Volume

1

A-WIT TECHNOLOGIES INC.

… a passion for execution …

CS410000 LCD Display Reference Guide Manual

Version 1.5

A-WIT TECHNOLOGIES INC.

CS410000 LCD Display Reference Manual

Phone (800) 985-AWIT • Fax (800) 985-2948

Table of Contents

Chapter 1

Introduction to the CS410000 LCD Display

The CS410000 LCD Display requires only a 5 V power supply and the two connections related to transferring data to and from the LCD. Many useful text formatting functions are built into the operation of the LCD. The LCD also provides the capability of defining custom characters, which can be recalled at any later time during program execution as required. There is a 64 Byte FIFO buffer built into the LCD to ensure a minimum delay in writing to the display.

Registering Your C Stamp or C Stamp Related Product

At A-WIT Technologies we respect your privacy; however, we do ask you to register your C Stamp or C Stamp related product, so you can receive free of charge product updates. The registration procedure is simple. Just send an e-mail to tech_support@a-wit.com with the word "REGISTRATION x" in the subject line, where "x" is the product number that you purchased. If you purchased more than one product, send an e-mail for each different product.

Introduction to the CS410000 LCD Display

The CS410000 4 rows x 20 columns I^2C Serial LCD display with backlight and 3 rows x 4 columns keypad interface is a very functional and low-cost LCD that can be easily controlled by a C Stamp. The LCD display consists of four lines by 20 characters and provides basic text wrapping so that your text looks right on the display. The LCD display is compatible with the C Stamp microcomputer's supplies and signal levels. Communicating with the LCD is made easy with A-WIT's supplied software commands LCDCMD, LCDIN, and LCDOUT that use the I^2C capability of the C Stamp. This simple three command interface is all that is required to communicate with the LCD. In addition, this I^2C Serial LCD also provides you with full control over all of its advanced LCD features, allowing you to move the cursor anywhere on the display with a single instruction and turn the display on and off in any configuration.

The CS410000 LCD display requires only a 5 V power supply and the two connections related to transferring data to and from the LCD via the I^2C protocol. Many useful text formatting functions are built into the operation of the LCD, as described with the software functions used to communicate with the LCD. The LCD also provides the capability of defining up to eight custom characters, which can be recalled at any later time during program execution as required. There is a 64 Byte FIFO buffer built into the LCD to ensure a minimum of delay in writing to the display. With this display and the C Stamp users can design a professional-looking text user interface on any microcontroller application, supply an easy-to-use serial debugging interface that does not require a PC, and provide real-time sensor data output on autonomous robotics applications.

FIFO

Because the I^2C communication interface operate faster than the display can accept data, all commands and text you send to the display are placed in the FIFO (First In, First Out) buffer. This data is sent to the display as fast as it will accept it. The FIFO is 64 bytes in length. When sending lots of text to the display, you should check the timing during your application implementation so you don't overflow the buffer. If the buffer does overflow, the excess bytes are ignored.

I^2C Bus Operation

The I^2C LCD display is located on the I^2C bus at address 0xC6. The SCL and SDA lines should have pull-up resistors on them somewhere on the bus. The value of these resistors should be anything from 1.3 KΩ to 10 KΩ. You only require 1 pair of resistors for the whole I^2C bus, not specifically for the LCD. These resistors are already supplied in the C Stamp that acts as a master controller for the LCD, so any other device that connects to the I^2C bus from the C Stamp does not need to have these resistors. The C Stamp has 1.3 KΩ resistors to provide the best noise immunity. The jumper on the back of the LCD should be removed.

