Diagnosing and recovering the DJI Phantom 2 Vision+ Wi-Fi module from lack of FPV (black screen)

There are many instances of DJI P2V+ owners losing the FPV in the DJI Vision app. Although telemetry and camera control operate ok there is a permanent black screen in the live video feed window.

If the failure mode can be confirmed as failure to boot of the WM301_DM368 PCB software within the Wi-Fi module there is a good chance that it can be recovered to restore FPV functionality.

Diagnosis

Follow the steps below to diagnose if your Wi-Fi module could be faulty due to the WM301_DM368 PCB failing to boot:

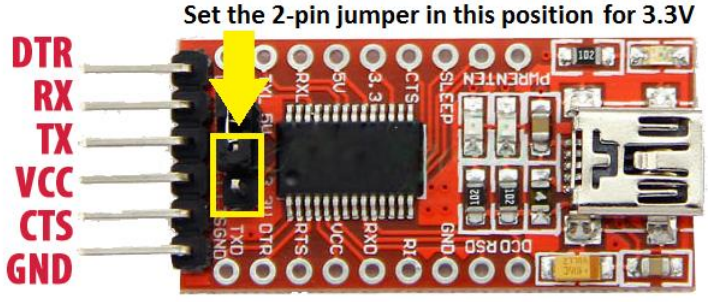
1. Turn on your P2V+, remote control, and extender and connect to your P2V+ Wi-Fi network.
2. Run the DJI Vision app and confirm you are connected to the P2V+ Wi-Fi network.
3. Using a network ping app, such as *Ping by Lipinic*, ping IP address 192.168.1.2 – if successful your Wi-Fi extender is alive and can connect to the P2V+ Wi-Fi, i.e., binding is ok. If ping fails there is another issue so go no further.
4. Ping IP address 192.168.1.1 – if successful the top PCB within the Wi-Fi module is working. If ping fails there is another issue so go no further.
5. Ping IP address 192.168.1.10 – if this fails the bottom PCB (WM301_DM368) within the Wi-Fi module could be suffering from the failure to boot issue and you should continue reading further. If successful then there is another issue so go no further.

If you are at step 5 above and the ping test to IP address 192.168.1.10 fails then there is a good chance you can recover the module assuming there are no other problems with the WM301_DM368 PCB.

The procedure now described can easily be undertaken if you are confident with making modifications to electronic PCBs and have a basic understanding of electronics.

Equipment required

- USB to serial port adapter, for example, the FT232RL adapter, a Google search reveals many places you can pick this up from such as Amazon (https://www.amazon.co.uk/XCSOURCE-FT232RL-Adapter-Arduino-TE203-x/dp/B00HSX3CXE). If you do use the FT232RL adapter please ensure you set the 2-pin jumper to 3.3V mode, see the picture below:



- 11V DC power supply (to power the WM301_DM368 PCB).

Recovery solution

The recovery solution is to reprogram the bootloader and application software into the NAND flash on the WM301_DM368 PCB using a serial port adapter (such as the FT232RL adapter). This connects to the UART port on the WM301_DM368 PCB.

