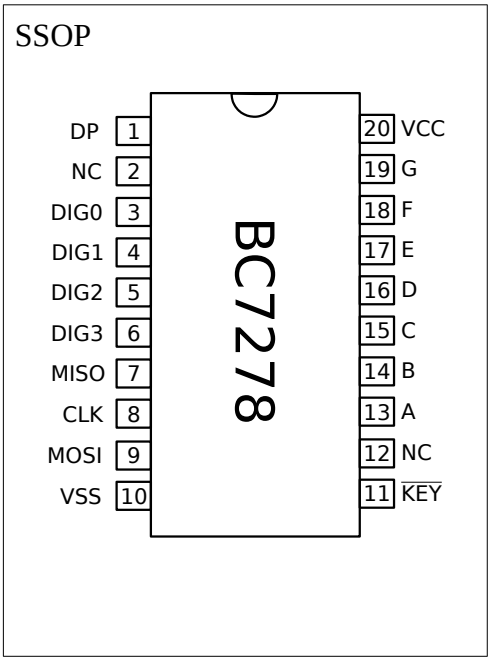
BC7278

4 Digits 32 Segments LED Display

With 16-key Keyboard Interface

Features:

- Drives 4 7-segment numeric or 28 individual LEDs
- Blink function for individual LED or whole character
- Adjustable blink frequency
- Individual control for each segment
- Separate control for decimal point for numeric display
- Direct write to display register (display user characters)
- Keyboard supports combined keys and long-press
- 2 or 3 wire SPI interface
- SSOP20 package
- Software compatible with other BC727x series



Abstract

The BC7278 features a single chip 4-digit LED display management and 16-key keyboard interface. By supporting segment addressing, each display segment can be controlled independently so it is also very suitable for driving individual LEDs, up to 32 individual LEDs can be driven. The BC7278 supports blinking display function with adjustable blinking speed, each LED can be controlled independently for blinking property, or blinking can be controlled by digit for a numeric display.

The BC7278 provides an internal decoding function that allows the user to write a value directly into the decoder register to obtain the corresponding numeric display. The decimal point segment is not affected by the decoding, so the user only needs to update the figure data and does not need to consider about refreshing the decimal point, which is particularly convenient for users who has fixed decimal point and will use that segment as a separate LED. It also supports writing to display register directly, which can be used for displaying some special characters such as ‘L’, ‘H’, ‘P’.

The keyboard interface can support up to 16 keys with built-in debounce feature, can support any combination of keys, long-press keys, and can support normally open or normally closed switches.

The SPI port supports communication rate up to 64Kbps. The unique internal SPI port reset logic allows the SPI interface to be reset when the communication is idle, ensuring error-free communication when CS is not present. A separate key status change indication output, $\overline{\text{KEY}}$ pin, can be used as an interrupt trigger to the controller.

A driver library is provided, using the library the display and keyboard functionalities can be done in a few lines without knowing the underlying details of the chip.

Absolute Ratings

（Note: Exceeding the listed range may cause permanent damage to the device）

Storage Temperature	-65 to +150°C
Working Temperature	-40 to +85°C
Voltage on any pin	-0.5 to 6.0V

Electrical Characteristics

（Unless otherwise specified, T_A=25°C, Vcc=5.0V）

Characteristic	Min	Typical	Max	Units	Comments
Supply Voltage	2.7	5.0	5.5	V	
Working Current		4.9		mA	
Low Level Input Voltage			1.4	V	
High Level Input Voltage	3.7			V	1.9V when Vcc=3V
Output Low Level Voltage			0.1	V	
Output High Level Voltage	4.4			V	
Display Scan Period		14		mS	

Pin Descriptions

Pin Name	Pin Number	Function
DIG0-DIG3	3,4,5,6	Digit drive output, connected to the common cathode pins of 7-segment numeric LED display
$\overline{KEY}$	11	Keyboard status, change when keyboard status is changed
MISO	7	Data output, connected to the SPI data input of MCU, push-pull output
CLK	8	SPI clock, input
MOSI	9	SPI data input, connected to data output of MCU
GND	10	Ground
A-G, DP	1,13,14,15, 16,17,18,19	Segment drive output, connected to the anode pins of LEDs or 7-segment displays.
VCC	20	Power supply, 2.7-5.5V

Registers

There are 13 internal registers, including 4 display registers, and 9 control registers. The address range is 00H-1DH, where 00H-0FH are display registers and the rest are control registers.

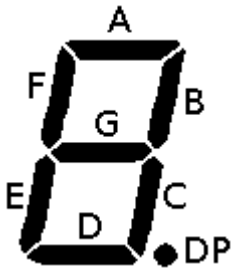
Address	Name	Default	Comments
00H	Display register for digit 0	FFH	Each bit is mapped to a display segment
01H	Display register for digit 1	FFH	
02H	Display register for digit 2	FFH	

© Bitcode Technology

Address	Name	Default	Comments
03H	Display register for digit 3	FFH	
04H-0FH	(not used)		
10H	Blink control for digit 0	00H	Each bit is mapped to a display segment
11H	Blink control for digit 1	00H	
12H	Blink control for digit 2	00H	
13H	Blink control for digit 3	00H	
14H-17H	(not used)		
18H	Character blink control	00H	Bit0-bit3 are mapped to digit 0-3, 1=blink,0=normal
19H	(not used)		
1AH	Blink frequency control	10H	Smaller value for faster blinking
1BH	Hexadecimal decoder	-	Value written to this register is displayed as a 0-F hexadecimal character on a 7-segment display
1CH	Segment control	-	To control each segment individually
1DH	Global write	FFH	Value will be written to all display registers

Display Registers：Address 00H-0FH

The display register is directly mapped to each LED display segment, and the mapping relationship between the bits and each LED on the 7-segment is as shown in the figure



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DP	G	F	E	D	C	B	A

User can directly change the contents of the display register to update the display, which is mainly used for displaying special characters that are not in the decoding table, such as 'H', 'L', 'P', '-' etc.
If a bit in the display register is set to 0, the display segment is ON. After reset, the contents of all display registers are set to FFH.

Blink Control Registers：Address 10H-13H

Similar to the display registers, the blink control registers also uses the same bit mapping scheme to control the blink of the LEDs, each bit corresponds to a display segment.

