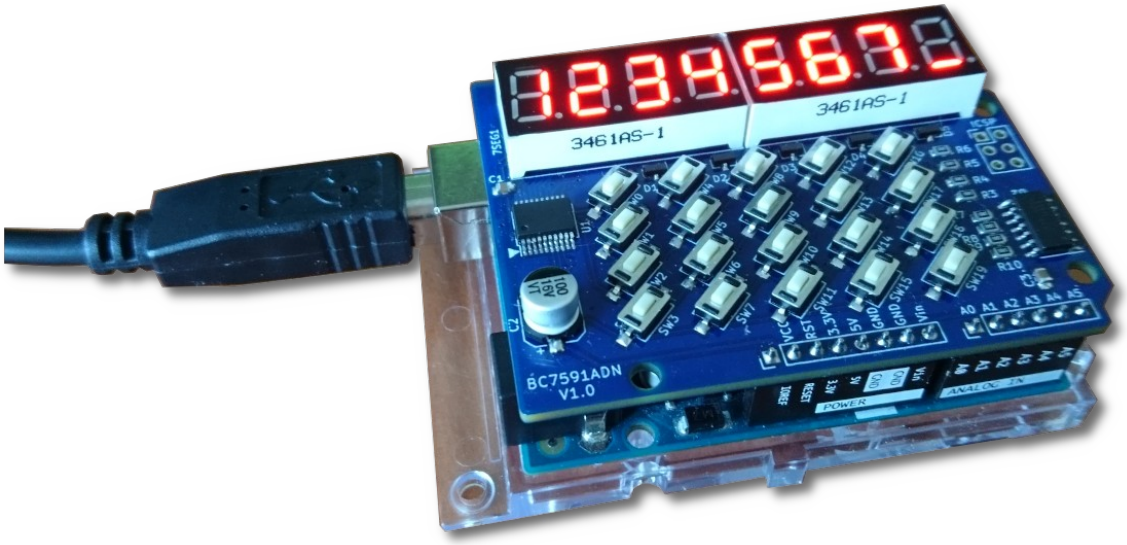
BC7591ADN

8 Digits Display & 20-key

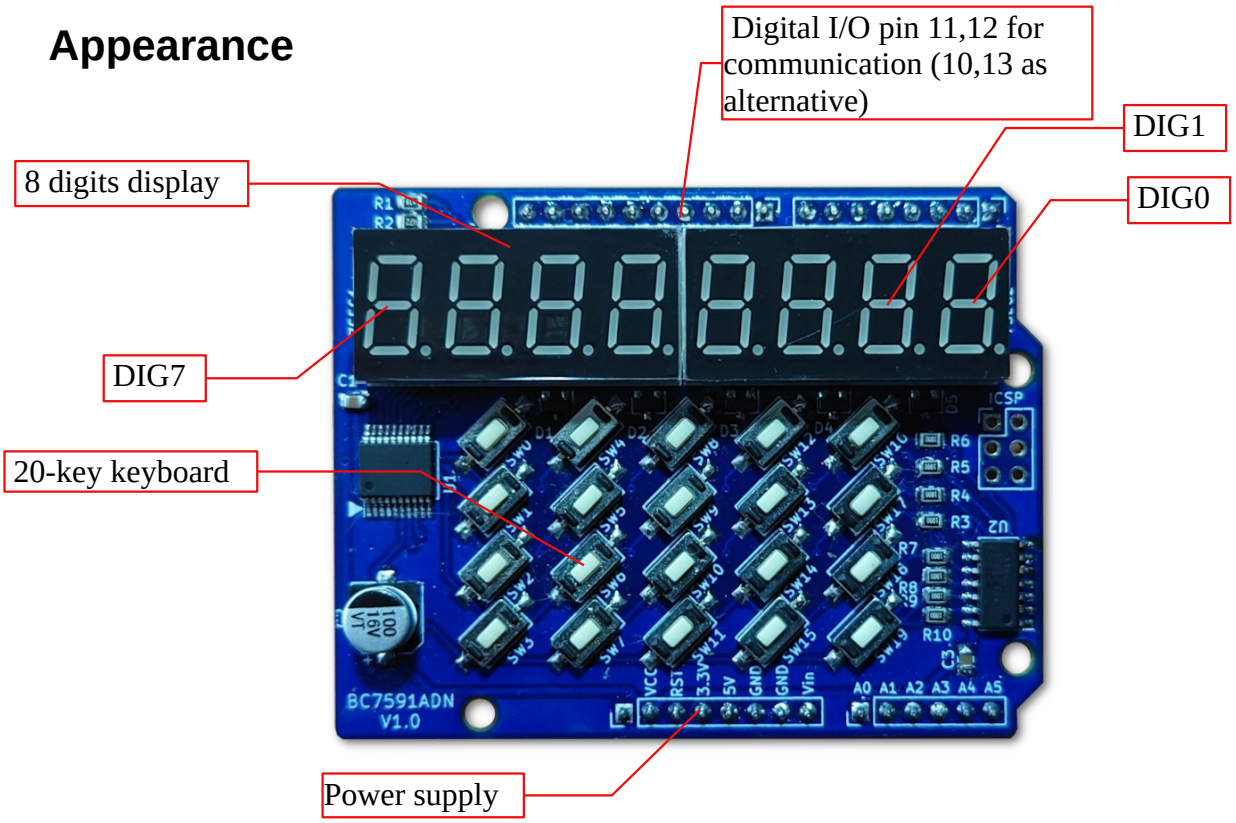
Arduino Shield

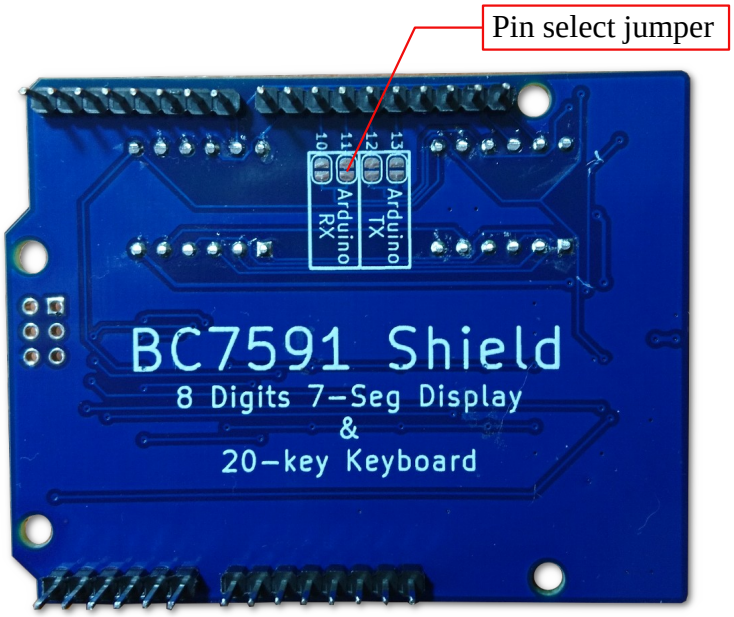


- Libraries provided and could be installed directly in the Arduino IDE
- Works with any UNO and MEGA sized Arduino board
- Only two digital I/O pins used, configurable pin numbers
- All segments individually blinkable, flash frequency adjustable
- Decimal and hexadecimal display functions
- Display brightness adjustable
