HAL1284

HAL 1284

Created by

Anders Faber Mygind

With Special thanks to:

dan14

Scott Lawrence - BleuLlama

maniacbug

... for the software

and

Avamander

Mark Stanley

Alexander Brevig

... for libraries

Full source code can be found on:

https://github.com/PlainOldAnders/HAL1284

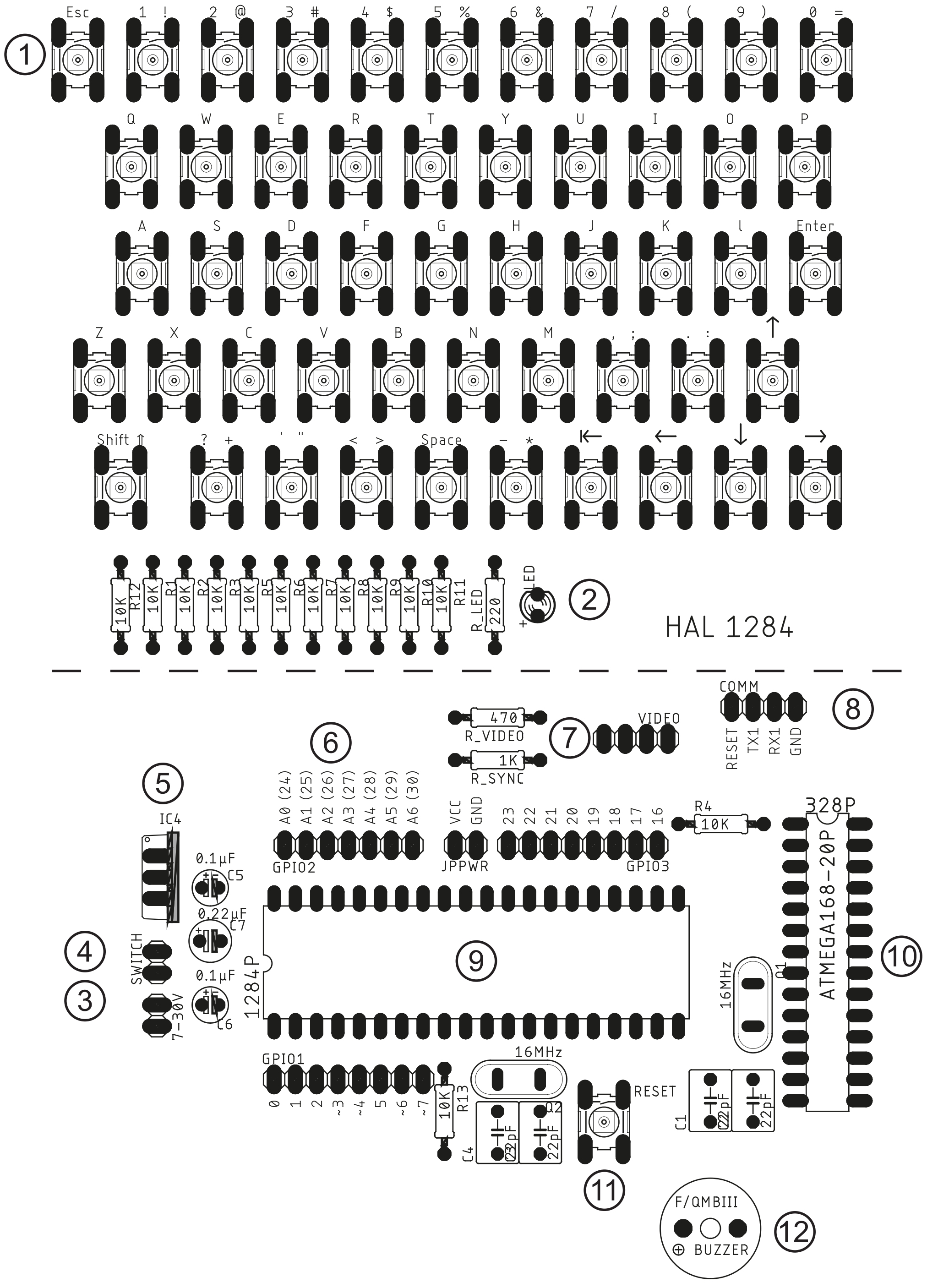
Created Winter ‘20

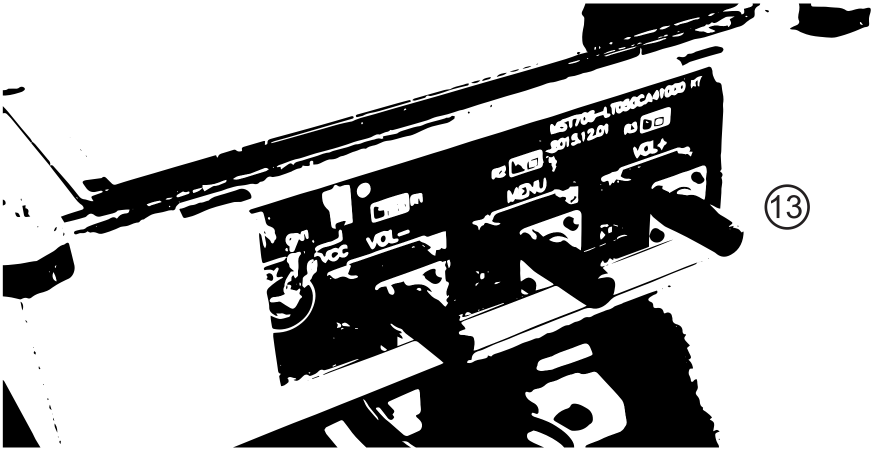
Index

Diagram

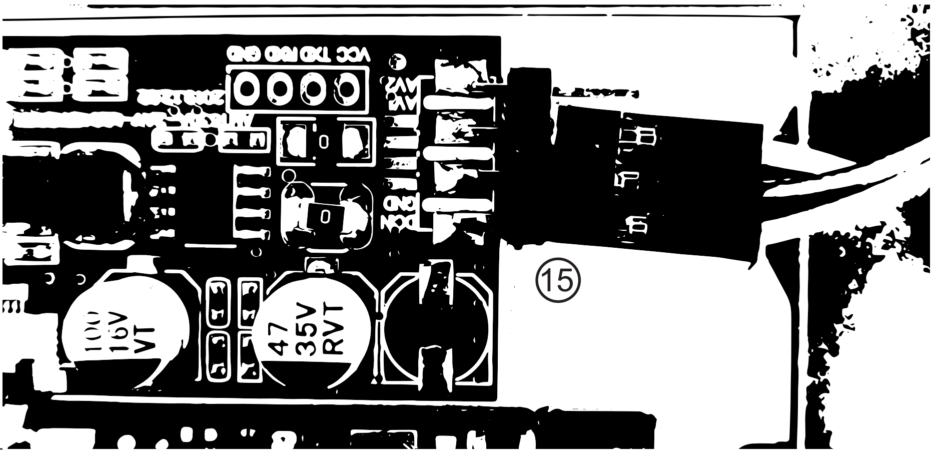
1 - Keybaord
2 - Power LED
3 - Power Input. Between 7 and 30 Volts DC. Power ~3.5W
4 - On/Off Switch. Shorting is On
5 - Voltage Regulator. Gets stupidly hot
6 - I/O Pins. Also has 5V and GND.
7 - Video Pins. From left: Power, Ground, Signal, Unused.
8 - Serial COM pins. From left: Unused, TX, RX, Ground.
9 - Main CPU. ATmega1284P
10 - Keyboard controller. ATmega328P
11 - Reset Button
12 - Onboard Buzzer. Connected to GPIO 15
13 - Control buttons for screen-options (Brightness, ratio, etc.)
14 - On/Off Switch. (On is up)
15 - Back of screen pins. From top:
Unused, Video Signal, Ground, Power

Top diagram is front view of board and bottom Diagram is back view of board.

