Modtronix Engineering

Modular Electronic Solutions

LCD2S Revision 1, Firmware V1.40

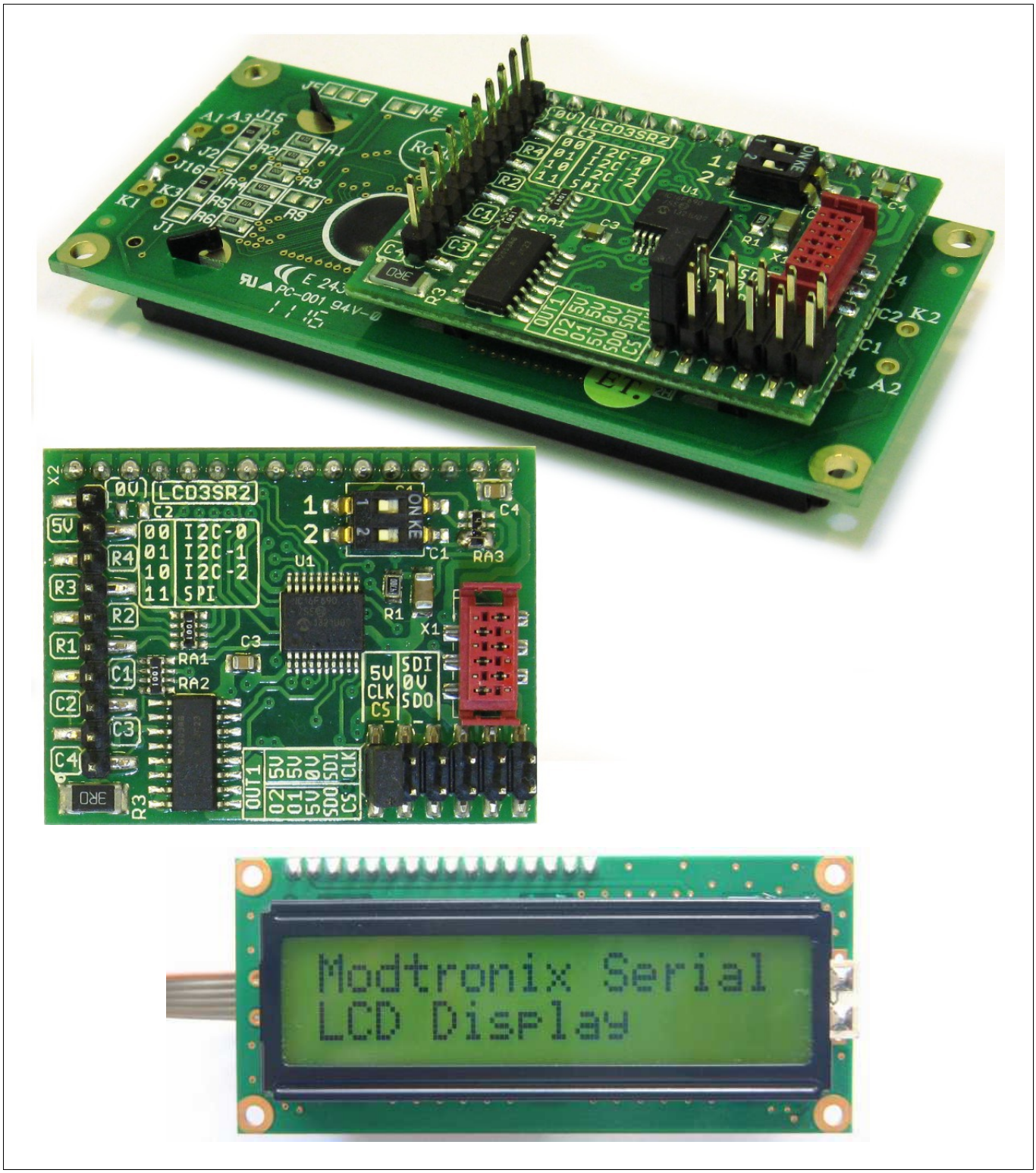
Serial LCD Display

Table of Contents

1 Introduction

The following documentation is for the LCD2S Hardware Revision 1, and Firmware V1.40. It is marked on the PCB as LCD2S REV1!

Note that a LCD display **IS NOT** included with the standard LCD2S daughter board! The images below showing the LCD2S combined with a LCD display are for reference only. The picture in the middle shows the actual LCD2S board.



2 Features

- Can be controlled via SPI or I2C bus
- Assembled with 2x16 or 4x20 line character displays, with or without back lighting
- Two high current, open collector outputs (OUT1 and OUT2), with output current of 1000mA each
- 2 General purpose input/output pins (GPIO1 and GPIO2) that can be configured as:
  - Analog input with 10 bit resolution
  - Digital Input, or Digital Output (1kΩ output impedance)
- 1 General purpose Digital Input pin (GPIO3).
- Keypad encoder for a keypad up to 16 keys (4 rows by 4 columns)
- Keypad has a configurable button repeat delay and repeat rate
- Can be configured to sound a buzzer (connected to OUT2) each time a button is pressed
- LCD Backlighting controlled via software, has 254 brightness levels
- LCD Contrast controlled via software, has 254 contrast levels
- User configurable start up screen
- 80 Byte buffer for messages received via serial interface
- Large number of LCD commands for moving cursor, moving display and more
- Up to 8 custom characters can be defined
- Built in commands for drawing Bar Graphs
- When using a 4x20 LCD, large characters spanning 3x2 characters can be drawn
- 6 Pin Micro Match connector with I2C, SPI and power signals
- 10 Pin (10 pins x 1 row), 2.54mm connector for connecting a keypad of up to 4 rows by 4 columns
- 10 Pin (5 pins x 2 rows), 2.54mm connector with user outputs, I2C, SPI and power signals
- DIP switch for setting serial mode and I2C address
- Low current consumption - without LCD backlight is about 4mA
- Assembled with brand name, quality components. For example, electrolytic capacitor used is extra long life rated, which is 5 times more than standard!
- PIC processor can be programmed with custom firmware via 5 in circuit serial programming pins.

3 Connectors

The LCD2S has 3 connectors with serial interface, power and I/O signals.

3.1 Connector X1 - Micro Match 6 Pin Connector

The LCD2S has a 6 pin Micro Match type connector for controlling it via the SPI or I2C interface.

X1 – 6 pin micro match type connector		
Pin	*I2C Signal*	*SPI Signal*
1	SDA – Serial Data	SDI – Serial data input
2	+5V	+5V
3	GND	GND
4	SCL – Serial Clock	SCK – Serial Clock
5		SDO – Serial Data Output
6	I2C Interrupt	CS – Chip select, active low

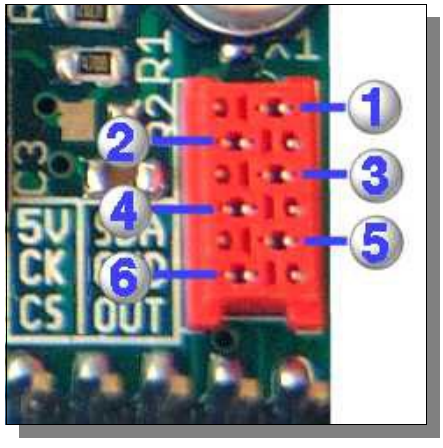


Figure 1

3.2 Connector X2 – Keypad and GPIO1 to GPIO3

The LCD2S has a 10 pin, 2.54mm connector that can be configured for the following functions:

- Keypad encoder up to 16 keys (4 rows by 4 columns)
- 2 Analog inputs with 10 bit resolution
- 3 General Purpose Digital I/O lines that can be used for inputs or outputs (1k output impedance).

X2 – 10 pin, 2.54mm header connector	
Pin	*Signal*
1	Gnd
2	Vcc – 5V supply
3	Row 4
4	Row 3
5	Row 2
6	Row 1
7	Column 1 or GPIO3 (General Purpose Digital Input)
8	Column 2 or GPIO2 (General Purpose I/O) or AN2 (Analog Input)
9	Column 3 or GPIO1 (General Purpose I/O) or AN1 (Analog Input)
10	Column 4

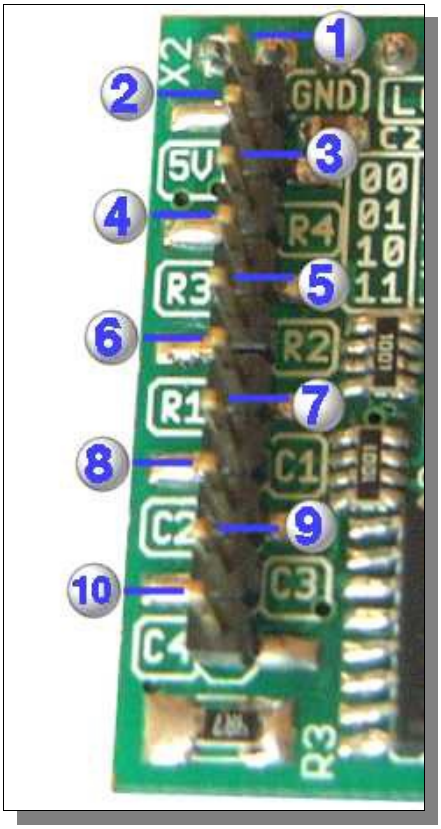


Figure 2

3.3 Connector X3 – SPI bus, I²C bus, Power, OUT1 and OUT2

The LCD2S has a 2x6 pin, 2.54mm connector with SPI, I²C and user output signals, and the OUT1 jumper. This can be used as an alternative to the 6 pin Micro Match connector for accessing the SPI and I²C signals.
Additionally this connector has two user controlled 1000mA outputs.

X3 – 10 pin, 2.54mm header connector		
Pin	***Signal***	
1	**OUT2** – 1000mA open collector output. When off, output is high impedance. When on, output is pulled low to GND.	
2	**+5V**	
3	**OUT1** – 1000mA open collector output. When off, output is high impedance. When on, output is pulled low to GND.	
4	**+5V**	
5	**+5V**	
6	**GND**	
	I²C Signal	***SPI Signal***
7		SDO – Serial Data Output
8	SDA – Serial Data	SDI – Serial data input
9	I²C Interrupt	CS – Chip select, active low
10	SCL – Serial Clock	CLK – Serial Clock

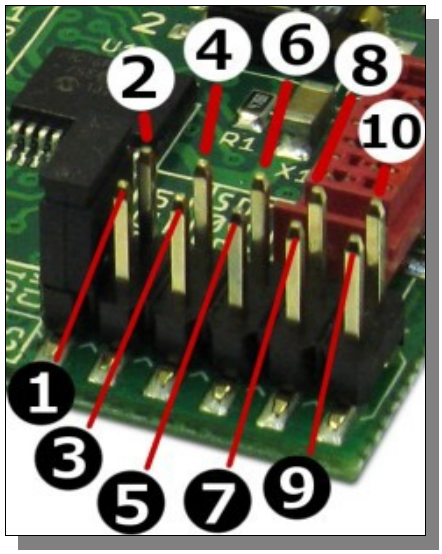


Figure 3

The pinouts on this connector have carefully been selected so that:
- Pin 1 and 2 are next to each other, and can be used for OUT1output
- Pin 3 and 4 are next to each other, and can be used for OUT2 output
- Pin 6, 8 and 10 are next to each other, and have the two I2C signals and ground on them

4 OUT1 and OUT2 outputs

The LCD2S has 2 general purpose open collector outputs. These outputs can supply up to 1000mA, and have protection circuitry for driving relays. Both OUT1 and OUT2 are enabled by **default**. They are available via the X3 connector.

Output OUT2 does not share port pins with any other function, and is always enabled.