When the bit is 1, the corresponding LED has the blink property ON. Blinking occurs only when that display segment is lit. If that display segment is OFF (the corresponding display register bit is set to 1), nothing is displayed for that display segment. When the content of the display register is cleared (the corresponding bit is set to 1), its blinking property is not affected, and when that segment(LED) is set to 0(ON) again, it will still be a blinking display.

After reset, the blink control registers are all cleared to zero (non-blinking).

Character Blink Control Register: Address 18H

The character blink control register controls the overall blink property of a 7-segment character, each bit in the register corresponds to a display character, the correspondence is as follows

18H							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
-	-	-	-	Digit 3	Digit 2	Digit 1	Digit 0

When a bit is 1, the character is in blinking mode. Blinking occurs only if that character has some display content; if that character is blank (all bits of the display register are 1), there will be no blinking. When reset, the character blinking control register is set to 00H (no blinking).

Blink Frequency Control Register: Address 1AH

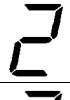
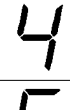
The BC7278 has an adjustable blink frequency, and it can be easily controlled by simply changing the blink frequency control register. The larger the value in the register, the slower the blink speed. After reset, the value of this register is 10H, and at this value, the blink rate is about 2Hz.

Hexadecimal Decoder Register: Address 1BH

Through the decoding register, the user can get the hexadecimal number displayed on a 7-segment directly by feeding in the value, saving the user from the trouble of preparing the mapping table by himself. The format of the data written into the decoding register is as follows.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
A_3	A_2	A_1	A_0	d_3	d_2	d_1	d_0

Where A3:A0 is the character address, which determines the position of the hexadecimal digit display. The d3:d0 is the value to be displayed. When the address is 0000b, the decoding result is displayed on digit 0, and when it is 0011b, it is displayed on digit 3. A3 : A0 has the value range of 0000-0011. d3:d0 is the value to be displayed. The decoding table is as follows.

d_3	d_2	d_1	d_0	d_3:d_0 (hex value)	Display
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	2	
0	0	1	1	3	
0	1	0	0	4	
0	1	0	1	5	
0	1	1	0	6	

d_3	d_2	d_1	d_0	d_3:d_0 (hex value)	Display
0	1	1	1	7	
1	0	0	0	8	
1	0	0	1	9	
1	0	1	0	A	
1	0	1	1	B	
1	1	0	0	C	
1	1	0	1	D	
1	1	1	0	E	
1	1	1	1	F	

Updating the display through the decoder register will not affect the decimal point, i.e. the state of the DP segment will remain unchanged. With this feature, when data with decimal point is displayed, or in the case of using the decimal point as independent indicator, the data update will be very convenient, because it can be done without considering refreshing the decimal point. The control of the decimal point can be done by segment control register.

Segment Control Registers: Address 1CH

The display can be controlled on a segment (individual LED) basis, which is achieved through segment control register.
By assigning an address to each display segment (LED), the segment control registers can be used to control the ON and OFF of each display segment. The address of each segment is as follows.

Digit	DP	G	F	E	D	C	B	A
0	07H	06H	05H	04H	03H	02H	01H	00H
1	0FH	0EH	0DH	0CH	0BH	0AH	09H	08H
2	17H	16H	15H	14H	13H	12H	11H	10H
3	1FH	1EH	1DH	1CH	1BH	1AH	19H	18H

The data format written to the segment control register is as follows.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Seg	A_6	A_5	A_4	A_3	A_2	A_1	A_0

Where A_0-A_6 is the segment address. Seg is the data written to the segment. when Seg is 0, the LED is ON, and when Seg is 1, the LED is OFF.

Global Write Register: Address 1DH

The global write register is a special register, the value written to this register is copied to all display registers at the same time. For purpose of clearing the display or turning on all LEDs, it can be easily achieved by writing FFH or 00H to this register. Note that writing to this register does not change other registers such as blink control, so if a display segment or character is set to blink before writing, even if FFH is written to the global write register, all the blink settings will remain unchanged.

SPI Port

The BC7278 uses an SPI serial interface that can be connected to a standard 3-wire SPI port (without using the CS chip select signal). If the keyboard is not used, the MISO can be left unconnected, so that at least two wires are needed to form the interface with BC7278. The bit rate is up to 64Kbps. For MCUs without hardware SPI resources, it can also be implemented with I/O ports in a very short code (bit banging).

1. Data Format

All the BC7278 commands are formed by 2 bytes. The first is the register address, and the second is the data. The most significant bit (MSB) comes first when it's transmitted. The data structure is as follows.

Register Address								Data Byte							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	A_4	A_3	A_2	A_1	A_0	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
MSB →															LSB

The lower 5 bits A_4-A_0 of the command byte are the register address, and the valid address range is 00H-1DH. If the value is over the valid range, that command will be ignored. Sometimes this feature can be used to send a 'pseudo command' to the BC7278 with the sole purpose of getting the keyboard status mapping output.

While receiving the command, the BC7278 outputs 16bit data on the MISO pin, which is the keyboard mapping data. The input and output are simultaneous. Each time the CLK pulse comes, the data is sampled by the BC7278 during the low level of the CLK pulse, and the keyboard mapping data is output at the same time. When using the hardware SPI in the micro controller, the SPI mode should be set to sample the MISO data on the rising edge of CLK. See the timing diagram for details.

The keyboard mapping data is also MSB first, see below.

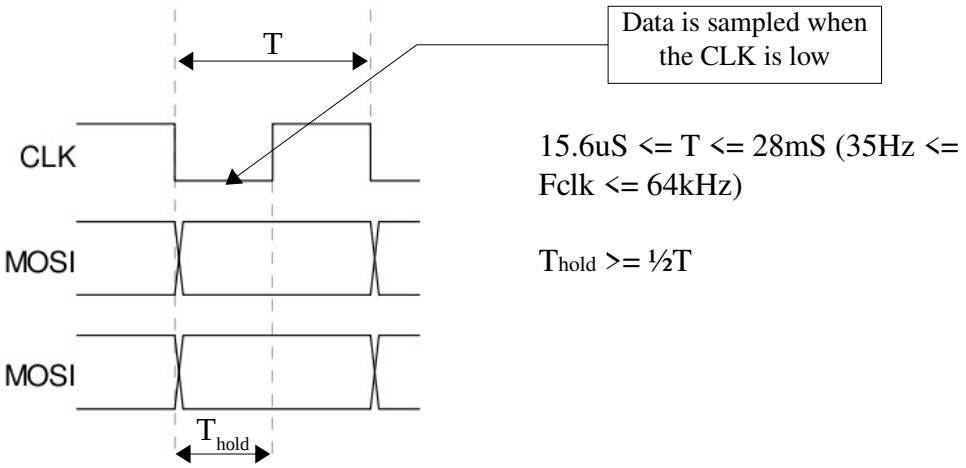
Keyboard mapping high byte								Keyboard mapping low byte							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
MSB →															LSB

When a key is in open state, the corresponding mapping bit will be 1. In other words, if no keypad is connected, the output value will be 0XFFFF. The correspondence between the data bits and the keys is shown in the following table, please refer to the schematic in the keyboard section later.

