

Volume

1

A-WIT TECHNOLOGIES INC.

... a passion for execution ...

CS410100 VFD Display Reference Guide Manual

Version 1.1

A-WIT TECHNOLOGIES INC.

CS410100 VFD Display Reference Manual

© A-WIT Technologies Inc.
Phone (800) 985-AWIT • Fax (800) 985-2948

Table of Contents

Registering Your C Stamp or C Stamp	
Related Product	1
Introduction to the CS410100 VFD Display	1
Connectivity and More Information	2
Notices	3
Getting Support	4
Installing the Microchip MPLAB and C	
Compiler Software	4
Installing the A-WIT C Stamp Quick	
Programmer	4
Installing the USB Software	6
Setting Up the C Stamp Software	
Templates	6
Documentation	6
Creating your VFD C Stamp Program	7
Downloading and Running Your Program	9
VFDCMD_CS410100	11
VFDOUT_CS410100	18
Terms & Conditions	19

Introduction to the CS410100 VFD Display

The CS410100 VFD Display requires only a 5 V power supply and the two connections related to transferring data to and from the VFD. Many useful text formatting functions are built into the operation of the VFD. The VFD also provides the capability of defining custom characters, displaying images, and playing animations.

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@awit.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.

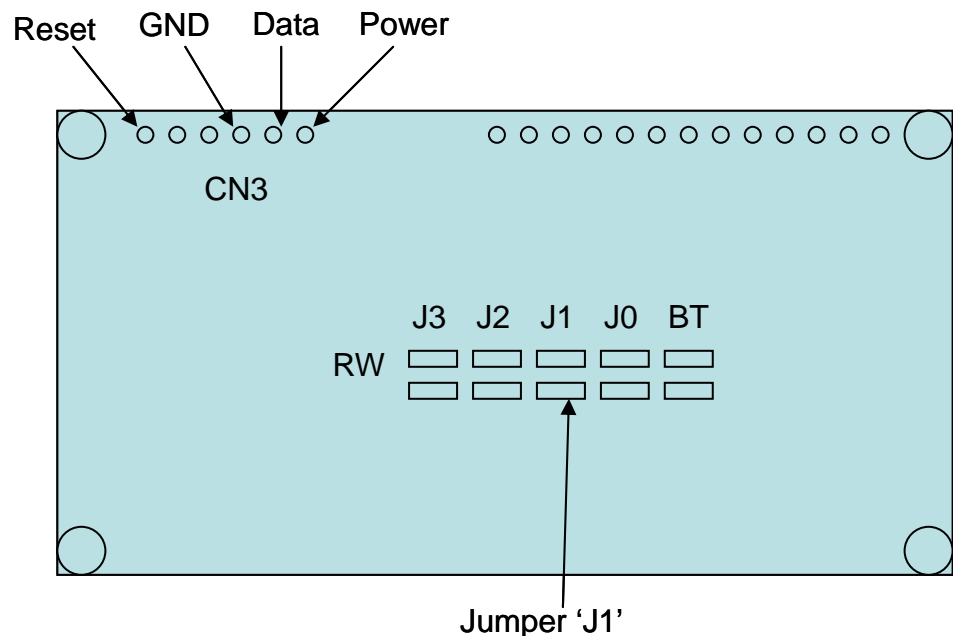
Introduction to the CS410100 VFD Display

The CS410100 2 rows x 16 columns CMOS Asynchronous Serial VFD display is a very functional and low-cost VFD that can be easily controlled by a C Stamp. The VFD display consists of two lines by 16 characters and provides basic text wrapping so that your text looks right on the display. The VFD display is compatible with the C Stamp microcomputer's supplies and signal levels. Communicating with the VFD is made easy with A-WIT's supplied software commands VFDCMD, and VFDOUT that use the CMOS Asynchronous capability of the C Stamp. This simple two command interface is all that is required to communicate with the VFD. In addition, this CMOS Asynchronous Serial VFD also provides you with full control over all of its advanced VFD 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 CS410100 VFD display requires only a 5 V power supply and the two connections related to

transferring data to and from the VFD. Many useful text formatting functions are built into the operation of the VFD, as described with the software functions used to communicate with the VFD. The VFD also provides the capability of defining up to sixteen custom characters, which can be recalled at any later time during program execution as required. 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. The VFD also allows for the display of images and animations through the A-WITT supplied functions.