Output OUT1 shares a port pin with the keypad encoder, and can only be used when the keypad encoder is configured for a 4x3 keypad (4 rows by 3 columns = 12 keys) or smaller. It is enabled by inserting a jumper on the end of the 2x6 pin header (marked OUT1 on PCB). It is enabled by default, and this jumper is present. When the keypad encoder is configured for a 4x4 keypad, this jumper **must be** removed! Output OUT1 can **not** be used when using a 16 key, 4x4 keypad!

5 GPIO1 to GPIO3 General Purpose I/Os

The LCD2S has 3 general purpose pins, GPIO1 to GPIO3. GPIO1 and GPIO2 can be used as Digital Inputs, Digital Outputs or Analog Inputs, and GPIO3 as a Digital Input. They are available on the X1 connector. They all share microcontroller port pins with the keypad encoder. The full keypad encoder (16 keys) and all GPIO ports can thus not all be used at the same time. By **default**, GPIO1 to GPIO3 are disabled, and the keypad encoder is enabled! For details on configuration, see Chaper 7. “Keypad and I/O Configuration“.

All these pins have a 1kΩ output impedance (1kΩ resistor between CPU port pin and GPIO pin). This means they can not supply lots of current. For example, when set to output high (5V), and supplying a current of 1mA, the output will only be = 5V – (0.001Ax1000Ω) = 5 – 1 = 4V (1V voltage drop over output resistor).

6 Keypad

A keypad of up to 16 keys (4 rows by 4 columns) can be connected to the LCD2S. All keys are filtered via a configurable software debounce filter. Each time a key is pressed, a key code from 'a' to 'f' is added to the 16 byte keypad FIFO buffer. See Figure 4 for a wiring diagram and key codes. The keypad buffer can be read via the "*Read Keypad Data*" command.

In I2C mode there is an I2C interrupt line that will be pulled low by the LCD2S if it has pending data. Currently pending data will always be keypad data. If the I2C interrupt line is not used, the LCD2S has to be polled each couple of ms to see if it has any keypad data available. This is done by reading the status byte with the "*Read Device Status Byte*" command. The status byte will indicate if there is any pending keypad data.

7 Keypad and I/O Configuration

The keypad can be configured via software and jumpers for a couple of different modes. The reason for this is that some port pins are shared for use by the keypad, the OUT1 output, and the GPIO1 to GPIO3 general purpose I/Os of the LCD2S. The maximum number of keys that can be decoded are 16, when the keypad is configured as 4 rows by 4 columns. The following summarizes when the OUT1 output and GPIO1 to GPIO3 general purpose I/Os are available. A √ indicates the function is available.

Keypad Configuration	OUT1	GPIO1 / AN1	GPIO2 / AN2	GPIO3
4x4				
4x3 **(Default Configuration)**	√			
4x2	√	√		
4x1	√	√	√	
none	√	√	√	√

By **default** (at delivery), the keypad is configured for a 12 key (4 rows by 3 column) keypad, OUT1 is enabled, GPIO1 to GPIO3 are disabled, and AN1 to AN2 are disabled. The OUT1 jumper (on 2x6 pin header) is also inserted, which is required when output OUT1 is enabled.

The keypad can be configured for the following modes:

7.1 Keypad with up to 16 Keys

In this mode, a keypad of up to 4 rows by 4 columns can to be used. To configure the LCD2S for a 16 key keypad, the following must be done:

- The OUT1 jumper (on 2x6 pin header) header must be **removed**!
- The keypad must be configured for a 4x4 keypad via the "*Configure Keypad and IO*" command. This command will also disable the OUT1 output and the GPIO1, GPIO2 and GPIO3 general purpose inputs/outputs.
- The OUT1 output and GPIO1, GPIO2 and GPIO3 general purpose inputs/outputs are not available when using a 16 key keypad!

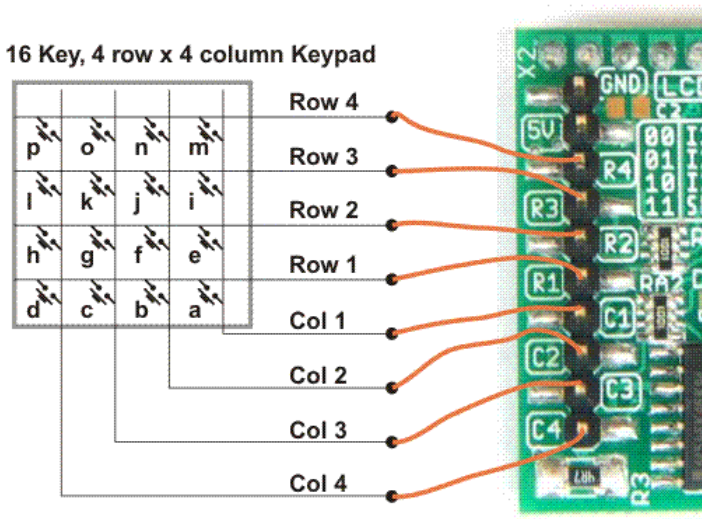


Figure 4: Wiring diagram and key codes for 4x4 keypad

7.2 Keypad with up to 12 Keys (Default)

This is the **default** mode. In this mode, a keypad of up to 4 rows by 3 columns can to be used. To configure the LCD2S for a 12 key keypad, the following must be done:

- The keypad must be configured for a 4x3 keypad via the "*Configure Keypad and IO*" command. This command will also disable the GPIO1, GPIO2 and GPIO3 general purpose inputs/outputs.
- The GPIO1, GPIO2 and GPIO3 general purpose inputs/outputs are not available when using a 12 key keypad! The OUT1 output is available (jumper OUT1 on 2x6 pin header must be inserted).
- The keypad must be connected to R1 to R4 (rows 1-4) and C1 to C3 (column 1-3) of the X2 connector.

7.3 Keypad with up to 8 Keys

In this mode, a keypad of up to 4 rows by 2 columns can to be used. To configure the LCD2S for a 8 key keypad, the following must be done:

- The keypad must be configured for a 4x2 keypad via the "*Configure Keypad and IO*" command. This command will also disable the GPIO2 and GPIO3 general purpose inputs/outputs.
- The GPIO2 and GPIO3 general purpose inputs/outputs are not available when using a 8 key keypad! The OUT1 output (jumper OUT1 on 2x6 pin header must be inserted) and GPIO1 general purpose input/output are available in this mode.
- The keypad must be connected to R1 to R4 (rows 1-4) and C1 to C2 (column 1-2) of the X2 connector.

7.4 Keypad with up to 4 Keys

In this mode, a keypad of up to 4 rows by 1 column can to be used. To configure the LCD2S for a 4 key keypad, the following must be done:

- The keypad must be configured for a 4x1 keypad via the "*Configure Keypad and IO*" command. This command will also disable the GPIO3 general purpose input.
- he GPIO3 general purpose input is not available when using a 4 key keypad! The OUT1 output(jumper OUT1 on 2x6 pin header must be inserted) and GPIO1 and GPIO2 general purpose inputs/outputs are available in this mode.
- The keypad must be connected to R1 to R4 (rows 1-4) and C1 (column 1) of the X2 connector.

7.5 Keypad with other number of keys

Any keypad with up to 16 keys, and a matrix of not more than 4 rows by 4 columns can be used. For example, a 4 key keypad with 2 rows by 2 columns can be used when the LCD2S is configured for a 16 (4x4), 12 (4x3) or 8 (4x2) key keypad.

8 Communication Protocols

The LCD2S board has a standard set of commands that can be sent to it via the I^2C or SPI interface. The communication protocol used is configured via a 2 position DIP switch. The DIP switch has two switches that can be on (1) or off (0), and thus provides four combinations: 00, 01, 10 and 11.

I^2C communication is enabled for the first three settings (00, 01 and 01) of the DIP switch. The I^2C address for each of the 3 positions is different, allowing 3 LCD2S boards to be connected to a single I^2C bus.

SPI communication is enabled for the last setting (11) of the DIP switch. Standard SPI mode 0 communication protocol is used.

9 Communication via SPI

9.1 Overview

This communication method is selected by setting the DIP switch to 11 (both switches are in the on position). SPI mode 0 is used. This method uses 4 signals for communication, CS, CLK, SDI and SDO. The SPI CS signal is used to enable communication with the LCD2S. Whenever the CS signal goes low, the SPI port on the LCD2S becomes active. In this state the SDO line will be an output, and the SDI and CLK lines will be inputs. Applying a clock signal to the CLK input will clock data into the LCD2S via the

SDI line, and data out via the SDO line.

Figure 5 below shows the timing diagram for this mode of communication.

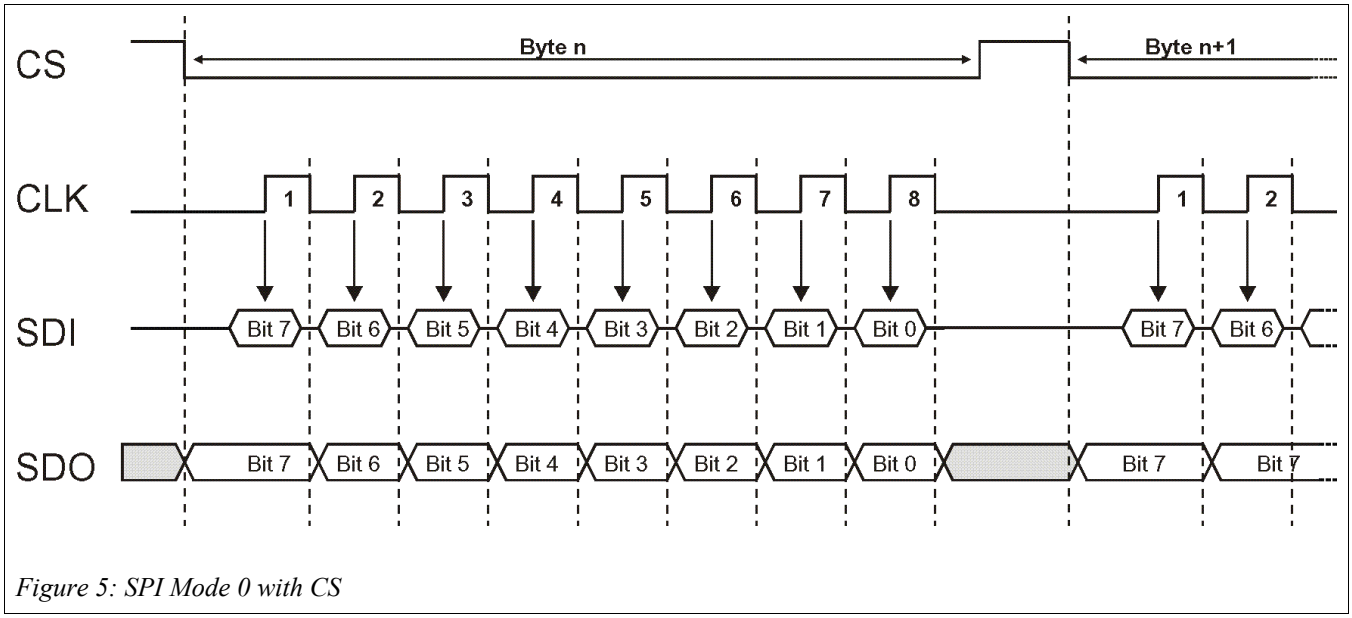


Figure 5: SPI Mode 0 with CS

When the CS signal is set high, the SPI port is disabled. In this state, the SDO line is tri-stated and the SDI input is disabled. The LCD2S will not respond to any data on the SPI bus, and will also not interfere with data been send between any other nodes that potentially share the SPI bus with it.

