



4D SYSTEMS

TURNING TECHNOLOGY INTO ART

ViSi Updating the CC3000 Firmware

Document Date: 3rd November, 2014

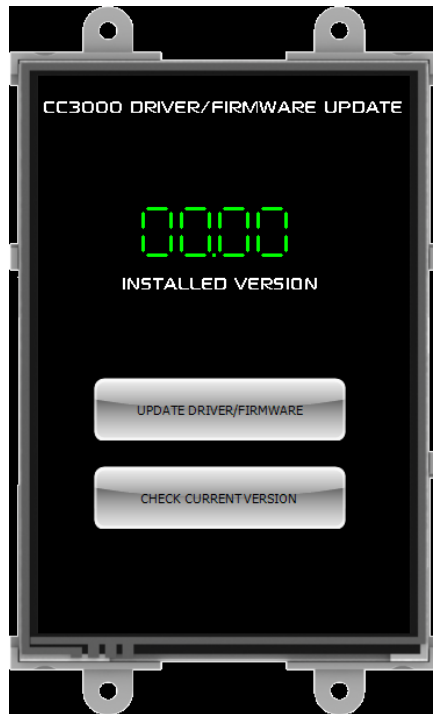
Document Revision: 1.0

Description

This application note shows how to update the driver and firmware of the CC3000 WiFi chip. This application note comes with a project specially made to perform the update.

** Updates must only be done if the user fully understands the process and only if deemed necessary. For more information on TI CC3000 please refer to the link below.

<http://processors.wiki.ti.com/index.php/CC3000>



Before getting started, the following are required:

- The target module can also be a Diablo16 display

[uLCD-35DT](#)

[uLCD-70DT](#)

Visit www.4dsystems.com.au/products to see the latest display module products that use the Diablo16 processor.

- [4D Programming Cable](#) or [uUSB-PA5](#)
- [micro-SD \(uSD\)](#) memory card
- [Workshop 4 IDE](#) (installed according to the installation document)
- When downloading an application note, a list of recommended application notes is shown. It is assumed that the user has read or has a working knowledge of the topics presented in these recommended application notes.
- 2-GB microSD
- [CC3000 WiFi Shield](#)

Content

Description	2
Content.....	3
Application Overview	4
Setup Procedure	4
Design of the Project	5
<i>Layout of the Project</i>	5
<i>The ViSi-based CC3000 Firmware Update Program</i>	5
The sub-routines	5
startPage() sub-routine	5
updatePage() sub-routine	6
disableTouch() sub-routine	6
drive() sub-routine	6
firmware() subroutine	7
The DRIVER.HEX and FWARE.HEX files	8
Proprietary Information	9
Disclaimer of Warranties & Limitation of Liability.....	9

Application Overview

Updates are provided for so many devices to pave the way for loading modification, enhancements, or patches to a device. This application note shows how to update the Texas Instrument CC3000 WiFi chip.



Setup Procedure

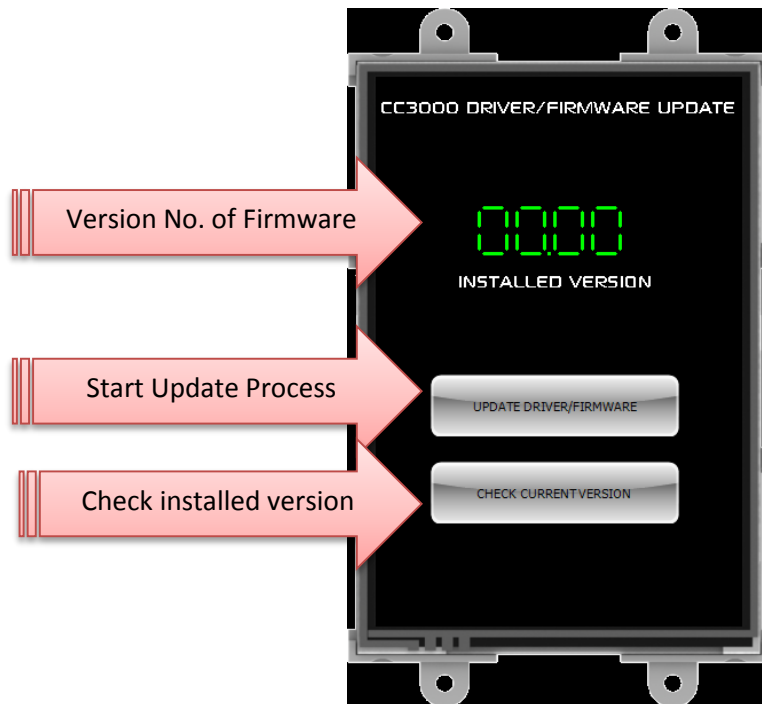
For instructions on how to launch Workshop 4, how to open a **ViSi** project, and how to change the target display, kindly refer to the section “**Setup Procedure**” of the application note

[ViSi Getting Started - First Project for Picaso and Diablo16](#)

Design of the Project

Layout of the Project

In this project couple of objects were used to provide a simple user interface form for the CC3000. The user interface have buttons that are used to check current firmware version and also to initiate the update.



