Application Note

Interfacing Z8 Encore! XP® MCUs with an I^{2}C-Based Character LCD

AN014902-1207

Abstract

This Application Note describes APIs for interfacing one or more I^{2}C-based character LCDs with Zilog's Z8 Encore! XP® microcontroller that utilizes its I^{2}C peripheral in MASTER mode. The slave I^{2}C controller inside the LCD module converts the commands issued by the Z8 Encore! XP MCU into appropriate drive signals for the LCD.

The main advantage of using an I^{2}C-based LCD is that only two I^{2}C pins are used to carry all the commands and data. Although the Z8 Encore! XP MCU's I^{2}C peripheral operates at 400 kHz for faster data transfer rates, it also supports slower I^{2}C slave devices that exhibit a standard data transfer rate of 100 kHz.

Z8 Encore! XP Flash Microcontrollers

Z8 Encore! XP products are based on the new eZ8 CPU and introduce Flash memory to Zilog's extensive line of 8-bit microcontrollers. Flash memory in-circuit programming capability allows for faster development time and program changes in the field. The high-performance register-to-register based architecture of the eZ8 core maintains backward compatibility with Zilog's popular Z8 MCU.

The new Z8 Encore! XP microcontrollers combine a 20 MHz core with Flash memory, linear-register SRAM, and an extensive array of on-chip peripherals. These peripherals make the Z8 Encore! XP suitable for various applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Discussion

This section briefly describes the I^{2}C peripheral block on Z8 Encore! XP MCU, along with a description of a typical PCF2116-based character LCD module with an I^{2}C interface.[1]

Overview of an I^{2}C Block in Z8 Encore! XP MCU

Inter-integrated circuit (I^{2}C) is a serial communication protocol that uses a two-wire bus to transport data bidirectionally between two or more I^{2}C-enabled devices. The Z8 Encore! XP series features an I^{2}C on-chip hardware peripheral that supports the standard I^{2}C protocol.

For more details on the I^{2}C protocol and its implementation on the Z8 Encore! XP MCU, refer to *Using the Z8 Encore! MCU as an I^{2}C Bus Master (AN0126)*, available for download at www.zilog.com.

Overview of an I^{2}C-Based Character LCD Controller

A liquid crystal display (LCD) module that displays alphanumeric characters can be used in a microcontroller-based product to provide a visual interface for viewing a product's internal parameters. A protocol-converting device is essential for converting the I^{2}C signals transmitted by the Z8 Encore! XP MCU into proper LCD signals.

In this demonstration, the PCF2116 device acts as an I^{2}C slave and converts the serial I^{2}C signals into appropriate LCD signals that directly drive the segments of the LCD.

1.The PCF2116 LCD controller is manufactured by Philips Ltd.

Figure 1 displays the instruction set for PCF2116 device. LCD modules featuring an I^{2}C interface and a built-in PCF2116 (or compatible) controller can be addressed using this instruction set.