The 0xF5 (245 decimal) character is a special SYNC character used during this mode of communication. Whenever the LCD2S receives this character, it will reset it's command state machine, and assume the next character received is the start of a new command.
If for some reason a 0xF5 character has to be sent to the LCD2S as part of a command, and it should **not** be interpreted as a SYNC character, two 0xF5 characters have to be send after each other. Whenever the LCD2S sees two 0xF5 characters after each other, it will replace it with a single 0xF5 character, and the command state machine will **not** be reset!

9.2 Writing to the LCD2S

To send any of the commands listed in the commands section of this document, a 0xF5 character has to be added to the command sequence. For example, to send a "*Backlight On*" command to the LCD2S via this mode of communication, the following two bytes have to be sent:

0xF5, 0x28 (or in decimal: 245, 40)

Figure 6 below shows the timing diagram for sending a "*Backlight On*" command to the LCD2S via SPI.

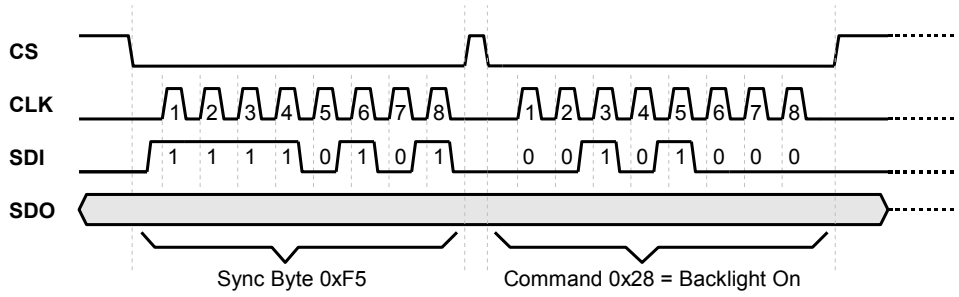


Figure 6: Timing diagram for sending a "*Backlight On*" command to the LCD2S via SPI.

For SPI timing diagrams see Chapter 14. "Specifications".

9.3 Reading from the LCD2S

When using the SPI communication mode, the command byte must always be preceded by a 0xF5 Sync character. Additionally a dummy byte (value is normally 0) has to be written after the sync byte. The dummy byte is required so that the LCD2S has time to prepare the requested data. Following the dummy byte, the requested bytes can be read from the LCD2S. The bits will be shifted out with the most significant bit (MSB) first, and bit 0 last. The output bits will be valid during the high level of the CLK signal.

For example, to read the "*Status Byte*" of the LCD2S via this mode of communication, we have to send three bytes and read one:

Send 0xF5 Sync byte, send 0xD0 command, send 0x00 dummy, read Status byte (or in decimal: 245, 208, 0)

Figure 7 below shows the timing diagram for reading the "*Status Byte*" of the LCD2S.

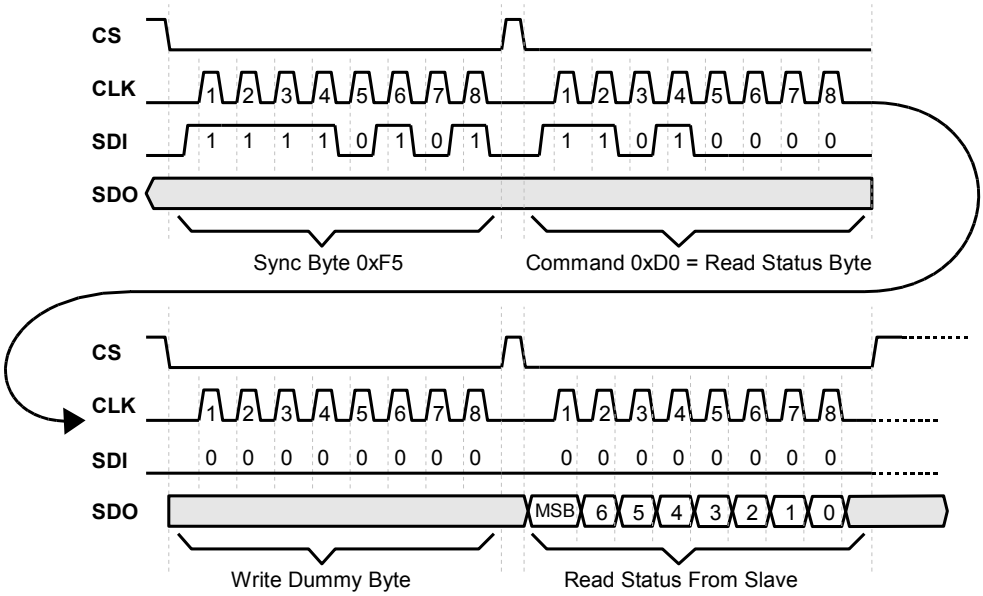


Figure 7: Timing diagram for reading the "*Status Byte*" from the LCD2S via SPI.

For SPI timing diagrams see Chapter 14. "Specifications".

10 Communication via I²C

10.1 Overview

The LCD2S can be configured to be an I²C slave device by setting the DIP switches to 00, 01 or 10. The bidirectional I²C bus consists of the serial clock (SCL) and serial data (SDA) lines. Both lines must be connected to a positive supply via a pullup resistor. The LCD2S doesn't have any pullup resistors, and they will normally be provided by the bus master. A typical value is between 2k2 and 1k. Additionally there is also an I²C interrupt line that will be pulled low by the LCD2S if it has pending data. Currently pending data will always be keypad data. If the I²C interrupt line is not used, the LCD2S has to be polled each couple of ms to see if it has any keypad data available.

I²C communication with this device is initiated by a master sending a Start condition, a high-to-low transition on the SDA input/output while the SCL input is high, see Figure 8. After the Start condition, the device address byte is sent, MSB first, including the data direction bit (R/W). This device does not respond to the general call address.

After receiving the valid address byte, this device responds with an ACK, a low on the SDA input/output during the high of the ACK-related clock pulse.

On the I²C bus, only one data bit is transferred during each clock pulse. The data on the SDA line must remain stable during the

high pulse of the clock period, as changes in the data line at this time will be interpreted as a Start or Stop condition, see Figure 8.

A Stop condition, a low-to-high transition on the SDA input/output while the SCL input is high, is sent by the master, see Figure 8.

Any number of data bytes can be transferred from the transmitter to the receiver between the Start and the Stop conditions. Each byte of eight bits is followed by one ACK bit. The transmitter must release the SDA line before the receiver can send an ACK bit. The device that acknowledges must pull down the SDA line during the ACK clock pulse so that the SDA line is stable low during the high pulse of the ACK-related clock period. When a slave receiver is addressed, it must generate an ACK after each byte is received. Similarly, the master must generate an ACK after each byte that it receives from the slave transmitter. Setup and hold times must be met to ensure proper operation.

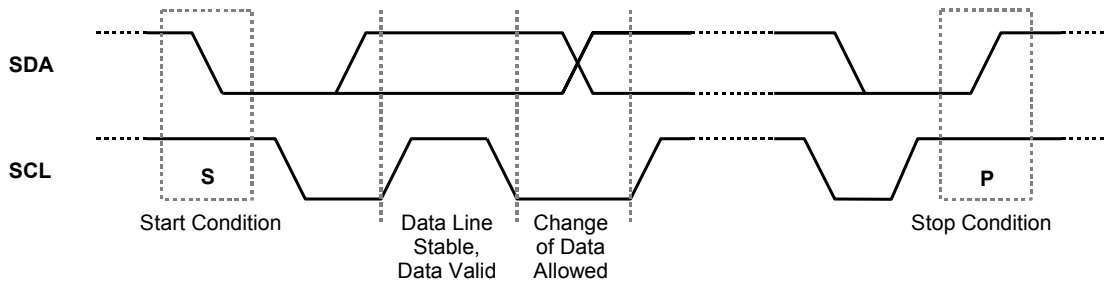


Figure 8: I2C Start condition, Stop condition, and Data

Figure 9 below shows the timing diagram for writing a command to the LCD2S.

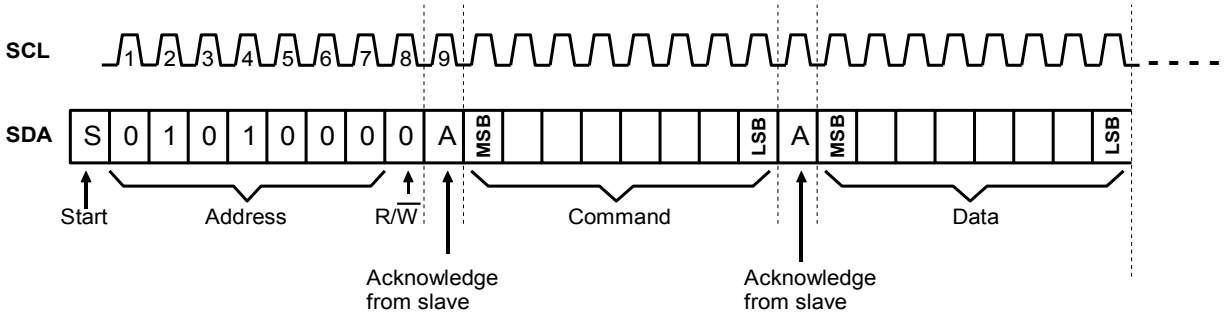


Figure 9: I2C write command

Reading data from an I2C slave device is a two stage process. Following the start condition, an address and command byte is written to the slave device. The R/W bit of the address byte is set for writing. This is followed by a repeated start condition. A repeated start condition is when a start is sent following a previous start with no stop condition send between them. Following the repeated start condition, an address byte is written to the slave (R/W bit of address is set for reading) followed by consecutive byte reads. The master will acknowledge each byte read, except the last one. The slave will thus continue sending data as long as the master acknowledges it.

Figure 10 below shows the timing diagram for reading a command from the LCD2S.

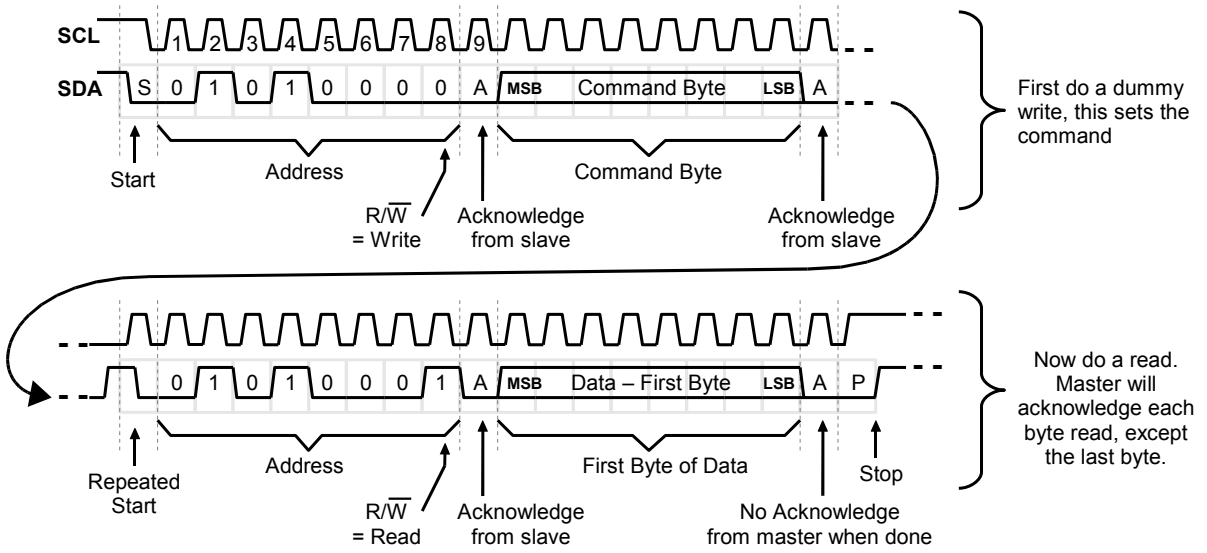


Figure 10: I2C read command

10.2 I2C Address

The LCD2S can be configured to be an I2C slave device by setting the DIP switches to 00, 01 or 10. Each position will configure the LCD2S to have an address relative to the I2C base address. The default I2C base address is 0x50 (decimal 80).