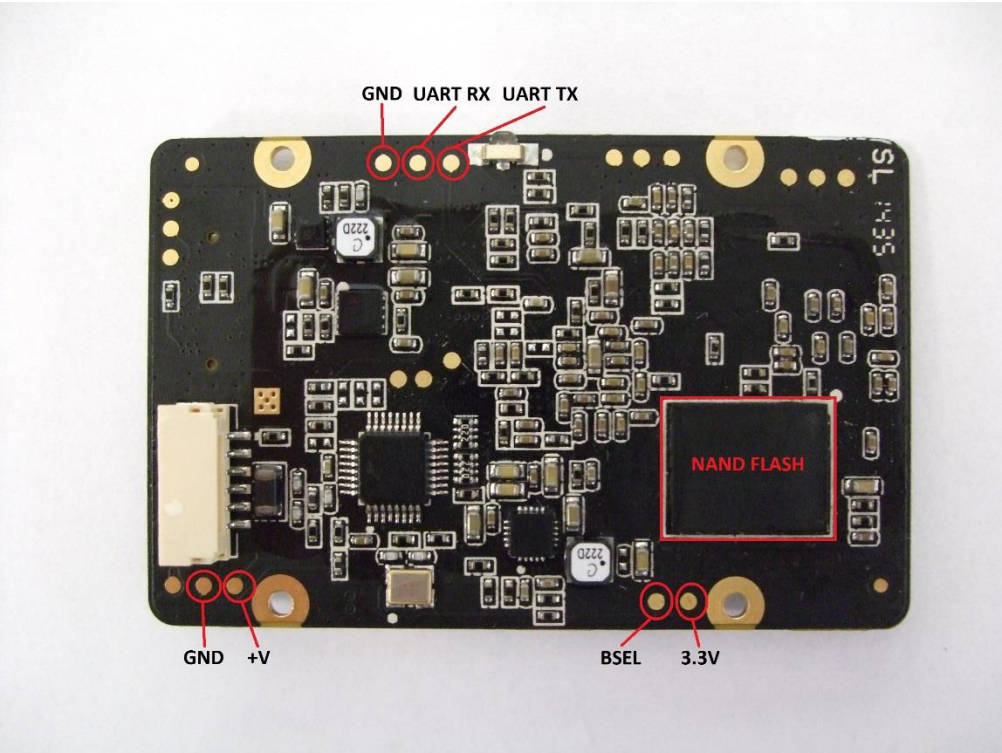
1. Remove the complete Wi-Fi module from the P2V+; there are a few videos on YouTube of how to do this so I won't bore you with detailing this here.
2. You will need to open up the Wi-Fi module in order to remove the WM301_DM368 PCB, which is the black coloured PCB in the bottom of the module; again there are a few videos on YouTube of how to do this.

   Tip A: take care when you remove the copper tape around the outside of the Wi-Fi module as it is not easy to remove in one piece, in my case it split so I had to replace it.

   Tip B: when you first open up the module the green coloured PCB in the top of the Wi-Fi module has a lot of white thermal paste spread over it, be careful not to touch the paste and don't remove any – just set that part of the Wi-Fi module aside until you need to put it back together.

3. Turn the WM301_DM368 PCB over so that board side facing you is the same as in the following picture – note that in the picture I have already removed the thermal pad located on the right hand side. On some PCBs there is white thermal paste spread over the NAND flash device, be careful not to touch the paste and don't remove any.

   Tip C: when removing the thermal pad use a flat piece of plastic to leverage it off the component underneath otherwise it may tear (as mine did)



4. Connect power and GND wires on to the +V and GND labelled pads located in the bottom left hand corner of the PCB in the picture above but DO NOT apply power to the PCB yet.
5. Connect the serial port adapter TX pin (i.e., FT232RL adapter) to the UART RX pin in the picture above.
6. Connect the serial port adapter RX pin (i.e., FT232RL adapter) to the UART TX pin in the picture above.
7. Connect the serial port adapter GND pin (i.e., FT232RL adapter) to the GND pin next to the UART pins in the picture above.

After the steps above have been completed you are now ready to power on the WM301_DM368 PCB and confirm 100% that the NAND flash image is failing to boot, follow the sequence below:

1. Connect the power and GND wires from the PCB to your 11V power supply but DO NOT apply power yet.
2. Connect your serial port adapter (i.e., FT232RL adapter) to your PC.
3. Run a terminal emulator program, such as TeraTerm, which can be downloaded from https://osdn.net/projects/ttssh2/releases/ and configure your serial port for 115200bps, 8 bit data, no parity, 1 stop bit, no flow control.
4. Power on the PCB and you should see the bootloader debug from the serial port displayed in the terminal emulator window – if it shows the same debug as below then your module issue is 100% caused by the NAND flash failing to run the application software

```
DM36x initialization passed!
UBL Product Vesion : DJI-UBL-1.0-rc2
Dji UBL Version: 1.51(Jan 24 2014 - 04:27:19)
Booting Catalog Boot Loader
BootMode = NAND
Starting NAND Copy...
Valid magicnum, 0xA1ACED66, found in block 0x00000019.
Valid magicnum, 0xA1ACED66, found in block 0x0000001B.
Valid magicnum, 0xA1ACED66, found in block 0x0000001D.
Valid magicnum, 0xA1ACED66, found in block 0x0000001F.
No valid boot image found!
NAND Boot failed.
Aborting...
```