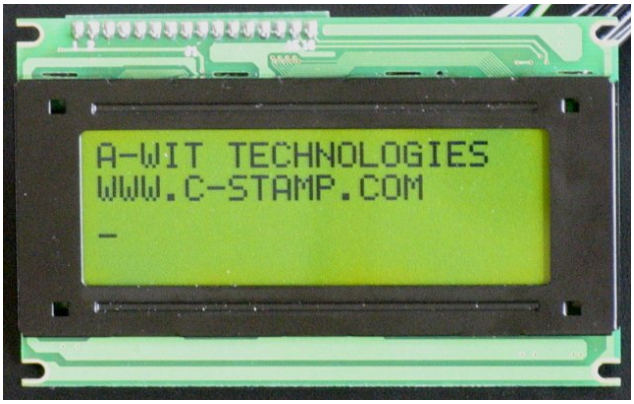
Keypad Input

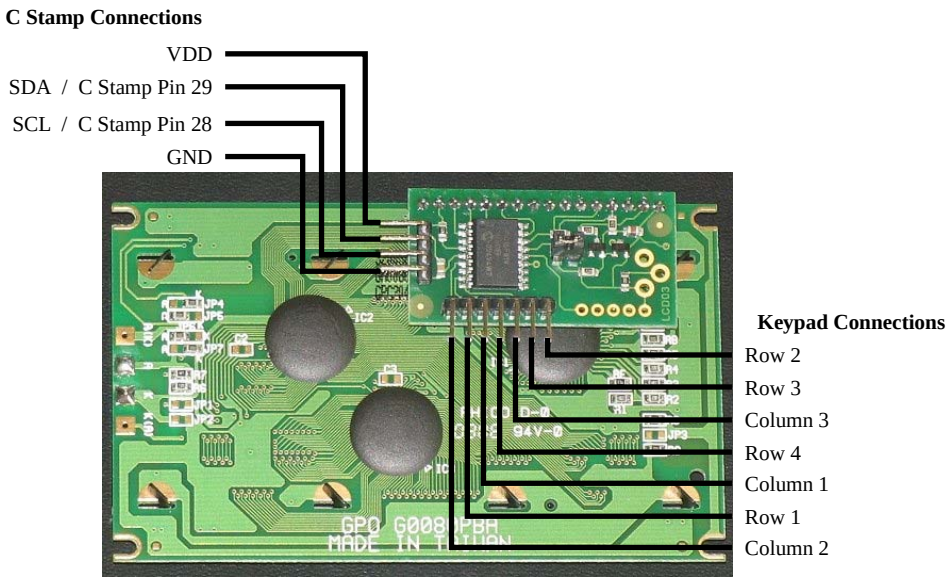
An added feature of the LCD module is the ability to connect to a 4 Rows x 3 Columns Keypad; such as the CS410001. The module will automatically scan the status of the keys at regular intervals. The result is then provided via the C Stamp LCDIN command.

Connectivity and More Information

More information on fully interfacing with the LCD in a very easy manner via the A-WIT supplied functions is provided in Chapter 3, where these functions are described in detail. The jumper on the back of the LCD should be removed.

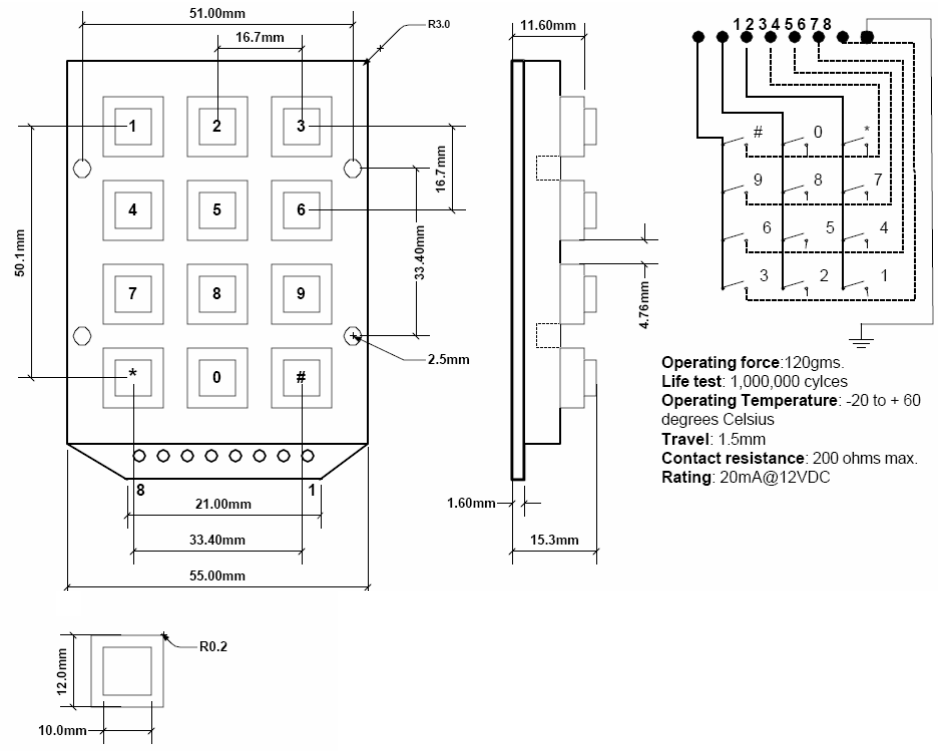
The following figures describe the LCD and its pin-out connectivity.