- For the default I2C base address, DIP switch setting 00 configures the I2C address to be 0x50 (decimal 80) for writing to the LCD2S, and 0x51 (decimal 81) for reading from it. When the I2C base address is modified via the "Set I2C base address" command, the read address is the base address, and the write address is the base address + 1.
- For the default I2C base address, DIP switch setting 01 configures the I2C address to be 0x52 (decimal 82) for writing to the LCD2S, and 0x53 (decimal 83) for reading from it. When the I2C base address is modified via the "Set I2C base address" command, the read address is the base address + 2, and the write address is the base address + 3.
- For the default I2C base address, DIP switch setting 10 configures the I2C address to be 0x54 (decimal 84) for writing to the LCD2S, and 0x55 (decimal 85) for reading from it. When the I2C base address is modified via the "Set I2C base address" command, the read address is the base address + 4, and the write address is the base address + 5.

Without modifying the I2C base address, it is thus possible having 3 LCD2S devices on a single I2C bus. By modifying the I2C base address, many more can be used. The exact number will depend on the length and speed of the bus, seeing that I2C can not work over very long distances at high speeds.

10.3 Writing to the LCD2S

Writing a command to the LCD2S via I2C follows the standard I2C format for doing a byte write. It is initiated with an I2C start condition. The I2C Start condition will reset the internal command state machine of the LCD2S. The next byte received will be seen as the first byte of a new command. The current command is ended as soon as a stop condition is detected. To send any of the commands listed in the commands section of this document, it has to be preceded by a start condition and the device address. For example, to send a "*Backlight On*" command to a LCD2S with an I2C write address of 0x50 (both DIP switches set to off), the following bytes have to be sent:

[I2C Start Condition] [0x50 byte] [0x28 byte] [I2C Stop Condition]

This is shown in Figure 11 below.

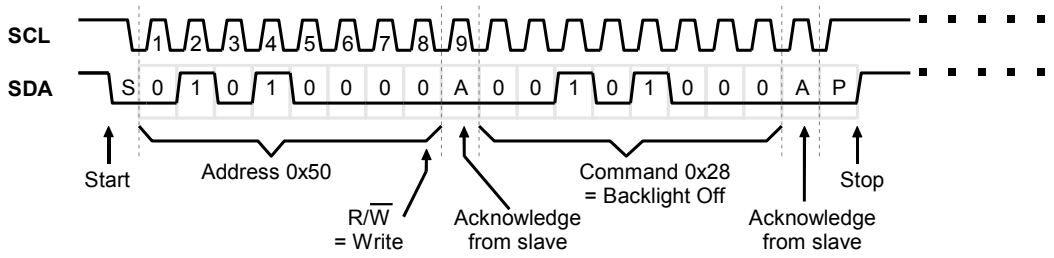


Figure 11: I2C backlight off command to LCD2S with write address 0x50

10.4 Reading from the LCD2S

Reading from the LCD2S via I2C follows the standard I2C format for doing a byte write followed by a byte read. It is thus a two stage process. During the byte write, only the address and command bytes are written to the LCD2S. During the second stage, the address byte is written again, followed by consecutive reads, until all required data has been obtained.

Following the start condition, an address and command byte is written to the slave device. The R/W bit of the address byte is set for writing. This is followed by a repeated start condition. A repeated start condition is when a start is sent following a previous start with no stop condition send between them. Following the repeated start condition, an address byte is written to the slave (R/W bit of address is set for reading) followed by consecutive byte reads. The master will acknowledge each byte read, except the last one. The slave will thus continue sending data as long as the master acknowledges it.

For example, to read the status byte of a LCD2S with an I2C write address of 0x50 (both DIP switches set to off), the following commands have to be sent:
[I2C Start Condition] [0x50 byte] [0xd0 byte] [I2C Repeated Start Condition] [0x51 byte] [Read Status byte] [I2C Stop Cond.]
The second address byte will be 0x51, seeing that the R/W bit (position 0) is set for a read. All bytes are sent from the master, except for the last byte that is sent from the LCD2S.

Figure 12 below shows the timing diagram for reading a command from the LCD2S.

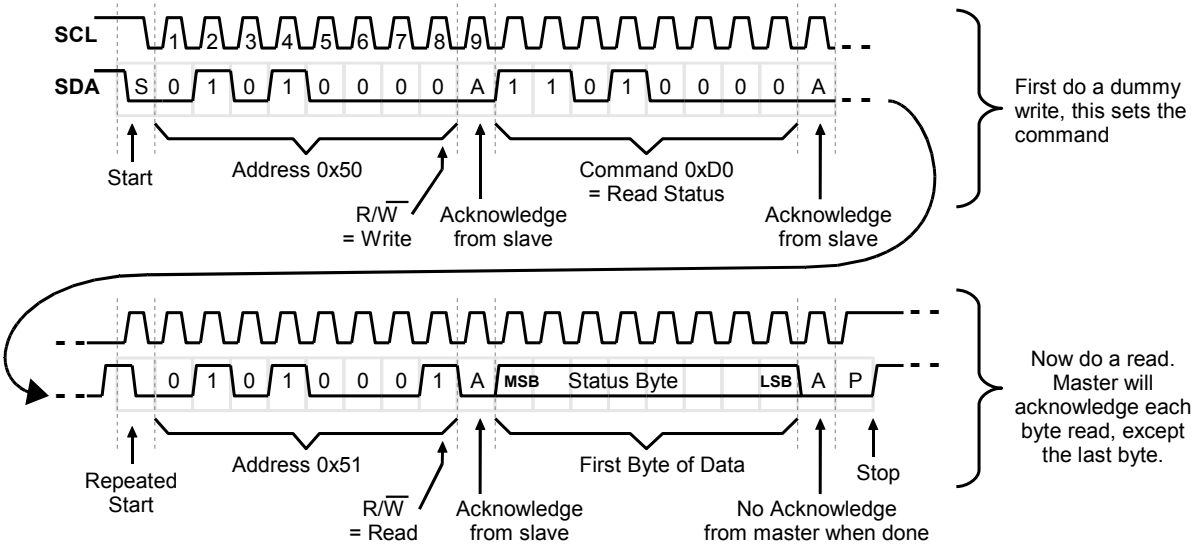


Figure 12: I2C read status byte from LCD2S with write address 0x50

11 Display Format

The LCD display has two rows of 16 characters each. When writing to it, characters are written to the current cursor position, after which the cursor is incremented or decremented (depending on how it is configured). The cursor can be configured to be an underline (non blinking) cursor, a blinking block cursor, both or off. When configured to move forward, the cursor will wrap around from the last position in the first row to the first position of the second row, and from the last position in the second row to the first position in the first row. To cursor can be moved to any location on the display by issuing either the "*Set Cursor Address*" command or the "*Set Cursor Location*" command. The address of each position on the display is shown in figure 13 below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Figure 13: Display Addresses

The column numbers are shown in figure 14 below.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Figure 14: Column Numbers

12 Commands

The LCD2S can accept commands via any of it's communication interfaces.

- Hex values are shown in 0xnn format, where nn is the hexadecimal value.
- Strings are shown as "string", and represents a string of bytes that are mapped to the Character Set shown in Character Set.
- Byte parameters to a command are shown as [value]
- Bytes read from the LCD2S are shown as <value>

12.1 State On Power-up

On power up, configuration data from the non volatile memory (EEPROM) is used to configure the LCD2S. This configuration data can be modified by the user by issuing a "Remember Command" followed by the command that is to be remembered. Only the command following the "Remember Command" will be remembered. To remember multiple command, the "Remember Command" has to be issues before each one. Only certain commands can be remembered. Each command's description will specify if it can be remembered or not.

The "Remember Command" should not be used unnecessarily for the following reasons:

- It is very slow. Each byte that is stored to non volatile memory takes about 6ms.
- The non volatile memory can only be written to a limited (about 1,000,000 times) number of times.

Because a remember command is very slow (about 6ms) the user should check how many bytes are still available in the receive buffer (Read Status Byte command) before issuing multiple remember commands! All commands written after a remember command will also only be executed (will remain in the receive buffer) once the remember command has been executed.

12.2 General Commands

12.2.1 Remember Command

Command: 0x8D	**Default:** Yes	**Can be remembered:** Yes

The command following this command will be remembered (stored in non volatile memory), and will be restored next time the unit is powered up. Only certain commands can be remembered. Each command's description will specify if it can be remembered or

not. See Chapter 7. “State On Power-up“ for details.

12.2.2 Read Device Status Byte

Command I²C: 0xD0, <read value> **Command SPI:** 0xD0, 0x00, <read value>	**Default: -**	**Can be remembered:** No

This command reads the current device status register. When executing this command via the SPI port, a dummy byte must be sent after the 0xD0 command! The bits of this register have the following meaning:

Bit 7 – Keypad Buffer Contains Data: When set, this bit indicates that the keypad buffer contains data. The keypad data can be read with the “*Read Keypad Data*” command.

Bits 0-6 – Receive buffer space available: Bits 0-6 indicate how much of the 80 byte receive buffer is still available. It is very important that the receive buffer does not overflow! The user should never write more bytes to the LCD2S then it has space for in it's receive buffer.

12.3 Configuration Commands

12.3.1 Configure Device

Command: 0x95, [Display], [Contrast and Brightness], [keypad and IO], [Keypad and Buzzer]	**Default: -**	**Can be remembered: -**

This is a new command, and is **only available in V1.40** and higher devices! Provides a single command for configuring most of the LCD2S. Not all possible configuration combinations are possible with this command, and other commands might be required to to fine tune some settings. This command can be used at power up to configure the LCD2S. It provides a single command that can be used in stead of sending many separate commands, like the “Configure Keypad and IO“, “Configure GPIO1“, “Backlight On“, ….. and many other commands that are issued by this single command.

The configuration values given in the third and fourth bytes (keypad and IO, Keypad and Buzzer) of this command are remembered, and will be restored at the next power up. The settings contained in the first and second byte (Display, Contrast and Brightness) are not remembered at the next power up. Use separate Display, Contrast and Brightness configuration command , (preceded by the “Remember Command“) to achieve this.

The **first byte** is used to configure the **display**. The “Dflt” column indicates the default value of the LCD2S.

Bit	Dflt	Description
6-7	0	**Not Used**
5	1	**Configure Interrupt pin output type – see Configure Interrupt Pin As Open Collector Output for details** 0 = Push-pull output type 1 = Open Collector output type
4	0	**Backlight On/Off** 0 = Backlight off 1 = Backlight on – brightness will be value last set with “Set Backlight Brightness” command
3	1	**Display On/Off** 0 = Display off - the displays is still powered, but nothing is displayed on it. It is blank 1 = Display on - turns the display on. Will return to what was displayed before display was turned off
2	0	**Show Cursor** 0 = Cursor off 1 = Cursor on – an underline cursor is displayed at the current cursor position. It does not blink.
1	0	**Show Blinking Block Cursor** 0 = Off 1 = On – A large blinking, block cursor is displayed at the current cursor position
0	1	**Cursor Direction** 0 = Cursor moves backwards 1 = Cursor Moves forward

The **second byte** is used to configure the **contrast and brightness**. The value of bits 0-5 are used for the contrast setting, and bits 6-7 for the backlight.