	D	C	B	A
E	S3	S2	S1	S0
F	S7	S6	S5	S4
G	S11	S10	S9	S8
DP	S15	S14	S13	S12

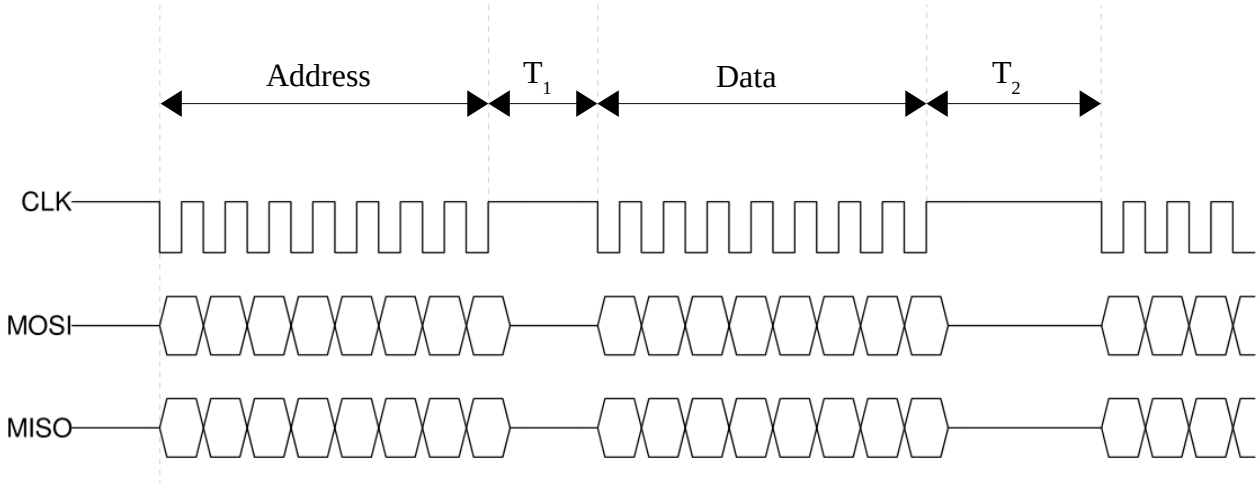
2. Timing

The SPI can be connected in a standard 3-wire fashion, including data (MOSI, MISO) and clock (CLK). If the keyboard is not used, the MISO connection can be omitted.



If you are using hardware SPI, please set the port to the following parameters: CLK idle high, data changed at the first clock edge (falling edge), sampling happened at the second clock edge (rising edge) with the data rate ≤ 64Kbps.

If using the bit banging method, since the data is sampled by the BC7278 on average during the CLK low level, it is necessary to set the data line before the CLK level changes to low and maintain the data line stable for at least half the clock cycle or more (Thold>½T).



T1 is the time from the rising edge of the last clock pulse of the address byte to the falling edge of the first clock pulse of the data byte in a command. T1 should not be shorter than ½T. When using the

hardware SPI, this time is guaranteed by the hardware, but if your controller cannot generate this time, or if the SPI is emulated by software, it is necessary to manually added a delay here. If the SPI can supports 16-bit mode, the address byte and data byte can be combined into a single 16-bit data transmission and the T1 is guaranteed.

T2 is the delay between the rising edge of the last clock pulse of the previous command and the falling edge of the first clock pulse of the next command, which is required to be not shorter than ½ clock cycle T. If hardware SPI is used, this time will be guaranteed by hardware.

If T2 is greater than two scan cycles (approximately 28ms), the BC7278 internal SPI will be reset. It is recommended that the user program ensure that there are periodic intervals greater than 28ms between commands to keep the SPI clock and data remain synchronized therefore to prevent communication errors. Since human eyes can not catch any meaningful data that changes more than 10 times a second, it is recommended that the refresh period of the display be no less than 100ms, which will guarantee there is a regular idle interval on the SPI bus so it can get reset between commands. This will also save processor resources.

