Using CrystalFontz CFA533-KC as an HMI

Associated Files:

QuickSilver Controls has implemented an I2C interface which uses IO2 and IO3 on our X-Series SilverNuggets and SilverMax units. This allows for data display and interaction with the operator via a keyboard. The display and keyboard are easily accessed using subroutines in the QCP example

This example is using a Crystalfontz series of I2C displays with entry pads.
https://www.crystalfontz.com/product/cfa533tfhkc-16x2-i2c-lcd-display
They also have several I2C displays only, as well as larger displays. This particular unit is approximately $50 at quantity one, from their web site (as of the writing of this application note) They also carry bezels and have cad drawing for the display and keyboard bezels.

Electronic Interface

The I2C interface connections are made through a 10 pin header on the back side of the display:

Signal Name	CF 10 Pin connector	X-series Signal
V+ (3.3-5v)	Pin 7	+5v
GND	Pin 9	Logic Ground
SDA (Data)	Pin 3	IO2
SCL (Clock)	Pin 5	IO3

Note: The internal programmable pullups in the X-Series controllers are automatically turned on when the I2C mode is configured, providing the required pullup resistors without additional hardware.

The display also has 4 additional I/O available through J8 connector. Some of these have options set by jumpers or resistors. See the data sheet for details. Contact Crystalfontz to have these configured, including having connectors installed.

I2C COMMANDS

Three new modes have been added to the SET MODE (SMD) command. The Send and Read back commands are specific to the CrystalFontz I2C communications protocol as the Send command appends two CRC bytes to the data stream, while the Read function verifies the received CRC values received from the display.

SMD [28] Enable I2C on IO2(SDAA) and IO3(SCLA), set the I2C address of display. The address is 7 bits (the 8th bit is a RD/WR* bit)
SMD [29] Send to I2C from buffer address. The first register holds the Command byte in the high word, and the byte count of the data portion in the low word. Any data must be placed in the following registers, again high word, then low word of successive registers. If only one byte is used in a register, it must be the high word. The CRC is automatically calculated and sent after the data. Note, the transaction is not closed by the send command, allowing a following read back command to be used, either to verify the sent command or to read information back from the display (such as keyboard information). The maximum size of the transaction is 32 bytes (not including the CRC).
SMD [30) Read Back from I2C to buffer address. The data read is returned to the buffer, with the number of bytes determined by the byte count from the display. CRC is automatically checked. A maximum of 32 bytes (not including the CRC) may be returned in a transaction (16 registers), although most commands do not return that much information.

The Enable I2C command should begin each transaction. The enable command is configured to send a long stop (9 clocks) to clear any existing state. A minimum of 1ms should follow the Enable command to allow it to finish before any new command is issued. (Register 64039 can also be checked – done with controller processing if register 64039=0). A minimum 6 ms delay should follow each Send or Read command to the I2C to allow the command to be sent and for the display to process the command before the next read or write command is sent. See the CrystalFontz manual for details, as some commands require longer times to complete.

Next the wanted command, byte count, and any data need to be written to the registers which will be used as the transmit buffer. The Send to I2C command is used to send these to the display. Again, the appropriate wait time for the command (most are 5ms) needs to be waited for the display to process the command before any new command is sent or the results of the just sent command are read back.

If the command is sending long blocks of data, additional time may be needed. Each byte (sent at 50kHz clock rate) takes approximately 180uS, and the packed must include command, count, data, and 2 bytes of CRC.

The status of the command sent (from the controller’s view) may be verified by looking at registers 64038 and 64039 (details below). Additionally, optionally, the command results from the display may be queried by use of the Read Back command. In the case of a query command (read keypad, read memory, etc.), the Read Back command must be used to retrieve the queried data.

The Enable command should be sent prior to the next Send command. A 1ms delay should be sufficient for the enable command to complete.

Register 64039 holds the I2C internal state (1=enable, 2=write, 3=readback) in the upper word, and the substate in the lower word. Value of 0 (both words) means the I2C state machine has completed its current operation.
Register 64038 holds status bits:
Bit 0 = success
Errors: Bit 15 set plus:
Bit 1 = bad receive (data size too big for buffer)
Bit 2 = bad CRC received
Bit 3 = illegal state detected in state machine (should never occur)

Checking these status bits is optional as long as sufficient time is provided for the transaction. (Checking status bits and optionally the Read Back for each command may help detect and diagnose errors in communications to the display).

POWER UP STATE OF DISPLAY

The display has its own non-volatile memory which includes IO configuration (4 IO lines), display data, user defined special character data, cursor type and location, and several other options. (See CrystalFontz Manual). The current configuration may be saved with command 4. (Note this can take up to 50ms, and the display should be operating at 5v to ensure success.) The standard display, as shipped has CrystalFontz banner and revision showing. Use command 6 to clear display and then write any startup message you may want before saving the power up state.