Keypad

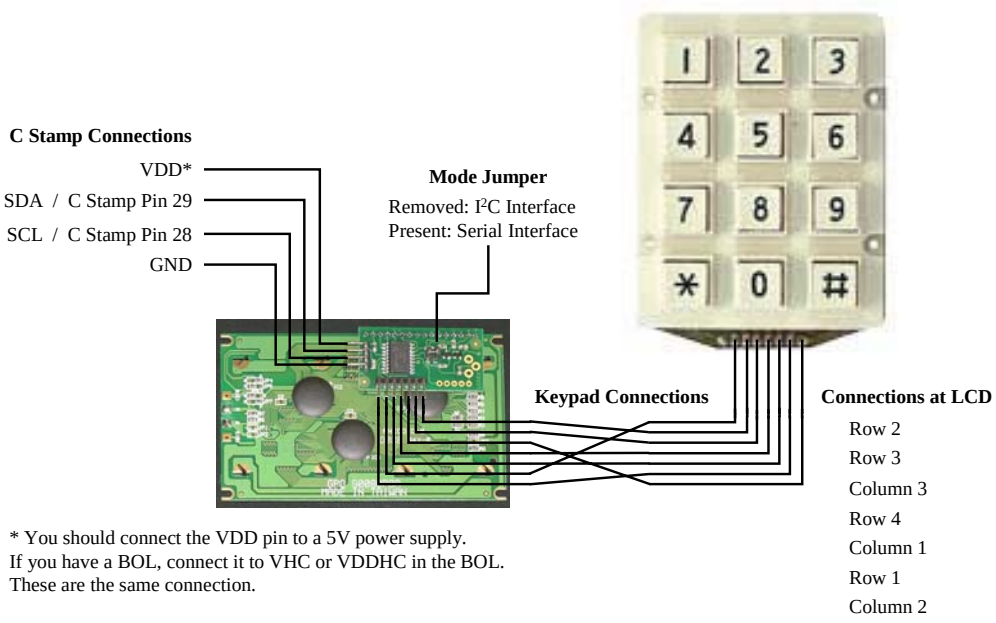
The following figures describe the CS410001 Keypad that can be interfaced with the CS410000 LCD module, and its pin-out connectivity.



Keypad Pin-Out Connectivity	
Pin	***Description***

Keypad Pin-Out Connectivity	
Pin	***Description***
1	Column 3
2	Column 2
3	Column 1
4	Row 4
5	Row 3
6	Row 2
7	Row 1
8	GND (optional connection)

The following figure clarifies further the connectivity between the C Stamp, the LCD, and the Keypad.



Chapter

2

Getting Started

This chapter is a quick start guide to using the CS410000 LCD and the CS410001 keypad with the C Stamp. This assumes you have a C Stamp and an appropriate connection kit or development board with the RESET and START circuitry, the LCD properly connected to the C Stamp, and the keypad properly connected to the LCD module. You will also need a programming cable, power supply, PC running Windows® 2000/XP/Media, with a quantity of RAM recommended for the OS, sufficient free hard disk drive space for the software installations, CD-ROM drive, Internet access (recommended only), and available port compatible with your programming cable.

Notices

DAMAGES OR OTHER LIABILITY ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR FIRMWARE OR THE USE OF OTHER DEALINGS IN THE SOFTWARE OR FIRMWARE.

Getting Support

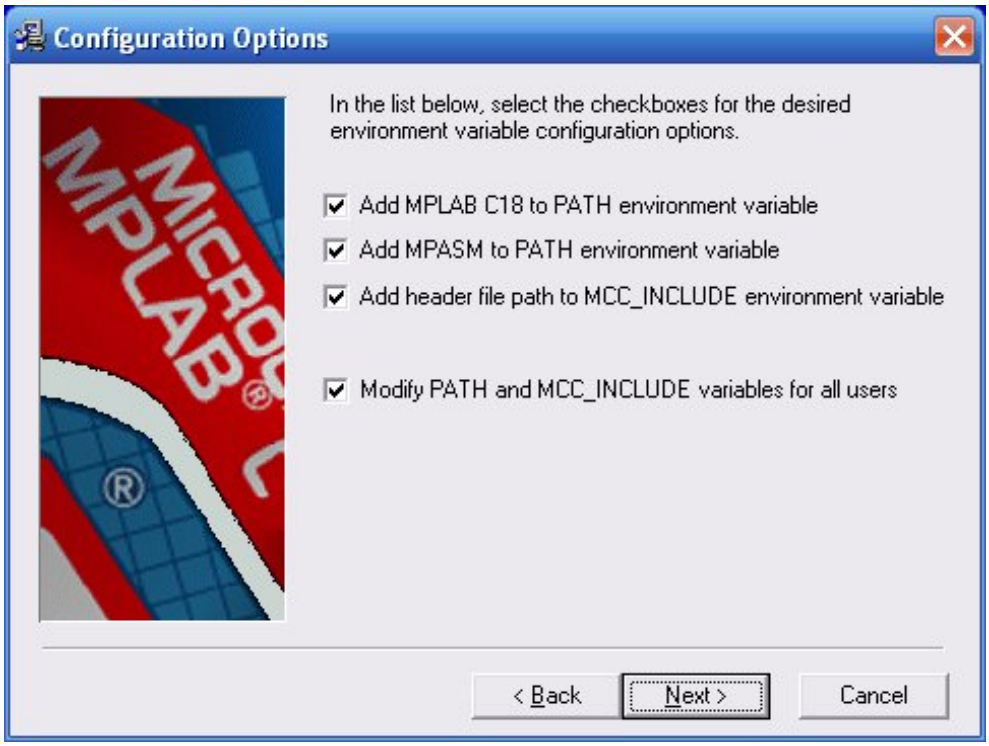
If possible, please check the C Stamp website www.c-stamp.com under SUPPORT for any updates to documentation, changes, or notices that may have become available since your Installation CD was produced. If you continue to have any issues for which a solution is not found in the aforementioned website, please e-mail tech_support@a-wit.com for help.