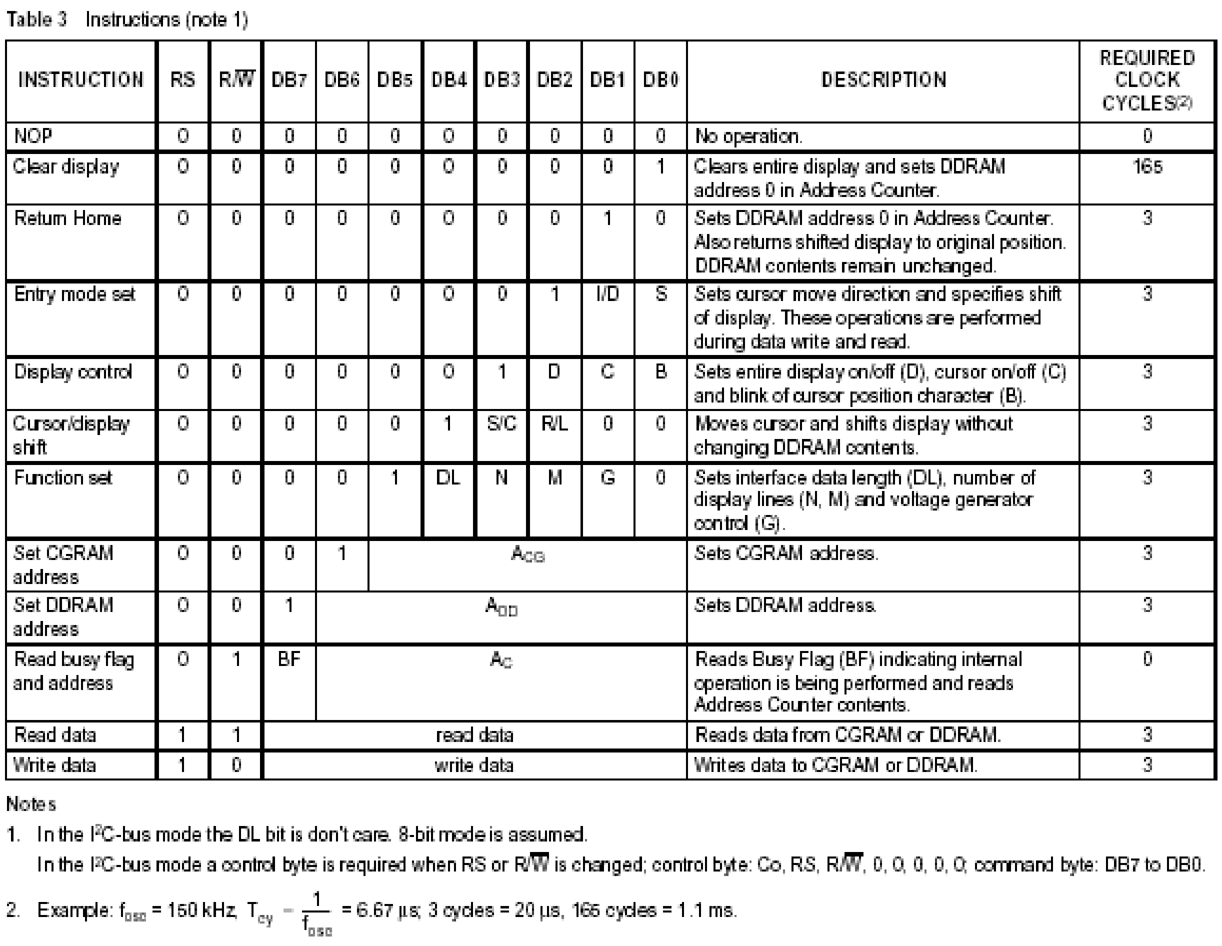

Table 3 Instructions (note 1)

INSTRUCTION	RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION	REQUIRED CLOCK CYCLES(2)
NOP	0	0	0	0	0	0	0	0	0	0	No operation.	0
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in Address Counter.	165
Return Home	0	0	0	0	0	0	0	0	1	0	Sets DDRAM address 0 in Address Counter. Also returns shifted display to original position. DDRAM contents remain unchanged.	3
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display. These operations are performed during data write and read.	3
Display control	0	0	0	0	0	0	1	D	C	B	Sets entire display on/off (D), cursor on/off (C) and blink of cursor position character (B).	3
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	0	0	Moves cursor and shifts display without changing DDRAM contents.	3
Function set	0	0	0	0	1	DL	N	M	G	0	Sets interface data length (DL), number of display lines (N, M) and voltage generator control (G).	3
Set CGRAM address	0	0	0	1	A_{CG}						Sets CGRAM address.	3
Set DDRAM address	0	0	1	A_{DD}							Sets DDRAM address.	3
Read busy flag and address	0	1	BF	A_C							Reads Busy Flag (BF) indicating internal operation is being performed and reads Address Counter contents.	0
Read data	1	1	read data								Reads data from CGRAM or DDRAM.	3
Write data	1	0	write data								Writes data to CGRAM or DDRAM.	3

Notes

1. In the I^{2}C-bus mode the DL bit is don't care. 8-bit mode is assumed.
   In the I^{2}C-bus mode a control byte is required when RS or R/$\overline{W}$ is changed; control byte: Co, RS, R/$\overline{W}$, 0, 0, 0, 0, 0; command byte: DB7 to DB0.
2. Example: f_{osc} = 150 kHz, $T_{cy} = \frac{1}{f_{osc}}$ = 6.67 µs; 3 cycles = 20 µs, 165 cycles = 1.1 ms.

Figure 1. Instruction Set for the PCF2116 LCD Controller (© Philips Ltd.)

[Figure 2](#) displays the font set supported by the PCF2116 LCD controller.