The next stage of the process is to set the board into NAND flash programming mode so you can reprogram the software. Follow the sequence below:

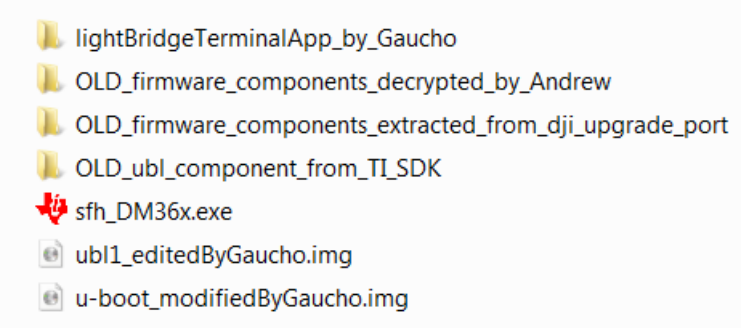
1. Power off the PCB.

2. Connect a wire link between the BSEL and 3.3V labelled pads in the picture above.
3. Power on the PCB and you should now see the boot debug in the terminal emulator window change to the following:

```
BOOTME BOOTME BOOTME BOOTME BOOTME BOOTME BOOTME BOOTME BOOTME BOOTME BOOTME BOOTME BOOTME
```

4. If you don't see the BOOTME debug then stop at this point and double check that the BSEL and 3.3V pads are connected together – without seeing BOOTME you will not be able to reprogram the NAND flash.
5. Power off the PCB.

If you have got this far great! You are now ready to reprogram the software. Before this can be done you need to download the software from the web; download the LB_hack.zip file from http://www.tr3ma.com/Dati/LB_hack.zip and unzip it to a local folder on your hard drive. This folder should contain the following files:



The *sfh_DM36x.exe* application is the software programming tool. The *ubl1_editedByGaucho.img* file is the application software and the *u-boot_modifiedByGaucho.img* file is the bootloader software. All other folders and files are not required and can be deleted. So the next stage in the process is to use the *sfh_DM36.exe* application to reprogram the software. I used a 32-bit Windows 7 PC in order to do this, I don't know if 64-bit versions of Windows work so you will just have to go ahead and try it. Follow the sequence below:

1. Close the terminal emulator program you have been using up to this point.
2. Open up a windows command window (DOS) and navigate to the folder where the *sfh_DM36x.exe* application and *ubl1_editedByGaucho.img* & *u-boot_modifiedByGaucho.img* files are located.
3. Run the following command within the command window:

```
sfh_DM36x.exe –nandflash –v –p "COM1" ubl1_editedByGaucho.img u-boot_modifiedByGaucho.img
```

Tip D: you may need to replace “COM1” above with the number of the COM port that the serial adapter is using on your PC, for example, if your COM port number is COM3 replace “COM1” in the line above with “COM3”.

4. Power on the PCB. The flash programming procedure starts but it will not be complete at this stage. You should see the following debug start to be printed:

```
Waiting for SFT on the DM36x…
Target:    BOOTME
Target:    BOOTME
Target:    BOOTME
Target:    BOOTME
Target:    BOOTME
```