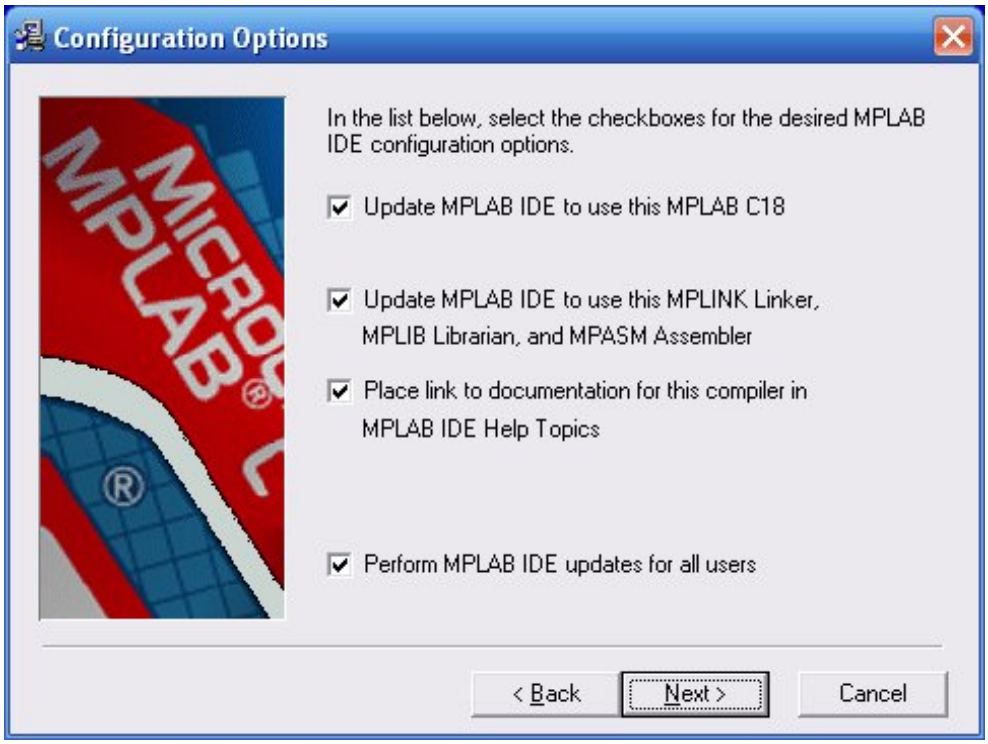
Installing the Microchip MPLAB and C Compiler Software

The first step is to install the Microchip MPLAB software that you will use to develop your programs.

Insert your A-WIT provided Installation CD in your CD drive. Go to the MPLAB directory in the CD and double click on the "MPLAB vX.XX Install" file in that directory. Follow the installation steps, prompts, and directions provided by the installer software, accepting all the default options.

After the MPLAB installation is complete, switch to the C18 directory in the CD, and double click on the file in that directory. Follow the installation steps, prompts, and directions provided by the installer software, accepting all the default options. The only exceptions to accepting all the default options is that on the 5th and 6th windows of the installation process for the C18 Compiler, you have to select everything as shown in the figures below. This will ensure that MPLAB is configured to use the C18 Compiler.


Configuration Options
In the list below, select the checkboxes for the desired environment variable configuration options.
Add MPLAB C18 to PATH environment variable
Add MPASM to PATH environment variable
Add header file path to MCC_INCLUDE environment variable
Modify PATH and MCC_INCLUDE variables for all users
< Back
Next >
Cancel


Configuration Options
In the list below, select the checkboxes for the desired MPLAB IDE configuration options.
Update MPLAB IDE to use this MPLAB C18
Update MPLAB IDE to use this MPLINK Linker, MPLIB Librarian, and MPASM Assembler
Place link to documentation for this compiler in MPLAB IDE Help Topics
Perform MPLAB IDE updates for all users
< Back
Next >
Cancel

Installing the A-WIT C Stamp Quick Programmer

To install the A-WIT C Stamp Quick Programmer, switch to the CSTAMPQP directory in the CD using Windows Explorer, and double click on the file in that directory. Follow the installation steps, prompts, and directions provided by the installer software, accepting all the default options.

Installing the USB Software

If you purchased a product with a USB download cable, make sure that the A-WIT provided CD is in the CD drive of your PC and insert the USB cable in the USB port of your PC. Windows auto detects the new USB device. If Windows prompts you to install drivers for the USB cable device, follow the installation steps, prompts, and directions provided by the installer software, accepting all the default options.

After the USB adapter has been installed, open a Windows Explorer window from the Accessories sub-menu in the Start menu, and right click on My Computer. Proceed to select Properties, and then select the Hardware tab. Click on the Device Manger button, and expand the Ports (COM & LPT) branch. Make a note of the COM port that has been assigned to the USB-to-Serial adapter. This is the port that should be selected in the C Stamp programmer software.

Setting Up the C Stamp Software Templates

To set up the C Stamp Software Templates, switch to the CSTAMP_Template directory in the CD using Windows Explorer, and double click on the file in that directory. Follow the installation steps, prompts, and directions provided by the installer software, accepting all the default options.