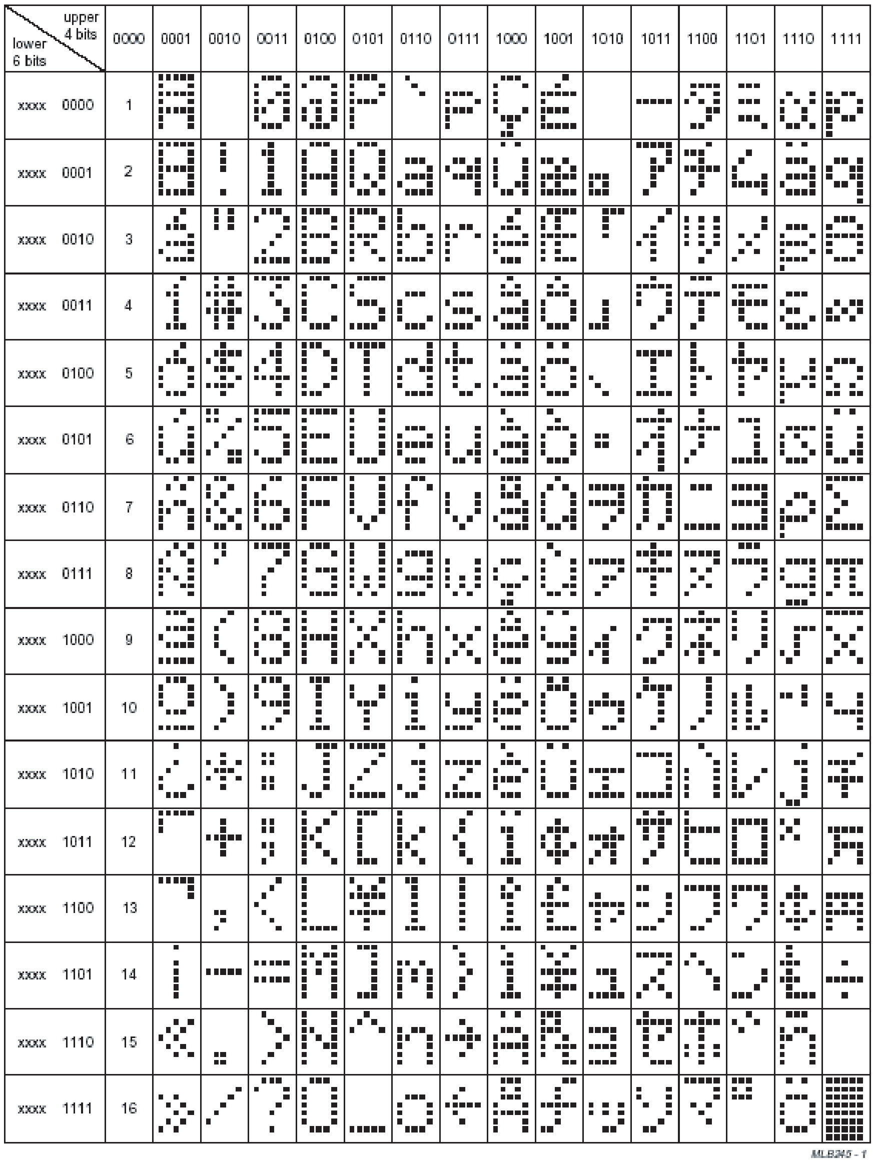
upper 4 bits
lower 6 bits
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
xxxx 0000
xxxx 0001
xxxx 0010
xxxx 0011
xxxx 0100
xxxx 0101
xxxx 0110
xxxx 0111
xxxx 1000
xxxx 1001
xxxx 1010
xxxx 1011
xxxx 1100
xxxx 1101
xxxx 1110
xxxx 1111
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
MLB345 - 1

Figure 2. Font Set and Relevant Addresses in the PCF2116 LCD Controller

Figure 3 displays the internal connections of the demonstration I^{2}C-based LCD module.