Connectivity and More Information

More information on fully interfacing with the VFD in a very easy manner via the A-WITT supplied functions is provided in Chapter 3, where these functions are described in detail. Prior to operation a few connections must be made on the VFD module before operation. The two pads, top and bottom, of jumper 'J1' on the back of the VFD must be soldered together. Ensure this connection has been made before operation with the A-WITT supplied functions. Pins 1, 2, 3 and 6 from 'CN3' will be used to interface the VFD to the CStamp. Pin 1 is the power pin, pin 2 is the data pin, pin 3 is the ground pin, and pin 6 is the reset pin. Jumper wires may be soldered to these pins to easily connect the VFD module to the C-Stamp, BOL and breadboard. The image below displays the location of these four pins, and the jumper J1, on the back of the VFD module.



Getting Started

This chapter is a quick start guide to using the CS410100 VFD 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 VFD properly connected to the C Stamp. 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

CSTAMP™ and CSTAMP™ Related Hardware Products, Software Products and Documentation are developed and distributed by A-WIT Technologies, Inc. All rights reserved by A-WIT Technologies, Inc. A-WIT SOFTWARE OR FIRMWARE AND LITERATURE IS PROVIDED “AS IS,” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL A-WIT BE LIABLE FOR ANY CLAIM, 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.

MPLAB C-18 and MPLAB C-18 Users Guide is reproduced and distributed by A-WIT Technologies, Inc. under license from Microchip Technology Inc. All rights reserved by Microchip Technology Inc. MICROCHIP SOFTWARE OR FIRMWARE AND LITERATURE IS PROVIDED “AS IS,” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL MICROCHIP BE LIABLE FOR ANY CLAIM, 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

