_CHIP_PRG

  Last value for programmable chips.

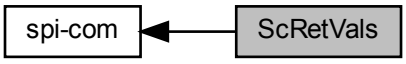
6.11.1 Detailed Description

Programmable flash chips.

6.12 ScRetVals

Return values for this module.

Collaboration diagram for ScRetVals:



Macros

- #define SC_OK MPSSE_OK

  OK status (0)
- #define SC_ERROR MPSSE_FAIL

  Error status (-1)

6.12.1 Detailed Description

Return values for this module.

Chapter 7

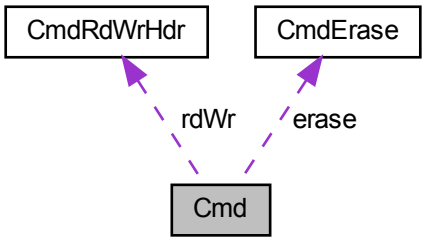
Data Structure Documentation

7.1 Cmd Union Reference

Generic command request.

`#include <cmd.h>`

Collaboration diagram for Cmd:



Data Fields

- uint8_t data [CMD_MAXLEN]

  Raw data (32 bytes max)
- uint8_t command

  Command code.
- CmdRdWrHdr rdWr

  Read/write request.
- CmdErase erase

  Erase request.

7.1.1 Detailed Description

Generic command request.

Definition at line 68 of file cmd.h.

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

- cmd.h

7.2 CmdErase Struct Reference

Command header for erase commands.

`#include <cmd.h>`

Data Fields

- uint8_t cmd
  Command code.
- uint8_t sectAddr [3]
  Address to erase, Full chip if 0xFFFFFF.

7.2.1 Detailed Description

Command header for erase commands.

Definition at line 62 of file cmd.h.

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

- cmd.h

7.3 CmdFlashId Struct Reference

Flash chip identification information.

`#include <cmd.h>`

Data Fields

- uint8_t manId
  Manufacturer ID.
- uint8_t devId [3]
  Device ID.

7.3.1 Detailed Description

Flash chip identification information.

Definition at line 76 of file cmd.h.

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

- cmd.h

7.4 CmdRdWrHdr Struct Reference

Command header for memory read and write commands.

```
#include <cmd.h>
```

Data Fields

- uint8_t cmd

  Command code.
- uint8_t addr [3]

  Address to read/write.
- uint8_t len [2]

  Length to read/write.

7.4.1 Detailed Description

Command header for memory read and write commands.

Definition at line 55 of file cmd.h.

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

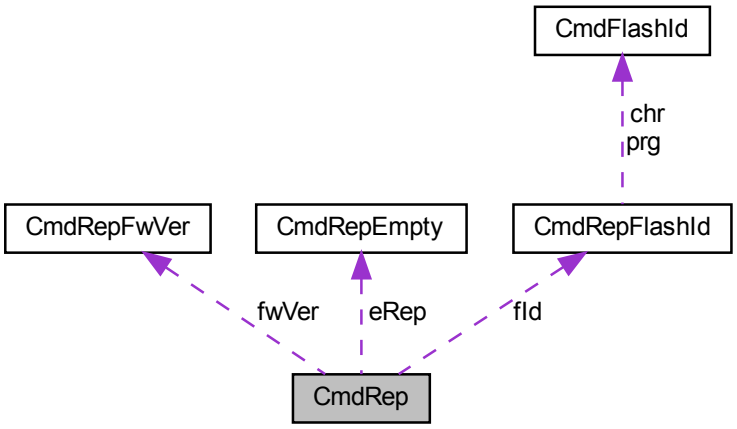
- cmd.h

7.5 CmdRep Union Reference

Generic reply to a command request.

`#include <cmd.h>`

Collaboration diagram for CmdRep:



Data Fields

- uint8_t data [CMD_MAXLEN]
  Raw data (up to 32 bytes)
- uint8_t command
  Command code.
- CmdRepEmpty eRep
  Empty command response.
- CmdRepFlashId fId
  Flash ID command response.
- CmdRepFwVer fwVer
  Firmware version command response.

7.5.1 Detailed Description

Generic reply to a command request.

