Experiment 3
Bus Snooping Attack
on HaHa v3.0 Board

The goal of this experiment is to carry out a bus snooping attack at board level which leads to retrieving secret information from a system through physical access.

Instructor: Dr. Swarup Bhunia
Co-Instructors/TAs: Reiner Dizon-Paradis and Shuo Yang

Case Study

The goal of this experiment is to carry out a bus snooping attack at board level which leads to retrieving secret information from a system through physical access.

Bus sniffing or bus snooping is a technique used in distributed shared memory systems and multiprocessors to achieve cache coherence. Although there is one main memory, there are several caches (one or more per processor), and unless preventative steps are taken, the same memory location may be loaded into two caches and given two different values. To prevent this, every cache controller monitors the bus, listening for broadcasts which may cause it to invalidate its cache line.

Each cache line is in one of the following states: "dirty" (has been updated by the local processor), "valid", "invalid" or "shared". The set of operations is thus: a value can be read or written. Writing it changes the value. Each value is either in main memory (which is very slow to access) or in one or more local caches (which is very fast). When a block is first loaded in the cache it is marked "valid".

On a read miss to the local cache, the read request is broadcast on the bus. All cache controllers monitor the bus. If one has cached that address and it is in the state "dirty", it changes the state to "valid" and sends the copy to requesting node. The "valid" state means that the cache line is current. On a local write miss (an attempt to write that value is made, but it's not in the cache), bus snooping ensures that any copies in other caches are set to "invalid". "Invalid" means that a copy used to exist in the cache, but it is no longer current.

Such an attack can also be done at printed circuit board (PCB) level to extract secret information (such as encryption key, firmware, sensed data) from a PCB through physical access.

Theory Background

Bus snooping attack can be implemented without any help of the software portion of the system. It involves eavesdropping or even altering of data that are transferred between the two or more components of a system.

Usually, in a PCB, there are numerous traces and other wires and ports that are passing signal which can be secret data or commands which are crucial to the system operation. Snooping the traces by soldering additional components or wires allows the attacker to leak those sensitive data. The system can be forced to shut down or do something malicious just by altering the critical signals. In fact, this is how the original Microsoft Xbox was initially hacked. The link between the Southbridge and EEPROM on the Xbox was observed during boot up and modified to allow the hackers access to the protected region of the Xbox's hard drive. In mobile devices, the successful snooping attack may allow the attacker to interfere with data between SoC and DRAM or SoC and NAND Flash. Furthermore, the adversary can capture and alter code and data which is written from SoC to memory. One of those tools was a USB cable with embedded hardware called Cottonmouth-I—a cable that can turn the computer's USB connections into a remote wiretap or even a remote control.

Experiment Set-up: Configuration

The instruments and software needed for this experiment are:

1. The HaHa Board
2. A USB A to B cable
3. An oscilloscope with two probes (or Analog Discovery 2)
4. A computer.
5. Atmel FLIP
6. (Optional) Waveforms 2015

Figure 1 shows the instruments connections. Follow the instructions provided in the next page to set up the connections.

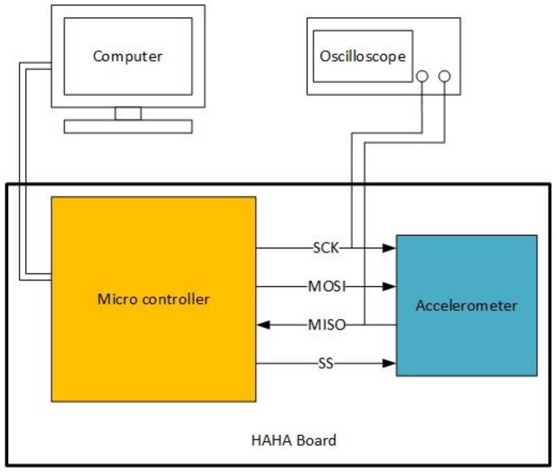


Figure 1 Experiment Set-up

If doing the experiment at home, use the Analog Discovery 2 (Figure 2) instead of an oscilloscope. Analog Discovery 2 is a multi-function instrument that can measure, visualize, and record mixed-signal circuits. Edge students can download the software WaveForms to your computer and connect the computer to the Analog Discovery 2. They together will work as your "oscilloscope" at home. Analog Discovery has 2 analog channels and 16 digital channels (Figure 3). For this experiment, you can use the digital channels to make things easier as we are snooping digital signals. Connect them to the correct pins on the HaHa Board, add the corresponding channels in the WaveForms software and do the measurement. You can see the manual [3] of this device for more detailed reference.