5. Press CTRL+C to stop the *sfh_DM36x.exe* application but DO NOT turn power off on the PCB.
6. Run again the same `sfh_DM36x.exe -nandflash -v -p "COM1" ubl1_editedByGaucho.img u-boot_modifiedByGaucho.img` command within the command window. This time the flash programming procedure should complete. Note that this will take about a minute to complete – DO NOT power off the PCB whilst this is in progress, be patient with it. You should see the following debug during programming and at the end of programming the *sfh_DM36x.exe* application will automatically exit:

```
-----------------------------------------------------

   TI Serial Flasher Host Program for DM36x

   (C) 2009, Texas Instruments, Inc.

   Ver. 1.50

-----------------------------------------------------


Flashing NAND with ubl1_editedByGaucho.img and u-boot_modifiedByGaucho.img.


Attempting to connect to device COM4...

Press any key to end this program at any time.


Waiting for the DM36x...

        Target:  BOOTME

BOOTME commmand received. Returning ACK and header...

ACK command sent. Waiting for BEGIN command...

        Target:   BEGIN

BEGIN commmand received. Sending CRC table...

 100% [                                                          ]

                          CRC table sent....
```

```
Waiting for DONE...
        Target:    DONE
DONE received.  Sending the UBL...
 100% [                                                                 ]
                              UBL sent....



        Target:    DONE
DONE received.  UBL was accepted.
UBL transmitted successfully.

Waiting for SFT on the DM36x...
        Target: Starting UART Boot...
        Target: BOOTUBL
BOOTUBL commmand received. Returning CMD and command...
CMD value sent.  Waiting for DONE...
        Target:    DONE
DONE received. Command was accepted.
Sending the UBL image
Waiting for SENDIMG sequence...
        Target: SENDIMG
SENDIMG received. Returning ACK and header for image data...
ACK command sent. Waiting for BEGIN command...
        Target:   BEGIN
BEGIN commmand received.
 100% [                                                                 ]
                            Image data sent...


Waiting for DONE...
        Target:    DONE
DONE received.  All bytes of image data received...
        Target: Writing UBL to NAND flash
        Target: Unprotecting blocks 0x00000001 through 0x00000018.
        Target: Number of blocks needed for header and data: 0x0x00000001
        Target: Attempting to start in block number 0x0x00000001.
        Target: Erasing block 0x00000001 through 0x00000001.
        Target: Writing header and image data to Block 0x00000001, Page 0x000000
00
        Target: Erasing block 0x00000002 through 0x00000002.
```

```
        Target: Writing header and image data to Block 0x00000002, Page 0x000000
00
        Target: Erasing block 0x00000003 through 0x00000003.
        Target: Writing header and image data to Block 0x00000003, Page 0x000000
00
        Target: Erasing block 0x00000004 through 0x00000004.
        Target: Writing header and image data to Block 0x00000004, Page 0x000000
00
        Target: Erasing block 0x00000005 through 0x00000005.
        Target: Writing header and image data to Block 0x00000005, Page 0x000000
00
        Target: Erasing block 0x00000006 through 0x00000006.
        Target: Writing header and image data to Block 0x00000006, Page 0x000000
00
        Target: Erasing block 0x00000007 through 0x00000007.
        Target: Writing header and image data to Block 0x00000007, Page 0x000000
00
        Target: Erasing block 0x00000008 through 0x00000008.
        Target: Writing header and image data to Block 0x00000008, Page 0x000000
00
        Target: Erasing block 0x00000009 through 0x00000009.
        Target: Writing header and image data to Block 0x00000009, Page 0x000000
00
        Target: Erasing block 0x0000000A through 0x0000000A.
        Target: Writing header and image data to Block 0x0000000A, Page 0x000000
00
        Target: Erasing block 0x0000000B through 0x0000000B.