Definition at line 102 of file cmd.h.

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

- cmd.h

7.6 CmdRepEmpty Struct Reference

Empty command response.

`#include <cmd.h>`

Data Fields

- uint8_t code

  Response code (OK/ERROR)

7.6.1 Detailed Description

Empty command response.

Definition at line 82 of file cmd.h.

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

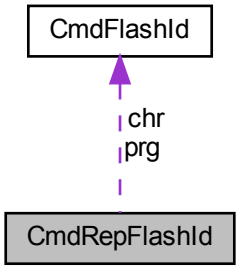
- cmd.h

7.7 CmdRepFlashId Struct Reference

Flash ID command response.

`#include <cmd.h>`

Collaboration diagram for CmdRepFlashId:



Data Fields

- uint8_t code
  Command code.
- uint8_t pad
  Padding.
- CmdFlashId prg
  PRG Flash information.
- CmdFlashId chr
  CHR Flash information.

7.7.1 Detailed Description

Flash ID command response.

Definition at line 94 of file cmd.h.

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

- cmd.h

7.8 CmdRepFwVer Struct Reference

Firmware version command response.

```
#include <cmd.h>
```

Data Fields

- uint8_t code
  Response code (OK/ERROR)
- uint8_t ver_major
  Major version number.
- uint8_t ver_minor
  Minor version number.

7.8.1 Detailed Description

Firmware version command response.

Definition at line 87 of file cmd.h.

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

- cmd.h

7.9 Flags Union Reference

Supported option flags.

Data Fields

- uint32_t all
  Access to all flags.
- struct {
  uint8_t fwVer:1
  Get firmware version.
  uint8_t verify:1
  Verify flashed files.
  uint8_t verbose:1
  Verbose operation.
  uint8_t flashId:1
  Get flash chip IDs.
  uint8_t chrErase:1
  Erase CHR flash.
  uint8_t prgErase:1
  Erase PRG flash.
  uint8_t dry:1
  Dry run.
  };

7.9.1 Detailed Description

Supported option flags.

Definition at line 116 of file main.c.

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

- main.c

7.10 MemImage Struct Reference

Definition of a file representing a memory image.

Data Fields

- char * file
  Image file name.
- uint32_t addr
  Memory address of the image.
- uint32_t len
  Length of the memory image.

7.10.1 Detailed Description

Definition of a file representing a memory image.

Definition at line 109 of file main.c.