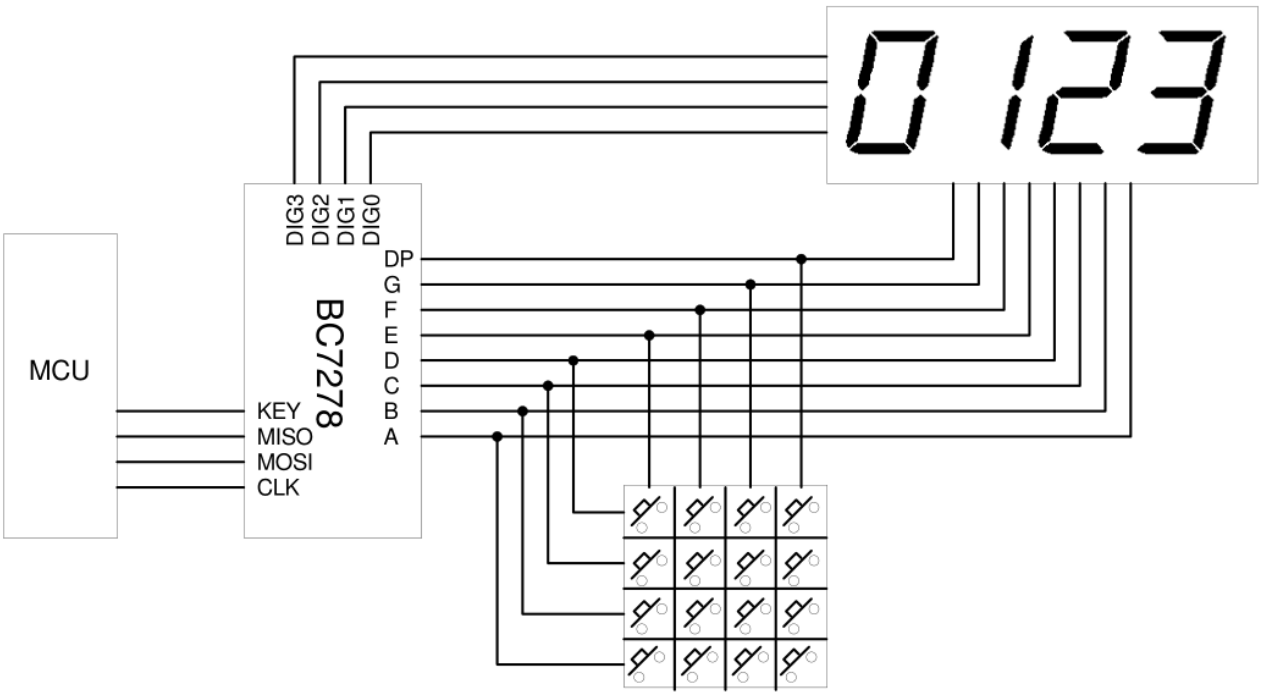
3. Interface Reset Mechanism

The BC7278 has a unique time reset mechanism that resets the SPI interface when the clock line is "silent" for more than 2 display scan cycles (about 28ms). In other words, when there is no level change on the CLK pin for more than 2 scan cycles, the currently received data will be discarded and the next bit clocked in will be treated as the MSB of a new command. This is important as there's no CS signal to reset the interface.

With the time reset mechanism, even if the previous command is corrupted (out of sync), as long as the time interval between the next 2-byte command and the previous one is greater than two scan cycles (during which there must be no clock pulse on the CLK line), the BC7278 will reset the internal receiving buffer so that the next instruction can still be received correctly.

Typical Circuit

Only minimum components are needed to form a multi-digit LED display and keyboard input circuit. A typical circuit is as follows: (schematic)



1. SPI Port

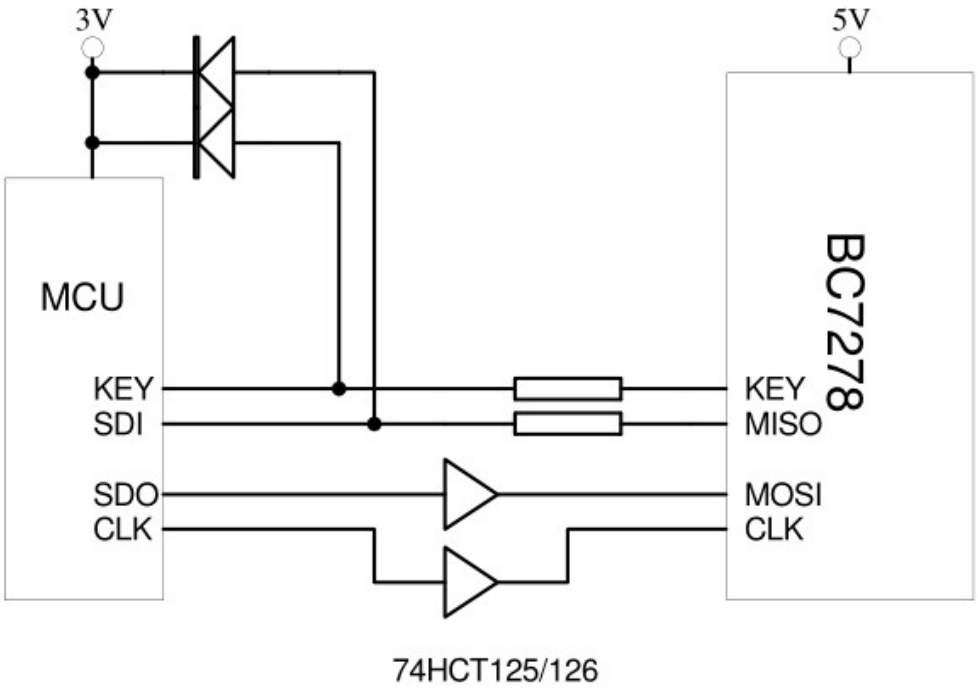
BC7278 has a SPI interface, the maximum bit rate is 64Kbps. It can have MISO, MOSI, CLK, 3-wire connection, or in 2-wire mode when the keyboard is not used, which needs only the CLK and MOSI.

Generally the SPI port on the micro controller can be set to different modes. When connected to the BC7278, it should be set to host (Master) mode, with CLK idle high and data passed on the first clock edge and sampled on the second clock edge. The clock frequency must not be above 64KHz.

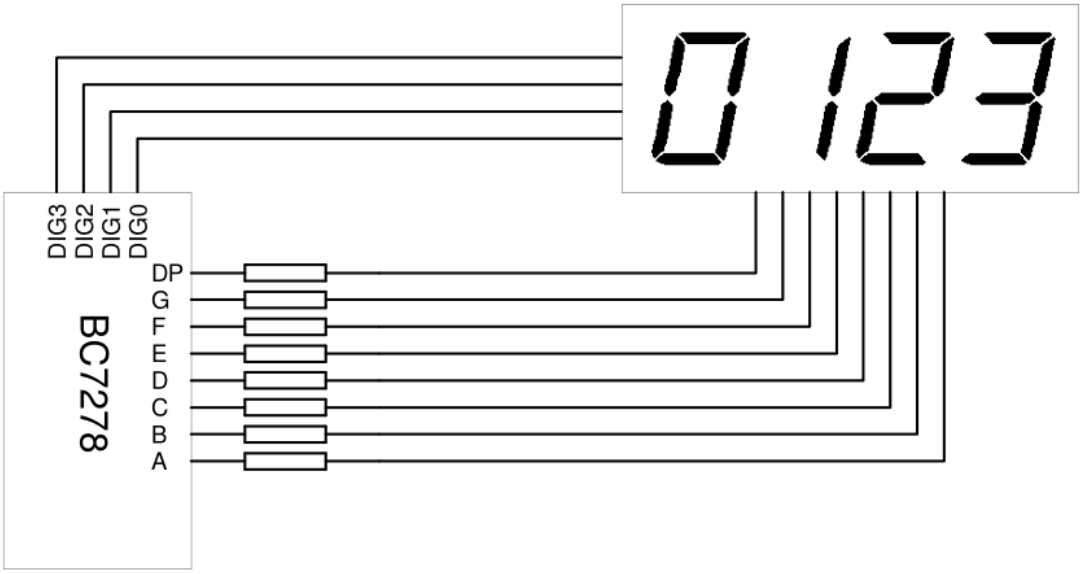
The MISO port is a push-pull output. You can connect the MISO of the BC7278 directly to the MISO pins on the MCU without pull-up resistors.