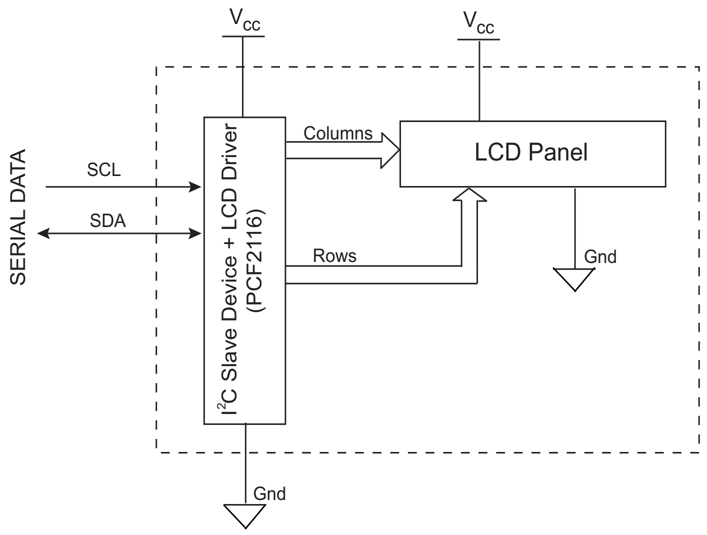

Figure 3. Internal Construction of an I^{2}C-Based LCD Module (PCF2116)

The main points in the I^{2}C-based LCD device include:

- The data bytes are received via the I^{2}C bus and are directly converted into LCD drive voltages
- The I^{2}C slave device operates in 8-bit data mode for LCD-related instructions
- The parallel data bus is not utilized in I^{2}C control mode and therefore remains unconnected
- The SA0 pin on the PCF2116 device can be used as a select pin when two I^{2}C-based LCD modules are connected to the Z8 Encore! XP® MCU (see Figure 4).

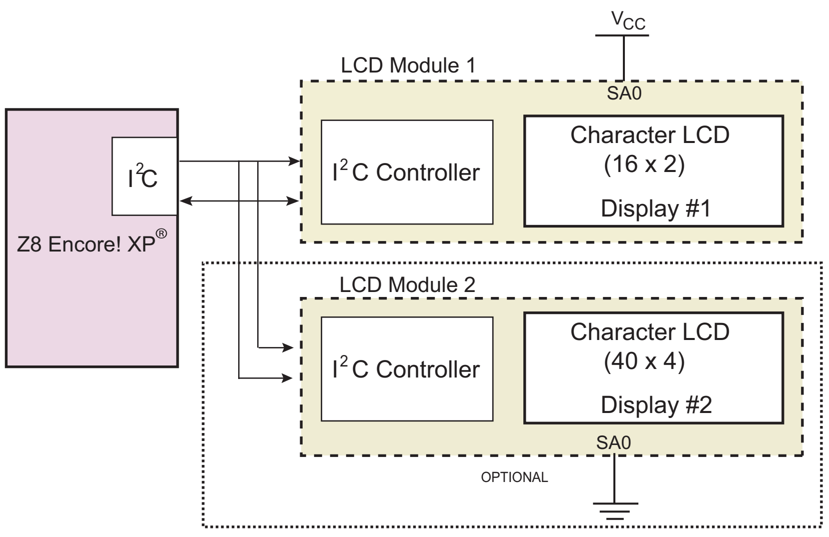

Figure 4. A Typical Arrangement of I^{2}C-Based LCD Modules with Z8 Encore! XP MCU

The advantage of using an I^{2}C-based character LCD module is that only two GPIO pins on the Z8 Encore! XP® microcontroller are utilized for transferring data to the LCD module instead of the usual eleven pins in 8-bit mode, or seven pins in 4-bit mode. Consequently, you can connect a number of I^{2}C-based LCD modules, in parallel, to a single Z8 Encore! XP microcontroller without much additional hardware or software overhead (see Figure 4 on page 4).

For details on parallel interfacing to a character LCD module with the Z8 Encore! XP MCU, refer to *Character LCD Interface for the Z8 Encore!® MCU (AN0143).*

Software Implementation

The APIs developed for this demonstration are generic and built with respect to the PCF2116 device. The instruction set is framed as a data structure according to the Philips Product Specifications sheet for the PCF2116 device.

Initialization

The process of Initializing a PCF2116-based LCD module with a 1-row x 24-character size is achieved using the `I2CLCD_init()` API, which performs the following operations:

- Initializes the Z8 Encore! XP on-chip I^{2}C peripheral
- Initializes the PCF2116 I^{2}C peripheral

To change the initialization mode for a 2 x 24 type of PCF2116-based LCD module, the `FS2` instruction is used instead of `FS1` in the `I2CLCD_init()` API.