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

- main.c

Chapter 8

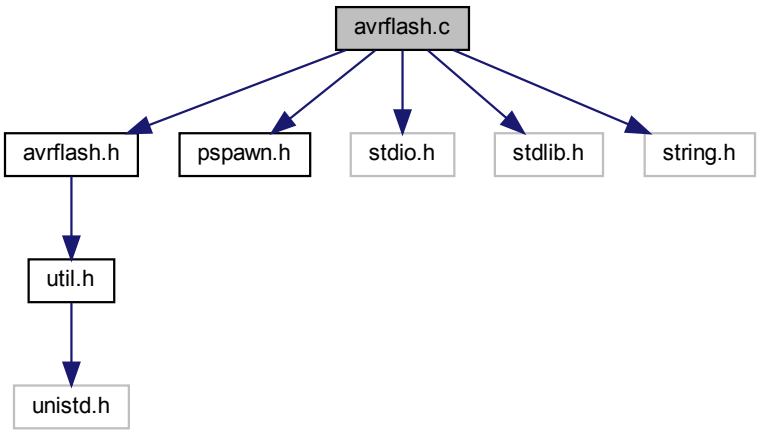
File Documentation

8.1 avrflash.c File Reference

Flashes elf files to programmer MCU or cart CIC.

```
#include "avrflash.h"
#include "pspawn.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for avrflash.c:



Macros

- #define AVR_AVRDUDE_POS 0

  Position in avrflash argument array of executable file.
- #define AVR_FLASH_MCU_POS 2

Position in avrflash argument array of the MCU to flash.

- #define AVR_FLASH_CFG_POS 4

  Position in avrflash argument array of the programmer configuration file.

- #define AVR_FLASH_PROG_POS 6

  Position in avrflash argument array of the programmer to use.

- #define AVR_FLASH_CMD_POS 8

  Position in avrFlash argument array of the flash command.

- #define AVR_FUSEH_CMD_POS 10

  Position in avrFlash argument array of the high fuse command.

- #define AVR_FUSEL_CMD_POS 12

  Position in avrFlash argument array of the low fuse command.

- #define AVR_FLASH_CMD "flash:w:"

  avrdude command to write to flash

- #define AVR_FUSEH_CMD "hfuse:w:"

  avrdude command to write higher fuse

- #define AVR_FUSEL_CMD "lfuse:w:"

  avrdude command to write lower fuse

Functions

- int AvrFlash (const char path[ ], const char cfg[ ], const char mcu[ ], const char file[ ], const char prog[ ])

8.1.1 Detailed Description

Flashes elf files to programmer MCU or cart CIC.

Author

Jesus Alonso (doragasu)

Date

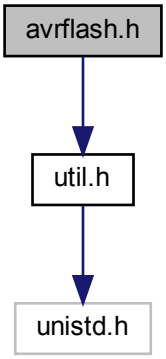
2016

8.2 avrflash.h File Reference

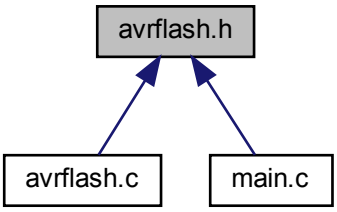
Flashes elf files to programmer MCU or cart CIC.

```
#include "util.h"
```

Include dependency graph for avrflash.h:



This graph shows which files directly or indirectly include this file:



Functions

- int AvrFlash (const char path[ ], const char cfg[ ], const char mcu[ ], const char file[ ], const char prog[ ])

8.2.1 Detailed Description

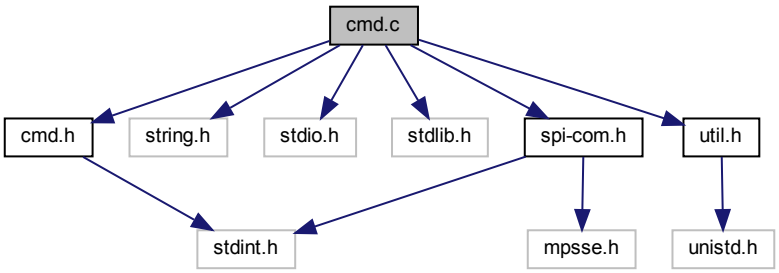
Flashes elf files to programmer MCU or cart CIC.

8.3 cmd.c File Reference

Allows sending commands to the programmer, and receiving results.

```
#include "cmd.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "spi-com.h"
#include "util.h"
```

Include dependency graph for cmd.c:



Functions

- int CmdInit (unsigned int channel)
- int CmdSend (const Cmd *cmd, uint8_t cmdLen, CmdRep **rep)
- int CmdSendLongCmd (const Cmd *cmd, uint8_t cmdLen, const uint8_t *data, int dataLen, CmdRep **rep)
- int CmdSendLongRep (const Cmd *cmd, uint8_t cmdLen, CmdRep **rep, uint8_t *data, int recvLen)

8.3.1 Detailed Description

Allows sending commands to the programmer, and receiving results.

Author

Jesus Alonso (doragasu)

Date

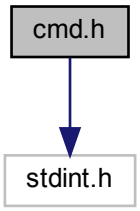
2016

8.4 cmd.h File Reference

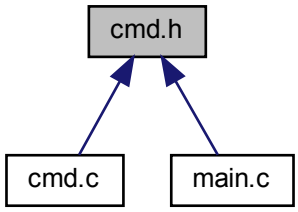
Allows sending commands to the programmer, and receiving results.