The ViSi-based CC3000 Firmware Update Program

In this part of the application note statements that were written in 4DGL will be shown. These statements are used to produce the functionality desired for this project application.

The sub-routines

The statements presented here are called upon in the main function later in this document. It is best to present the purpose of each sub-routine for us to be able to easily understand the main program statements.

startPage() sub-routine

To initially display the ViSi project objects is the purpose of this sub-routine. It also contains the statements that clear the attributes for the objects there enabling the object to be valid for touch feature of the display. In these set of statements the touch attributes of objects not included in the page is cleared so as to avoid conflict of detection with the objects included in this routine.

```
func startPage()
    disableTouch("updatePage");
    img_ClearAttributes(hnd1, iWinbutton1, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
    img_ClearAttributes(hnd1, iWinbutton2, I_TOUCH_DISABLE); // set to enable touch, only need to do this once

    img_SetWord(hnd1, iWinbutton2, IMAGE_INDEX, 0); // where state is 0 for up and 1 for down
    img_SetWord(hnd1, iWinbutton1, IMAGE_INDEX, 0); // where state is 0 for up and 1 for down

    gfx_Panel(PANEL_RAISED, 36, 84, 240, 119, WHEAT);

    img_Show(hnd1, iStatictext2);
    img_Show(hnd1, iWinbutton2); // show button, only do this once
    img_Show(hnd1, iWinbutton1); // show button, only do this once
    img_Show(hnd1, iStatictext1);
    img_Show(hnd1, iWinbutton1);
    img_Show(hnd1, iLeddigits1); // show all digits at 0, only do this once
    ledDigitsDisplay(versionValue, iLeddigits1+1, 88, 4, 3, 33, 0);

    return;
endfunc
```

updatePage() sub-routine

This sub-routine is made to initially display the buttons for the updatePage form and also disable the touch related attributes of the other objects not belonging to this form.

```
func updatePage()
    disableTouch("startPage");
    // gfx_Panel(PANEL_RAISED, 0, 0, 320, 480, 0x9010);
    // gfx_Panel(PANEL_RAISED, 8, 12, 303, 455, WHITE);
    img_Show(hnd1, iStatistic3);

    img_ClearAttributes(hnd1, iWinbutton3, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
    img_ClearAttributes(hnd1, iWinbutton4, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
    img_Show(hnd1, iWinbutton3);
    img_Show(hnd1, iWinbutton4);
endfunc
```

disableTouch() sub-routine

This routine was written to simply manage the setting and clearing of touch related attributes. When working with touch feature on objects, it is always best to disable this feature on non-members of a form to avoid detection problems to occur.

```
func disableTouch(var page)
    if(page == "startPage")
        img_SetAttributes(hnd1, iWinbutton1, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
        img_SetAttributes(hnd1, iWinbutton2, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
    else if(page == "updatePage")
        img_SetAttributes(hnd1, iWinbutton3, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
        img_SetAttributes(hnd1, iWinbutton4, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
    endif
endfunc
```

drive() sub-routine

This sub-routine is dedicated to processing the driver.HEX file that is saved on the microSD. In the later parts of this application note, this HEX file will be described in detail.

```
func drive()
    WIFI_begin(NO_PATCH);
    hci_cmd_set_event_mask(unsol_keep_alive|unsol_ping_report|unsol_init|unsol_connect|u
    hci_cmd_sp_version();

    var aa,ss:=0;
    for(aa:=0; aa <= 11; aa++)
        fatTable[ss++] := (FAT_1[aa]&0xFF) + 1;
        fatTable[ss++] := (FAT_1[aa]&0xFF00) >> 8;
        fatTable[ss++] := (FAT_2[aa]&0xFF);
        fatTable[ss++] := (FAT_2[aa]&0xFF00) >> 8;
    next

    print("\n");
    for(aa:=0; aa < 48; aa++)
        if(aa%8 == 0)
            print("\n");
        endif
        //print(" ",[HEX2Z] fatTable[aa]);
    next

    hci_cmd_nvmem_write(16,4,0,LS);
    hci_cmd_nvmem_write(16,24,4,fatTable);
    hci_cmd_nvmem_write(16,24,28,&fatTable[24]);

    var xmp;
    for(xmp:=0; xmp<4;xmp++)
        hci_cmd_nvmem_write(NVMEM_RM_FILEID,32,32*xmp,&RM[32*xmp]);
    next

    patch_write(NVMEM_WLAN_DRIVER_SP_FILEID, "DRIVER.HEX",32);

    hci_cmd_nvmem_write(NVMEM_MAC_FILEID,6,0,MAC_);
    print("\nDRIVER DOWNLOAD SUCCESSFULL");
    stop_wifi();
    pause(2000);
endfunc
```