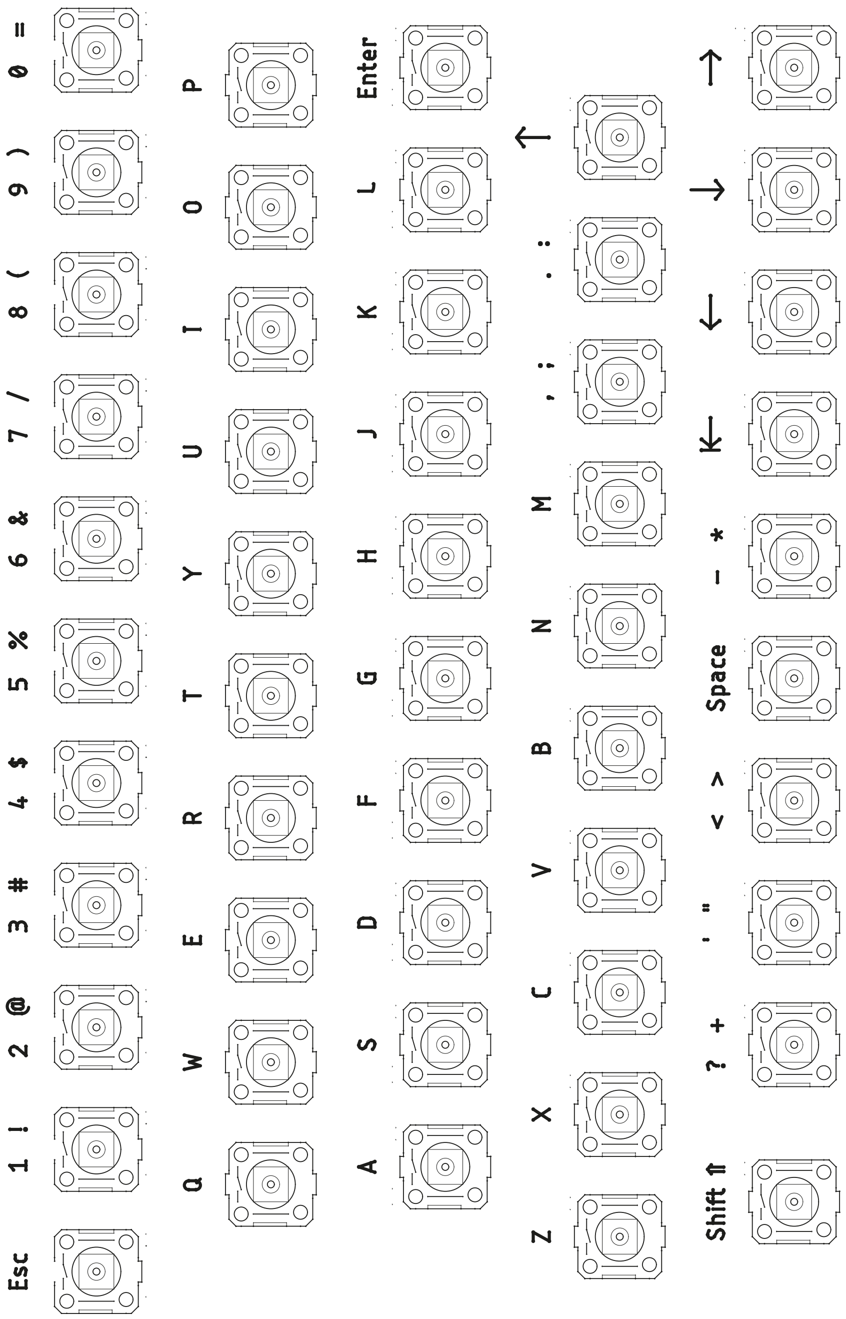

Esc
1 !
2 @
3 #
4 $
5 %
6 &
7 /
8 (
9)
0 =
Q W E R T Y U I O P
A S D F G H J K l Enter
Z X C V B N M , ; . : ↑
Shift ⇑
? +
' "
< >
Space
– *
⇤ ← ↓ →
R12 R1 R2 R3 R5 R6 R7 R8 R9 R10 R11 R_LED
10K
220
LED
HAL 1284
COMM
RESET TX1 RX1 GND
470
R_VIDEO
1K
R_SYNC
VIDEO
A0 (24) A1 (25) A2 (26) A3 (27) A4 (28) A5 (29) A6 (30)
VCC GND
23 22 21 20 19 18 17 16
R4
10K
328P
GPIO2
JPPWR
GPIO3
IC4
0.1µF
C5
0.22µF
C7
0.1µF
C6
SWITCH
7–30V
1284P
ATMEGA168-20P
16MHz
Q1
GPIO1
0 1 2 ~3 ~4 5 ~6 ~7
R13
10K
16MHz
Q2
C4
22pF
RESET
C1
22pF
F/QMBIII
⊕ BUZZER
1 2 3 4 5 6 7 8 9 10 11 12