```
#include <stdint.h>
```

Include dependency graph for cmd.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct CmdRdWrHdr
  Command header for memory read and write commands.
- struct CmdErase
  Command header for erase commands.
- union Cmd
  Generic command request.
- struct CmdFlashId
  Flash chip identification information.
- struct CmdRepEmpty
  Empty command response.
- struct CmdRepFwVer
  Firmware version command response.
- struct CmdRepFlashId
  Flash ID command response.
- union CmdRep
  Generic reply to a command request.

Macros

- #define CMD_OK 0

  Function completed successfully.
- #define CMD_ERROR -1

  Function completed with error.
- #define CMD_MAXLEN 32

  Maximum length of a command.
- #define CMD_SRAM_MAXLEN 8∗1024

  Maximum SRAM length is 8 KiB.
- #define CMD_REP_OK 0

  OK reply code.
- #define CMD_FW_VER 1

  Get programmer firmware version.
- #define CMD_CHR_WRITE 2

  Write to CHR flash.
- #define CMD_PRG_WRITE 3

  Write to PRG flash.
- #define CMD_CHR_READ 4

  Read from CHR flash.
- #define CMD_PRG_READ 5

  Read from PRG flash.
- #define CMD_CHR_ERASE 6

  Erase CHR flash (entire or sectors)
- #define CMD_PRG_ERASE 7

  Erase PRG flash (entire or sectors)
- #define CMD_FLASH_ID 8

  Get flash chips identifiers.
- #define CMD_RAM_WRITE 9

  Write data to cartridge SRAM.
- #define CMD_RAM_READ 10

  Read data from cartridge SRAM.
- #define CMD_MAPPER_SET 11

  Configure cartridge mapper.
- #define CMD_REP_ERROR 255

  Error reply code.
- #define CmdRepFree(pRep) free(pRep)

Enumerations

- enum CmdMapper { CMD_MAPPER_MMC3X = 0, CMD_MAPPER_TKROM }

  Supported mappers.

Functions

- int CmdInit (unsigned int channel)
- int CmdSend (const Cmd ∗cmd, uint8_t cmdLen, CmdRep ∗∗rep)
- int CmdSendLongCmd (const Cmd ∗cmd, uint8_t cmdLen, const uint8_t ∗data, int dataLen, CmdRep ∗∗rep)
- int CmdSendLongRep (const Cmd ∗cmd, uint8_t cmdLen, CmdRep ∗∗rep, uint8_t ∗data, int recvLen)

8.4.1 Detailed Description

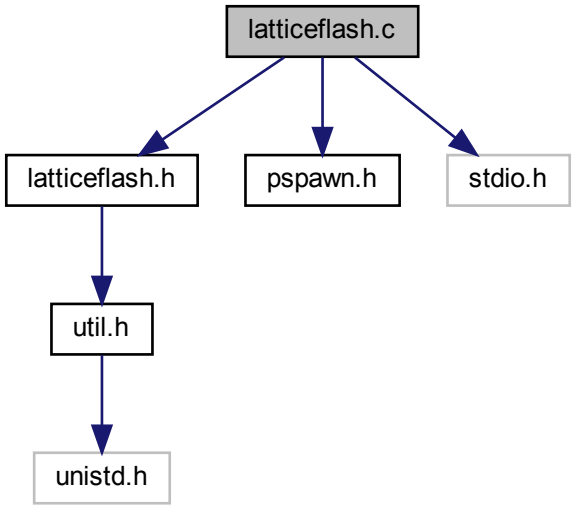
Allows sending commands to the programmer, and receiving results.

8.5 latticeflash.c File Reference

Flash specified xcf file to FPGA.

```
#include "latticeflash.h"
#include "pspawn.h"
#include <stdio.h>
```

Include dependency graph for latticeflash.c:



Macros

- #define LATTICE_FLASH_BIN_POS 2

  Position in latFlash argument array of the flasher binary/script.
- #define LATTICE_FLASH_XCF_POS 4

  Position in latFlash argument array of the input XCF file.

Functions

- int LatticeFlash (const char prgcmd[ ], const char xcf[ ])

8.5.1 Detailed Description

Flash specified xcf file to FPGA.

Author

Jesus Alonso (doragasu)

Date

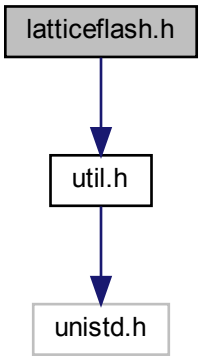
2016

8.6 latticeflash.h File Reference

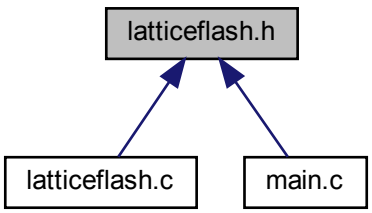
Flash specified xcf file to FPGA.