LIGHTS

The LEDs on for the display and keyboard are independent and can be set to 0..100%. It is suggested to only illuminate the keyboard when an input is being requested, but that is entirely up to your choice. (See command 14 - 0x0E)

KEYPAD

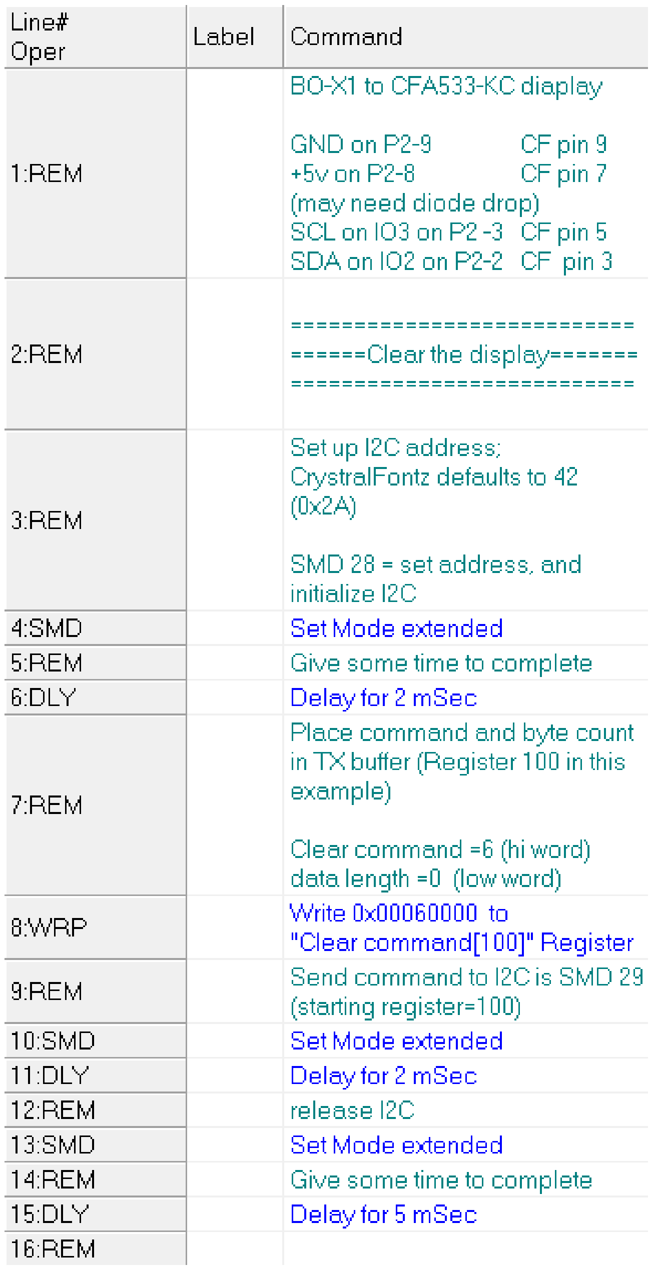
The keypad is scanned in the background by the display. There are a couple of optional modes that can generate IO either on a key push, or on a particular combination or holding of buttons (i.e. pulse one of the onboard IO if the exit (x) button is kept pushed for 4 seconds); see the display manual for details. The standard use is to poll the keyboard results using the Read Keypad 24 – 0x18 command. This returns 3 bytes, the currently pressed, pressed since last read keypad command, and the buttons released since last read keypad command. These are mapped below. Each button reports its status as a separate bit in the three bytes returned.

Bit0 (0x01)	= Up
Bit1 (0x02)	= enter (check mark)
Bit2 (0x04)	= Cancel (X button)
Bit3 (0x08)	= Left
Bit4 (0x10)	= Right
Bit5 (0x20)	= Down

EXAMPLES

Clear I2C Display.qcp
This routine will clear the display to all blanks. (Note, if you have a cursor set, that must be cleared separately by setting cursor type to 0).

First we set the address and initialize the I2C interface. Next we set the command and data length for the Clear Display command: Command 6, data length 0. Finally we send the command using the SMD 29 command with the same register to which we wrote the command and data length. The appropriate delays are inserted between commands.