Since the minimum high-level input voltage of the interface is 3.7V, when the BC7278 supply voltage is 5V, it is not possible to interface directly with a 3V or 3.3V system, but since the BC7278 itself can work from a 3V supply voltage, it is recommended that the same power supply voltage be used for the BC7278 when the system micro controller is powered by 3V or 3.3V.

If the BC7278 needs to be powered by 5V to get higher brightness while the MCU still maintains 3V or 3.3V supply, an additional level conversion circuit is required. Resistors and diodes to the MCU VCC pin on the output ports MISO and KEY of the BC7278 prevent these pins from damaging by high voltage. 74HCT series logic circuits, when be powered by 5V, has a high level minimum input voltage of 2V so can accept input from the 3V system directly. The level matching problem can be solved by using such buffers in the communication line. As shown in the diagram below.



2. 7-segment Displays



The BC7278 can drive 7-segment numeric displays of common cathode types. DIG0-DIG3 are connected to the common cathode pin, while A-DP are connected to the anode pins through current limiting resistors.

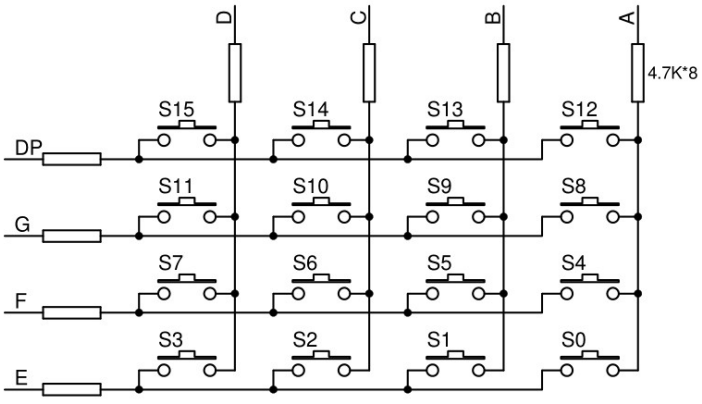
The resistors can be calculated using the following equation.

$$R = 67*(VCC-0.7-U_{LED}) – 50$$

Where VCC is the supply voltage on the BC7278, U_{LED} is the forward voltage drop of the LED (normally @10mA), the voltage unit is volts, and the calculated resistance unit is in ohms. For example, when VCC = 5V and U_{LED} is 1.8V, the calculated resistance is about 118Ω, can take an approximate value of 100Ω. When VCC is 3.3V and U_{LED} is 1.8V, the calculated current limiting resistance is 3.6Ω, you can use an approximate value of small resistance, or just omit. When VCC is 3V, the calculation output is negative, indicating that the current-limiting resistance can be omitted in this situation, and the brightness might be compromised.

3. Keyboard

The 16 keys are connected as a 4x4 matrix and connected to the pin A-DP, see the diagram below for the connection.



Because the keyboard and LED driver pins are shared, resistors are required to be put in between the matrix rows and columns and the BC7278 pins, the purpose of these resistors are to prevent short circuit between the driver outputs. A 4.7K resistor will be suitable for most applications.

Keyboard status is mapped into a register. All kinds of normally open and normally closed switches can be used with BC7278. However, since the default value of the mapping register is 0xFFFF at power-up, which corresponds to the open status of all switches, there will be a level change on the $\overline{\text{KEY}}$ pin at power-up if some of the keys are normally closed.

The $\overline{\text{KEY}}$ pin can be in any state at power-up, and the keyboard matrix is scanned every two scan cycles (about 28ms). In other words, the shortest change period of $\overline{\text{KEY}}$ pin, is two scan cycles.

The $\overline{\text{KEY}}$ pin level changes every time the keyboard status changes, but this does not mean for every individual key change there will be a level change on $\overline{\text{KEY}}$. If two or more keys' status are changed simutaneously within two scan cycles, the KEY pin level will only change once.

The level on $\overline{\text{KEY}}$ pin is not a reflection of the keyboard state. User should care about the changing of level on $\overline{\text{KEY}}$ pin, not the specific high or low level of the pin, because the same level can correspond to any keyboard state, and $\overline{\text{KEY}}$ pin can be high or low when no key is pressed. For example: if all switches are open at a certain moment and $\overline{\text{KEY}}$ pin level is high, at the next moment if two keys are pressed at the same time (within 2 scan cycle) and become on, then $\overline{\text{KEY}}$ pin will become low. Then after some time, one of the two pressed keys is released and become open circuit again, then $\overline{\text{KEY}}$ pin will return to high, it's original state, but the keyboard will still have 1 switch in conduct state. After this if the left switch is back to off, then all switches are in off state now, that is the initial state. But the $\overline{\text{KEY}}$ pin will flip once more to low level again. At the end the state of the keyboard is the same as the initial moment, but the state of $\overline{\text{KEY}}$ pin has changed from high to low.

