Airbus A3xx FCU Custom LCD

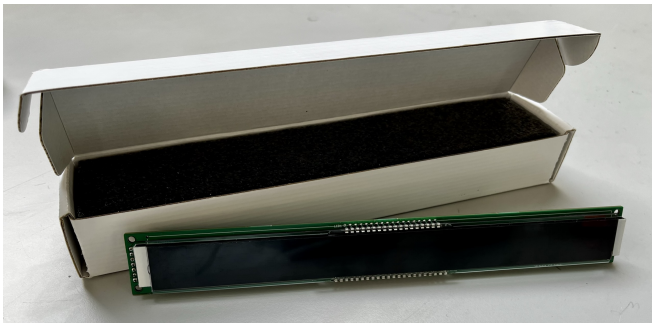
Kav Simulations A3xx FCU LCD

Thanks for your purchase of our bespoke Airbus A3xx FCU LCD. This document outlines the details of interacting with the component using our library.

The library consists of a '`.cpp`' and a '`.h`' file which gives you access to functions to display the required information on the display. This document is designed to talk you through how to use and call the provided functions.

If you wish to use the device 'plug-and-play' with MobiFlight, then this document may go into more detail than you need. But, if you want to understand a bit more about how the display works, then it's probably worth a read!

Ok, let's get started!



Installation

1. Copy the provided .cpp and .h files into your Arduino project folder.

2. Our library depends on the 'HT1621.h' library to communicate with the HT1621 LCD driver. You need to include the 'HT1621.h' and 'HT1621.cpp' files in your project as well. The files can be found at https://macduino.blogspot.com/2015/02/HT1621.html

Usage

Initialisation

First, you need to connect the device as described in the Interface Configuration section below, and then initialise the LCD object and attach it to the correct pins. **If you're not using MobiFlight, you need to remove the '`#include "commandmessenger.h"`' line from the libraries if it is there. It has been removed from the demo code already.**

```
#include "KAV_A3XX_FCU_LCD.h"

KAV_A3XX_FCU_LCD lcd(CS_PIN, CLK_PIN, DATA_PIN);

void setup() {
  lcd.attach(CS_PIN, CLK_PIN, DATA_PIN);
}
```

Replace 'CS_PIN', 'CLK_PIN', and 'DATA_PIN' with the appropriate pin numbers for your setup.

We first create the FCU object with the required pins. Then, we call the attach function which sets up the device using the `HT1621` library with the required settings, then puts the device into it's start state.

Display Functions

These functions are used to display various values on the LCD.

```
void setSpeedMode(uint16_t value);
void setMachMode(uint16_t value);
void showHeadingValue(uint16_t value);
void showAltitudeValue(uint32_t value);
void showVerticalValue(int16_t value);
void showFPAValue(int8_t value);
```

Display Functions Example:

```
lcd.setSpeedMode(250);
lcd.setMachMode(80);
lcd.showHeadingValue(180);
lcd.showAltitudeValue(35000);
lcd.showVerticalValue(-800);
lcd.showFPAValue(-30);
```

Control Functions

These functions are used to control the appearance of the LCD.

```
void setSpeedDashes(int8_t state);
void setHeadingDashes(int8_t state);
void setAltitudeDashes(int8_t state);
void setVrtSpdDashes(int8_t state);
void setSpeedDot(int8_t state);
void setHeadingDot(int8_t state);
void setAltitudeDot(int8_t state);
void toggleTrkHdgMode(int8_t state);
```

Control Functions Example:

```
lcd.setSpeedDashes(1);
lcd.setHeadingDashes(0);
lcd.setAltitudeDashes(1);
lcd.setVrtSpdDashes(1);
lcd.setSpeedDot(1);
lcd.setHeadingDot(0);
lcd.setAltitudeDot(1);
lcd.toggleTrkHdgMode(1);
```

Command Handling

The library also includes a function to handle MobiFlight commands. You can pass a string containing a command and it will be executed.

```
void handleMobiFlightCmd(char *cmd);
```

Example:

```
char command[] = "setSpd=250";
lcd.handleMobiFlightCmd(command);
```

Complete Example

This is just an example showing how functions are called and used. When you first plug the device in from new, it might take a few moments for the display to initialise.

```
#include "KAV_A3XX_FCU_LCD.h"

KAV_A3XX_FCU_LCD fcu(CS_PIN, CLK_PIN, DATA_PIN);

void setup() {
  fcu.attach(CS_PIN, CLK_PIN, DATA_PIN)
}

void loop() {
  fcu.setStartLabels();

  // Demo State 1
  fcu.toggleTrkHdgMode(0);
  fcu.setSpeedMode(0);
  fcu.setSpeedDashes(1);
  fcu.setSpeedDot(1);
  fcu.setHeadingDashes(1);
  fcu.setHeadingDot(1);
  fcu.showAltitudeValue(5000);
  fcu.setVrtSpdDashes(1);
  delay(4000);

  // Demo State 2
  fcu.setSpeedDashes(0);
  fcu.setSpeedDot(0);
  fcu.setHeadingDashes(0);
  fcu.setHeadingDot(0);
  fcu.setVrtSpdDashes(0);
  fcu.setMachMode(80);
  fcu.showHeadingValue(245);
  fcu.showAltitudeValue(12000);
  fcu.showVerticalValue(-1200);;
  delay(3000);
  fcu.showVerticalValue(800);
  delay(3000);

  // Demo State 3
  fcu.setSpeedMode(250);
  fcu.toggleTrkHdgMode(1);
  fcu.showHeadingValue(210);
  fcu.showAltitudeValue(16000);
  fcu.showFPAValue(-30);
  delay(4000);

  // Clear
  fcu.clearLCD();
}
```

API Reference

'void attach(byte CS, byte CLK, byte DATA)'

Attaches the LCD object to the specified pins and initialises the display. Parameters:

- 'CS': Chip Select pin
- 'CLK': Clock pin
- 'DATA': Data pin

'void detach()'

Detaches the LCD object and disables the display.