To support other types of LCD modules, appropriate changes are made in the `LCD_type` structure. When multiple LCD modules are connected to a single Z8 Encore! XP microcontroller, the `I2CLCD_init()` API is used to initialize each module by entering appropriate slave addresses as parameters.

Note: *The initialization sequence is specific to the PCF2116 device. Therefore, it must be modified when used with other controllers, such as the Hitachi HD66717.*

The instructions, as specified in the data sheet from the manufacturer, must be added in the data structure `LCD_type`. In addition to the instruction Op Codes, this data structure also contains information such as the I^{2}C slave address of the device and the number of rows and columns in the LCD display. The structure is initialized according to the values specified in the `main.c` file.

Writing to I^{2}C-Based Character LCD Device

While writing to the I^{2}C-based character LCD slave device, the slave address is passed as the first parameter to activate the module. Therefore, the instructions following the slave address are ignored by the other I^{2}C devices connected to the I^{2}C bus. The set of instructions that follow the address byte consist of the control byte for a Write operation and the ASCII symbol to be printed to the LCD display panel. The selected slave device then responds by sending an I^{2}C acknowledge signal back to the Z8 Encore! XP I^{2}C Master and displaying the ASCII symbol.

When a continuous string of characters is to be displayed, the `I2CLCD_prints()` API is used. When an ASCII symbol is received and displayed by the selected LCD module, the cursor position is advanced by one position automatically and the next character is displayed in the next location. To stop string transmission, an I^{2}C stop signal is sent by the Z8 Encore! XP Master device using the I^{2}C`_stop()` API.

See Figure 6 on page 8 for the flowchart illustrating the sequence of operations while writing a byte to the I^{2}C slave device.

Reading a Byte from the I^{2}C-based Character LCD Device

While reading a byte from the I^{2}C-based character LCD Slave device, the slave address is transmitted first, to be followed by the control byte for a Read operation.

The Slave device enters READ DATA mode and sends the requested data via the I^{2}C bus. After receiving the required data bytes, the I^{2}C_`stop()` API is executed to end the communication.

See Figure 7 on page 11 for the flowchart illustrating the sequence of operations while reading a byte of data from the I^{2}C slave device.

Summary

Character LCD modules that feature an I^{2}C interface are convenient to use in a product design because of simple hardware and reduced PCB complexity. This simplicity is achieved by using only two GPIO pins instead of the seven or eleven pins normally used to transmit and receive data in LCDs. This Application Note presents the concept of a device driver software application used with a character LCD device. The APIs are built around the PCF2116 device, which works both as the I^{2}C slave device and as an LCD segment driver. These APIs make use of a data structure to access the instruction set of the controller, which can be easily modified for use with any other I^{2}C-based LCD controller.

References

The documents associated with Z8 Encore! XP® MCUs, eZ8 CPU, and Philips PCF2116 LCD device are provided below:

- Z8 Encore! XP® 64K Series Flash Microcontrollers Product Specification (PS0199)
- eZ8 CPU User Manual (UM0128)
- PCF2116, Dot Matrix Liquid Crystal Controller/Driver Data Sheet (www.philips.com)

Appendix A—Flowcharts

This appendix contains flowcharts which illustrate the implementation of an I^{2}C-based character LCD interface with the Z8 Encore! XP® MCU.

Figure 5 displays the steps involved in initializing an I^{2}C-based character LCD using the instruction set of the Philips PCF2116 controller.

