Technická dokumentace

Extreme Gcode Analyzer

Úvod

Extreme Gcode Analyzer je program běžící na přípravku MicroZed APO schopný otevřít a analyzovat gcode soubory pro 3D tiskárny. Kromě obecných informací o modelu dokáže na LCD displeji zobrazovat jednotlivé vrstvy. Zdrojový kód je dostupný v git repozitáři na github.com/humpljir/mzapo.

Architektura aplikace

Blokové schéma jednotlivých modulů a naznačení komunikace mezi nimi.

**MZAPO\_PARLCD**

**MZAPO\_PLUGS**

**HWIO**

**GCODE**

**LCD**

**GUI**

**MAIN**

Funkce

LCD

bool lcd\_init(void);

bool lcd\_destroy(void);

void lcd\_test(uint16\_t color);

void lcd\_paint(uint16\_t color);

void lcd\_write\_pixel(disp\_pos\_t pixel, uint16\_t color);

void lcd\_paint\_buffer(uint16\_t color);

void lcd\_draw\_line(\*\*\*);

void lcd\_print\_char(\*\*\*);

void lcd\_print\_string(\*\*\*);

unsigned char lcd\_get\_char\_width(\*\*\*);

int lcd\_get\_string\_width(\*\*\*);

void lcd\_print\_frame\_buffer(void);

bool lcd\_print\_from\_file(char \*filename);

Přiřazení funkcí podle jejich účelu ke konkrétním modulům programu.

LCD

void gcode\_init\_cmd(command \*cmd);

bool gcode\_parse\_line (command \*cmd, char \*line, int size);

void gcode\_process\_cmd(command \*cmd);

void gcode\_print\_cmd(command \*cmd);

void gcode\_move(command \*cmd);

void gcode\_print\_stats(void);

bool gcode\_init\_file(char \*filename);

void gcode\_close\_file(void);

void gcode\_set\_defualt\_vals(void);

bool gcode\_set\_layer(layer\_t \*layer);

bool gcode\_set\_layer\_by\_num(int layer\_num);

layer\_t \*gcode\_get\_layer(int layer\_num);

void gcode\_free\_layer(layer\_t \*layer);

void gcode\_print\_layer(layer\_t \*layer);

bool gcode\_get\_cmd(command \*cmd, bool move\_only);

model\_t \*gcode\_get\_model(void);

file\_t \*gcode\_get\_file\_info(void);

GUI

void gui\_win\_right(void);

void gui\_win\_left(void);

void gui\_win\_manual(void);

void gui\_win\_manual\_exit(void);

void gui\_layer\_up(void);

void gui\_layer\_down(void);

void gui\_layer\_up\_4x(void);

void gui\_layer\_down\_4x(void);

void gui\_layer\_up\_16x(void);

void gui\_layer\_down\_16x(void);

void gui\_apply\_state(void);

void gui\_r\_click(void);

void gui\_r\_incr(void);

void gui\_r\_decr(void);

void gui\_g\_click(void);

void gui\_g\_incr(void);

void gui\_g\_decr(void);

void gui\_b\_click(void);

void gui\_b\_incr(void);

void gui\_b\_decr(void);

void gui\_set\_quit(void);

void print\_win1(void);

void print\_win2(void);

void print\_win3(void);

void print\_win\_man(void);

void gui\_refresh\_ledstrip(void);

bool gui\_start(char \*filename);

bool gui\_init\_file(char \*filename);

bool gui\_destroy(void);

void gui\_print\_layer(void);

disp\_pos\_t gui\_map\_extruder\_to\_disp(pos\_t pos);

bool gui\_quit(void);

HWIO

bool hw\_init(void);

void hw\_check(void);

bool hw\_r\_released(void);

bool hw\_g\_released(void);

bool hw\_b\_released(void);

bool hw\_r\_incremented(void);

bool hw\_g\_incremented(void);

bool hw\_b\_incremented(void);

bool hw\_r\_decremented(void);

bool hw\_g\_decremented(void);

bool hw\_b\_decremented(void);

void hw\_test\_out(void);

void hw\_write\_ledstrip(uint32\_t bits);

void hw\_write\_led1(uint8\_t red, uint8\_t green, uint8\_t blue);

void hw\_write\_led2(uint8\_t red, uint8\_t green, uint8\_t blue);

MAIN

int main(int argc, char \*\*argv);0