Figure 2 Analog Discovery 2

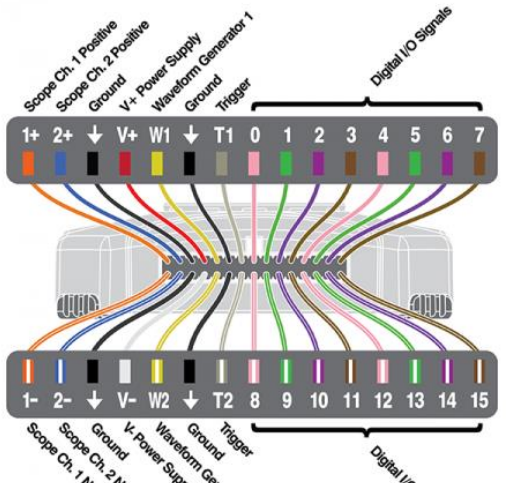


Figure 3 Pinout Diagram

Experiment Set-up: Instructions

1) Download the two hex files needed for this experiment.
   Note: You will only need one of them to run for this experiment.
2) Connect the HaHa Board to the computer through the USB A-to-B cable.
3) Turn ON the HaHa Board, and then program the Microcontroller with the hex file. Refer to Appendix A for programming details.
   a. If your accelerometer was working from the Self-Test, program "U_1.hex"
   b. Otherwise, program "U_2.hex"

   NOTE: You may need to tap *MCU RST* button multiple times before the LEDs show the correct value and change as you move the board.

4) Turn OFF the HaHa Board.
5) Clip the oscilloscope probes to the "MISO" and "SCK" pins of the SPI header (P2) on the HaHa Board. Connect both ground pins to the screws labeled "GND". (Refer to the HaHa User Manual to find the right signal).
6) Turn ON the HaHa Board.
7) Observe the pattern of traffic on the two lines. Refer to Appendix B for details on using Waveforms 2015.
   a. If your accelerometer was working from the Self-Test, tilt the PCB around and see how the traffic changes as well.
   b. Otherwise, you would not need to do anything with the board.

Measurement, Calculation, and Question

Answer the following questions:

1) Which hex file did you use to run this experiment?
2) Which line would you say is the clock line? How fast is the clock running at?
3) Which line would you say is the data line?
4) Is it possible to set up the oscilloscope to both trigger and decode this data line bus? Please submit a screenshot of the scope triggering on a start condition and successfully decoding the transmitted data.
5) Can you describe the general data packet format?
   a. For U_1.hex:
      i. If you tilt the PCB in one direction (try different angles) how does the data packet change? How does it stay the same?
   b. For U_2.hex:
      i. What is the pattern in the data? Does it ever remain the same? Why would a pattern like this show up on the bus?
6) If you didn't have conveniently labeled traces to probe, how you would figure out which traces would you probe if you needed to figure out how the board worked?
7) How would you prevent such attacks? If the data transmitted over this bus was critically important and could not be intercepted, how would you secure the communications? How would you then debug any issues that arose because of your fix?

Optional Follow-up

The previous bus was not the only data communication line on the PCB. There is another much larger bus that enables the Atmel IC and the Altera IC to communicate with each other. Use the oscilloscope, attach the probe ends to all of the pins at the HaHa Board, as well as the pin labeled GND at the top.

Now, repeat the same process as above (moving the board around and observing the data). Answer the following questions:

1) How does this data differ from that of the other bus?
2) How is the bus laid out? What is the data packet like?
3) What is the advantage of a faster, smaller bus over a larger, slower bus? Are there any disadvantages? Provide a real-world example of this difference.

☞ Lab Report Guidelines

1. In your report, give a photo of your whole experiment set up.
2. Give answers to all the questions in the Question part.
3. Give screenshots (or photos) of the oscilloscope when you are bus-snooping the signals and describe what you observed.