```
#include "util.h"
```

Include dependency graph for latticeflash.h:



This graph shows which files directly or indirectly include this file:



Functions

- int LatticeFlash (const char prgcmd[ ], const char xcf[ ])

8.6.1 Detailed Description

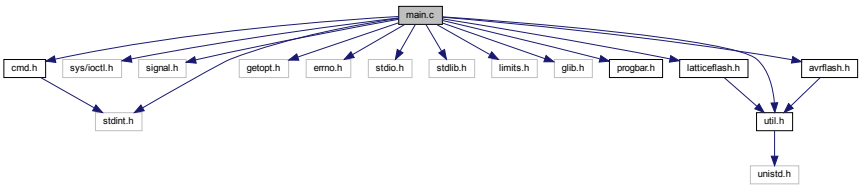
Flash specified xcf file to FPGA.

8.7 main.c File Reference

Command Line Interface for the mojo-nes-mk3 programmer.

```
#include "util.h"
#include <sys/ioctl.h>
#include <signal.h>
#include <stdint.h>
#include <getopt.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <glib.h>
#include "progbar.h"
#include "cmd.h"
#include "avrflash.h"
#include "latticeflash.h"
```

Include dependency graph for main.c:



Data Structures

- struct MemImage

  Definition of a file representing a memory image.

- union Flags

  Supported option flags.

Macros

- #define VERSION_MAJOR 0x00

  Major version of the program.
- #define VERSION_MINOR 0x04

  Minor version of the program.
- #define MAX_FILELEN 255

  Maximum file length.
- #define PROG_CHIP_CHR 0

  Character ROM chip (CHR ROM)
- #define PROG_CHIP_PRG 1

  Program ROM chip (PRG ROM)
- #define PROG_CHIP_MAX PROG_CHIP_PRG

  Last value for programmable chips.
- #define PROG_ERASE_FULL 0xFFFFFF

  Return value for Erase operation error.
- #define PROG_SRAM_BASE 0x6000

  SRAM base address.
- #define PROG_SRAM_LEN (8 ∗ 1024)

  SRAM length.
- #define AVR_CHIP_MCU "m8515"

  Definition of the chip of the programmer (ATMEGA8515)
- #define AVR_CHIP_CIC "t13"

  Definition of the CIC chip (ATTINY13)
- #define AVR_PATH "/usr/bin/avrdude"

  avrdude binary path
- #define AVR_PROG_CFG "/usr/share/mk3-prog/mk3prog.conf"

  Configuration file of the programmer to use.
- #define AVR_PROG_MCU "mk3prog-mcu"

  MCU programmer defined in mk3prog.conf.
- #define AVR_PROG_CIC "mk3prog-cic"

  CIC programmer defined in mk3prog.conf.
- #define LATT_PROG_PATH "/usr/local/diamond/3.7_x64/bin/lin64/pgrcmd"

  Path to the FPGA programmer program/script.
- #define CMD_SET_ADDR(field, addr)

  Copies the specified address to a byte array field.
- #define CMD_SET_LEN(field, len)

  Copies the command length to the specified byte array field.
- #define CondPrintf(cond, ...) do{if(cond) printf(__VA_ARGS__);}while(0)

  Printf-like macro that prints only if condition is TRUE.
- #define PrintVerb(...) do{if(f.verbose) printf(__VA_ARGS__);fflush(stdout);}while(0)

  Printf-like macro that prints only if f.verbose is TRUE.
- #define try(code, error)

Functions

- uint8_t ∗ AllocAndRead (uint8_t chip, MemImage ∗f, unsigned int cols)
- uint8_t ∗ AllocAndRamWrite (MemImage ∗f)
- uint8_t ∗ AllocAndRamRead (MemImage ∗f)
- int CmdMapperCfg (CmdMapper mapper)
- int main (int argc, char ∗∗argv)

8.7.1 Detailed Description

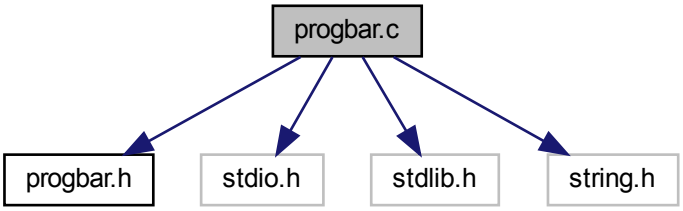
Command Line Interface for the mojo-nes-mk3 programmer.

8.8 progbar.c File Reference

Draw progress bars for command line applications.