- Supports long key press detection and key combinations

BC7591ADN shield provides an 8 digits 7-segment numeric display with decimal points and a 20-key keyboard with key press/release detection, and supports long key press and key combination.

Appearance





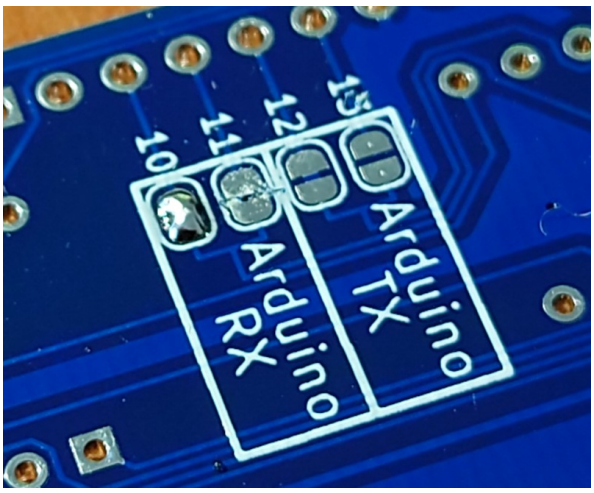
Communication Pin Configuration

The BC7591ADN Shield is based on the BC7591 chip, which is a LED display and keyboard interface chip with up to 32 7-segment displays and 96-key keyboard driving capability.