Driver Library

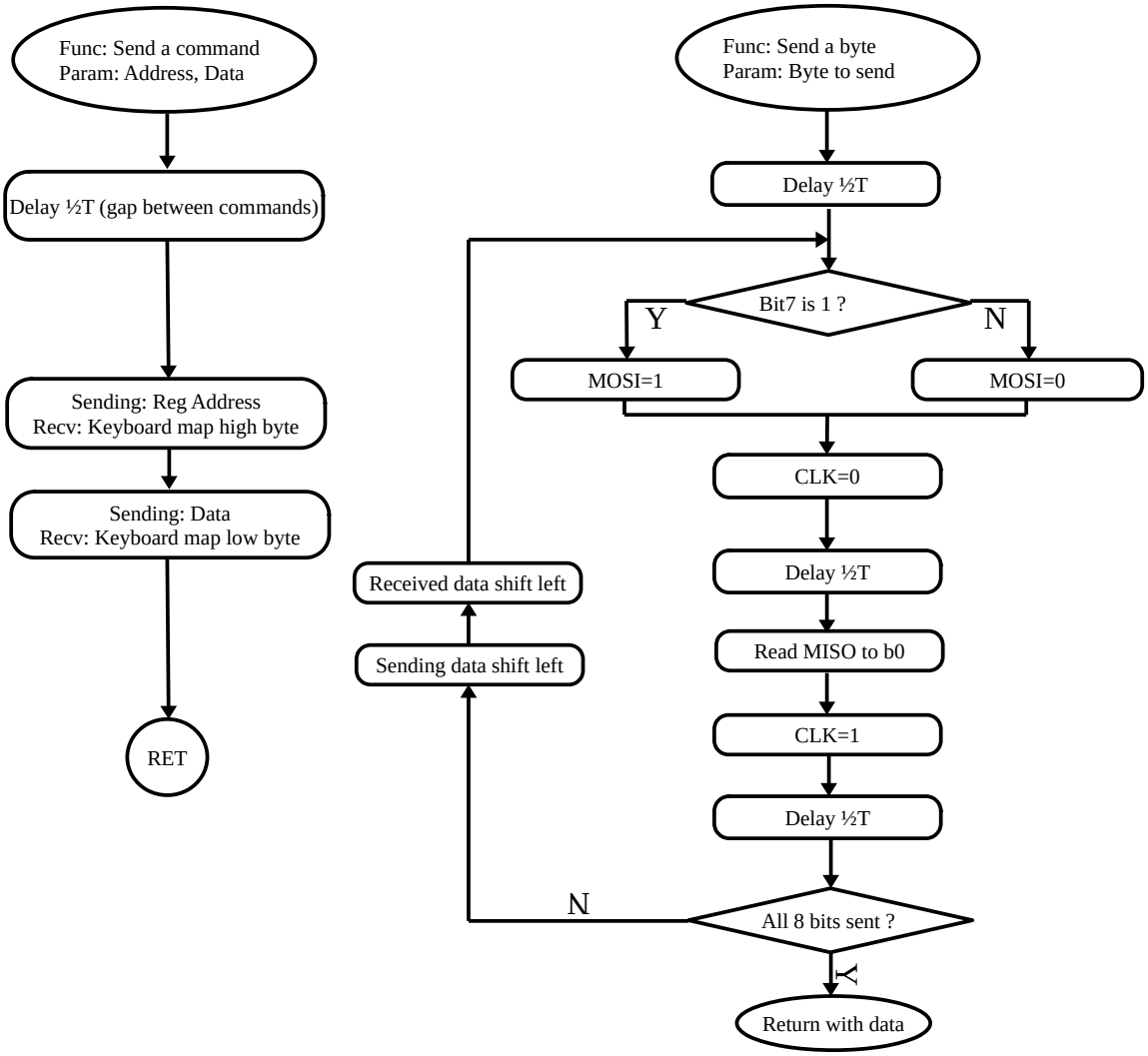
BC7278 is supported by the "BC727x series chip driver library". By using the driver library, users can have decimal, hexadecimal or float number displayed and complex keyboard functionalities by calling functions, including supporting for such as long-press keys and key combinations, without the need to know the internal registers of the chip. What the user needs to provide is the underlying SPI read/write functions, and the rest can be done by calling library functions. For details, please refer to the user manual of the driver library.

Program Flowcharts

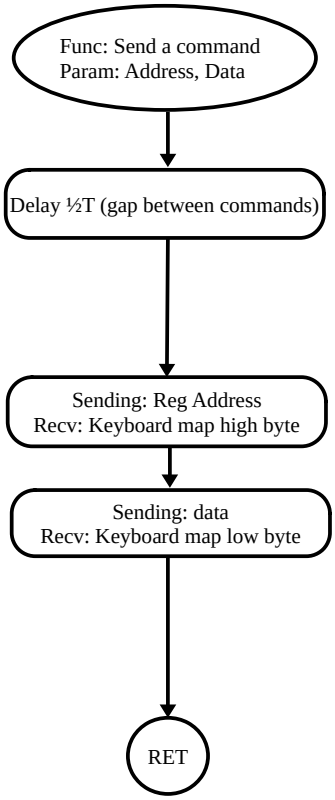
For users who do their own coding instead of using the driver library, please refer to the following flowcharts, which is divided into three types: software simulated(bit banging) SPI, hardware SPI interface + query method and hardware SPI + interrupt method.

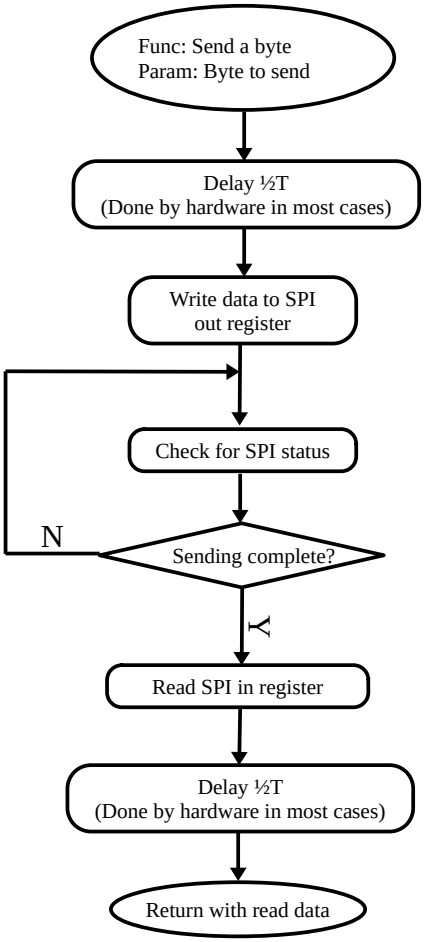
When doing the programming yourself, the keyboard handling may require some special considerations. The main caution should be taken is to prevent keyboard handling from interrupting display commands sending from main program, see later for details.