```
#include "progbar.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for progbar.c:



Functions

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

8.8.1 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)

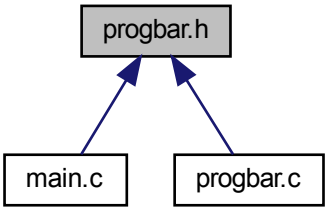
Date

2015

8.9 progbar.h File Reference

Draw progress bars for command line applications.

This graph shows which files directly or indirectly include this file:



Functions

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

8.9.1 Detailed Description

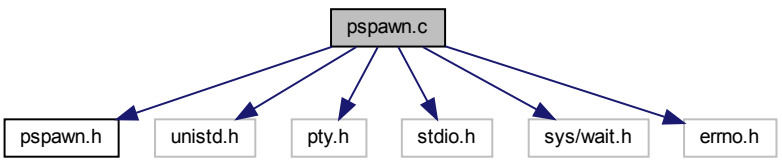
Draw progress bars for command line applications.

8.10 pspawn.c File Reference

Spawns a child process using a pseudo terminal.

```
#include "pspawn.h"
#include <unistd.h>
#include <pty.h>
#include <stdio.h>
#include <sys/wait.h>
#include <errno.h>
```

Include dependency graph for pspawn.c:



Macros

- #define PSPAWN_LINE_BUF_LEN 256

  Lenght of the line buffer.

Functions

- int pspawn (const char *file, char *const arg[ ])

8.10.1 Detailed Description

Spawns a child process using a pseudo terminal.

Spawns a child process using a pseudo-terminal to avoid input/output buffering. The standard output of the child process is redirected to the standard output of the parent process.

Author

Jesus Alonso (doragasu)

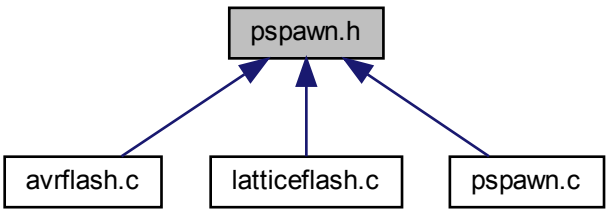
Date

2016

8.11 pspawn.h File Reference

Spawns a child process using a pseudo terminal.

This graph shows which files directly or indirectly include this file:



Functions

- int pspawn (const char *file, char *const arg[ ])

8.11.1 Detailed Description

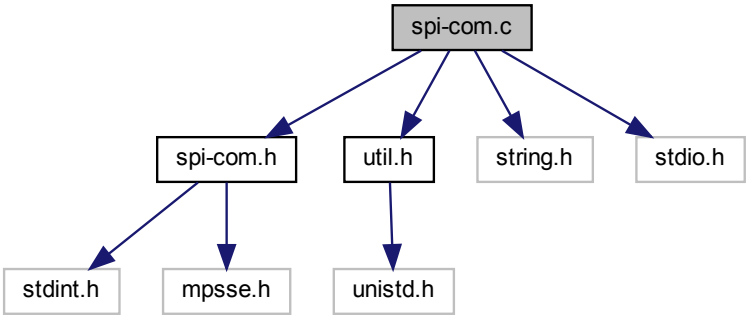
Spawns a child process using a pseudo terminal.