        Target: Writing header and image data to Block 0x0000000B, Page 0x000000
00
        Target: Erasing block 0x0000000C through 0x0000000C.
        Target: Writing header and image data to Block 0x0000000C, Page 0x000000
00
        Target: Erasing block 0x0000000D through 0x0000000D.
        Target: Writing header and image data to Block 0x0000000D, Page 0x000000
00
        Target: Erasing block 0x0000000E through 0x0000000E.
        Target: Writing header and image data to Block 0x0000000E, Page 0x000000
00
        Target: Erasing block 0x0000000F through 0x0000000F.
```

```
        Target: Writing header and image data to Block 0x0000000F, Page 0x000000
00
        Target: Erasing block 0x00000010 through 0x00000010.
        Target: Writing header and image data to Block 0x00000010, Page 0x000000
00
        Target: Erasing block 0x00000011 through 0x00000011.
        Target: Writing header and image data to Block 0x00000011, Page 0x000000
00
        Target: Erasing block 0x00000012 through 0x00000012.
        Target: Writing header and image data to Block 0x00000012, Page 0x000000
00
        Target: Erasing block 0x00000013 through 0x00000013.
        Target: Writing header and image data to Block 0x00000013, Page 0x000000
00
        Target: Erasing block 0x00000014 through 0x00000014.
        Target: Writing header and image data to Block 0x00000014, Page 0x000000
00
        Target: Erasing block 0x00000015 through 0x00000015.
        Target: Writing header and image data to Block 0x00000015, Page 0x000000
00
        Target: Erasing block 0x00000016 through 0x00000016.
        Target: Writing header and image data to Block 0x00000016, Page 0x000000
00
        Target: Erasing block 0x00000017 through 0x00000017.
        Target: Writing header and image data to Block 0x00000017, Page 0x000000
00
        Target: Erasing block 0x00000018 through 0x00000018.
        Target: Writing header and image data to Block 0x00000018, Page 0x000000
00
        Target: Protecting the entire NAND flash.
        Target:    DONE
Sending the Application image
Waiting for SENDIMG sequence...
        Target: SENDIMG
SENDIMG received. Returning ACK and header for image data...
ACK command sent. Waiting for BEGIN command...
        Target:   BEGIN
BEGIN commmand received.
 100% [                                                          ]
```

```
                            Image data sent...


Waiting for DONE...
        Target:    DONE
DONE received.  All bytes of image data received...
        Target: Writing APP to NAND flash
        Target: Unprotecting blocks 0x00000019 through 0x00000032.
        Target: Number of blocks needed for header and data: 0x0x00000003
        Target: Attempting to start in block number 0x0x00000019.
        Target: Erasing block 0x00000019 through 0x0000001B.
        Target: Writing header and image data to Block 0x00000019, Page 0x000000
00
        Target: Erasing block 0x0000001C through 0x0000001E.
        Target: Writing header and image data to Block 0x0000001C, Page 0x000000
00
        Target: Erasing block 0x0000001F through 0x00000021.
        Target: Writing header and image data to Block 0x0000001F, Page 0x000000
00
        Target: Erasing block 0x00000022 through 0x00000024.
        Target: Writing header and image data to Block 0x00000022, Page 0x000000
00
        Target: Erasing block 0x00000025 through 0x00000027.
        Target: Writing header and image data to Block 0x00000025, Page 0x000000
00
        Target: Erasing block 0x00000028 through 0x0000002A.
        Target: Writing header and image data to Block 0x00000028, Page 0x000000
00
        Target: Erasing block 0x0000002B through 0x0000002D.
        Target: Writing header and image data to Block 0x0000002B, Page 0x000000
00
        Target: Erasing block 0x0000002E through 0x00000030.
        Target: Writing header and image data to Block 0x0000002E, Page 0x000000
00
        Target: Protecting the entire NAND flash.
        Target:    DONE
        Target:    DONE


Operation completed successfully.
```

```
D:\DJI\fix\software\LB_hack>
```