Flowchart For Software SPI:

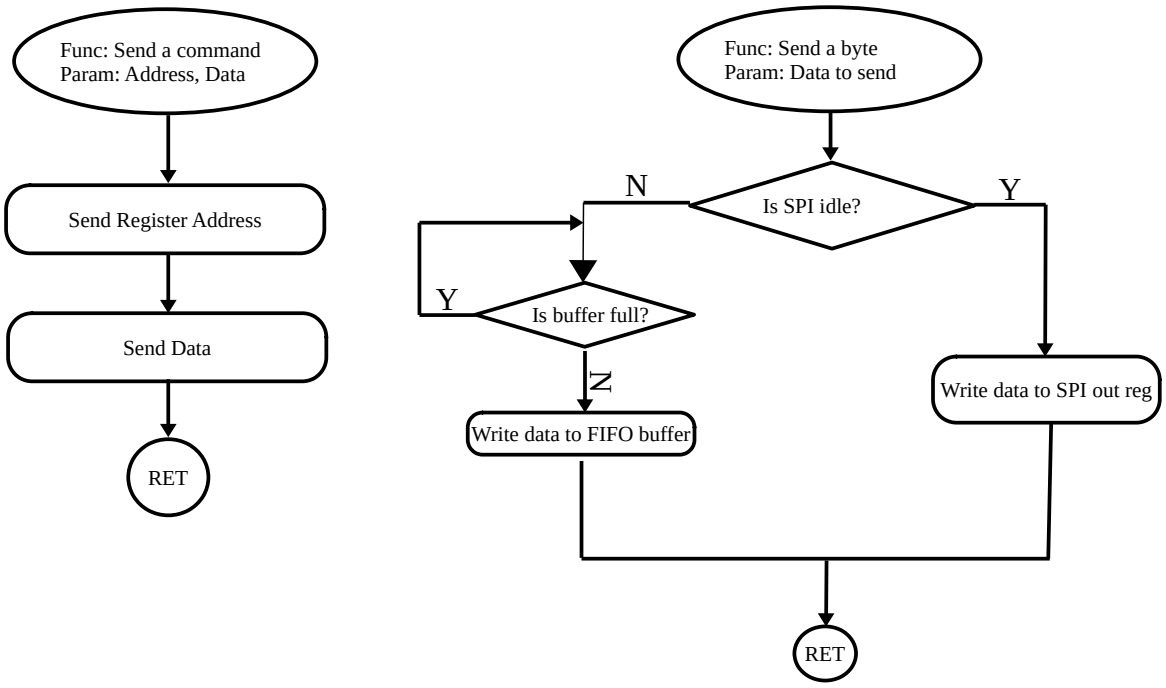


Flowchart For Hardware SPI+Query

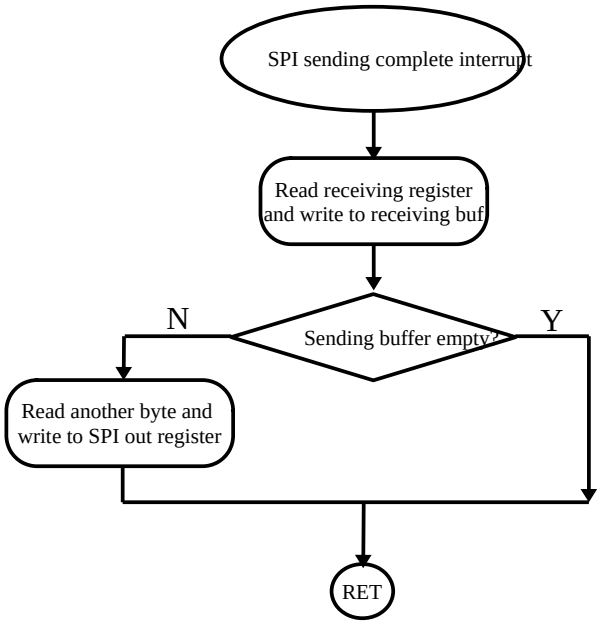




Flowchart For Hardware SPI+Interrupt:



ISR(Interrupt Service Routine):



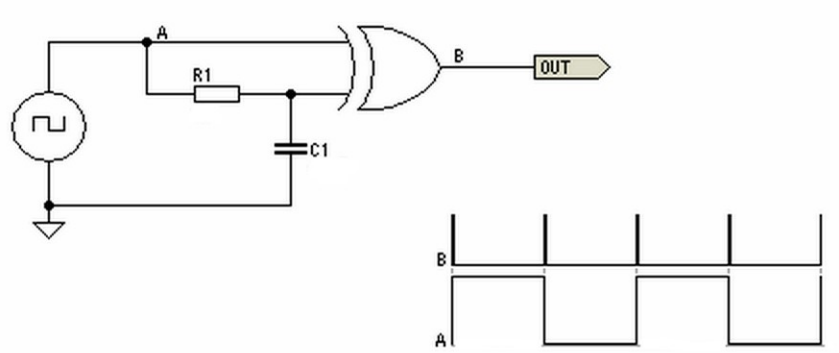
Note: In the interrupt mode, due to the sending buffer, there is a problem that the call of the "send a command" subroutine and the actual time when the data is sent on the SPI port are not synchronized, and for the same reason, the "send a byte" subroutine cannot return with the read-in data. Therefore, when using the interrupt method, a receive buffer is needed to store the received keyboard mapping data along with the send buffer. The SPI interface buffer increases the difficulty of keyboard processing.

Keyboard Processing

The BC7278 has a keyboard status change indication output $\overline{KEY}$, when the keyboard state changes, the level of $\overline{KEY}$ pin will be flipped, so monitoring the change of $\overline{KEY}$ pin is the best way to know the keyboard event. However, users need to pay attention that only the level change of $\overline{KEY}$ itself represents the change of keyboard status, and when its level is maintained at high or low level, it only means that there is no change of keyboard status, and it does not mean "no key pressed" or "a key pressed".

The simplest way to process the keyboard, if no interrupt is involved, is to keep checking the status of the $\overline{KEY}$ pin in the main loop, and if it is found to have a change, the current keyboard mapping can be obtained while the next BC7278 command is sent (usually a pseudo instruction 0xff, 0xff is sent in the keyboard handler). Because the shortest change period of the keyboard mapping is two scan cycles (about 28ms), and the duration of a human operation (pressing) on the keyboard will be in a few hundred ms of time, so as long as the polling time of the $\overline{KEY}$ pin is shorter than this time, no keyboard event will be missed.

There are many people who are accustomed to using external interrupts to handle keyboard events. When applied to the BC7278 specifically, the following two aspects need to be noted: Firstly, the MCU interrupt pins connected to the $\overline{KEY}$ pin need to be able to accept both rising and falling edge interrupts, because each level change represents a change in keyboard state. If the microcontroller does not have the setting to accept both rising and falling edge trigger, then the interrupt trigger edge setting must be dynamically changed in the interrupt service routine, for example after the falling edge trigger, the interrupt setting should be changed to rising edge trigger before the ISR returns. If the microcontroller hardware only has the ability to trigger interrupts in one direction, or if you do not want to use software, you can use a XOR gate with an RC circuit to convert the level change into a pulse signal. As shown below, the width of the pulse T = 0.8RC.