Bit	Dflt	Description
6-7		**Backlight setting,** 00 = off, 01 = 100, 10=200, 11=full
0-5	0	**Contrast setting**, the value given here is multiplied by 4. For example, if bit 0-5 have a value of 40 (0x28), the contrast will be set to 160.

The **third byte** is used to configure the **keypad and I/Os**. This command will also turn OUT2, and any other IO (OUT1, GPIO1 and GPIO2) configured as an output off (set value to 0V). Jumper OUT1 (on 2x6 pin header) must be present when OUT1 is enabled, and removed when disabled. For additional information, see Keypad and I/O Configuration.

Value	Keypad	OUT1	GPIO1 / AN1	GPIO2 / AN2	GPIO3
0x0f	4x4	x	x	x	x
0x40	4x3	√	x	x	x
0x80	4x2	√	Digital Input	x	x
0x81	4x2	√	Digital Output	x	x
0xc0	4x1	√	Digital Input	Digital Input	x
0xc1	4x1	√	Digital Output	Digital Input	x
0xc4	4x1	√	Digital Input	Digital Output	x
0xc5	4x1	√	Digital Output	Digital Output	x
0x00	x	√	Digital Input	Digital Input	Digital Input
0x01	x	√	Digital Output	Digital Input	Digital Input
0x04	x	√	Digital Input	Digital Output	Digital Input
0x05	x	√	Digital Output	Digital Output	Digital Input

The **fourth byte** is used to configure the **keypad and buzzer**. The "Dflt" column indicates the default value.

Bit	Dflt	Description
6-7	01	**Keypad Buzzer Period**, for details see Set Keypad Buzzer Period 00 = Do not change current setting 01 = Off 10 = 80ms 11 = 160ms
4-5	10	**Keypad Repeat Rate**, for details see Set Keypad Repeat Rate 00 = Do not change current setting 01 = Set to 192ms 10 = Set to 320ms 11 = Set to 448ms
2-3	10	**Keypad Repeat Delay**, for details see Set Keypad Repeat Delay 00 = Do not change current setting 01 = Set to 512 ms 10 = Set to 1 second 11 = Set to 1.5 seconds
0-1	01	**Keypad Debounce Time**, for details see Set Keypad Debounce Time 00 = Do not change current setting 01 = Set keypad debounce time to 32ms 10 = Set keypad debounce time to 64ms 11 = Set keypad debounce time to 96ms

12.3.2 Configure Interrupt Pin As Open Collector Output

Command: 0x2A	**Default:** Yes	**Can be remembered:** Yes

This command only affects the I2C communication mode. It configures the CS signal (used to indicate a pending interrupt on the LCD2S) to be an open collector output. This means that an external pull-up resistor is required on the CS signal. It has the advantage that multiple nodes can share the CS line to indicate they have a pending interrupt available. If only a single device will be connected to the CS line, then configure the interrupt pin to be a push-pull output. The following conditions will cause the

interrupt pin to be active:

- If there is any keypad data available to be read

12.3.3 Configure Interrupt Pin As Push-Pull Output

Command: 0x22	**Default:** No	**Can be remembered:** Yes

This command only affects the I2C communication mode. It configures the CS signal (used to indicate a pending interrupt on the LCD2S) to be a push-pull type output. No external pull-up resistor is required on the CS line, and only a single device (a LCD2S in this case) can be connected to it.

12.3.4 Set I2C base address

Command: 0x91	**Default:** 0x50	**Can be remembered:** Required

This command only affects the I2C communication mode, and is **only** executed if preceded by the remember command! It sets the I2C base address to the given value. The I2C base address is used to determine the I2C address when reading from, or writing to the LCD2S. See the "I2C Address" section of this document for details.

12.3.5 Configure Keypad and IO

Command: 0xE0, [value]	**Default:** 1 = 4x3 keypad, OUT1 on, GPIO1 to GPIO3 disabled, AN1 and AN2 disabled	**Can be remembered:** Required

This command configures the keypad encoder, the OUT1 open collector output, and the 3 general purpose inputs/outputs GPIO1 to GPIO3. This command is **only** executed if preceded by the remember command! Because these functions share CPU port pins, they can not all be enabled at the same time. The available configuration options are:

	Keypad	OUT1	GPIO1 / AN1	GPIO2 / AN2	GPIO3
0x00	4x4				
0x01	4x3	√			
0x03	4x2	√	√		
0x07	4x1	√	√	√	
0x0f	none	√	√	√	√

√ *indicates enabled*

When OUT1 is configured to be disabled, the OUT1 jumper (on 2x6 pin header) must be removed. When OUT1 is configured to be enabled, the OUT1 jumper must be inserted!

At delivery, the **default configuration** is for a 4x3 keypad, OUT1 enabled, GPIO1 to GPIO3 disabled and AN1 and AN2 disabled. The OUT1 jumper (on 2x6 pin header) is also inserted. For details on configuration, see Chaper 7. "Keypad and I/O Configuration".

Seeing that this command is written to the LCD2S's non volatile memory (remembered), it **takes about 6ms to be executed**! Be sure to check how many bytes are still available in the receive buffer (Read Status Byte command), so that the receive buffer does not overflow! Additionally the non volatile memory can only be written to a limited (about 1,000,000 times) number of times, so **don't call this command too often**!

12.3.6 Configure GPIO1

Command: 0xE3, [value]	**Default:** Digital Input	**Can be remembered:** Required

The command configures GPIO1 to be an Analog Input, Digital Input or Digital Output. This command is **only** executed if preceded by the remember command! Currently the Analog Input function has **not been implemented** (will be implemented in future version)! following values are defined:

– 0x00 = Digital Input

- 0x01 = Digital Output
- 0x02 = Analog Input **(not implemented yet)**
- 0x04 = Digital Input with Pull-up resistor enabled
- 0x06 = Analog Input with Pull-up resistor enabled **(not implemented yet)**

To be able to use GPIO1, the keypad must be configured for 8 keys (4 row by 2 column) or less. For details on configuration, see Chaper 7. “Keypad and I/O Configuration“.

Seeing that this command is written to the LCD2S's non volatile memory (remembered), it **takes about 6ms to be executed**! Be sure to check how many bytes are still available in the receive buffer (Read Status Byte command), so that the receive buffer does not overflow! Additionally the non volatile memory can only be written to a limited (about 1,000,000 times) number of times, so **don't call this command too often**!

12.3.7 Configure GPIO2

Command: 0xE4, [value]	**Default:** Digital Input	**Can be remembered:** Required

The command configures GPIO2 to be an Analog Input, Digital Input or Digital Output. This command is **only** executed if preceded by the remember command! Currently the Analog Input function has **not been implemented** (will be implemented in future version)**!** The following values are defined:

- 0x00 = Digital Input
- 0x01 = Digital Output
- 0x02 = Analog Input **(not implemented yet)**
- 0x04 = Digital Input with Pull-up resistor enabled
- 0x06 = Analog Input with Pull-up resistor enabled **(not implemented yet)**

To be able to use GPIO2, the keypad must be configured for 4 keys (4 row by 1 column) or less. For details on configuration, see Chaper 7. “Keypad and I/O Configuration“.

Seeing that this command is written to the LCD2S's non volatile memory (remembered), it **takes about 6ms to be executed**! Be sure to check how many bytes are still available in the receive buffer (Read Status Byte command), so that the receive buffer does not overflow! Additionally the non volatile memory can only be written to a limited (about 1,000,000 times) number of times, so **don't call this command too often**!

12.4 LCD Display Commands

12.4.1 Write Parsed String

Command: 0x80, “String”	**Default: -**	**Can be remembered:** No

Writes the given string to the LCD. All characters are mapped to the Character Set shown in Error: Reference source not found. The string is parsed for escape sequence characters that perform the following actions:

- 0x0a ('\n') = Go to beginning of next line.
- 0x0c ('\f') = Clear display and go to beginning of first line.
- 0x0d ('\r') = Go to beginning of first line
- 0x08 ('\b') = Cursor left
- 0x09 ('\t') = Cursor Right

The escape sequence addresses don't store any characters in the Character Set, so all possible characters can still be displayed.

For example, to write “Hello” to line 1, and “world” to line two, the following string can be written to the LCD2S:
`"\fHello\nworld"`

12.4.2 Set Startup Screen

Command: 0x90, [line], “String”	**Default: -**	**Can be remembered:** Always

Sets a single line of the startup screen. The startup screen is displayed when the LCD is switched on.

The *line* parameter of this command gives the LCD line that the following string is for. For a two line display (16 by 2 lines for example), the line parameter can be 1 or 2. For a 4 line display (20 by 4 lines for example), the line parameter can be 1, 2, 3 or 4.

The *string* parameter of this command should be a maximum of 20 characters long. For a 2 by 16 line display, the string will be 16 characters long. For a 4 by 20 line display the string will be 20 characters long.

For example, to set line one of a 2 by 16 line display to “This is a really” and line two to “cool LCD Display”, the following two commands have to be sent to the LCD2S:
The first command writes “This is a really” string to line 1:

Command Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Command	0x90	0x01	T	h	i	s		i	s		a		r	e	a	l	l	y

The second command writes “cool LCD Display” string to line 2:

Command Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Command	0x90	0x02	c	o	o	l		L	C	D		D	i	s	p	l	a	y

Because this command stores the given string to the LCD2S's internal EEPROM, it is very slow. About 6ms should be allowed for each character of the given string. This means, that if a 16 character string is written, the command will take about 16 x 6ms = 96ms to be processed.

12.4.3 Backlight On

Command: 0x28	**Default:** Yes	**Can be remembered:** Yes

Switches the display on. The display brightness will be the value last set with the “*Set Backlight Brightness*” command.

12.4.4 Backlight Off

Command: 0x20	**Default:** Yes	**Can be remembered:** Yes

Switches the display off. The display brightness will not be modified, and will be restored the next time the “*Display On*” command is issued.

12.4.5 Set Backlight Brightness

Command: 0x81, [value]	**Default:** Yes	**Can be remembered:** Yes

Sets the current brightness level of the backlight. The “value” parameter has to be a value from 0 to 255, where 0 is minimum (not off) and 255 maximum brightness. Additionally the value has to be less than or equal to the *maximum backlight brightness* value set with the “*Set Maximum Backlight Brightness*” command. This is to protect certain displays that could get damaged if their backlight value is set too high!

12.4.6 Set Maximum Backlight Brightness

Command: 0xA3, [value]	**Default:** Depends on display type	**Can be remembered:** Required

Sets the maximum allowed backlight value that can be set with the “*Set Backlight Brightness*” command. This command is **only** executed if preceded by the remember command! For bare (without display board) LCD2S boards, the default value is 255, and has no affect on the “*Set Backlight Brightness*” command. For LCD2S boards combined with displays, this value is set to the correct value for that display, and should normally not be changed!

Seeing that this command is written to the LCD2S's non volatile memory (remembered), it **takes about 6ms to be executed**! Additionally the non volatile memory can only be written to a limited number of times (about 1,000,000 times), so **don't call this command too often**!

12.4.7 Cursor moves forward

Command: 0x09	**Default:** Yes	**Can be remembered:** Yes

Causes the cursor (or display) to move one position forward each time a character is written to the display. The default mode of the LCD2S is for the cursor to move forward.

12.4.8 Cursor moves backwards

Command: 0x01	**Default:** No	**Can be remembered:** Yes

Causes the cursor (or display) to move one position backwards each time a character is written to the display.

12.4.9 Blinking Block Cursor On

Command: 0x18	**Default:** No	**Can be remembered:** Yes

Turns the blinking block cursor on. It is displayed at the current cursor position. It is a large, blinking block that is displayed at the current cursor position.