The BC7591 chip adopts serial communication with a baud rate of 9600, where the RX of BC7591 (connected to the TX of Arduino) is the display interface and the TX of BC7591 (connected to the RX of Arduino) is the keyboard interface. Two libraries are used for this shield: Key Scan Library and Display Library. The reason for separate into 2 libraries is because some chips like the BC6xxx series are scan only, and the separation also make the code for the display-keyscan chip a more streamlined program when keyboard is not used. The driver library itself supports both Arduino hardware and software serial port, because the hardware serial port on Arduino UNO is generally used to download program, so BC5791ADN Shield chooses to use software serial port, the digital I/O pin is user-selectable through the solder jumper on the back, the user can choose to use digital pin 10 or 11 as the RX of the Arduino software serial port, and 12 or 13 as the TX of the software serial port.

The board is shipped with digital pin 11 and 12 connected for communication with Arduino. If these I/Os are already used for other purposes, you can disconnect them to shield and use a jumper to connect 10 and 13 as alternatives. As shown below:

Cut the original connection

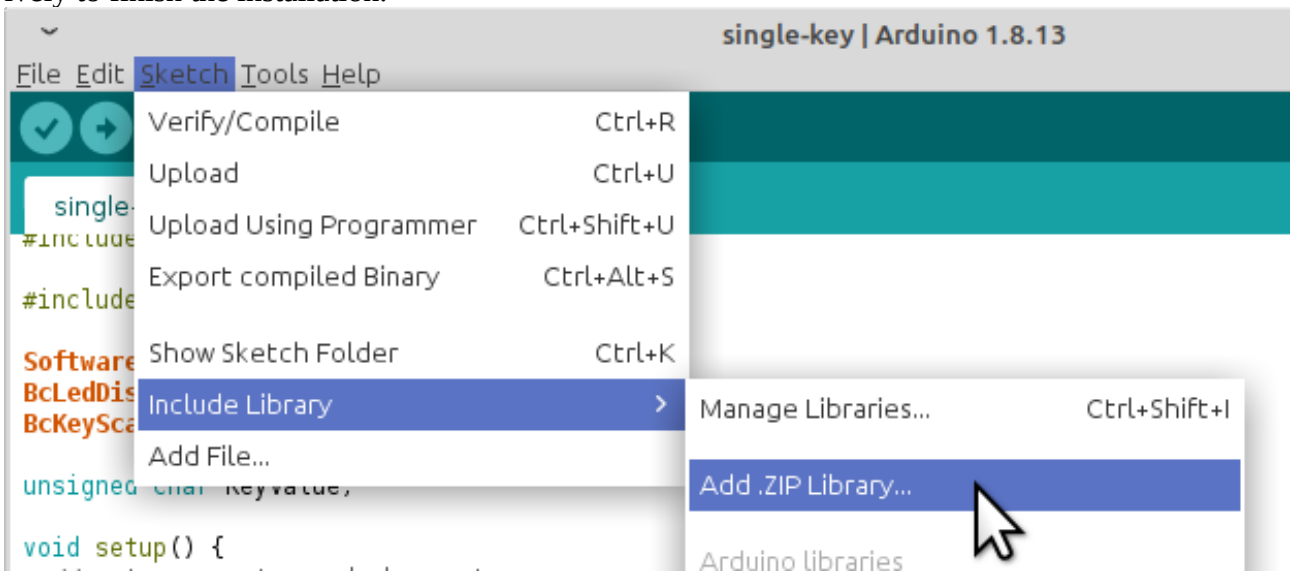


Use solder jumper to connect alternative I/O

Please note that after modifying the connection, the example Sketch source program must also be modified accordingly.

Library Installation

The driver library contains two files, BC_led_disp.zip and BC_key_scan.zip, which are the display driver library and the keyboard driver library respectively. To install, open Arduino IDE, select "Sketch -> Include Library -> Add .ZIP Library", and select these two .ZIP library files respectively to finish the installation.



The library manuals contain further instructions on how to install them manually, etc.

Usage

The following section explains the process of using the shield starting from creating a new project. It is recommended that first-time users do not create a new project from scratch, but take the four examples provided as templates and modify the program to meet their needs. Modifying only one point at a time, run and test it after every modification, so that the program can be debugged more easily.