Secondly, care needs to be taken not to let the ISR interrupt the main program's operation on the BC7278. If the operation of reading the keyboard mapping is put in the keyboard ISR, it must be noted that when the keyboard interrupt occurs, it is possible that the main program is already carrying out a read/write operation of the BC7278, and this can cause the data be corrupted.

To avoid this, we recommend that keyboard operations be performed in one of the following two ways.
1. Temporarily disable the keyboard interrupt before the main program performs the BC7278 operation and re-enabling it after the operation is completed can avoid this error. Since the time taken to write a BC7278 instruction is very short compared to the hold time of the keyboard map data(at 62.5kbps, a full instruction transfer is completed in less than 0.3ms), temporarily blocking the interrupt will not affect the correct reading of the keyboard map.
2.No keyboard map reading operation is done in the keyboard interrupt, only a flag bit is set to notify the main program of a keyboard event, and then the main program handles the keyboard. Because the keyboard event will generally lasts for several hundred ms, the processor should have enough time to read the keyboard.

The situation is more complicated if the SPI interface uses a buffer or DMA to do the data sending. It is

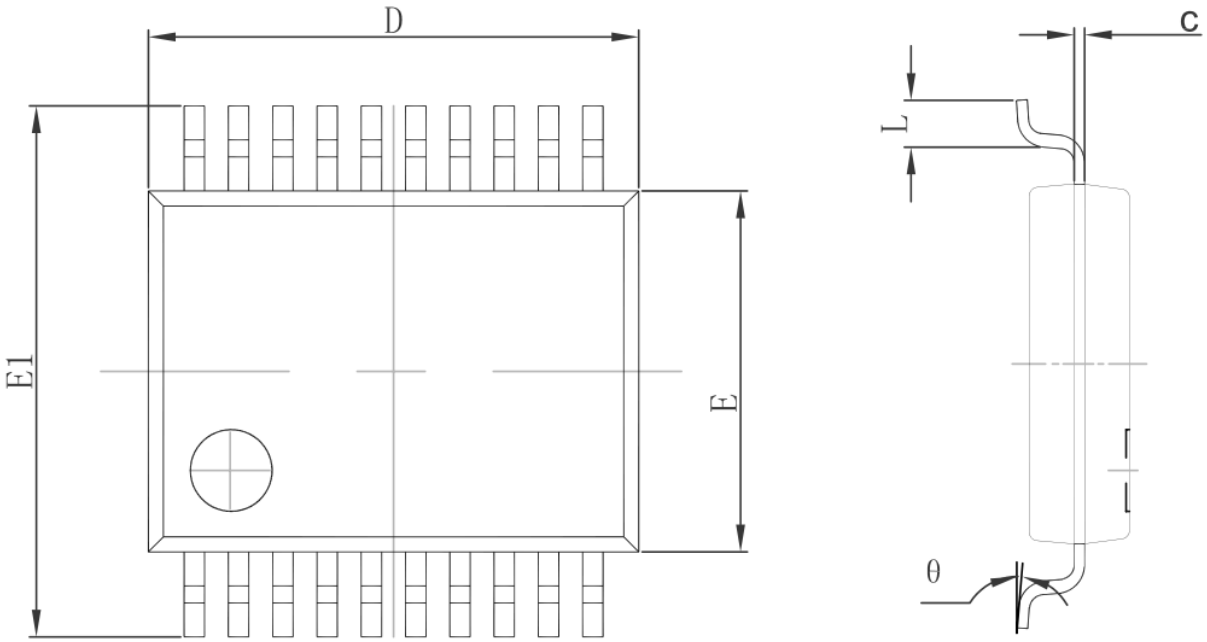
only appropriate to set the keyboard event flag in the keyboard ISR and then let the main program handle the keyboard reading.

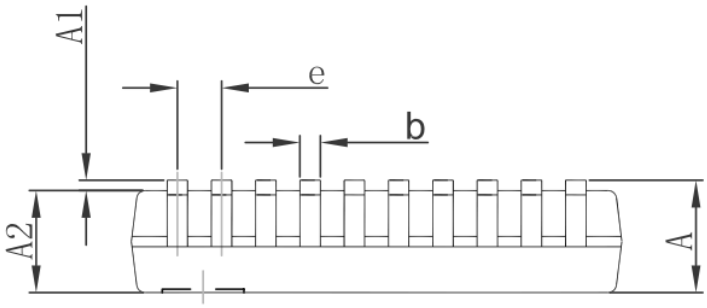
Since by each communication the microcontroller gets a keyboard mapping that reflects the state of all 16 keys, it is easy to check any combination of any number of key combinations.

Checking of long keystrokes is also very simple. If the $\overline{KEY}$ pin does not change for a certain period of time after a keystroke or a combination of keystrokes occurs, that means the keystroke remains in its current state. For example, if the $\overline{KEY}$ does not change for 2s, the keystroke is considered to have been held for 2s.

Packaging Information

BC7278 is in SSOP20 package, the dimensions are:





Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min	Max	Min	Max
A		1.730		0.068
A1	0.050	0.230	0.002	0.009
A2	1.400	1.600	0.055	0.063
b	0.220	0.380	0.009	0.015
c	0.090	0.250	0.004	0.010
D	7.000	7.400	0.276	0.291
E	5.100	5.500	0.201	0.217
E1	7.600	8.000	0.299	0.315
e	0.65(BSC)		0.026(BSC)	
L	0.550	0.950	0.022	0.037
θ	0°	8°	0°	8°

Ordering Information

Ordering number	Packaging	Quantity in each package
BC7278EC-T	tube	6600
BC7278EC-RS	tape & reel	1000
BC7278EC-RL	tape & reel	2000

Appendix

BC727X SERIES

	BC7275B	BC7276	BC7277	BC7278
Package	SSOP20	SSOP20	SSOP24	SSOP20
Digits Driving	6¾	8 or 16	9	4
Keyboard	no	16-key	16-key	16-key
Type	Common Cathode	Common Anode	Common Cathode	Common Cathode
External Driver	no need	transistor & shift reg	no need	no need
Large Display	no	yes	no	no

© Bitcode Technology