Documentation

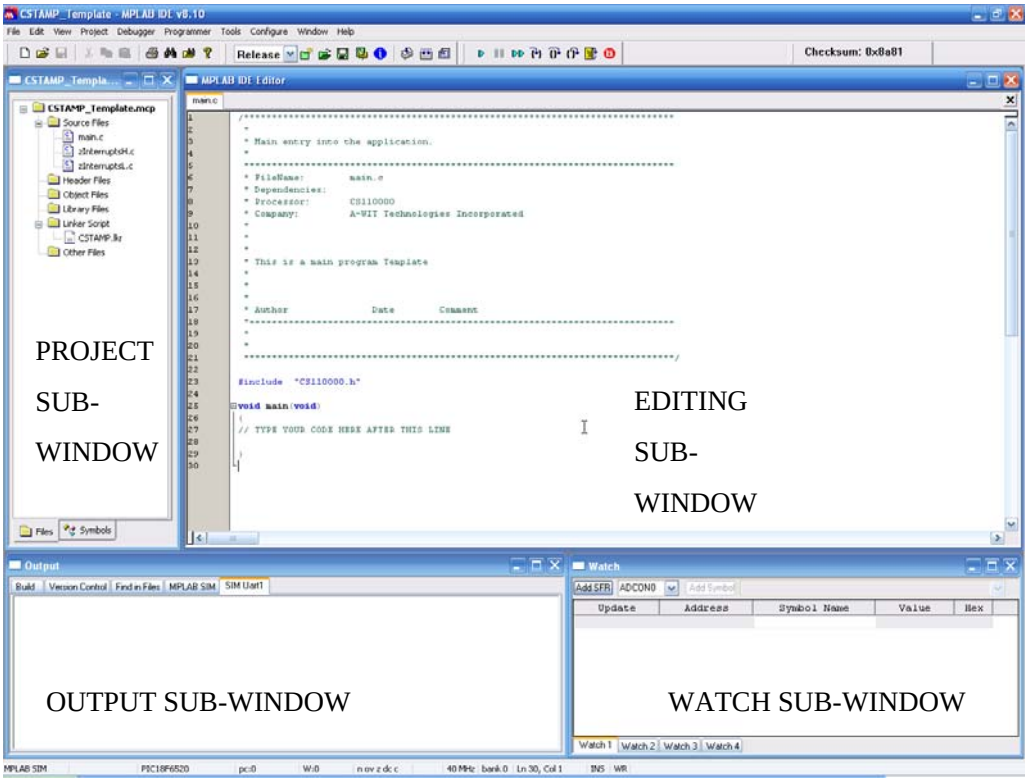
Copy the DOCS directory from the C Stamp Installation CD to your C:\A-WIT directory. This directory contains all the C Stamp related documentation in PDF format.

Creating your LCD/Keypad C Stamp Program

Create a directory where you want to have all the files for your program; for example UI_APP. We recommend making this directory under your C:\A-WIT directory, so you can have all your CSTAMP related files in one place.

Copy the all files in your C Stamp Software Templates directory C:\A-WIT\CSTAMP_Template to the directory you just made.

Open the Microchip MPLAB IDE application. As shown the following figure, the IDE has several sub-windows. Depending on the resolution of your screen, your sub-windows may have a different layout. However; you can move and resize these into the position that you want to fit your screen, and your layout for that particular project will get saved upon answering yes to the prompt of saving the workspace when you exit the software development environment.



Go to the "File" menu to "Open Workspace…". Then navigate to your program directory, and open CSTAMP_Template.mcw.

Right click on CSTAMP_Template.mcp in the "Project" sub-window, and "Save as…" the name of your program project after you have navigated to your program

directory. For example, your program project could be named "UI_APP". Now when you open the Microchip MPLAB IDE (Integrated Development Environment), and go to your program directory to open the workspace for your program, you will see a .mws file with the name of your program preceding it. This is the file that you should open any time you want to work on your program.

Double click on the main.c source and type the following code fragment where it is indicated. You can omit the comments for brevity, as they are written here to offer clarifications of what the code does. Do pay attention; however, to the indentation of the code blocks between curly brackets for loops, if statements, etc. Although indenting the code is not a requirement for the compiler to parse your code (i.e. any blank spaces are ignored by the compiler), it does help tremendously to make your code much more readable, and consequently, it makes finding any errors easier. Keywords and function names in the code fragment below are bolded.