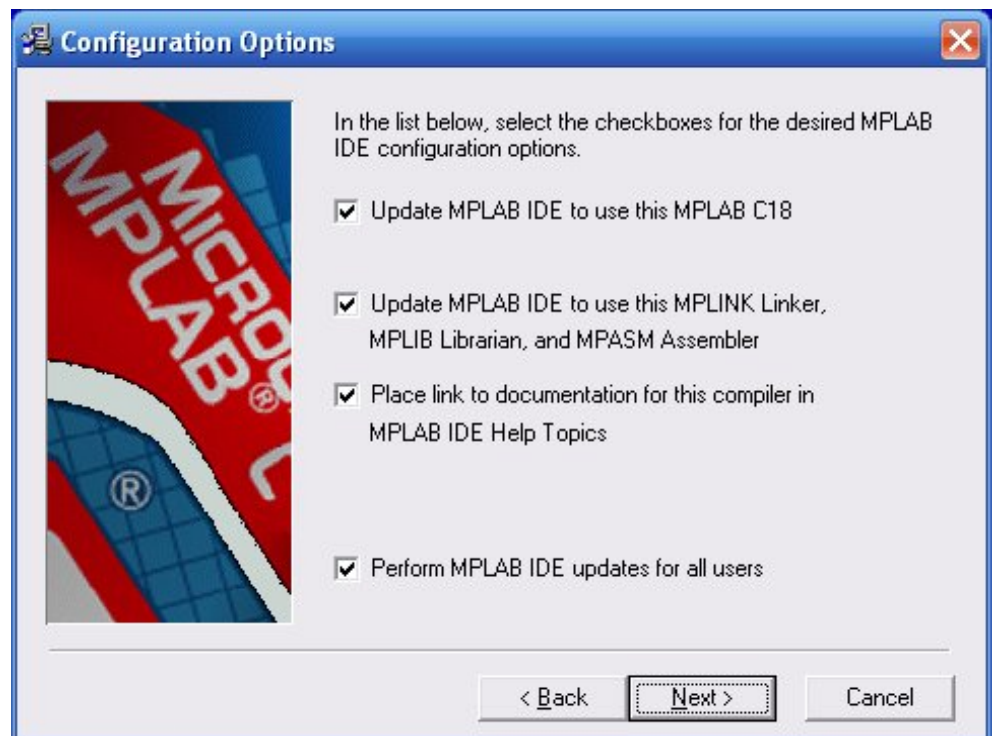
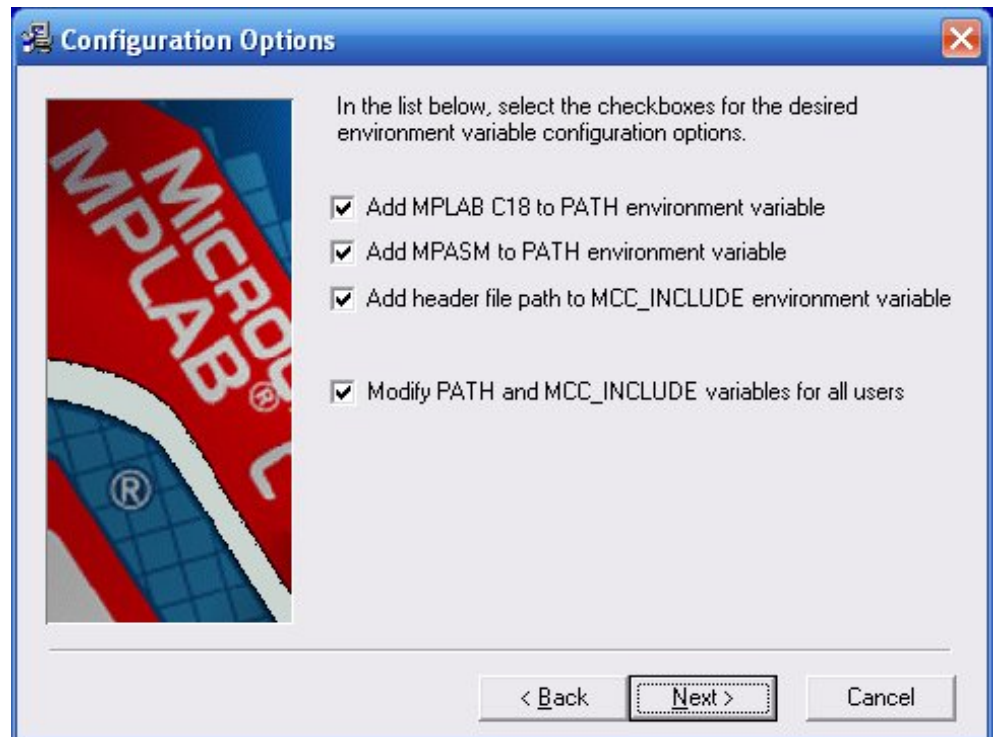
If possible, please check the C Stamp website www.c-stamp.com under SUPPORT for any updates to documentation, changes, or notices that may have become available since your Installation CD was produced. If you continue to have any issues for which a solution is not