References and Further Reading

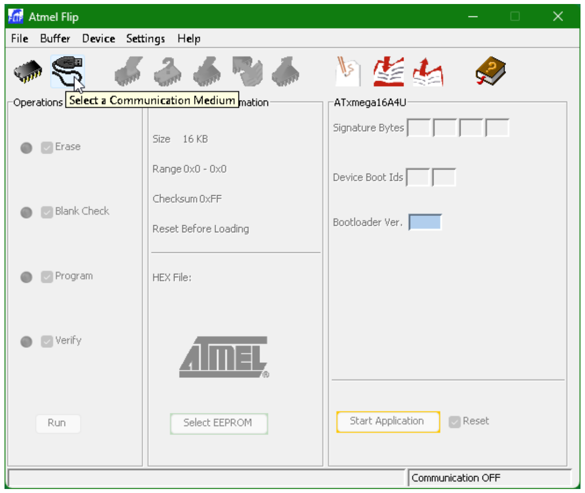
[1] Huang, A. "Hacking the Xbox: An Introduction to Reverse Engineering," No Starch Press, 2003.

[2] Steil, M. "17 Mistakes Microsoft Made in the Xbox Security System," presented at the 22nd Chaos Communication Congress, Berlin, 2005.

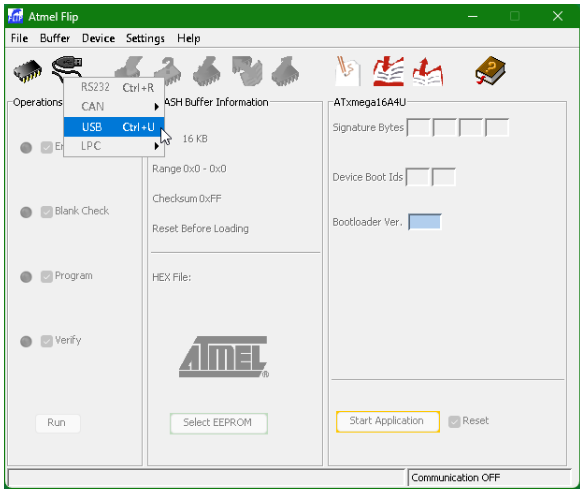
[3] https://reference.digilentinc.com/reference/instrumentation/analog-discovery-2/reference-manual

Appendix A: Programming using Atmel FLIP

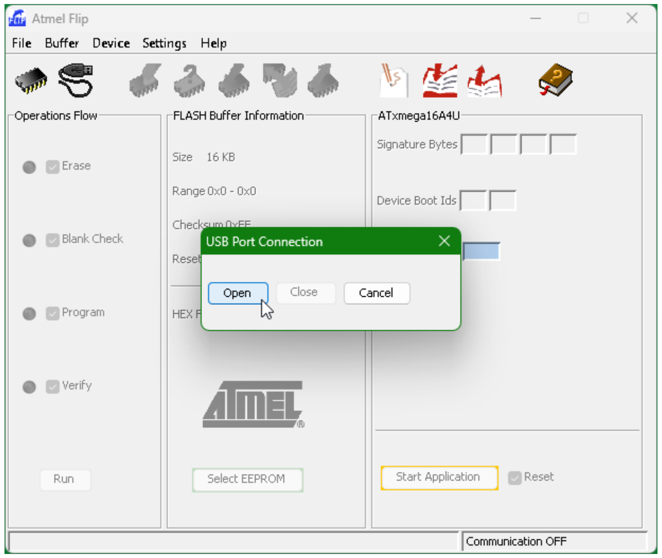
1. Make sure the switch next to the *MCU RST* button is switched down to *BOOT*. Double tap the *MCU RST* button.
2. Open Atmel FLIP. Click on the button (**Select a Communication Medium**) shown below.