After you START the C Stamp in user mode as explained in the "Downloading and Running Your Program" section (this will not be the RESET/BOOT/DOWNLOAD mode), the program will run. This program assumes that your LCD and keypad are connected. The program will display the key that is pressed in the LCD. The program executes indefinitely until you restart it by pushing and releasing the RESET button while holding the START button and then letting go of the latter.

```
// Declare some necessary variables
  NIBBLE key;
  BYTE k0[]="0"; BYTE k1[]="1"; BYTE k2[]="2";
  BYTE k3[]="3"; BYTE k4[]="4"; BYTE k5[]="5";
  BYTE k6[]="6"; BYTE k7[]="7"; BYTE k8[]="8";
  BYTE k9[]="9"; BYTE k10[]="*"; BYTE k11[]="#";

// Initialize key to something else
  key=12;

  while(1){
// Get key
    key = LCDIN_CS410000();

// Display key
    switch(key){
      case 0:  LCDOUT_CS410000(k0, 1); break;
      case 1:  LCDOUT_CS410000(k1, 1); break;
      case 2:  LCDOUT_CS410000(k2, 1); break;
      case 3:  LCDOUT_CS410000(k3, 1); break;
      case 4:  LCDOUT_CS410000(k4, 1); break;
      case 5:  LCDOUT_CS410000(k5, 1); break;
      case 6:  LCDOUT_CS410000(k6, 1); break;
```

```
      case 7:  LCDOUT_CS410000(k7, 1); break;
      case 8:  LCDOUT_CS410000(k8, 1); break;
      case 9:  LCDOUT_CS410000(k9, 1); break;
      case 10: LCDOUT_CS410000(k10, 1); break;
      case 11: LCDOUT_CS410000(k11, 1); break;
    }
  }
```

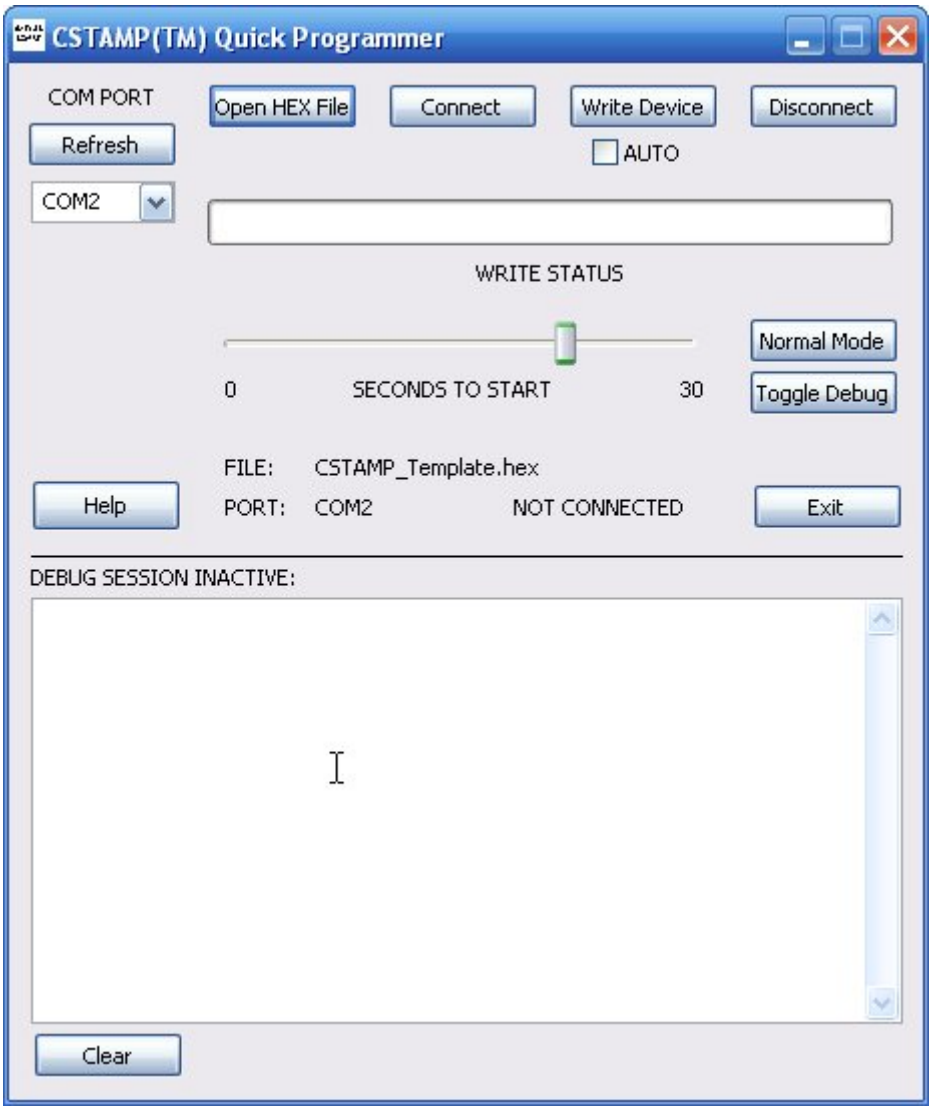
Save your program from the "File" menu or by clicking on the appropriate icon in the tool bar. Then "Build All" from the "Project" menu or from the tool bar.

If the code was typed correctly, you will have a file in your program directory with the name of your program project and a .HEX extension. An example is UI_APP.HEX. This is the file that you will download to the C Stamp, as explained up ahead.