12.4.10 Blinking Block Cursor Off

Command: 0x10	**Default:** Yes	**Can be remembered:** Yes

Turns the blinking block cursor off. It is displayed at the current cursor position. It is a large, blinking block that is displayed at the current cursor position.

12.4.11 Underline Cursor On

Command: 0x19	**Default:** No	**Can be remembered:** Yes

Turns the underline cursor on. It is displayed at the current cursor position. This is not the blink cursor, and can not be configured to blink. It is a line that appears under the character at the current cursor position. The cursor is displayed using 5 dots in the 8th line.

12.4.12 Underline Cursor Off

Command: 0x11	**Default:** Yes	**Can be remembered:** Yes

Turns the underline cursor off. It is displayed at the current cursor position. This is not the blink cursor, and can not be configured to blink. It is a line that appears under the character at the current cursor position.

12.4.13 Display On

Command: 0x1A	**Default:** Yes	**Can be remembered:** Yes

Turns the entire display on. If the display is turned off, nothing is visible. This command does not change the contents of the *Display RAM*! It also does not change the value of the display contrast set by the "*Set display Contrast*" command.

12.4.14 Display Off

Command: 0x12	**Default:** No	**Can be remembered:** Yes

Turns the entire display off. Nothing will be displayed on the display – it will be blank. This command does not change the

contents of the *Display RAM*! When off, the display data remains in the *Display RAM*, and can be displayed instantly by issuing the “Display On” command.

12.4.15 Set Display Contrast

Command: 0x82, [value]	**Default:** Yes	**Can be remembered:** Yes

Sets the current contrast level of the LCD display. The “value” parameter has to be a value from 0 to 254, where 0 is off and 254 will be maximum contrast.

12.4.16 Move Cursor Right

Command: 0x83	**Default:** -	**Can be remembered:** No

Moves the cursor one position forward (right). This command does not change the contents of the *Display RAM*! When shifting the cursor right along the first line, it will move to the second line when passing the 40th character of line 1. The next character written to the LCD will be at the new location of the cursor. Please note that it is possible shifting the cursor to a *Display RAM* address that is not currently visible!

12.4.17 Move Cursor Left

Command: 0x84	**Default:** -	**Can be remembered:** No

Moves the cursor one position backwards (left). This command does not change the contents of the *Display RAM*! When shifting the cursor left along the first line, it will move to *Display RAM* address 83 when passing the first character of line 1. When shifting the cursor left along the second line, it will move to *Display RAM* address 40 when passing the first character of line 2. The next character written to the LCD will be at the new location of the cursor. Please note that it is possible shifting the cursor to a *Display RAM* address that is not currently visible!

12.4.18 Shift Display Right

Command: 0x85	**Default:** -	**Can be remembered:** No

Moves the contents of the display one position right. The last character of both rows will shift off the display, and be lost. The cursor does not move. The first characters of both rows will be filled with space characters. Doing a “Shift Display Left” after this command will NOT restore the display to it's original condition!
For example, if the display contains the text “This is a really cool LCD Display”, and this command is issued, the following will happen:

Figure 15: Before command

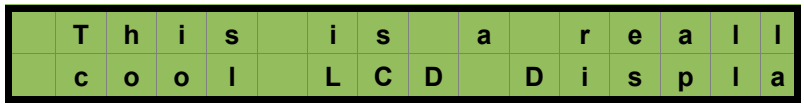


Figure 16: After command

12.4.19 Shift Display Left

Command: 0x86	**Default:** -	**Can be remembered:** No

Moves the contents of the display one position left. The first character of both rows will shift off the display, and be lost. The cursor does not move. The last characters of both rows will be filled with space characters. Doing a “Shift Display Right” after this command will NOT restore the display to it's original condition!
For example, if the display contains the text “This is a really cool LCD Display”, and this command is issued, the following will happen:

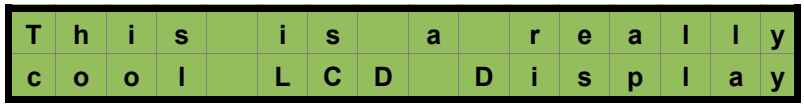


Figure 17: Before command

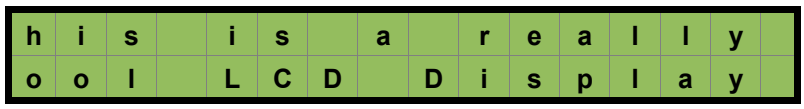


Figure 18: After command

12.4.20 Shift Display Up

Command: 0x87	**Default:** -	**Can be remembered:** No

Copied the contents of row 2 to row 1, and fills row 2 with space characters. The contents of row 1 is lost after this command, and issuing a "*Shift Display Down*" command after this command will NOT restore the display to it's original condition! The cursor does not move. For example, if the display contains the text "This is a really cool LCD Display", and this command is issued, the following will happen:

Figure 19: Before command

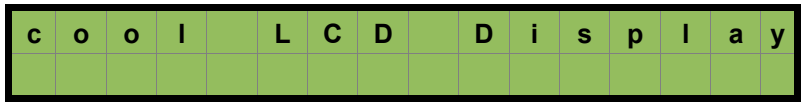


Figure 20: After command

12.4.21 Shift Display Down

Command: 0x88	**Default:** -	**Can be remembered:** No

Copied the contents of row 1 to row 2, and fills row 1 with space characters. The contents of row 2 is lost after this command, and issuing a "*Shift Display Up*" command after this command will NOT restore the display to it's original condition! The cursor does not move. For example, if the display contains the text "This is a really cool LCD Display", and this command is issued, the following will happen:

Figure 21: Before command

Figure 22: After command

12.4.22 Set Cursor Address

Command: 0x89, [cursor position]	**Default:** -	**Can be remembered:** No

Sets the cursor location to the given *Display RAM* address. If no display shift commands have been issues:

The following is valid for a **2x16** line display:
- The first character of line 1 will be at *Display RAM* address 0, and the last character at address 15
- The first character of line 2 will be at *Display RAM* address 64, and the last character at address 79

The following is valid for a **4x20** line display:
- The first character of line 1 will be at *Display RAM* address 0, and the last character at address 19

- The first character of line 2 will be at *Display RAM* address 64, and the last character at address 83
- The first character of line 3 will be at *Display RAM* address 20, and the last character at address 39
- The first character of line 4 will be at *Display RAM* address 84, and the last character at address 103

If display shift commands have been issued, the user will have to keep track of what *Display RAM* locations are currently visible!

12.4.23 Set Cursor Position

Command: 0x8A, [row], [column]	**Default:** -	**Can be remembered:** No

Moves the cursor to the given row and column location. The *row* parameter is a value from 1 to 2 for a 2x16 line display, and 1 to 4 for a 4x20 line display. The *column* parameter is a value from 1 to 16 for a 2x16 line display, and 1 to 20 for a 4x20 line display. This command will compensate for any display shifts that have been done!

Command: 0x8B	**Default:** -	**Can be remembered:** No

This command sets the cursor position to the first character of row 1.

12.4.24 Clear Display

Command: 0x8C	**Default:** -	**Can be remembered:** No

This command sets the cursor position to the first character of row 1 and writes space characters (0x20) to all Display RAM locations.

12.5 Bar Graphs and Custom Characters

The LCD2S has space for 8 custom characters. The following commands are used to define these custom characters to user defined values, or load them with predefined values.

These custom characters are mapped to addresses 0 to 7 of the Character Set. For example, to display custom characters 0,1,2 and 7 you could issue the following “Write Parsed String“ command:
0x80, 0x00, 0x01, 0x02, 0x07
The 0x80 is the code for the “Write Parsed String“ command, and the following 4 bytes give the addresses of the characters from the Character Set to display.

12.5.1 Define Custom Character

Command: 0x92, [adr], [8 byte code]	**Default:** -	**Can be remembered:** Yes

The LCD2S has space for 8 custom characters. Each custom character is 5 pixels wide by 8 pixels high. This command is used to define one of these 8 custom characters.

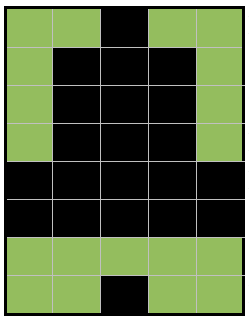
The *adr* parameter indicates which custom character this command is defining, and must have a value from 0 to 7, where 0 is for the first custom character, 1 for the second custom character, up to 7 which is for the eighth character.

Following the *adr* parameter are 8 bytes that define the custom character. Bits 0 to 4 of each byte will each define a pixel of the character. When the bit is 1, the pixel is on. When the bit is 0, the pixel is off. The first byte defines the pixels for the top row, the second byte defines the pixels for the second row, up to the eighth byte which defines the pixels for the last row.

4	3	2	1	0	First byte. Bits 0 to 4 define pixel state. Bits 5 to 7 are not used, and can be 1 or 0.
4	3	2	1	0	Second byte. Bits 0 to 4 define pixel state. Bits 5 to 7 are not used, and can be 1 or 0.
4	3	2	1	0	Third byte. Bits 0 to 4 define pixel state. Bits 5 to 7 are not used, and can be 1 or 0.
4	3	2	1	0	Fourth byte. Bits 0 to 4 define pixel state. Bits 5 to 7 are not used, and can be 1 or 0.
4	3	2	1	0	Fith byte. Bits 0 to 4 define pixel state. Bits 5 to 7 are not used, and can be 1 or 0.
4	3	2	1	0	Sixth byte. Bits 0 to 4 define pixel state. Bits 5 to 7 are not used, and can be 1 or 0.
4	3	2	1	0	Seventh byte. Bits 0 to 4 define pixel state. Bits 5 to 7 are not used, and can be 1 or 0.
4	3	2	1	0	Eighth byte. Bits 0 to 4 define pixel state. Bits 5 to 7 are not used, and can be 1 or 0.

A Custom Character Calculator with a graphical user interface for calculating the 8 bytes is at the following web page:
www.modtronix.com/info/cgramcalc

For example, to define the second custom character (address 0x01) to be a picture of a bell, we will send the following command to the LCD2S:
0x80, 0x01, 0x04, 0x0e, 0x0e, 0x0e, 0x1f, 0x1f, 0x00, 0x04
The values of the 8 bytes are calculated as follows:



First byte = 0x04 hex, or 4 decimal.
Second byte = 0x0e hex, or 14 decimal.
Third byte = 0x0e hex, or 14 decimal.
Fourth byte = 0x0e hex, or 14 decimal.
Fith byte = 0x1f hex, or 31 decimal.
Sixth byte = 0x1f hex, or 31 decimal.
Seventh byte = 0x00 hex, or 0 decimal.
Eighth byte = 0x04 hex, or 4 decimal.

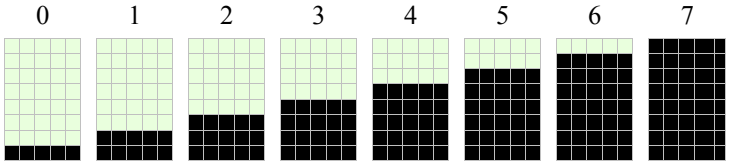
12.5.2 Load Custom Character Set

Command: 0x8E, [character set code]	**Default:** -	**Can be remembered:** No

This command is used to load the 8 custom characters with a set of 8 predefined characters. Some commands, like the bar graph commands, require that a specific custom character set be loaded prior to it's use. The following custom character sets are defined:

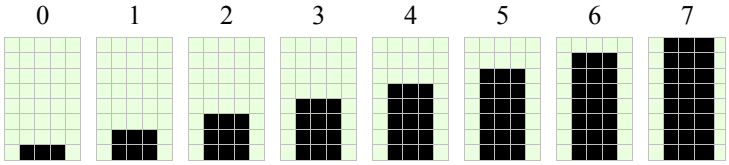
Character Set Code: 0 **Name:** Vertical Bar Graph

Defines character set for displaying numbers that are 3 character high, and 2 character wide. This character set has to be loaded prior to using the “Write Large Number String to LCD“ command.