found in the aforementioned website,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



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

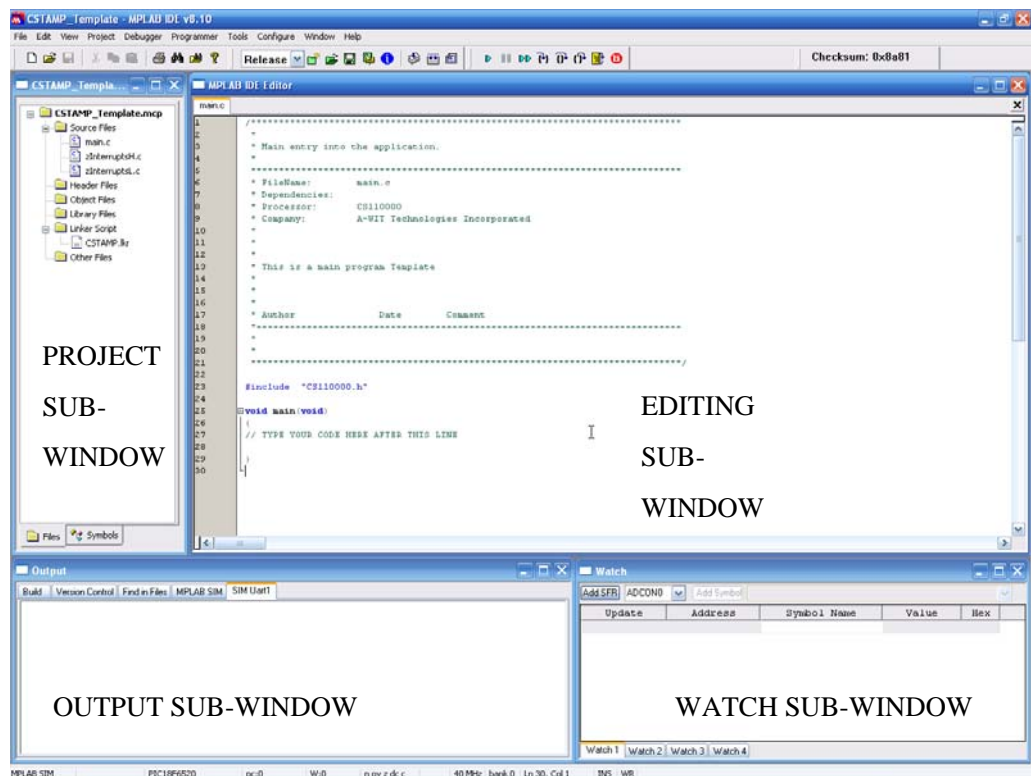
Copy the DOCS directory from the C Stamp Installation CD to your C:\A-WIT directory. This directory contains all the C Stamp related documentation in PDF format.