If you get an error message or an indication that your program did not build successfully in the "Output" sub-window of the IDE, you probably have one or more syntax error(s). Double click on the line of the "Output" sub-window that mentions the error, and the program line that most likely contains the error will be indicated in the sub-window where you were editing your program. Correct as necessary, and "Build All" again until you get a successful .HEX file output.

Downloading and Running Your Program

Power up your KIT, and connect the KIT to the PC with the provided cable. Upon power up, the C Stamp will be in RESET/BOOT/DOWNLOAD mode. To go back to this mode at any time, just push and let go of the RESET button. Then open the A-WIT C Stamp Quick Programmer application shown in the next figure.



The first step is to choose the serial port that you are using from the drop-down menu. Then click on "Refresh", so that the program registers your selection. Your selection should show in the status area of the program next to "PORT:". Then click on "Open HEX File" and load/select the HEX file that you had previously created during the

development of your program. The status are should indicate that the file has been loaded successfully. This is what will be downloaded to the C Stamp. Then click on "Connect", and the PC will be connected to the KIT, and the status area should indicate so. To download the HEX file to the C Stamp, just click on "Write Device", and you should see the progress bar after a few seconds, as the HEX file is downloaded. At this point, you can click on "Disconnect" to disconnect the PC from the KIT, disconnect the serial cable from both the PC and the KIT, and start your program manually at the KIT. To do this just push and let go of the RESET button while pushing the START button. Then you can let go of the START button. Alternatively, you can click on "Normal Mode" to start your program from the PC. This will also disconnect the program/PC from the KIT. Then you can disconnect the serial cable from the PC and the KIT. You can also instruct the CSTAMP™ Quick Programmer to wait several seconds before starting your program from the PC and disconnecting by adjusting the "SECONDS TO START" slide. This feature is useful in case you want to keep the PC connected with the serial cable, but need time to manually set up something in a circuit that you have built. If this is not the case it can just be left at the default of "0", and your program will started from the PC right away. After you click "Normal Mode" and your program is started, the CSTAMP™ Quick Programmer will not be communicating with the C Stamp any longer, so if you want to reconnect, you must click on "Connect" again.

Chapter 3

Accessory Specific Functions and Commands Reference

This chapter describes the functions and commands that are specific to the software support of different types of accessories that are available from A-WIT Technologies to complement the function and projects developed with the C Stamp. The user should consult the manual for a specific accessory for full information on connectivity and usage.

LCDCMD_CS410000

```
BIT LCDCMD_CS410000(BYTE command, BYTE B0, BYTE B1,
                    BYTE B2, BYTE B3, BYTE B4,
                    BYTE B5, BYTE B6, BYTE B7);
```

The **LCDCMD** function sends a command to the CS410000 LCD display according to the table below. If the function is successful, it returns **TRUE**; otherwise, it returns **FALSE**. This could mean that there was an error in the arguments of the function or some other problem.

command is a variable/constant/expression (1 – 17) indicating the LCD command to send.

B0 – B7 are byte variables/constants/expressions that are used by different commands. Not all bytes are used by all commands. If a byte is not used by a command, any value can be passed to the function (e.g. **ZERO**), and it will be ignored. However; something always must be passed in all the arguments. If the usage of a byte is not specified in the table below, then that byte is not used and ignored by the function.

Symbol	*Value*	*Command*	*Description*

Symbol	*Value*	*Command*	*Description*
LCD_HOME	1	Cursor Home	Sets the cursor to the home position (top left).
LCD_SETC	2	Set Cursor (1 – 80)	Set cursor to a position specified by **B0**, where 1 is the top left and 80 is the bottom right.
LCD_SETCLC	3	Set Cursor (line, column)	Sets cursor using two bytes, where **B0** is the line and **B1** is the column.
LCD_HIDEC	4	Hide Cursor	Stops the position cursor from appearing on the display.
LCD_UNDLC	5	Show Underline Cursor	Changes the cursor to the underline type.
LCD_BLNKC	6	Show Blinking Cursor	Changes the cursor to the blinking type.
LCD_BACKS	7	Backspace	Deletes the preceding character from the current position on the display.
LCD_HTAB	8	Horizontal Tab (by Tab Set)	Moves the current position across by the tab space set by command **LCD_SETTA** (default tab space is 4).
LCD_SLF	9	Smart Line Feed	Moves the cursor down one line to the position beneath it in the same column.
LCD_VTAB	10	Vertical Tab	Moves the cursor up one line to the position above it in the same column.
LCD_CLRS	11	Clear Screen	Clears the screen and sets the cursor to the home position.
LCD_CR	12	Carriage Return	Moves the cursor to the start of the next line.
LCD_CLRC	13	Clear Column	Clears the contents of the current column and moves the cursor right by one column.

Symbol	*Value*	*Command*	*Description*
LCD_SETTA	14	Tab Set	Sets the tab size to **B0**. **B0** can be a size between 1 and 10.
LCD_BLON	15	Backlight On	Turns the backlight of the LCD on.
LCD_BLOFF	16	Backlight Off (default)	Turns the backlight of the LCD off.
LCD_CCHAR	17	Custom Character	Sends one of 8 possible custom characters to the LCD to be used later. **B0 – B7** define the custom character. See below.
LCD_DCCHAR	18	Display Custom Character	Displays a custom character at the current cursor position. **B0** has a value 0 – 7 that specifies which one of the custom characters is to be displayed.