7. Power off the PCB.
8. Remove the wire link between the BSEL and 3.3V labelled pads in the picture above.
9. Run the terminal emulator program again, such as TeraTerm, again with the same settings as you used previously (115200bps, 8 bit data, no parity, 1 stop bit, no flow control).
10. Power on the PCB and you should see the bootloader and application software debug from the serial port displayed in the terminal emulator window – if it shows the same debug as below then your module is now running the software and has been recovered:

```
DM36x initialization passed!
UBL Product Vesion : DJI-GSP-UBL-1.0-rc10(2014-08-15)
Dji UBL Version: 1.51(Aug 15 2014 - 17:05:12)
Booting Catalog Boot Loader
BootMode = NAND
Starting NAND Copy...
Valid magicnum, 0xA1ACED66, found in block 0x00000019.
Uboot Checksum:0x7E25B44B
Actua Checksum:0x7E25B44B

U-Boot Product Vesion : DJI-GSP-Uboot-1.0-rc4(2014-07-23)
U-Boot 2010.12-rc2-svn-Dji (Jul 23 2014 - 11:14:40)
Cores: ARM 432 MHz
DDR:   297 MHz
I2C:   ready
DRAM:  128 MiB
NAND:  128 MiB
Bad block table found at page 65472, version 0x01
Bad block table found at page 65408, version 0x01
*** Warning - bad CRC, using default environment

Net:   Ethernet PHY: GENERIC @ 0xff
DaVinci-EMAC
Press ESC to abort autoboot in 1 seconds

Loading from nand0, offset 0x4a0000
** Unknown image type
Wrong Image Format for bootm command
ERROR: can't get kernel image!
```