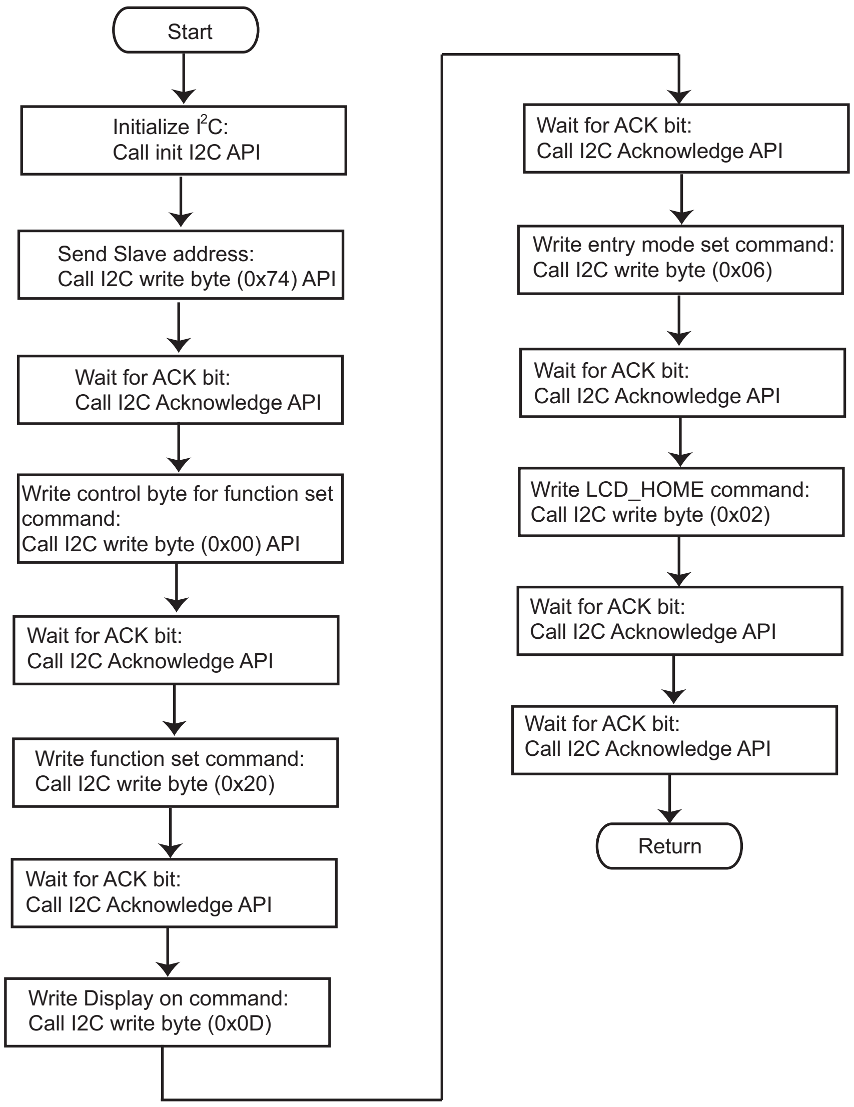
Start
Initialize I2C: Call init I2C API
Send Slave address: Call I2C write byte (0x74) API
Wait for ACK bit: Call I2C Acknowledge API
Write control byte for function set command: Call I2C write byte (0x00) API
Wait for ACK bit: Call I2C Acknowledge API
Write function set command: Call I2C write byte (0x20)
Wait for ACK bit: Call I2C Acknowledge API
Write Display on command: Call I2C write byte (0x0D)
Wait for ACK bit: Call I2C Acknowledge API
Write entry mode set command: Call I2C write byte (0x06)
Wait for ACK bit: Call I2C Acknowledge API
Write LCD_HOME command: Call I2C write byte (0x02)
Wait for ACK bit: Call I2C Acknowledge API
Wait for ACK bit: Call I2C Acknowledge API
Return

Figure 5. Reading a Data Byte from an I^{2}C-Based Character LCD

Figure 6 displays the steps involved in writing data to an I²C-based character LCD.