'void refreshLCD(uint8_t address)'

Refreshes the LCD display at the specified address. Parameters:

- 'address': The address to refresh on the display.

'void clearLCD()'

Clears the entire LCD display.

'void setStartLabels()'

Sets the default labels for the LCD display.

'void setHeadingMode()'

Sets the LCD display to Heading Mode.

'void setTrackMode()'

Sets the LCD display to Track Mode.

'void displayDigit(uint8_t address, uint8_t digit)'

Displays a digit at the specified address on the LCD. Parameters:

- 'address': The address to display the digit on the LCD.
- 'digit': The digit to display (0-9, 10 = dash, 11 = blank, 12 = small 0 for V/S).

The addresses are predefined at the top of the file as 'SPD_HUN', 'SPD_TEN', 'HDG_HUN', etc for easier use.

Easy API using MobiFlight Functions

The file was written to be able to work alongside 'MobiFlight', so you can also use this as a way of controlling the LCD. There is the option to utilise the '`handleMobiFlight...`' functions if you wish to pass multiple or single commands at once. We would recommend just calling the functions directly if you're using this file for your own firmware or software, but this is here if you want to use it. You can see the full list of available calls in the 'if else' statement of '`void handleMobiFlightCmd(char *cmd)`'.

A single MobiFlight command looks like this:
'`setSpd=100`'
Multiple MobiFlight commands look like this:
'`setSpd=100,setHdg=270,toggleTrkHdg=1`'

'`void handleMobiFlightRaw(char *cmds)`'

Handles multiple MobiFlight commands separated by commas. Parameters:

- 'cmds': A string containing MobiFlight commands separated by commas.

'`void handleMobiFlightCmd(char *cmd)`'

Handles a single MobiFlight command. Parameters:

- 'cmd': A string containing a MobiFlight command.

Hardware Level Specification

Our device comes pre-soldered on a PCB with a driver chip, and everything is set up in our library above to make it all work. You shouldn't need anything more than the above. If you do, please contact us and let us know and we will provide you with the information you need.

Interface Configuration

- Pin 1: CS (Serial Shift Pulse Input)
- Pin 2: CLK (Serial Shift Pulse Input)
- Pin 3: DI (Serial Data Input)
- Pin 4: VSS (Power Ground)
- Pin 5: VDD: (Power Anode)

Operating Data

- Operating Voltage: 4.8V 5.2V
- Operating Current: <300uA (5.0V)
- Operating Temperature: -10°C +60°C
- Storage Temperature: -20°C +70°C

Separate Backlight Power

The backlight is powered by the same input as the LCD and driver chip. This means that out of the box, the backlight cannot be dimmed. This was a design choice we had to make in order for the unit and PCB to be within the size constraints we needed. The intensity of the LCD is suitable, and it should be noted that as with any LCD, the viewing angle will change the intensity also. The LCD is designed to viewed straight on, and so viewing it from above or below will reduce the intensity.

If you really need to power the backlight independently, then it is possible. First, remove the resistors for the LED's from the PCB (shown in figure 2. Then for each backlight 'A' terminal (there are 2, one on either side of the board), you need to place two 200 Ohm resistors in parallel between your 5V power in and the 'A' terminal. This is shown in figure 1.

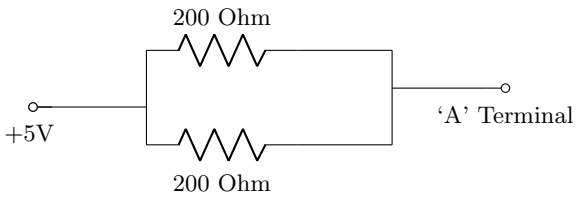


Figure 1: Circuit Diagram

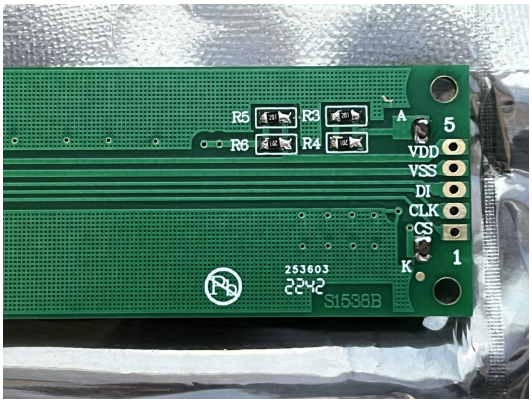


Figure 2: Back of PCB

Final Words

A fully working example Arduino project is included with the project for free. You can simply plug the device in and see it working on a simple loop. This should give you a good idea of how the device works and how to get it up and running. This can be downloaded on our website and through our GitHub.

Thank you for taking the time to read this document, and we hope that you found it useful. If you do still have any questions, then please contact us at info@jak-kav.co.uk.