Including Library

The BC7591ADN Shield uses Arduino's software serial port, so the Arduino SoftwareSerial library must be loaded at the same time. This library is one of the standard libraries for Arduino. Open the Arduino IDE, in the "Project -> Load Libraries" menu, load the following three libraries.

SoftwareSerial
BC_key_scan
BC_led_disp

Creating Instance

First, an instance of the software serial port needs to be created for use by the display and keyboard driver libraries. After the #include statement and before setup(), enter :

```
SoftwareSerial     swSerial(11, 12);
```

swSerial is the name given by the user to the software serial instance, it can be anything, as long as it conforms to the general Arduino variable name rules. 11 and 12 are the digital I/O port lines used by the software serial port, 11 is RX and 12 is TX respectively. These two values cannot be changed , they are determined by the hardware connection of this Shield and the Arduino board. The port number could be different on different Arduino board. For example on WeMos ESP8266 boards these same pins are numbered 13 and 12. Check with your Arduino board's specification to figure out the correct pin number.

Once you have an instance of the software serial port, you can create an instance of the display and keyboard, At the bottom of the statement that creates the software serial port instance, enter :

```
BcLedDisp          Disp(swSerial);
BcKeyScan          Keypad(swSerial);
```

In the above two statements, swSerial is the name of the instance of the software serial port created before, and Disp and Keypad are the names of the instances of display and keyboard respectively. As the serial port instance, they can be named arbitrarily by the user, as long as they conform to the naming rules of Arduino variable names. The meaning of these two statements is that both the display and the keyboard use the swSerial software serial port as the communication port.

Initialization

The serial port must be initialized before it can be used. Therefore, in the setup() section, the following serial port initialization operation must precede any display library operation that requires sending data.

```
swSerial.begin(9600);
```

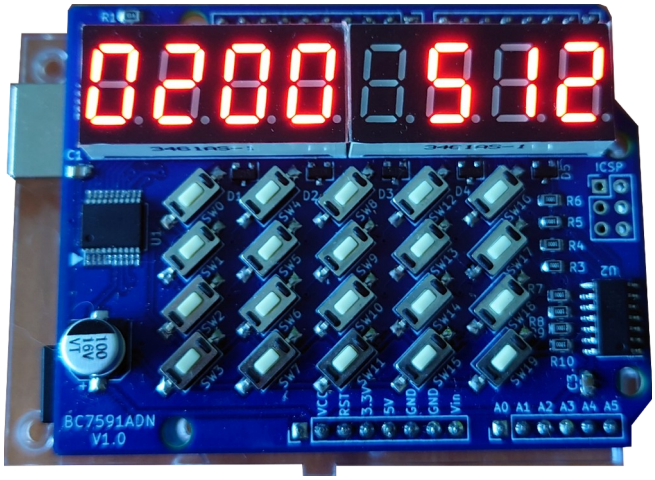
This operation makes the parameters of the serial port consistent with those required by the BC7591 chip.

The keyboard and display driver library itself can work without initialization, and the default settings of the driver library should be suitable for this Shield configuration. However, some

operations, such as clear(), can be placed in setup() to ensure that the display can return to a definite state after each reset. In addition, if you want to use the function of long-press keys and key combinations in the keyboard library, you need to do some long-press keys and key combinations definition work, which can also be done in setup().

Example Code Descriptions

1. Display Only Example (display-only.ino)



This example does not use the keyboard function, but only demonstrates the display driver function, and therefore only loads the display driver library.

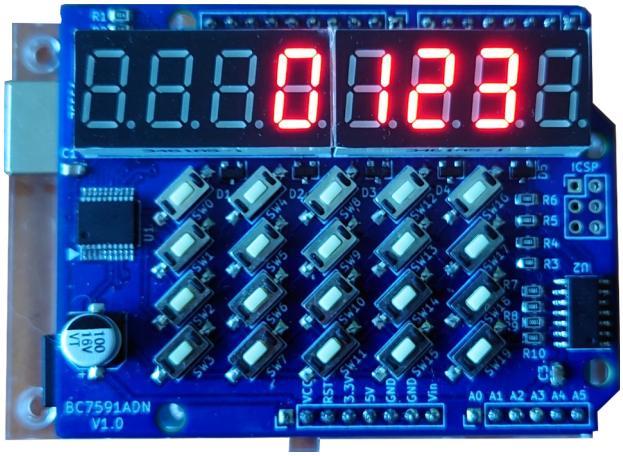
The program starts by displaying a fast counting, which is divided into two parts, decimal and hexadecimal, and is displayed simultaneously on the four digit blocks on the left and right side, in other words the same value is displayed in hexadecimal and decimal way. The counting starts from 0 and ends at 0x200, or 512 in decimal. This part mainly demonstrates the function of the hexadecimal and decimal display functions of the driver library.