3. Select the **USB** option.

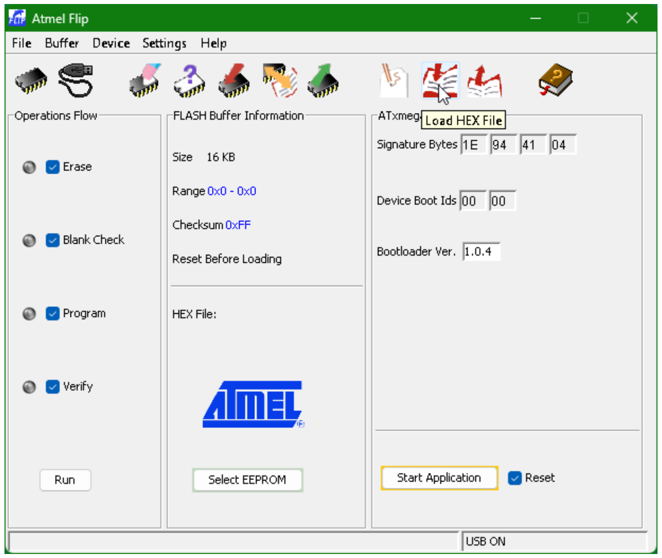


4. Click **Open**.



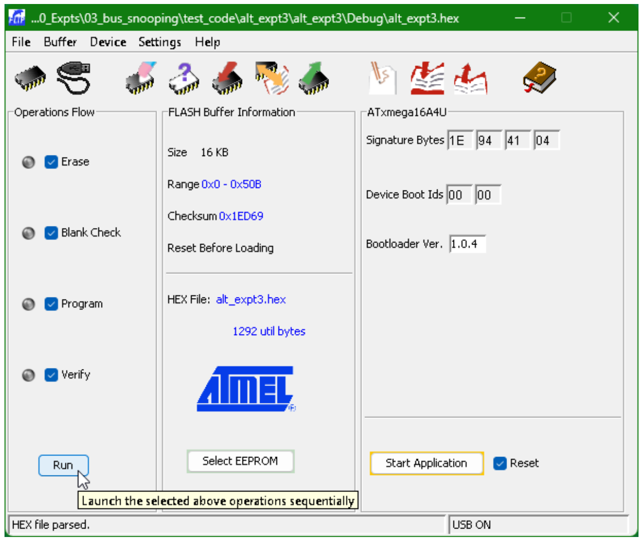
ATxmega16A4U
Signature Bytes
Device Boot Ids
Size 16 KB
Range 0x0 - 0x0
Checksum 0xFF
Communication OFF

5. Click on the button (**Load HEX File**) below. Locate the HEX file and Click **OK**.



Signature Bytes 1E 94 41 04
Device Boot Ids 00 00
Bootloader Ver. 1.0.4
Size 16 KB
Range 0x0 - 0x0
Checksum 0xFF
Reset Before Loading
HEX File:
USB ON

6. Click **Run**.



...0_Expts\03_bus_snooping\test_code\alt_expt3\alt_expt3\Debug\alt_expt3.hex

FLASH Buffer Information

Size 16 KB

Range 0x0 - 0x50B

Checksum 0x1ED69

Reset Before Loading

HEX File: alt_expt3.hex

1292 util bytes

ATxmega16A4U

Signature Bytes 1E 94 41 04

Device Boot Ids 00 00

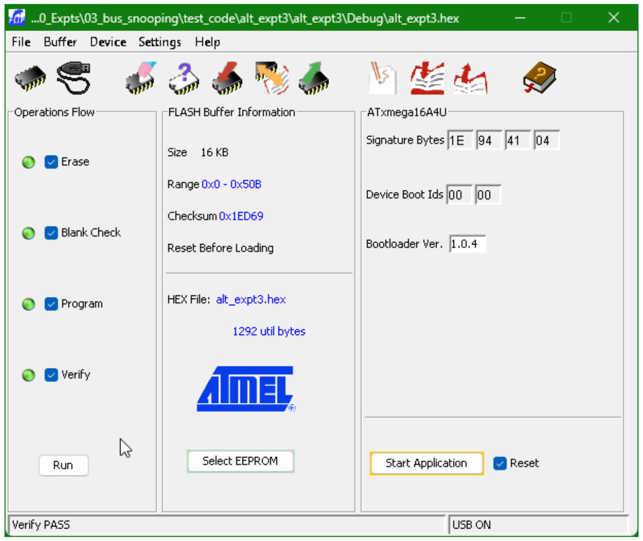
Bootloader Ver. 1.0.4

Launch the selected above operations sequentially

HEX file parsed.

USB ON

7. Your microcontroller program should be loaded. Then, the switch next to the *MCU RST* button should be switched up to *APP*. Double tap the *MCU RST* button.