8.12 spi-com.c File Reference

Handles SPI and framing for communications.

```
#include "spi-com.h"
#include "util.h"
#include <string.h>
#include <stdio.h>
```

Include dependency graph for spi-com.c:



Macros

- #define SCStopErr() do{Stop(mpsse);return SC_ERROR;}while(0)

  Signals frame stop and returns with SC_ERROR.
- #define SCStopNull() do{Stop(mpsse);return NULL;}while(0)

  Signals frame stop and returns NULL.

Functions

- struct mpsse_context * SCInit (unsigned int channel)
- int SCFrameSend (struct mpsse_context *mpsse, char *data, uint16_t datalen)
- char * SCFrameRecv (struct mpsse_context *mpsse, uint8_t *maxlen)

8.12.1 Detailed Description

Handles SPI and framing for communications.

Frames are delimited by the SOF/EOF characters. Following SOF, payload length in sent using 1 byte. Then data follows, and finally EOF character ends transmission.

Author

Jesus Alonso (doragasu)

Date

2016

Note

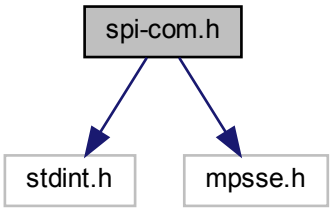
Uses open source mpsse library to interface FT2232 in MSPSSE mode.

8.13 spi-com.h File Reference

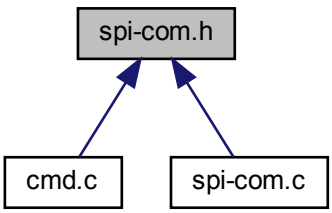
Handles SPI and framing for communications.

```
#include <stdint.h>
#include <mpsse.h>
```

Include dependency graph for spi-com.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define SC_SPI_MODE SPI0

  SPI mode for using with MPSSE library.
- #define SC_SPI_CLK 100000
- #define SC_SOF 0x7E

  Start of frame marker.
- #define SC_EOF 0x7D

  En of frame marker.
- #define SC_VID 0x0403

  USB Vendor ID of the programmer board.
- #define SC_PID 0x6010

  USB Device ID of the programmer board.
- #define SC_IFACE IFACE_B

  FT2232 interface used to communicate with the microcontroller.
- #define SC_MAX_DATALEN 32

  Maximum data payload is 32 bytes long.
- #define SC_OK MPSSE_OK

  OK status (0)
- #define SC_ERROR MPSSE_FAIL

  Error status (-1)

Functions

- struct mpsse_context ∗ SCInit (unsigned int channel)
- int SCFrameSend (struct mpsse_context ∗mpsse, char ∗data, uint16_t datalen)
- char ∗ SCFrameRecv (struct mpsse_context ∗mpsse, uint8_t ∗maxlen)

8.13.1 Detailed Description

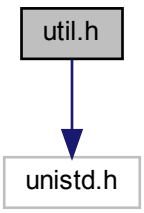
Handles SPI and framing for communications.

8.14 util.h File Reference

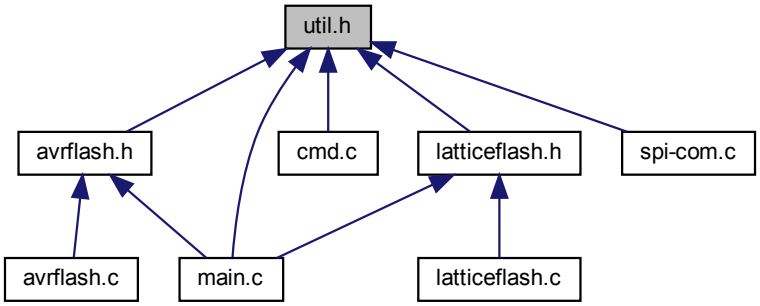
General purpose utiities.

```
#include <unistd.h>
```

Include dependency graph for util.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define TRUE 1

  TRUE value definition for logic comparisons.
- #define FALSE 0

  FALSE value definition for logic comparisons.
- #define MAX(a, b) ((a)>(b)?(a):(b))

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

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

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

  Delay the specified amount of milliseconds.

8.14.1 Detailed Description

General purpose utiities.

Index