When the counting is over, a light dot is displayed and does a rotation around the entire 7-segment area, first in a clockwise direction and then in a counterclockwise direction. This part of the program demonstrates the use of the sendCmd() function to directly send the segment addressing (SEG_ON, SEG_OFF) instructions of the BC7591 to control the ON and OFF of any display segment.

The third part of the demostration, displays the number "12345.678" with the decimal point blinking, showing how to use the sendCmd() function to control the decimal point and the blinking of any display segment.

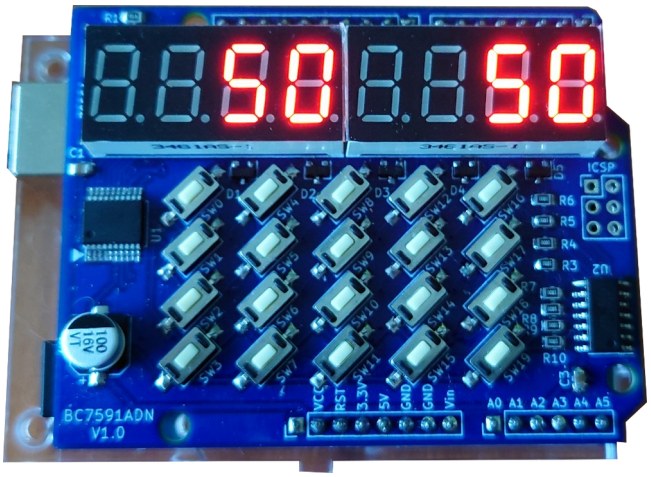
In the fourth part of the demo program, the characters "brt=" is displayed on the left side and blinking, a hexadecimal value behind the equal sign, while on the rightmost position displays a two-digit count. Hexadecimal value indicates the current brightness value, it changes every 50 counts an dranges from 0 to F, then back to 0. The user will see the brightness of the digital tube gradually change from highest to dark and then to highest. This part mainly demonstrates the special character display, the use of digitBlink() function, and the display brightness control.

2. Single Key Press Example (single-key.ino)



This example demonstrates simple key processing. The 20 swithes with key value 0-15 are defined as hexadecimal numbers 0-F, while the last 4 are defined as special characters "P", "H", "L", and space. If a switch is pressed, the corresponding character is displayed at the right most 7-segment, while the previous display is shifted to the left, and then the code is waiting for the next key press.

3. Full Featured Keyboard Example (full-feature.ino)



This example demonstrates the use of long-presses keys and key combinations, including long-presses of combination keys. The program displays 2 decimal number from 0 to 100 simultaneously on the left and right block of the 7-segment displays, in blinking, with an initial value of 50. When switch 17 (the '+' key) is pressed, the data on the right side is added by one, and when switch 18 is pressed, the data on the right side is subtracted by one. If you press and hold switch 17 or switch 18 for 2 seconds, the flashing of the number on the right side will stop and the number will start to increase or decrease rapidly until the key is released or the number reaches 100 or 0. If you press and hold switch 3 and then press switch 17 or switch 18 to form a key combination, the number on the left side will be adjusted instead, and the key combination also supports long-presses, and a long-press on the key combination will also stop the flashing of the number on the left side and increase/decrease rapidly.

This example demonstrates the use of long-press keys and key combinations. Please refer to the manual of the keyboard driver library for a detailed description of the usage of the long-press keys and key combinations.