...0_Expts\03_bus_snooping\test_code\alt_expt3\alt_expt3\Debug\alt_expt3.hex

FLASH Buffer Information

Size 16 KB

Range 0x0 - 0x50B

Checksum 0x1ED69

Reset Before Loading

HEX File: alt_expt3.hex

1292 util bytes

ATxmega16A4U

Signature Bytes 1E 94 41 04

Device Boot Ids 00 00

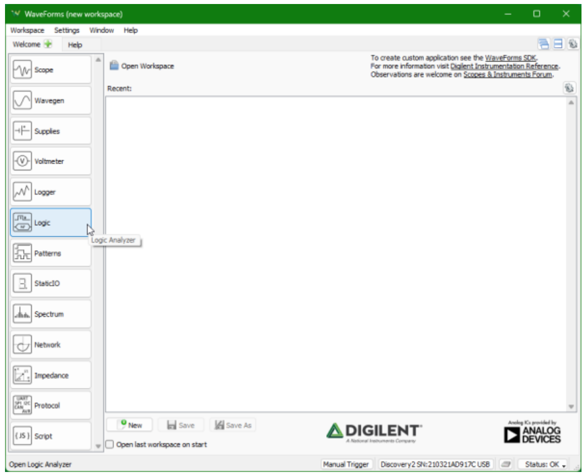
Bootloader Ver. 1.0.4

Verify PASS

USB ON

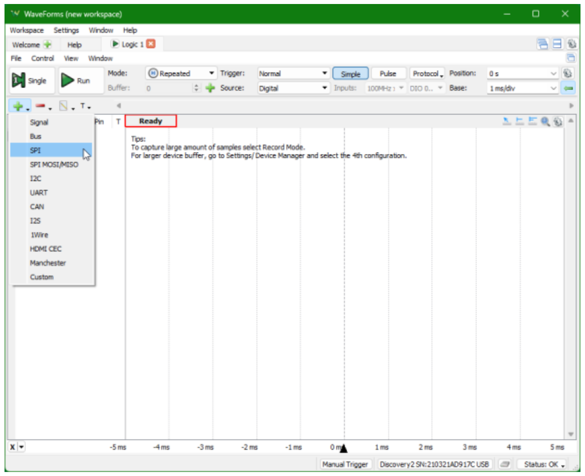
Appendix B: Setting up Waveforms 2015

1. Make sure the micro-USB connector is connected from the AD board to the computer. Connect your pins and ground.
2. Open Waveforms 2015. Click the **Logic** button on the left sidebar.



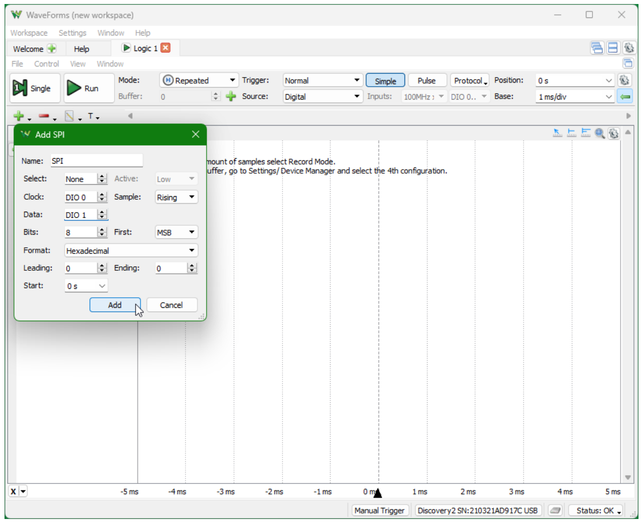
To create custom application see the WaveForms SDK.
For more information visit Digilent Instrumentation Reference.
Observations are welcome on Scopes & Instruments Forum.

3. A new tab will pop up. Click on the green plus sign. Select **SPI**.



Tips:
To capture large amount of samples select Record Mode.
For larger device buffer, go to Settings/ Device Manager and select the 4th configuration.