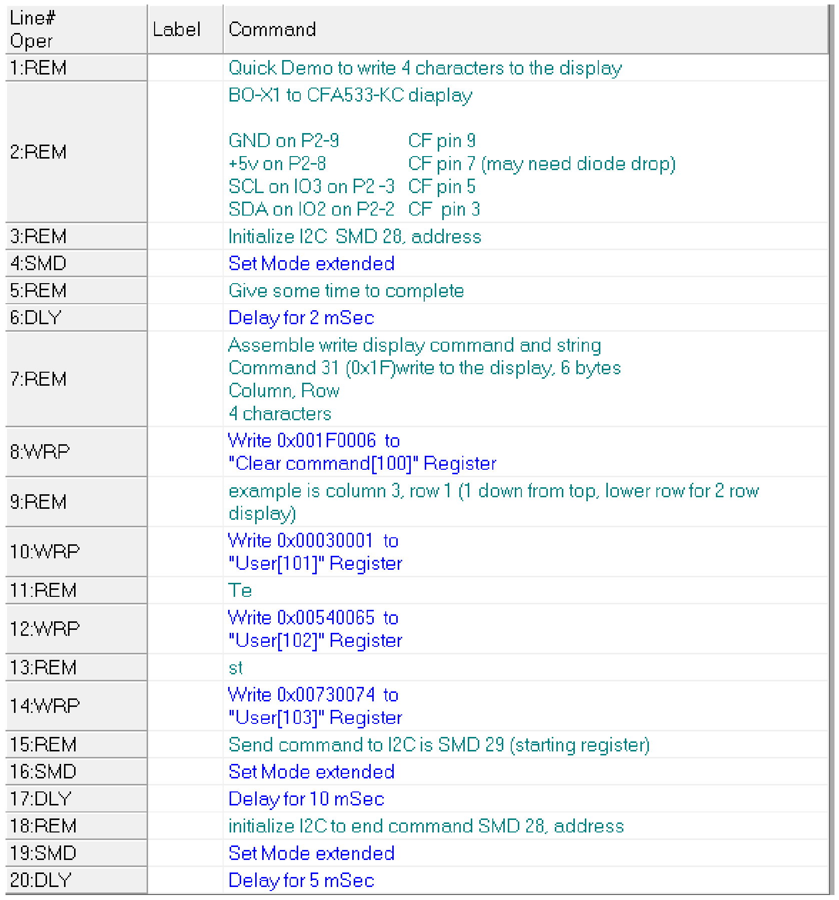


Line# Oper	Label	Command
1:REM		BO-X1 to CFA533-KC diaplay GND on P2-9 CF pin 9 +5v on P2-8 CF pin 7 (may need diode drop) SCL on IO3 on P2 -3 CF pin 5 SDA on IO2 on P2-2 CF pin 3
2:REM		=========================== ======Clear the display======= ===========================
3:REM		Set up I2C address; CrystalFontz defaults to 42 (0x2A) SMD 28 = set address, and initialize I2C
4:SMD		Set Mode extended
5:REM		Give some time to complete
6:DLY		Delay for 2 mSec
7:REM		Place command and byte count in TX buffer (Register 100 in this example) Clear command =6 (hi word) data length =0 (low word)
8:WRP		Write 0x00060000 to "Clear command[100]" Register
9:REM		Send command to I2C is SMD 29 (starting register=100)
10:SMD		Set Mode extended
11:DLY		Delay for 2 mSec
12:REM		release I2C
13:SMD		Set Mode extended
14:REM		Give some time to complete
15:DLY		Delay for 5 mSec
16:REM		

Test Write 4.qcp

This program writes 4 characters “Test” to column 4, row 1 as an example.

First we clear the I2C state machine.
We then assemble command 31 (0x1F), with 6 bytes of data:
Column, Row
“Test”
We write the command to the display, and then complete the bus cycle by initializing the I2C bus



Line# Oper	Label	Command
1:REM		Quick Demo to write 4 characters to the display
2:REM		BO-X1 to CFA533-KC diaplay GND on P2-9 CF pin 9 +5v on P2-8 CF pin 7 (may need diode drop) SCL on IO3 on P2 -3 CF pin 5 SDA on IO2 on P2-2 CF pin 3
3:REM		Initialize I2C SMD 28, address
4:SMD		Set Mode extended
5:REM		Give some time to complete
6:DLY		Delay for 2 mSec
7:REM		Assemble write display command and string Command 31 (0x1F)write to the display, 6 bytes Column, Row 4 characters
8:WRP		Write 0x001F0006 to "Clear command[100]" Register
9:REM		example is column 3, row 1 (1 down from top, lower row for 2 row display)
10:WRP		Write 0x00030001 to "User[101]" Register
11:REM		Te
12:WRP		Write 0x00540065 to "User[102]" Register
13:REM		st
14:WRP		Write 0x00730074 to "User[103]" Register
15:REM		Send command to I2C is SMD 29 (starting register)
16:SMD		Set Mode extended
17:DLY		Delay for 10 mSec
18:REM		initialize I2C to end command SMD 28, address
19:SMD		Set Mode extended
20:DLY		Delay for 5 mSec

ADDITIONAL EXAMPLES

Additional qcp example files are included in the zip file.

Set Startup Screen.qcp
This program (configured to be run-only as that is how it would likely be used) will clear the screen, set only the screen LED as lit, turn off the cursor, and set a greeting line on the top line of the display, in this example “QuickSilver”. It will then save this as the default startup screen for the display when power is applied to the display.

Set Greeting edit number.qcp
This program is written with various operations as subroutines to move the bulk of the operations out of the main routine. It clears the display and writes “QuickSilver” to the top line, and then displays a number on the lower line. The number limits are set, and then the keyboard is illuminated and the keyboard can be used to edit the number (within the limits). The number can be accepted (check key) or cancelled (x key) which reverts to the original number. The cursor is then cleared, and the keypad is turned off. This example program then delays 1 second before starting over as an example, but a real program could then display the next information to be edited before starting a routine.
The subroutines are all documented within the program