MST703-LT056CA41000 K7
2015.12.01
R3
VOL+
R2
MENU
VCC
VOL-
⑬

⑭


GND RXD TXD VCC
5V
3V
GND
DCIN
100 16V VT
47 35V RVT
⑮

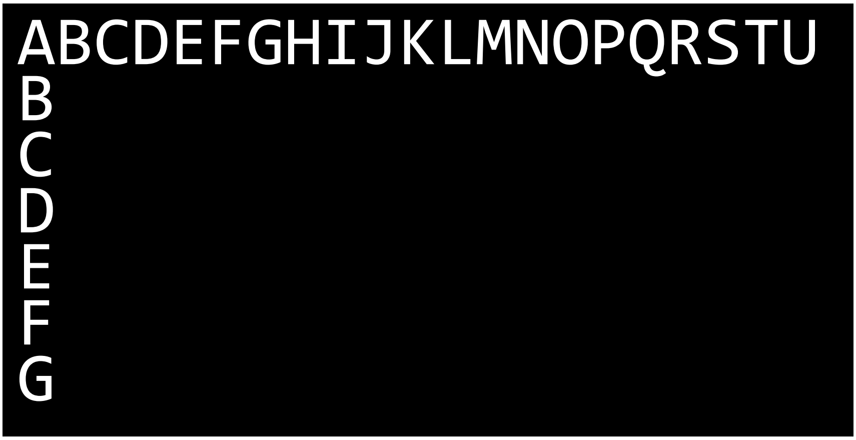
Keyboard Close Up



Screen Data

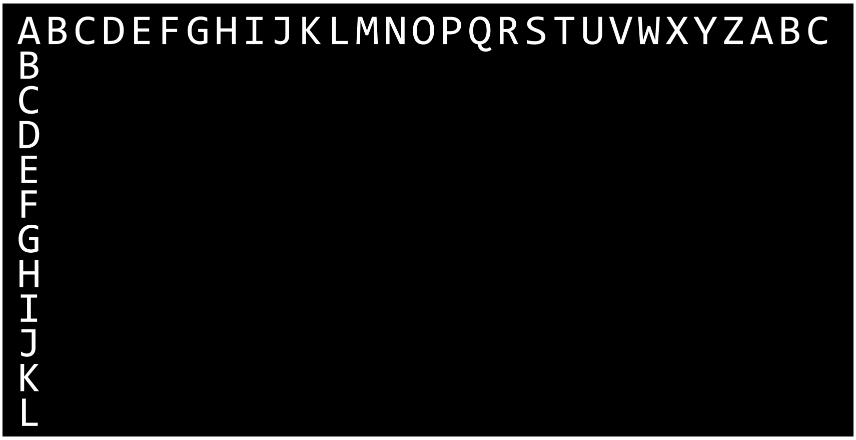
- Video signal ⑦ is regular composite video and can be connect-ed to any monitor with signal and Ground.

Screen Mode 1 (Default)



- Screen has a width of 128px and height of 56px.
- One character has width of 6px and height of 8px.
- There are 7 rows of text.
- There are 21 coloumns for each character.
- When Drawing, there are a 3-4 color options:
    - 0 = Black
    - 1 = White
    - 2 = Invert Color
    - 3 = (Only used for fill color) No fill

Screen Mode 2



- Screen has a width of 176px and height of 96px.
- One character has width of 6px and height of 8px.
- There are 12 rows of text.
- There are 29 coloumns for each character.
- When Drawing, there are a 3-4 color options:

  0 = Black

  1 = White

  2 = Invert Color

  3 = (Only used for fill color) No fill

Commands

LIST – Lists current program

BYE - Exits Basic, soft reboot on Arduino

END - Stops execution from the program, also “STOP”

MEM - Displays memory usage statistics

NEW - Clears the current program

RUN - Executes the current program

EFORMAT - Clears the EEProm memory (Takes a couple of seconds)

ELOAD - Load the program in from EEProm

ESAVE - Save the current program to the EEProm

ELIST - Print out the contents of EEProm

ECHAIN - Load the program from EEProm and run it

INPUT *variable* - Let the user input an expression equal to *variable*

PEEK* (*address*) - Get a value in memory

POKE* *address* - Set a value in memory

PRINT/? *expression* - Print out the *expression*. Put *expression* in quotes

REM *expression* - Remark/Comment. Put *expression* in quotes

A=V, LET A=V - Assign value *V* to a variable *A*

+, -, *, / - Math operations

<,<=,=,<>,!=,>=,> - Math comparisons

ABS(*expression*) - Returns the absolute value of the *expression*

RSEED(*v*) - Sets the random seed to *v*

RND(*m*) - Returns a random number from 0 to *m*

IF *expression statement* - Perform *statement* if *expression* is true

FOR *variable* = *start* **TO** *end* - Start for block

FOR *variable* = *start* **TO** *end* **STEP** *value* - Start for block with step

NEXT - End of for block

GOTO *linenumber* - Continue execution at this *line number*

GOSUB *linenumber* - Call a subroutine at this *line number*

RETURN - Return from a subroutine

DELAY *timems* - Wait *timems* (in milliseconds)

DWRITE *pin*, *value* - Set *pin* with a *value* (HIGH,HI,LOW,LO)

AWRITE* *pin*, *value* – Set PWM *pin* with analog *value* (0-255)

DREAD(*pin*) - Get the value of the *pin*

AREAD*(*analogPin*) - Get the value of the *analog pin*

TONE *freq*, *timems* - Play *freq* for *timems* (in milliseconds)

TONEW *freq*, *timems* - Same as TONE, but also waits for it to finish

NOTONE - Stop playback of all playing tones

SERCOM [A] - Toggle listen for Serial Communication. Defaults to false (not listening)

CLEAR [A] - Clears the TV Screen

NLIST *linenumber* [A] - Same as LIST, but prints only given *line number*

DRAWPIX* *x, y, c* [A] - Draw pixel at position *x, y* with color *c*

DRAWLINE* *x0, y0, x1, y1, c* [A] - Draw line from *x0, y0* to *x1, y1* with color *c*

DRAWROW* *r, x0, x1, c* [A] - Fills *r* row from *x0* to *x1* with color *c*

DRAWCOL* *co, y0, y1, c* [A] - Fills *co* column from *y0* to *y1* with color *c*

DRAWRECT* *x, y, w, h, c, fillC* [A] - Draw top left corner of rectangle at *x, y* with height *h* and width *w* with border color *c* and fill color *fillC*

DRAWCIRC* *x, y, r, c, fillC* [A] - Draw circle with center at *x,y* with radius *r* and border color *c* with fill color *fillC*

DRAWCHAR* *x, y, ch* [A] - Draw character *ch* at position *x, y*

GETPIX *x, y* [A] - Get pixel color at position *x,y*

TVTOG [A] - Toggles between two the two TV modes

SETTINGS *att, val* [A] - Set settings on startup. First is *att* (attribute, which setting) and second is the value (0 or 1). This requires restart.

0 = Serial Communication (0 is recommended here)

1 = Toggle between TV modes

2 = Show SplashScreen on startup

3 = Print message on startup

Esc Key - Esc-key used for break (CTRL + C)

Explanation of Commands

PEEK and **POKE** is untested, since I’m not sure what the addresses are pointed at.

AWRITE is using PWM (Pulse Width Modulation) pins are generating a pseudo analog value. Pins marked with ‘~’ on the board can generate analog values.

AREAD can only be used with analog pins marked with A# on the board. These pins can also be used for digital logic, but the number will be different.

Commands marked with [A], are written by me. I’m not sure about their functionality.

All **DRAW**-commands use a color. This color can be 0 (Black), 1 (White) or 2 (Invert).