Creating your VFD C Stamp Program

Create a directory where you want to have all the files for your program; for example UL_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 VFD is connected to the CStamp with the data pin being pin 22 and the reset pin being 23. The program will display the arrow image, as described in the image explanation section, at position 8 on the screen. The program executes once, and the arrow remains on the VFD display until you reset, or restart, the VFD module.

```
BYTE Image_Array[24] = {2, 11, 0x04, 0x00, 0x0C,
                        0x00, 0x1C, 0x00, 0x3F, 0xFF, 0x7F, 0xFF, 0xFF,
                        0xFF, 0x7F, 0xFF, 0x3F, 0xFF, 0x1C, 0x00, 0x0C,
                        0x00, 0x04, 0x00};
BYTE lImage_Array = 24;
BYTE Cursor_Set[1] = {8};

VFDCMD_CS410100(1, 22, 23, FALSE, FALSE);
VFDCMD_CS410100(2, 22, 23, Cursor_Set, 1);
VFDCMD_CS410100(13, 22, 23, Image_Array,
                 lImage_Array);

END( );
```

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.

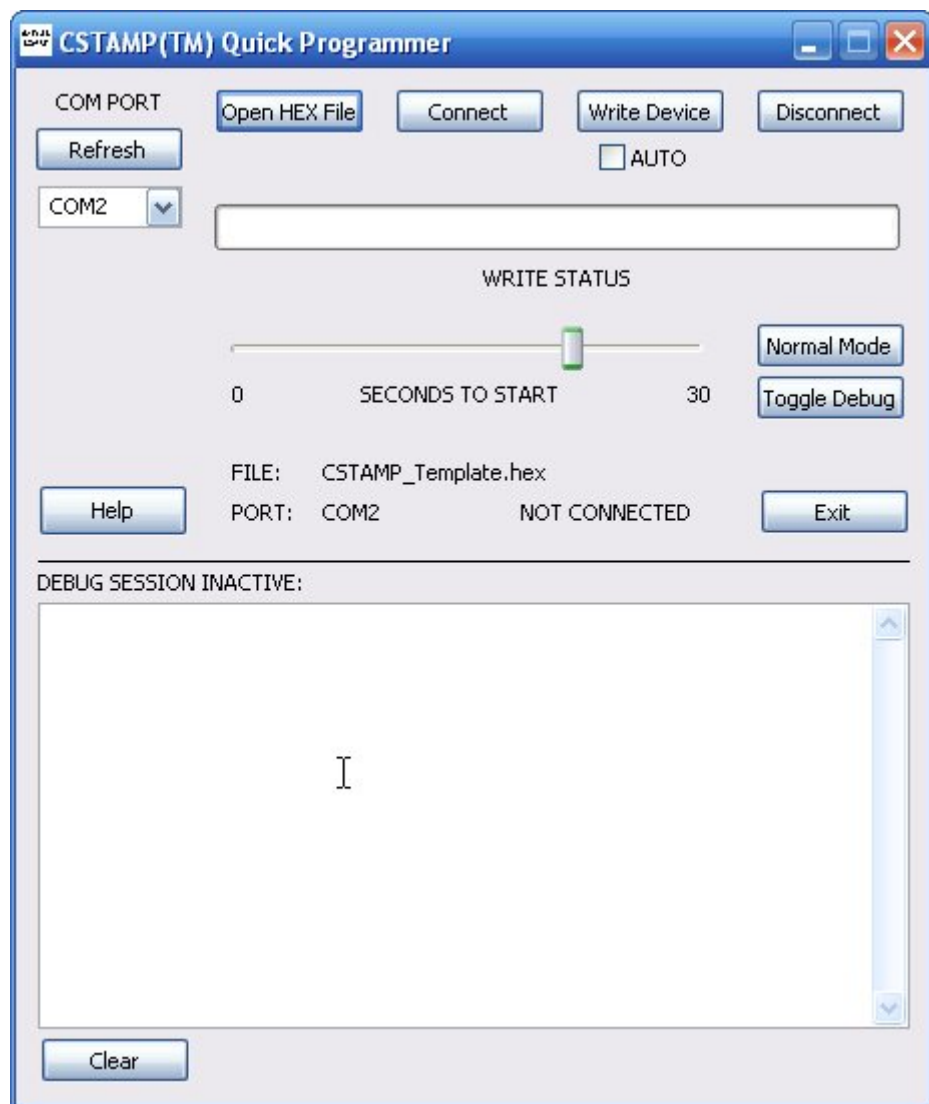
If the code was typed correctly, you will have a file in your program directory with the name of your program project and a .HEX extension. An example is UI_APP.HEX. This is the file that you will download to the C Stamp, as explained up ahead.

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 area 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 start 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.

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.

VFDCMD_CS410100

```
BIT VFDCMD_CS410100(BYTE command, BYTE Data_Pin,  
                    BYTE Reset_Pin,  
                    BYTE Argument_Array[],  
                    BYTE lArgument_Array);
```

The **VFDCMD** function sends a command to the CS410100 VFD 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 – 14) indicating the VFD command to send.

Data_Pin is a variable/constant/expression indicating which CStamp pin, the VFD data wire is connected to. **Data_Pin** must be a valid I/O pin on the CStamp as listed in the CStamp manual. This excludes pins 5-7, 24-26, and 48.

Reset_Pin is a variable/constant/expression indicating which CStamp pin, the VFD reset wire is connected to. **Reset_Pin** must be a valid I/O pin on the C

Stamp as listed in the C Stamp manual. This excludes pins 5 - 7, 24 - 26, 48, and **Data_Pin**.

Argument_Array is an array of type BYTE which contains the arguments specific to the command being sent to the VFD module. These arguments are specified by the command list in the table below. The number of elements in **Argument_Array** is also specific to the command being sent, refer to the table for the length used for a specific command.

lArgument_Array is a variable/constant/expression which gives the length of the array **Argument_Array**. The number of elements necessary is based upon which command is being issued, refer to the table below for details on the length of the array for each command. Although array indices begin at 0 in WC, **lArgument_Array** will be the number of elements in the array, i.e. if the **Argument_Array** as defined as:

```
BYTE TEST_ARRAY[10];
```

lArgument_Array should be assigned a value of 10. Commands which have an argument size of 0, as specified in the table below, do not require a true **Argument_Array** or **lArgument_Array**, for these commands **FALSE** may be used to replace both in the command, or any variable/constant/expression may be entered, because the value of both are not used in the function.