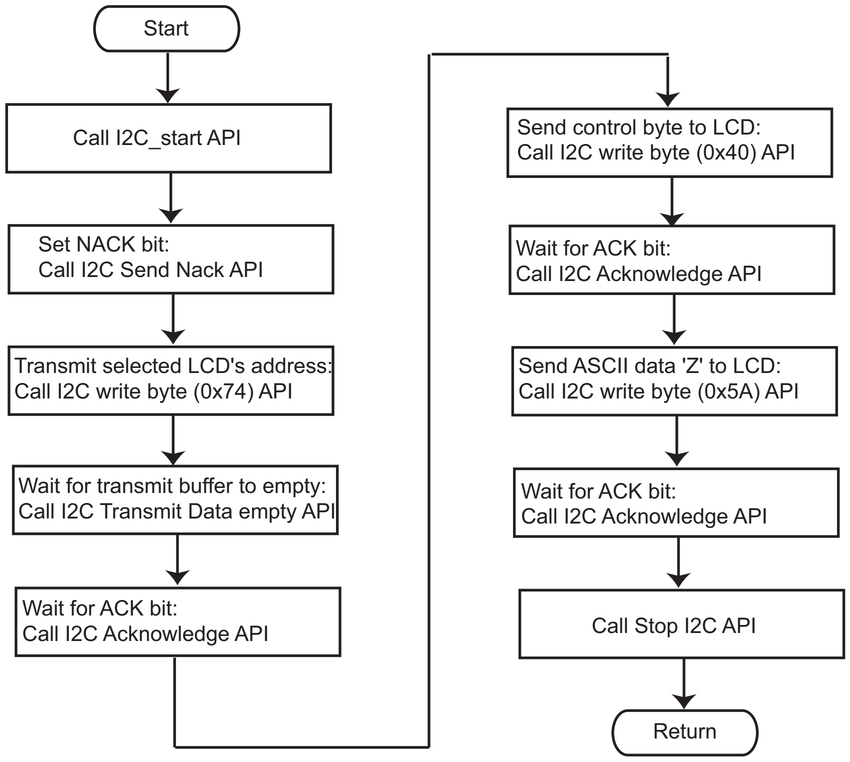
Start
Call I2C_start API
Set NACK bit:
Call I2C Send Nack API
Transmit selected LCD's address:
Call I2C write byte (0x74) API
Wait for transmit buffer to empty:
Call I2C Transmit Data empty API
Wait for ACK bit:
Call I2C Acknowledge API
Send control byte to LCD:
Call I2C write byte (0x40) API
Wait for ACK bit:
Call I2C Acknowledge API
Send ASCII data 'Z' to LCD:
Call I2C write byte (0x5A) API
Wait for ACK bit:
Call I2C Acknowledge API
Call Stop I2C API
Return

Figure 6. Writing Data to a I²C-Based Character LCD

Figure 7 displays the steps involved in reading a data byte from an I^{2}C-based character LCD.

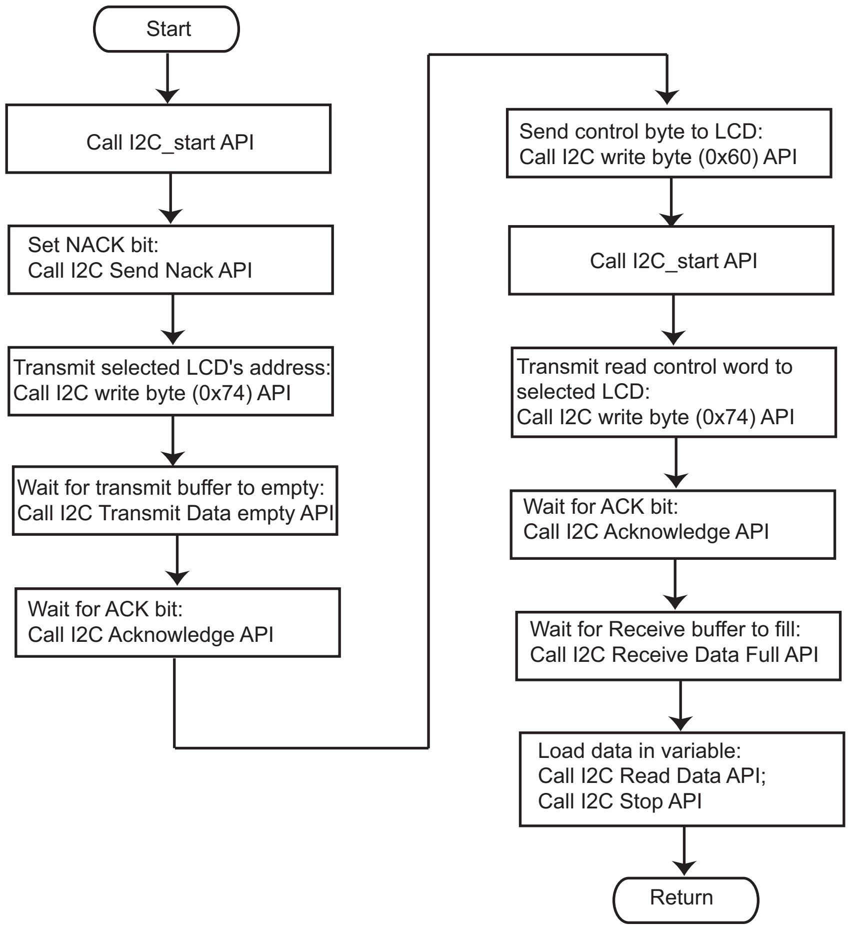
Start
Call I2C_start API
Set NACK bit:
Call I2C Send Nack API
Transmit selected LCD's address:
Call I2C write byte (0x74) API
Wait for transmit buffer to empty:
Call I2C Transmit Data empty API
Wait for ACK bit:
Call I2C Acknowledge API
Send control byte to LCD:
Call I2C write byte (0x60) API
Call I2C_start API
Transmit read control word to selected LCD:
Call I2C write byte (0x74) API
Wait for ACK bit:
Call I2C Acknowledge API
Wait for Receive buffer to fill:
Call I2C Receive Data Full API
Load data in variable:
Call I2C Read Data API;
Call I2C Stop API
Return