The process starts with processing the information related to the internal EEPROM of the CC3000. This section of the program updates the File Allocation Table (FAT) of the EEPROM according to the data included in the following image.

```

//*****Service Pack version P1.13.7.14.24****//
var RM[128]:=[ 0x03, 0x00, 0x01, 0x01, 0x14, 0x14, 0x00, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27, 0x27,
0x27, 0x27, 0x27, 0x27, 0x27, 0x23, 0x25, 0x25, 0x25, 0x25, 0x25, 0x25, 0x25, 0x25, 0x25, 0x23,
0x23, 0x23, 0x23, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50,
0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50,
0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x01, 0x77, 0x80, 0x1D, 0x1F,
0x22, 0x26, 0x28, 0x29, 0x1A, 0x1F, 0x22, 0x24, 0x26, 0x28, 0x16, 0x1D, 0x1E, 0x20, 0x24, 0x25,
0x1E, 0x2D, 0x01, 0x02, 0x02, 0x02, 0x00, 0x15, 0x15, 0x15, 0x11, 0x15, 0x15, 0x0E, 0x00];

var FAT_1[12] := [0x50, 0x1f0, 0x390, 0x1390, 0x2390, 0x4390, 0x6390, 0x6390, 0x63b0, 0x63f0, 0x6430, 0x6830];
var FAT_2[12] := [0x1a0, 0x1a0, 0x1000, 0x1000, 0x2000, 0x2000, 0x10, 0x10, 0x40, 0x40, 0x400, 0x200];
var MAC[6] := [0x08, 0x00, 0x28, 0x59, 0xDA, 0xE2];
//*****

```

The arrays above shows two sets of FAT arrays. These are parallel arrays that define the start location of a particular length of memory space. Also included in this segment of the project codes is the RM array. The RM array is a special set of data that are parameters to the CC3000 radio.

**** The MAC address specified on this section must first be known to the user and be changed accordingly.**

This section of the project reads the information saved on the driver.HEX file. The information is then transferred to the CC3000 according to the specified number of bytes. The entire sub-routine will write the data chunks-by-chunks at a speed compatible to the CC3000. Each time a segment of the driver is written to the CC3000 the numerical indication of successful writes is updated. At the end of writing the driver file, a notification is posted on the screen informing the user that is has ended. After the successful write of the driver, next will be the firmware update.

**** THE CC3000 REQUIRES THAT THE DRIVER MUST FIRST BE LOADED BE SUCCESSFULLY LOADED BEFORE THE FIRMWARE UPDATE.**

firmware() subroutine

This sub-routine contains a set of statements that execute the updating of the firmware for the CC3000.

```

func firmware()
  WIFI_begin(NO_PATCH);
  hci_cmd_set_event_mask(unsol_keep_alive || unsol_ping_report || unsol_init || unsol_connect || unsol_disconnect || unsol_DHCP || unsol_sp_version());

  patch_write(NVMEPI_WLAN_FW_SP_FILEID, "FWARE.HEX",16);
  print("\nFIRMWARE DOWNLOAD SUCCESSFULL");

  stop_wifi();
  pause(2000);
endfunc

```

The update statements starts with specifying the source of the HEX file. Seen in the set of statements above that the firmware data is contained on a file named – FWARE. It is important to name all the files same as the names specified on the project or to names that has at most 8 charactes for the filename and and extension of either TXT or HEX.

During the course of the update, a notification of successful writes is displayed on the screen. After the firmware update has finished the user will again receive a notification that the process completed successfully and shall return to the main start page of the project.

The DRIVER.HEX and FWARE.HEX files

The files specified in this application note was derived from the CC3000 Service Pack version 1.33. This update project will result to a firmware version that is identified as 1.24.

For further details on this information please refer to the link specified below.

http://processors.wiki.ti.com/index.php/CC3000_Wi-Fi_Downloads_1.12_and_1.13

Please note that the driver information and the firmware information must first be saved on two separate files – DRIVER.HEX and FWARE.HEX. Copy the supplied array from the service pack being used. When a new service pack version is available the user will have to create the files manually and should follow the same format for the content of the files.

Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.