<i>Symbol</i>	<i>Value</i>	<i>Command</i>	<i>Description</i>
VFD_RESET	0	Reset	Resets the module to the default state. This function will interrupt any operation currently being performed by the VFD module. Argument Size: 0
VFD_HOME	1	Cursor Home	Sets the cursor to the home position (top left). Argument Size: 0
VFD_SETC	2	Set Cursor (1-32)	Set the cursor to a position specified by the value of the argument array at index 0, where 1 is the top left and 32 is the bottom right. Argument Size: 1
VFD_SETCLC	3	Set Cursor (line, column)	Sets cursor using two positions in the argument array, where index 0 is the line and index 1 is the column. Argument Size: 2

<i>Symbol</i>	<i>Value</i>	<i>Command</i>	<i>Description</i>
VFD_HIDE	4	Hide Cursor	Stops the position cursor from appearing on the display. Argument Size: 0
VFD_SHOW	5	Show Cursor	Changes the cursor to the underline type. Argument Size: 0
VFD_BACK	6	Backspace	Deletes the preceding character from the current position on the display. Argument Size: 0
VFD_HTAB	7	Horizontal Tab	Moves the current position across by the space of one character. Argument Size: 0
VFD_SLF	8	Smart Line Feed	Moves the cursor down one line to the position beneath it in the same column. Argument Size: 0
VFD_VTAB	9	Vertical Tab	Moves the cursor up one line to the position above it in the same column. Argument Size: 0
VFD_CR	10	Carriage Return	Moves the cursor to the start of the next line. Argument Size: 0
VFD_CCHAR	11	Custom Character	Sends one of 16 possible custom characters to the VFD to be used later. Argument_Array defines the custom character. See table on custom characters below. Argument Size: User defined.
VFD_DCCHAR	12	Display Custom Character	Displays a custom character at the current cursor position. Argument_Array has a value 0 – 16, at index 0, which specifies the custom character to be displayed. Argument Size: 1
VFD_IMAGE	13	Display bit image	Displays a user defined image immediately once the command has been issued. Index 0 of Argument_Array gives the height of the image, in units of 8 pixels, and

<i>Symbol</i>	<i>Value</i>	<i>Command</i>	<i>Description</i>
			index 1 gives the width of the image, in units of 1 pixel. All subsequent indices contain the bit image data. Argument Size: User defined
VFD_MOVIE	14	Display bit image animation	Uses multiple bit images, or frames, to create an animated image. Index 0 of Argument_Array gives the height of the images, in units of 8 pixels, and index 1 gives the width of the images, in units of a single pixel. All frames must have the same dimensions. Index 2 gives the number of iterations the animation is to be played. Valid ranges are from 0-255, where 0 is an infinite loop. Index 3 specifies the speed of the animation 1-255 milliseconds delay inserted between each frame. The frames are placed in sequential order starting with index 4 holding the first byte of the first frame. See bitmap section below for details on creating the data corresponding to a single image. Argument Size: User defined.

Up to 16 custom characters can be stored in the VFD for subsequent use with the VFD command **VFD_CCHAR**. The command uses the first two elements of **Argument_Array** to select the character and its format; subsequent elements are used for the data which defines the character. The first table below shows how to define the custom character options, and the second table shows how to map the data for a character.

Custom Character Options	
Index of Argument_Array	Option
0	Select Character (1-16)
1	Select format; 5x7 (value of 0) or 7x8 (value of 1)

Custom Character Map					
	Indices 2 - 6 of Argument_Array				
	<i>Char Map</i>				
Bits	2	3	4	5	6
Bit 7	0	0	0	0	0
Bit 6	0	0	1	0	0
Bit 5	0	1	1	1	0
Bit 4	1	0	1	0	1
Bit 3	0	0	1	0	0
Bit 2	0	0	1	0	0
Bit 1	0	0	1	0	0
Bit 0	0	0	0	0	0

In this example the indices would have the following values given in binary, and in hexadecimal:

Index 2: binary 00010000 or hexadecimal 10

Index 3: binary 00100000 or hexadecimal 20

Index 4: binary 01111110 or hexadecimal 7E

Index 5: binary 00100000 or hexadecimal 20

Index 6: binary 00010000 or hexadecimal 10

These values are obtained by reading the bits down the index from bit 7 (MSB) to bit 0 (LSB).

In the 5x7 format, bit 0 is not eligible and should be kept at a value of 0 for each index. If the 7x8 format was selected bit 0 would be eligible for use, and the character map would occupy indices 2 - 8.