Up to 8 custom characters can be stored in the LCD for subsequent usage by sending an 8 byte map in bytes **B0 – B7** with the LCD command **LCD_CCHAR**. The example below shows how to define a byte map for a custom character.

Byte	*Char #*	*Bits of* **B0 – B7**								*Byte in Binary*	*Byte in Hex*
		Char # in Binary			*Char Map*						
B0	2	0	1	0	0	0	0	0	0	0100 0000	0x40
B1	2	0	1	0	0	0	1	0	0	0100 0100	0x44
B2	2	0	1	0	0	1	1	1	0	0100 1110	0x4E
B3	2	0	1	0	1	0	1	0	1	0101 0101	0x55
B4	2	0	1	0	0	0	1	0	0	0100 0100	0x44
B5	2	0	1	0	0	0	1	0	0	0100 0100	0x44
B6	2	0	1	0	0	0	1	0	0	0100 0100	0x44

Byte	*Char #*	*Bits of* **B0 – B7**								*Byte in Binary*	*Byte in Hex*
		Char # in Binary			*Char Map*						
B7	2	0	1	0	0	0	0	0	0	0100 0000	0x40

The bytes **B0 – B7** defined above would store an up-arrow as custom character # 2. To store this character in the LCD, we would call **LCDCMD** as:

```
result = LCDCMD_CS410000(LCD_CCHAR, 0x40, 0x44, 0x4E,
                         0x55, 0x44, 0x44, 0x44,
                         0x40);
```

Checking the result of calling the function is optional, although it is a good programming practice. Also, decimal values could be sent for **B0 – B7**, instead of Hexadecimal; so an alternative way of calling **LCDCMD** is:

```
LCDCMD_CS410000(LCD_CCHAR, 64, 68, 78, 85, 68, 68, 68,
                64);
```

Then to display this character at the current cursor position, we would call:

```
result = LCDCMD_CS410000(LCD_DCCHAR, 2, 0, 0, 0, 0, 0,
                         0, 0);
```

or

```
LCDCMD_CS410000(LCD_DCCHAR, 2, 0, 0, 0, 0, 0, 0, 0);
```

LCDIN_CS410000

```
NIBBLE LCDIN_CS410000(void);
```

The **LCDIN** function receives data from the CS410000 LCD display. This command supports the feature of the CS410000 LCD to host a standard 3 x 4 Matrix keypad, like the CS410001. **LCDIN** returns the number corresponding to the last key that is being pressed in the keypad (0 – 9, 10 for the * key, and 11 for the # key). If no key is being pressed or there is an error, the function returns 12.

LCDOUT_CS410000

```
BIT LCDOUT_CS410000(BYTE string[], BYTE n);
```

The **LCDOUT** sends ASCII bytes to the CS410000 LCD for displaying. Displaying of these characters will start at the current cursor position. If the function is successful, it returns **TRUE**; otherwise, it returns **FALSE**.

string is an array of the ASCII bytes to be displayed.

n is a variable/constant/expression that specifies how many bytes are in **string**.

Terms & Conditions

Quality Assurance

A-WIT has stringent quality control procedures in place to insure the best quality products.

90-Day Limited Warranty

A-WIT Technologies, Inc warrants its products against defects in materials and workmanship for a period of 90 days. If you discover a defect, A-WIT Technologies, Inc. will, at its option, repair, replace, or refund the purchase price. After 90 days, products can still be sent in for repair or replacement, but there will be a $10.00USD minimum inspection/labor/repair fee (not including return shipping and handling charges).

14-Day Money-Back Guarantee

If, within 14 days of having received your product, you find that it does not suit your needs, you may return it for a refund. A-WIT will refund the purchase price of the product in the form of a check, excluding shipping/handling costs, once the product is received. This refund does not apply if the product has been altered or damaged. If you decide to return the products after the 14-day evaluation period, a 20% restocking fee will be charged against a credit.

Disclaimer

Warranty does not apply if the product has been altered, modified, or damaged. A-WIT makes no other warranty of any kind, expressed or implied, including any warranty of merchantability, fitness of the product for any particular purpose even if that purpose is known to A-WIT, or any warranty relating to patents, trademarks, copyrights or other intellectual property. A-WIT shall not be liable for any injury, loss, damage, or loss of profits resulting from the handling or use of the product shipped.

How to Return a Product

When returning, you must first e-mail sales@a-wit.com for a Return Merchandise Authorization number. No packages will be accepted without the RMA number clearly marked on the outside of the package. After inspecting and testing, we will return your product, or its replacement using the same shipping method used to ship the product to A-WIT within 30 days. In your package, please include a daytime telephone number and a brief explanation of the problem.

Please contact our Sales Department at sales@a-wit.com if you have any questions regarding our warranty policy or if you are requesting an RMA number.