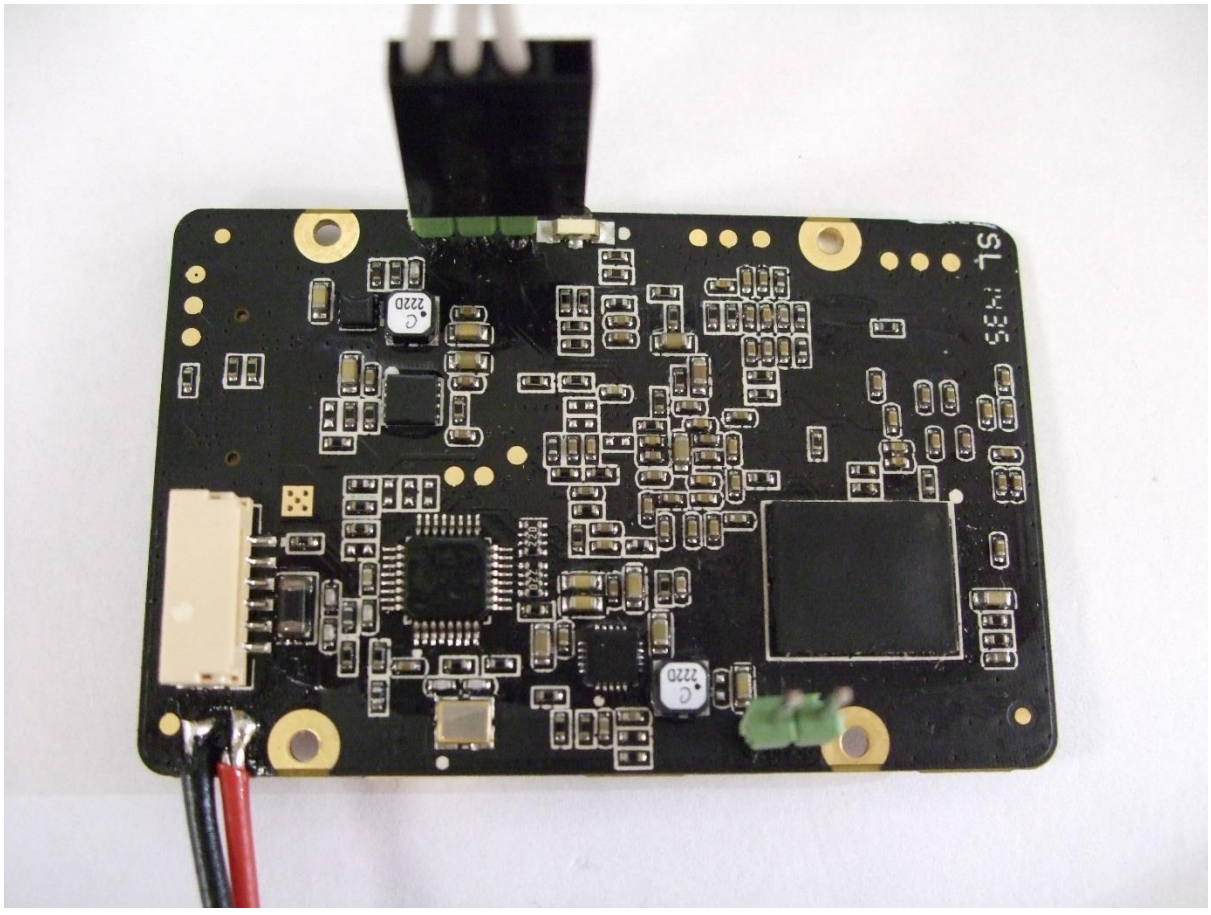
```
Loading from nand0, offset 0x900000
   Image Name:   Linux-2.6.32.17-davinci1
   Created:      2014-04-09  12:21:58 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3823424 Bytes = 3.6 MiB
   Load Address: 80008000
   Entry Point:  80008000
## Booting kernel from Legacy Image at 80700000 ...
   Image Name:   Linux-2.6.32.17-davinci1
   Created:      2014-04-09  12:21:58 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3823424 Bytes = 3.6 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Loading Kernel Image ... OK
OK


Starting kernel ...


[    0.000000] Kernel Product Vesion : DJI-Kernel-1.0-rc2
[    0.000000] Linux version 2.6.32.17-davinci1 (root@ubuntu) (gcc version 4.3.3 (Sourcery G++
Lite 2009q1-203) ) #6 PREEMPT Wed Apr 9 05:21:55 PDT 2014
[    0.000000] CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=00053177
[    0.000000] CPU: VIVT data cache, VIVT instruction cache
[    0.000000] Machine: DaVinci DM36x EVM
[    0.000000] Memory policy: ECC disabled, Data cache writeback
[    0.000000] DaVinci dm36x_rev1.2 variant 0x8
[    0.000000] Built 1 zonelists in Zone order, mobility grouping off.  Total pages: 12192
[    0.000000] Kernel command line: console=ttyS0,115200n8 rw dm365_imp.oper_mode=0
video=davincifb:vid0=0,10K:vid1=0,10K:osd0=1920x1080X16,8100K mem=48MB
davinci_enc_mngr.ch0_output=COMPOSITE davinci_enc_mngr.ch0_mode=pal ubi.mtd=2,2048
root=ubi0:rootfs rootfstype=ubifs ip=off lpj=1077248
[    0.000000] PID hash table entries: 256 (order: -2, 1024 bytes)
[    0.000000] Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
[    0.000000] Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
[    0.000000] Memory: 48MB = 48MB total
[    0.000000] Memory: 44784KB available (3456K code, 310K data, 112K init, 0K highmem)
[    0.000000] SLUB: Genslabs=11, HWalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] Hierarchical RCU implementation.
[    0.000000] NR_IRQS:245
```

```
[    0.000000] Console: colour dummy device 80x30
[    0.000000] Calibrating delay loop (skipped) preset value.. 215.44 BogoMIPS (lpj=1077248)
[    0.000000] Mount-cache hash table entries: 512
[    0.000000] CPU: Testing write buffer coherency: ok
[    0.000000] DaVinci: 8 gpio irqs
```

If you successfully get to this stage well done!

The next step is to remove all of the connections you have made to the PCB and put the Wi-Fi module back together and reinstall it into the P2V+ and hopefully you will have FPV restored.

The picture below shows the connections that I made to the WM301_DM368 PCB. I used 2.54mm pitch pin headers for the serial port connections and the BSEL 3.3V connections since the pads align neatly. The 2-pin header connected to BSEL and 3.3V makes it easy to fit a jumper between the pins so you don't have to solder and de-solder these connections if you need to repeat the programming sequence for any reason.



That's it, procedure complete – I hope you have success in recovering your Wi-Fi module ☺.