Figure 7. Reading a Data Byte from an I^{2}C-Based Character LCD

Appendix B—API Descriptions

This appendix contains description of I^{2}C-based character LCD driver APIs.

- I2CLCD_init()
- I2CLCD_on()
- I2CLCD_off()
- I2CLCD_printc()
- I2CLCD_prints()
- I2CLCD_clear()
- I2CLCD_home()
- I2CLCD_blink()
- I2CLCD_blink_off()
- I2CLCD_setposition()

I2CLCD_init()

Prototype

```
void I2CLCD_init(struct*)
```

Description

The `I2CLCD_init()` API initializes a selected LCD as per specifications.

Argument(s)

`struct*`	Pointer to the I^{2}C LCD structure.

Return Values(s)

void

Example

```
I2CLCD_init(&LCD1);// LCD1 is pointing to data
// structure struct Lcd_type *LCD1;
```

I2CLCD_on()

Prototype

```
void I2CLCD_on(struct*)
```

Description

The `I2CLCD_on()` API enables printing on a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I²C LCD structure.

Return Values(s)

void

Example

```
I2CLCD_on(&LCD1);
```

I2CLCD_off()

Prototype

```
void I2CLCD_off(struct*)
```

Description

The `I2CLCD_off()` API disables a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I^{2}C LCD structure.

Return Values(s)

void

Example

```
I2CLCD_off(&LCD1);
```

I2CLCD_printc()

Prototype

```
void I2CLCD_printc(struct*, unsigned char*)
```

Description

The `I2CLCD_printc()` API prints an ASCII symbol at the current cursor position on a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I^{2}C LCD structure.
`unsigned char*`	ASCII symbol.

Return Values(s)

void

Example

```
I2CLCD_printc(&LCD1,'A');
```

I2CLCD_prints()

Prototype

```
void I2CLCD_prints(struct*, unsigned char*)
```

Description

The `I2CLCD_prints()` API prints a string of ASCII symbols at current cursor position on a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I²C LCD structure.
`unsigned char*`	Pointer to string.

Return Values(s)

void

Example

```
I2CLCD_prints(&LCD1,"A String");// Print fixed string

I2CLCD_prints(&LCD1,*Astring);// Print from memory
```

I2CLCD_clear()

Prototype

```
void I2CLCD_clear(struct*)
```

Description

The `I2CLCD_clear()` API clears the display screen of a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I^2C LCD structure.

Return Values(s)

void

Example

```
I2CLCD_clear(&LCD1);
```

I2CLCD_home()

Prototype

```
I2CLCD_home(&LCD1);
```

Description

The `I2CLCD_home()` API brings the cursor to the first position on first line, without changing contents in the DDRAM of a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I^{2}C LCD structure.

Return Values(s)

void

Example

```
void I2CLCD_home(struct*)
```

I2CLCD_blink()

Prototype

```
I2CLCD_blink(&LCD1);
```

Description

The `I2CLCD_blink()` API blinks the symbol at the current cursor position on a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I^{2}C LCD structure.

Return Values(s)

void

Example

```
void I2CLCD_blink(struct*)
```

I2CLCD_blink_off()

Prototype

```
void I2CLCD_blink_off(struct*)
```

Description

The `I2CLCD_blink_off()` API stops the cursor blink on a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I²C LCD structure.

Return Values(s)

void

Example

```
I2CLCD_blink_off(&LCD1);
```

I2CLCD_setposition()

Prototype

```
void I2CLCD_setposition(struct*, unsigned char*)
```

Description

The `I2CLCD_setposition()` API relocates the cursor to a specified position on a selected LCD panel.

Argument(s)

`struct*`	Pointer to the I^{2}C LCD structure.
`unsigned char*`	Column number.

Return Values(s)

void

Example

```
I2CLCD_setposition(&LCD1,6);
```