The resulting **Argument_Array** for this example might look like the following:

```
BYTE Example_Array[7] = {1, 0, 0x10, 0x20, 0x7E, 0x20,
                        0x10};
```

The command would then look similar to the code fragment below:

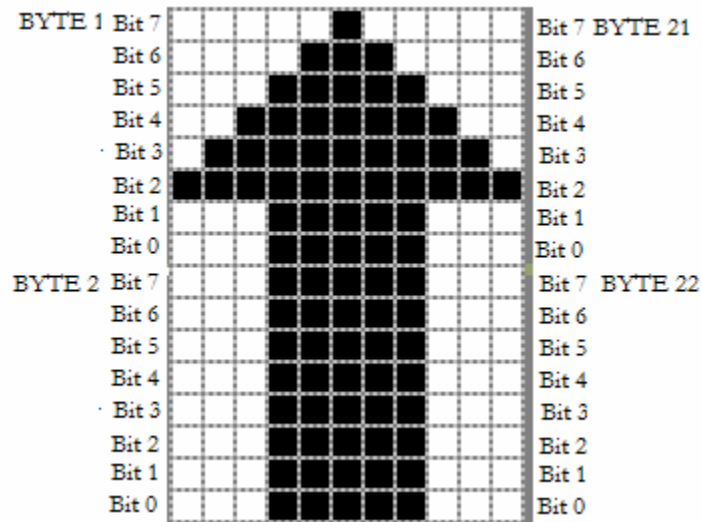
```
result = VFDCMD_CS410100(VFD_CCHAR, 18, 17,
                          Example_Array, 7);
```

Checking the result of calling the function is optional, although it is a good programming practice.

Then to display this character at the current cursor position, we would call the function as seen below:

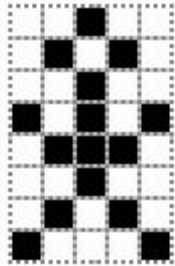
```
result = VFDCMD_CS410100(VFD_DCCHAR, 18, 17, 1, 1);
```

Bit image data is defined from top to bottom and left to right, where the most significant bit of the first byte of bit image data controls the top-left pixel, and the least significant bit of the last byte of bit image data controls the bottom-right pixel. The graphic below shows how to create a bitmap for an image. This image uses both rows of the display and 11 columns. The black spaces represent a '1' while the white spaces represent a '0'.

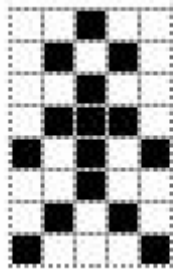


The bytes 3-20 run from left to right following the same pattern in which all odd numbered bytes control the top row and all even numbered bytes control the bottom row. This bitmap gives the resulting data which should be incorporated into **Argument_Array** as specified in the command table.

Animation on the VFD is accomplished through the use of multiple user defined bit images. The bitmap corresponding to each frame is listed in succession and looped through based on the arguments set by the user. Below is the first bitmap of a two frame animation.



As described previously, the bitmap is converted to the appropriate bytes in order from top to bottom, and left to right. In this example the five bytes for this image, given in hexadecimal format are 0x11, 0x4A, 0xBC, 0x4A and 0x11.



The second image is converted from bitmap to bytes in the same fashion. This bitmap corresponds to the following bytes in hexadecimal format, 0x09, 0x52, 0xBC, 0x52, and 0x09. To create the animation in which these two images are repeated indefinitely and with a delay of approximately 50ms between each frame, we define the following array, and call the VFDCMD function as seen below. Once the function is called the animation is displayed immediately on the VFD.

```
MOVIE_ARG[14] = {2, 5, 0, 50, 0x11, 0x4A, 0xBC, 0x4A,  
0x11, 0x09, 0x52, 0xBC, 0x52, 0x09};
```

```
VFDCMD_CS410100(14, Data_Pin, Reset_Pin, MOVIE_ARG,  
14);
```

VFDOUT_CS410100

```
BIT VFDOUT_CS410100(BYTE Data_Pin, BYTE ASCII_Array[],  
                    BYTE lASCII_Array);
```

The **VFDOUT** sends ASCII bytes to the CS410100 VFD 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**.

Data_Pin is a variable/constant/expression indicating which CStamp pin, the VFD data wire is connected to.

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

lASCII_Array is a variable/constant/expression that specifies how many bytes are in **ASCII_Array**.

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.