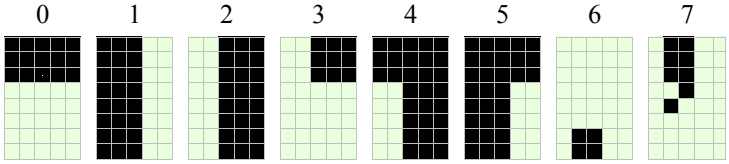
Character Set Code: 1 **Name:** Vertical Bar Graph Narrow

Defines character set for displaying numbers that are 3 character high, and 2 character wide. This character set has to be loaded prior to using the “Write Large Number String to LCD“ command.



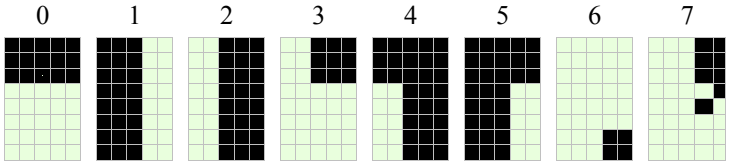
Character Set Code: 2 **Name:** Large 3x2 Number Character Set

Defines character set for displaying numbers that are 3 character high, and 2 character wide. This character set has to be loaded prior to using the “Write Large Number String to LCD“ command.



Character Set Code: 3 **Name:** Large 3x2 Number Character Set , variation B

Defines character set for displaying numbers that are 3 character high, and 2 character wide. This is the same as the previous character set, except that the last two characters are shifted right 2 pixels. These two characters are used for displaying the colon and comma characters when writing a “Large 3x2 Number Character” string.. This variation of the “Large 3x2 Number Character Set” is best to use when spaces are inserted after the colon and comma character, and will give a better visual effect. This character set has to be loaded prior to using the “Write Large Number String to LCD“ command.



12.5.3 Draw Vertical Bar Graph

Command: 0x93, [row], [column], [height]	**Default: -**	**Can be remembered:** No

Draws a vertical bar graph in the space of a single character position. Before using this command, a vertical bar graph character set has to be loaded with the “Load Custom Character Set“ command.

The *row* and *column* parameters gives the location of the bar graph, and are values from 1 to the maximum row and column count of the display used.

The *height* parameter gives the height of the bar graph to draw, and can have a range from 0 to 8. If 0, the given location is cleared. For a value between 1-8 a vertical bar graph is drawn in the given cell with that height.

12.5.4 Draw Tall Vertical Bar Graph

Command: 0x94, [row], [column], [height]	**Default: -**	**Can be remembered:** No

Draws a vertical bar graph spanning two rows in the same column. Before using this command, a vertical bar graph character set has to be loaded with the “Load Custom Character Set“ command.

The *row* and *column* parameters gives the location of the bottom of the bar graph, and are values from 1 to the maximum row and column count of the display used. For example, if the row is 2 and the column 3, then the bar graph will start in row 2 column 3, and end in row 1 column 3.

The *height* parameter gives the height of the bar graph to draw, and can have a range from 0 to 16. If 0, the given location is cleared. For a value between 1-16 a vertical bar graph is drawn in the given cell with that height.

12.5.5 Write Large Number String to LCD

Command: 0x8F, “String”	**Default: -**	**Can be remembered:** No

This command is **only valid when the display has 4x20 lines**. When using a 4x20 line display, the LCD2S can be configured for displaying large number fonts. In this mode, each number is 3 characters high, and 2 characters wide. The other characters (colon, comma and space) in this mode are 3 characters high, and 1 character wide. Best results are obtained by placing a space between each number. This allows a total of 7 numbers to be displayed next to each other, with a comma or colon between any of them if desired. This command writes the given large number (3 characters high, 2 characters wide) string to the LCD. It starts at the current cursor position, which is at the bottom left side of the string that will be displayed.

Before using this command, the “Large 3x2 Number Character Set” has to be loaded with the “Load Custom Character Set“ command. Only the following characters are accepted:

0 , **1** , **2** , **3** , 4 , **5** , **6** , **7** , **8** , **9** , **.** , **:** , space ,

The string is parsed for the following escape sequence characters that perform the following actions:

- 0x09 ('\t') = Cursor Right
- 0x0c ('\f') = Clear display and go to beginning of first line
- 0x0d ('\r') = Go to beginning of first line

The escape sequence addresses don't store any characters in the Character Set, so all possible characters can still be displayed.

For example, to write “12:55:04” in the top 3 lines of the display, the following command can be sent to the LCD2S:
`"\f1\t2:5\t5:0\t4"`
Note that we place a space (move cursor right with '\t' escape sequence) between consecutive number characters, this gives the best visual effect.

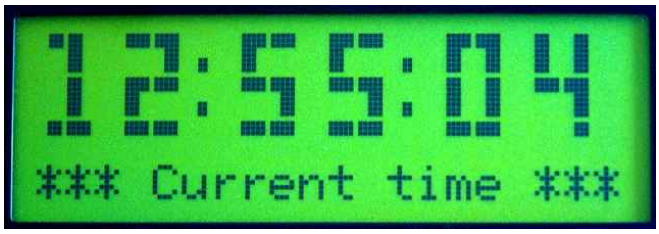


Figure 23: Shows a 4x20 line display with the time written to it using the “Write Large Number String to LCD“ command and the remaining text written to it using the “Write Parsed String” command.

12.6 Keypad Commands

12.6.1 Set Keypad Debounce Time

Command: 0xE1, [value]	**Default:** 3 = 24ms	**Can be remembered:** Required

Sets the keypad debounce time. This command is **only** executed if preceded by the remember command! This is the software filter time used for the keypad encoding, and has to be a value between 1 and 15. The debounce time is calculated as follow:

Debounce Time = value x 8ms

For example, a value of 1 sets the debounce time to 8ms, a value of 2 sets it to 16ms, ... , a value of 15 sets it to 120ms.

The default of 24ms (a value of 3) should be fine for most keypads. If a key press is registered twice by the LCD2S when a button is pressed, then the debounce time should be increased.

Seeing that this command is written to the LCD2S's non volatile memory (remembered), it **takes about 6ms to be executed**! Additionally the non volatile memory can only be written to a limited number of times (about 1,000,000 times), so **don't call this command too often**!

12.6.2 Set Keypad Repeat Delay

Command: 0xA0, [value]	**Default:** 62 = 1 seconds	**Can be remembered:** Required

Sets the keypad repeat delay. This command is **only** executed if preceded by the remember command! When holding a button in on the keypad, this is the initial delay before the keypad starts automatically repeating the pressed key. This time is calculated as follow:

Keypad Repeat Delay = value x 16ms

For example, a value of 10 sets the repeat rate to 160ms, a value of 100 sets it to 1600ms...

The default of 1 seconds (a value of 62) should be fine for most keypads.

Seeing that this command is written to the LCD2S's non volatile memory (remembered), it **takes about 6ms to be executed**! Additionally the non volatile memory can only be written to a limited number of times (about 1,000,000 times), so **don't call this command too often**!

12.6.3 Set Keypad Repeat Rate

Command: 0xA1, [value]	**Default:** 20 = 320ms	**Can be remembered:** Required

Sets the keypad repeat delay. This command is **only** executed if preceded by the remember command! When holding a button in

on the keypad, this is the rate at which the pressed key is automatically repeated (after the initial Keypad Repeat Delay). This time is calculated as follow:

Keypad Repeat Rate = value x 16ms

For example, a value of 10 sets the repeat rate to 160ms, a value of 20 sets it to 320ms...

The default of 320 ms (a value of 20) should be fine for most applications.

Seeing that this command is written to the LCD2S's non volatile memory (remembered), it **takes about 6ms to be executed**! Additionally the non volatile memory can only be written to a limited number of times (about 1,000,000 times), so **don't call this command too often**!

12.6.4 Set Keypad Buzzer Period

Command: 0xA2, [value]	**Default:** 0 = off	**Can be remembered:** Required

Sets the time that OUT2 will be activated for each time a button is pressed. This command is **only** executed if preceded by the remember command! This is useful when a buzzer is connected to output OUT2, and it is required to get activated each time a button is pressed. The time OUT2 is activated each time a button is pressed is calculated as follow:

Keypad Buzzer Period = value x 16ms

For example, a value of 10 activates OUT2 for 160ms, a value of 20 activates it to 320ms...

By default this function is switched off!

Seeing that this command is written to the LCD2S's non volatile memory (remembered), it **takes about 6ms to be executed**! Additionally the non volatile memory can only be written to a limited number of times (about 1,000,000 times), so **don't call this command too often**!

12.6.5 Read Keypad Data

Command I²C: 0xD1, <read key> **Command SPI:** 0xD1, 0x00, <read key>	**Default:** -	**Can be remembered:** No

Reads the next byte from the keypad buffer. When executing this command via the SPI port, a dummy byte must be sent after the 0xD1 command! If there is no key data in the keypad buffer, 0 is returned. If there is key data in the keypad buffer, a key code from 'a' to 'p' is returned, depending on what key was pressed. For details on what keys return what values, see Chapter 7. “Keypad and I/O Configuration“.

The LCD2S has a 16 byte keypad buffer. Each time a key is pressed, it's code is added to the keypad buffer.

12.7 Input/Output Commands

12.7.1 Set OUT1 and OUT2

Command: 0xE2, [value]	**Default:** -	**Can be remembered:** Yes

Sets the value of OUT1 and OUT2 general purpose outputs. Bit 0 of the given value sets the state of OUT1. Bit 1 of the given value sets the state of OUT2. For example:
- 0 (0x00) will turn OUT1 and OUT2 off
- 1 (0x01) will turn OUT1 on, and OUT2 off
- 2 (0x02) will turn OUT1 off, and OUT2 on
- 3 (0x03) will turn OUT1 and OUT2 on

12.7.2 OUT1 On

Command: 0x38	**Default:** No	**Can be remembered:** Yes

Turns OUT1 output on. For this command to work, the LCD2S has to be configured to use the OUT1 general purpose output. For

details on configuration, see Chaper 7. “Keypad and I/O Configuration“.

12.7.3 OUT1 Off

Command: 0x30	**Default:** Yes	**Can be remembered:** Yes

Turns OUT1 output off. For this command to work, the LCD2S has to be configured to use the OUT1 general purpose output. For details on configuration, see Chaper 7. “Keypad and I/O Configuration“.

12.7.4 OUT2 On

Command: 0x39	**Default:** No	**Can be remembered:** Yes

Turns OUT2 output on. For this command to work, the LCD2S has to be configured to use the OUT2 general purpose output. For details on configuration, see Chaper 7. “Keypad and I/O Configuration“.

12.7.5 OUT2 Off

Command: 0x31	**Default:** Yes	**Can be remembered:** Yes

Turns OUT2 output off. For this command to work, the LCD2S has to be configured to use the OUT2 general purpose output. For details on configuration, see Chaper 7. “Keypad and I/O Configuration“.

12.7.6 GPIO1 On

Command: 0x48	**Default:** No	**Can be remembered:** Yes

Turns GPIO1 output on. For this command to work, GPIO1 has to be configured as a digital output:

- GPIO1 must be enabled via the “*Configure Keypad and IO*” command.
- GPIO1 must be configured as a digital output with the “*Configure GPIO1*” command.