Some **DRAW**-commands use a Fill Color. This can also be 0, 1 and 2, but a value of 3, means no fill.

Serial Communication

Download
The software can be downloaded here: *https://github.com/PlainOldAnders/HAL1284/tree/main/HAL1284Com*

What?
Code can be uploaded to the HAL1284 using a USB to Serial chip. This happens through the board's UART1 port connected to the pins on ⑧. The Reset-pin is unused, and the TX1 (Transfer Pin), is technically not necessary.

Software
There is a program called HAL1284Com, that can be used for this communication. There is both a CLI and a GUI. They have the same capabilities: Write Directly to the board, Upload a file to the board (any file is accepted, and compilation happens on the board) and "Basicify" a file.

"Basicify"
Since line numbers in Basic is annoying, the Basicify-function takes a file with no linenumbers and adds them to the start of each line. Line 1 becomes 10, 2 becomes 20 and so on. Existing line numbers are not accounted for. This creates a .basic file in the next to the uploaded file.

SERCOM
SERCOM is a toggle command that enables/disables serical communication with the UART1 port. Having this enabled or having wires connected to the pins, can cause static to write empty characters.

Settings.xml
This is a file found in the root of the HAL1284Com software. It

holds the auto/manual port selection. If you experience errors, try setting port selection to manual.
It also holds write speed. This is a delay between each written line.

Supported Boards
There are many different boards that support USB to Serial commands: “Empty” Arduino Unos, PL2303HX, CP2102, CH340G esc. There should be one included in the package.

Known Bugs/Future Work

Arrow Keys
The Arrow-keys were put accidently and don’t serve much function. This makes navigation difficult

File I/O
There exists File I/O capabilities but these are not implemented. To combat this, the HAL1284Com software is able to upload files to the board. It is also possible to upload to the EEPROM.

PEEK/POKE
PEEK and POKE are untested, and I’m not sure what the addresses are.

Break
Break (Esc) has sometimes been messing with the entire code, when in infinite loops.

Pin 7
The PWM pin 7 can be difficult when trying to DWRITE High or AWRITE 255. It works fine with AWRITE 254.

Idiot
It would be ideal to also be able to upload via UART1 but upload is only possible via UART0, so upload will have to be done by extracting the main IC.

The Screen and Voltage Regulator ⑤ gets a bit too hot when running for a long time.

Examples

This is a small collection of examples to run on the HAL1284. These (and more) can be found on: *https://github.com/PlainOldAnders/HAL1284/tree/main/examples.*

Blink

```
5 REM “Hook up LED on pin 23”
10 FOR A=0 TO 10
20 DWRITE 23, HIGH
30 DELAY 250
40 DWRITE 23, LOW
50 DELAY 250
60 NEXT A
```

Fade

```
5 REM “Hook up LED on pin 3”
10 FOR A=0 TO 10
20 FOR B=0 TO 255
30 AWRITE 3, B
40 DELAY 10
50 NEXT B
60 FOR B=255 TO 0 STEP -1
70 AWRITE 3, B
80 DELAY 10
90 NEXT B
100 NEXT A
```

Tones

```
10 FOR T = 500 TO 1500 STEP 100
20 TONEW T, 200
30 NEXT T
```

Knob

```
5 REM “Potentiometeter on A2 and ground”
6 REM “Led pin 3 and ground.”
7 REM “If knob is at 0, it stops”
10 A = AREAD( 2 )
20 PRINT A
30 B = A / 4
40 AWRITE 3, B
50 IF A = 0 GOTO 100
60 GOTO 10
100 PRINT “Done.”
```

Random Squares

```
5 CLEAR
10 FOR I = 0 TO 5
20 X = RND(70)
30 Y = RND(40)
40 W = RND(25)
50 H = RND(16)
60 DRAWRECT X, Y, W, H, 1, 3
70 DELAY 200
80 NEXT I
```

User Input

```
5 REM “Enter number like 33 or var like ‘b’.”
10 A=0
15 B=999
20 PRINT “A is “, A
30 PRINT “Enter a new value “;
40 INPUT A
50 PRINT “A is now “, A
```