Appendix 1: Brief description of common functions of the driver library

Display Library：

clear() - clear display

This function clears everything that is displayed and clears all blinking attributes.

displayHex(Val, Pos, Width) - Display value in hexadecimal

The value Val is displayed in hexadecimal starting from the Pos digit, and the width is Width characters.

displayDec(Val, Pos, Width) - Display value in decimal

The value Val is displayed in decimal starting from the Pos digit, and the width is Width characters.

digitBlink(Digit, On/Off) - Digit blink

Control whether a 7-segment digit blinks, On/Off control whether to blink, 1 is blinking, 0 is not blinking

sendCmd(Cmd, Data) - Send command to BC7591

This function sends commands directly to the BC7591, Cmd is the BC7591 command, and Data is the data. All the library functions are working through this function. Through this function, you can control the BC7591 chip on the shield directly to reach some special features of BC7591 not provided by the library, such as controlling the display brightness, shifting the display content left/right, etc. Please refer to the BC7591 chip datasheet for detailed BC7591 instruction description.

Key-scan Library：

setDetectMode(Mode) - Set detect mode

When Mode is 0, only key press is detected and key release is ignored. When it is 1, both key press and release are detected.

checkChanges() - Update keyboard status

In loop(), checkChanges() should be called periodically, so that the library can keep track of the status of the keyboard.

isKeyChanged() - Check for new keyboard changes

The return value would be true (1) or false (0), and this query allows the user to know if the keyboard has new key events that have not yet been processed.

getKeyValue() - Get key value

This function returns the most recent key value. The keys here are not only include individual physical keys, but also user-defined key combinations and long-press keys.

setLongpressCount(CountLimit) - Set long-press count

The time counting of the long-press keystroke is implemented through an internal counter. The counter is considered to have reached the long-press time when it reaches a set limit, and this function is used to set this limit.

longpressTick() - Tick the long-press counter

When using a long-press key event, this function is called periodically in loop(), and when the set limit of times is reached, the long-press time is considered to be over and a long-press key event is generated.

defLongpressKey() - Define long-press keys

Before you can use long-press keys, they must be defined so the library would know which keys to monitor for long-press actions and what custom key values are assigned to those long-pressed keys. Please refer to the manual of the key-scan library for details on how to use it.

defCombinedKey() - Define combined keys

The use of key combinations is similar to that of long-press keys, it is necessary to define in advance which key combinations are to be detected by the library and to assign a special key value for each key combination. See the library manual for more details on the usage.

Appendix 2: Most common BC7591 commands

The sendCmd() function allows you to directly control the operation of the BC7591, the main chip on this shield, to perform many specific actions, such as controlling the display brightness, displaying special characters, and so on. For a complete description and usage of the BC7591 commands, please refer to the datasheet.

DIRECT_WT - Write directly to the display register, mainly used to display special characters, such as P, H, etc. Format:

```
sendCmd(DIRECT_WT+pos, data);
```

Where, `pos` is the display position, `data` is the bit mapped display data, segment is lit when corresponding bit is 1.

BLINK_WT_CLR - Segment blink clear. Format:

```
sendCmd(BLINK_WT_CLR+pos, data);
```

Where `pos` is the display position, the segment blink attribute corresponding to all the '1' bits will be cleared.

BLINK_WT_SET - Segment blink set. Format:

```
sendCmd(BLINK_WT_SET+pos, data);
```

Where `pos` is the display position, the segment blink attribute corresponding to all the '1' bits will be set.

SHIFT_H_WT - Insert towards higher position. Inserts the data into the display register while shifting the existing display toward the high display position. Command format:

```
sendCmd(SHIFT_H_WT+pos, data);
```

where `pos` is the display position and `data` is the bit map data of display content.

SHIFT_L_WT - Insert towards lower position. This command is similar to the previous command, but the insertion moves in the opposite direction.

SEG_ON - Segment ON, turn on a specific segment. Format:

```
sendCmd(SEG_ON, seg_number);
```

`seg_number` is the segment address, the segment A of digit 0 is assigned address 0, the segment DP is 7, the segment A of digit 1 is addressed 8, and so on.

SEG_OFF - Segment OFF, turn off a specific segment. Command format is the same as above.

WRITE_ALL - Write to all display registers. It can be used to turn on all segments, turn off display, etc. Usage format:

```
sendCmd(WRITE_ALL, data);
```

BLINK_SPEED - Blink speed control. The blink frequency can be roughly calculated using formula $F_{blink} = 92/(2*S)$. Usage format.

```
sendCmd(BLINK_SPEED, S);
```

DIM_CTL - Display brightness control. The control value ranges from 0 to F. When it is 0x00, the brightness is maximum and when it is 0x0F, it is the darkest. Usage format:

```
sendCmd(DIM_CTL, ctl);
```

Appendix 3: Schematic