12.7.7 GPIO1 Off

Command: 0x40	**Default:** No	**Can be remembered:** Yes

Turns GPIO1 output off. For this command to work, GPIO1 has to be configured as a digital output:

- GPIO1 must be enabled via the “*Configure Keypad and IO*” command.
- GPIO1 must be configured as a digital output with the “*Configure GPIO1*” command.

12.7.8 GPIO2 On

Command: 0x49	**Default:** No	**Can be remembered:** Yes

Turns GPIO2 output on. For this command to work, GPIO2 has to be configured as a digital output:

- GPIO2 must be enabled via the “*Configure Keypad and IO*” command.
- GPIO2 must be configured as a digital output with the “*Configure GPIO2*” command.

12.7.9 GPIO2 Off

Command: 0x41	**Default:** No	**Can be remembered:** Yes

Turns GPIO2 output off. For this command to work, GPIO2 has to be configured as a digital output:

- GPIO2 must be enabled via the “*Configure Keypad and IO*” command.
- GPIO2 must be configured as a digital output with the “*Configure GPIO2*” command.

12.7.10 Read GPIO1, GPIO2 and GPIO3 Inputs

Command I²C: 0xD3, <read key> **Command SPI:** 0xD3, 0x00, <read key>	**Default:** -	**Can be remembered:** No

Reads the current state of the GPIO1, GPIO2 and GPIO3 digital inputs. The value of GPIO1 is represented by bit 0 (right most bit - LSB), GPIO2 by bit 1, and GPIO3 by bit 2 of the returned byte. For example, if the returned byte is 00000101, than GPIO1 and GPIO3 are on (5V), and GPIO2 off (OV).

For this command to work, the desired GPIO ports must be configured as a digital inputs:

1. Use the "*Configure Keypad and IO*" command to configure what GPIO ports are to be used.
2. Use the "Configure GPIO1" and "Configure GPIO2" commands to set the desired GPIO ports as digital inputs.

After configuring the desired GPIO ports as digital inputs, their values can be read with this command.

13 Character Set

The LCD2S board has a standard English/European Character Set. This is shown below. CGRAM 1 to 8 are initialized with the bar graph characters! They can be customized with the “Load Custom Character Set“ command.

Figure 24 Character Set

14 Specifications

14.1 Absolute Maximum Ratings

The operating and storage temperature depends on the LCD display used. The LCD2S without and display

Item	Symbol	Min	Typ	Max	Unit
Operating Temperature of LCD2S without display	Topnd	-20		+85	°C
Storage Temperature of LCD2S without display	Tstnd	-55		150	°C

The standard operating temperature for an LCD display is 0°C to 50°C. The operating temperature for an wide operating temperature LCD display is -20°C to 70°C.

14.2 Electrical Characteristics

The following readings were done with a 2x16 line character LCD display, with an array LED backlight.

Item	Symbol	Condition	Min	Typ	Max	Unit
DC Supply Voltage:	Vdd	-	4.5		5.5	V
Typical Operating Current with backlight off	Idd	Vdd = 5V		4		mA
Typical Operating Current with backlight at 50%	Idd	Vdd = 5V		40		mA
Typical Operating Current with backlight at 100%	Idd	Vdd = 5V		76		mA

14.3 D.C. Characteristics of user I/O pins

Item	Symbol	Condition	Min	Typ	Max	Unit
OUT1 and OUT2 outputs	Iout		0		1000mA	mA
GPIO1 to GPIO3 Input Low Voltage	V_{IL}		0		1	V
GPIO1 to GPIO3 Input High Voltage	V_{IH}		2.05		5	V
GPIO1 and GPIO2 Output Low Voltage(1)	V_{OL}	VDD = 4.5V			0.6	V
GPIO1 and GPIO2 Output High Voltage(1)	V_{OH}	VDD = 4.5V	4.3			V
GPIO1 and GPIO2 Capacitive loading	C_{IO}			50		pF

14.4 D.C. Characteristics of I2C and SPI pins

Item	Symbol	Condition	Min	Typ	Max	Unit
Input Low Voltage - 0.2Vdd	V_{IL}	Vdd = 5V	Vss		1.0	V
Input High Voltage - 0.8Vdd	V_{IH}	Vdd = 5V	4.0		Vdd	V
Output Low Voltage	V_{OL}	IOL = 8.5 mA, VDD = 4.5V			0.6	V
Output High Voltage(2) - SPI mode only	V_{OH}	IOL = 8.5 mA, VDD = 4.5V	Vdd - 0.7			V

Note 1: The GPIO1 to GPIO3 pins have a 1kΩ resistor between CPU port pin and GPIO pin. This means they can not supply lots of current. For example, when set to output high (5V), and supplying a current of 1mA, the output will only be = 5V – (0.001Ax1000Ω) = 5 – 1 = 4V (1V voltage drop over output resistor).

Note 2: I^2C Outputs are open collector, meaning they do not output a high voltage. The bus is pulled high via external pull-up resistors.

14.5 SPI

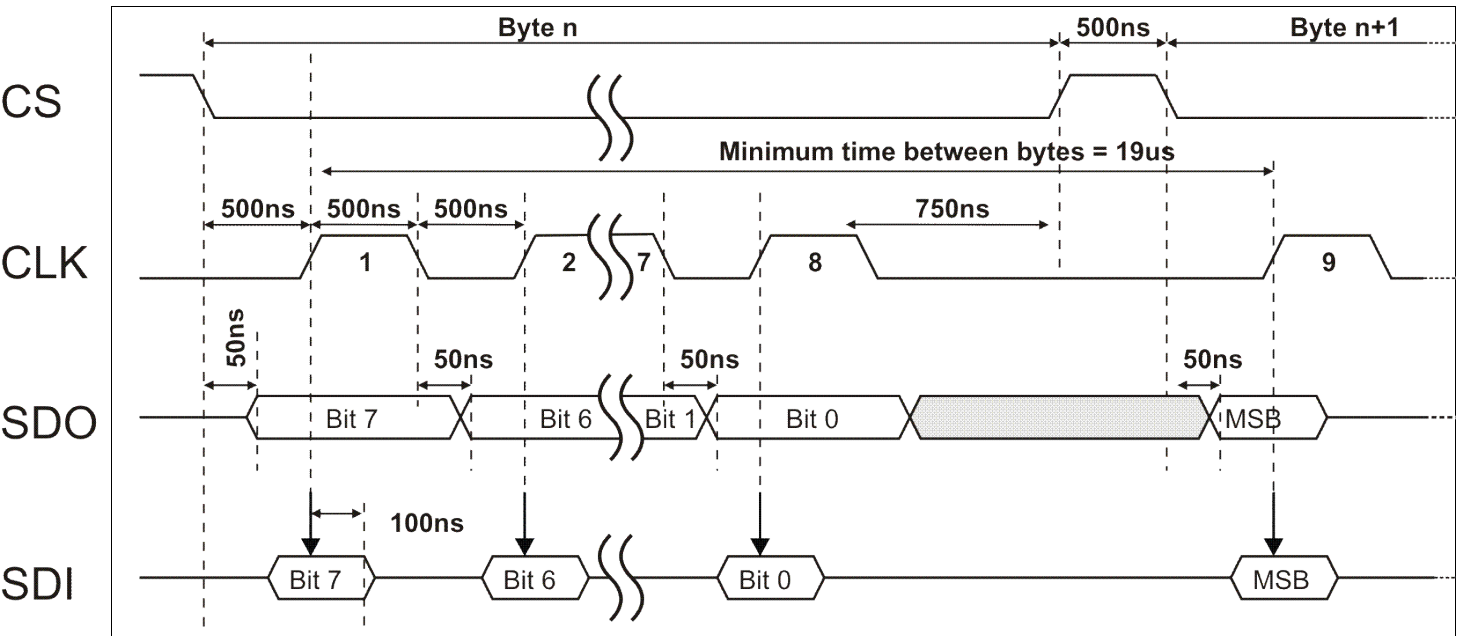


Figure 25: SPI Mode 0 with CS

15 Dimensions

The LCD2S dimensions as seen from the side are shown below.

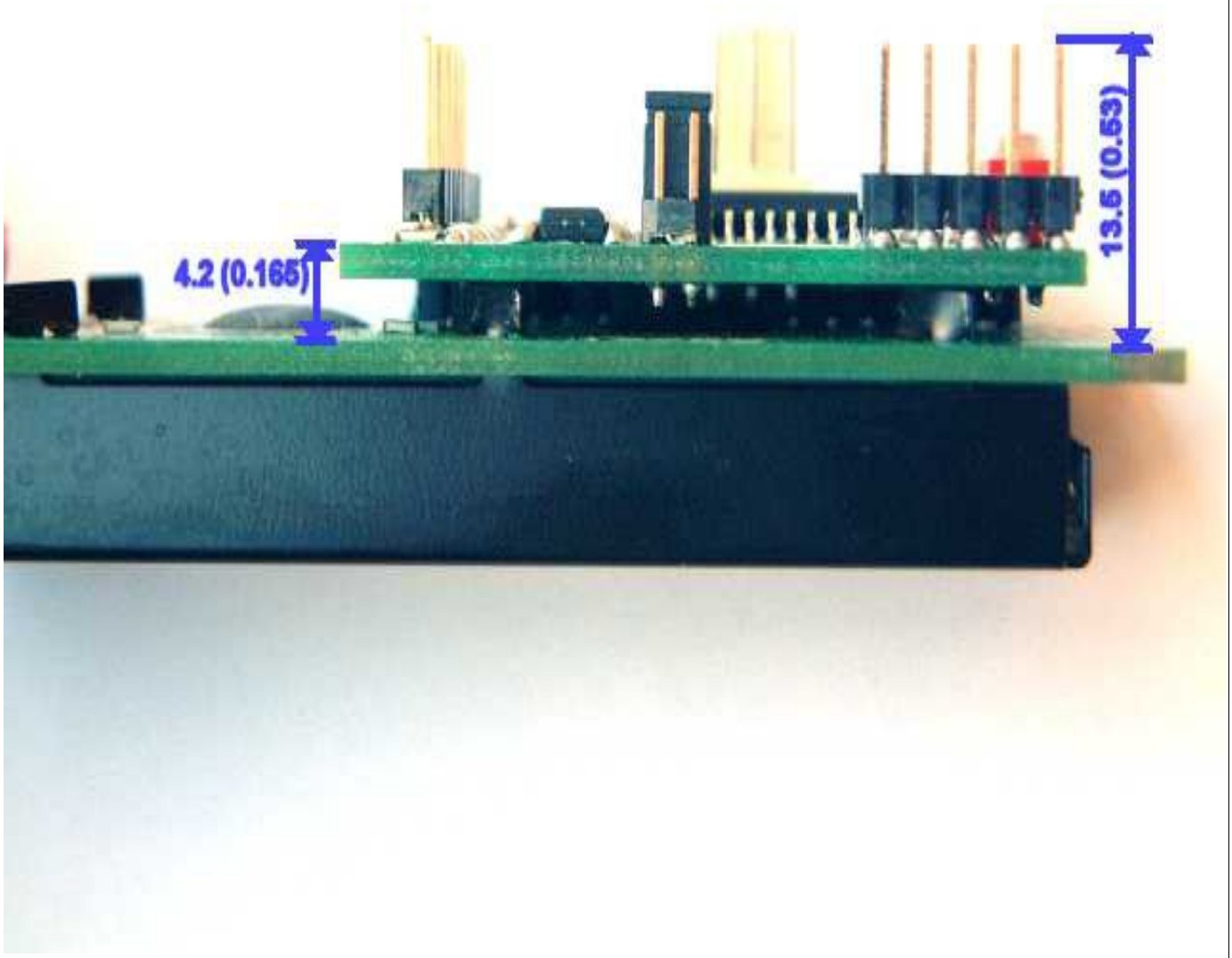


Figure 26