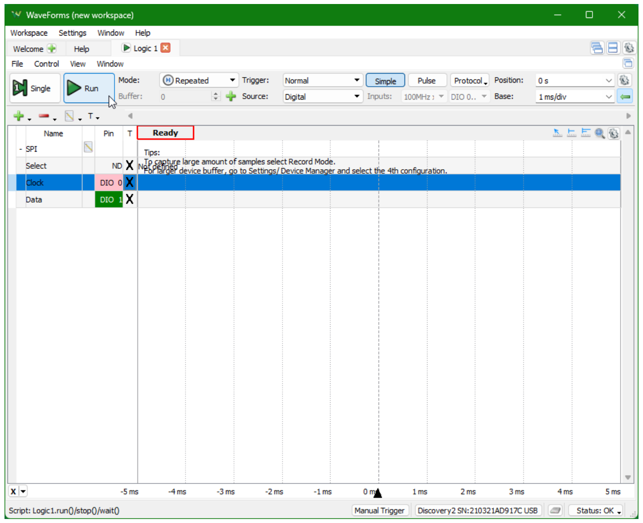
4. Setup the SPI pins to match which wires you chose. Click **Add**.



Add SPI

Name: SPI
Select: None Active: Low
Clock: DIO 0 Sample: Rising
Data: DIO 1
Bits: 8 First: MSB
Format: Hexadecimal
Leading: 0 Ending: 0
Start: 0 s

5. Click **Run**.



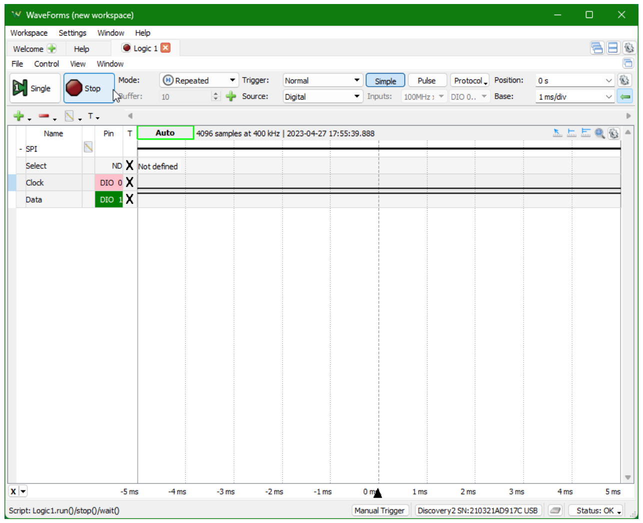
Name	Pin
SPI	
Select	ND
Clock	DIO 0
Data	DIO 1

Ready

Tips:
To capture large amount of samples select Record Mode.
For larger device buffer, go to Settings/ Device Manager and select the 4th configuration.

Script: Logic1.run()/stop()/wait()

6. At any time, you can stop reading the signals by pressing **Stop**.



Mode: Repeated Trigger: Normal Position: 0 s
Buffer: 10 Source: Digital Inputs: 100MHz DIO 0.. Base: 1 ms/div

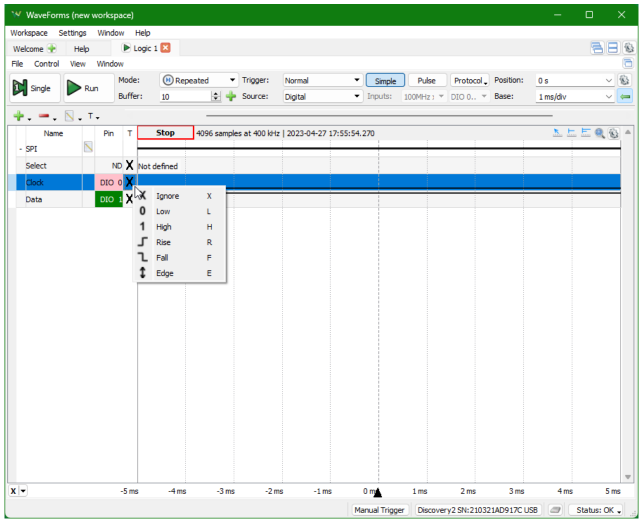
4096 samples at 400 kHz | 2023-04-27 17:55:39.888

Select: Not defined

Script: Logic1.run()/stop()/wait()

Discovery2 SN:210321AD917C USB Status: OK

7. If you need to adjust the trigger options for any signal. Click on the **X** next to the pin number of your signal. Options are shown below:



Mode: Repeated Trigger: Normal Position: 0 s
Buffer: 10 Source: Digital Inputs: 100MHz DIO 0.. Base: 1 ms/div

4096 samples at 400 kHz | 2023-04-27 17:55:54.270

Select: Not defined

Discovery2 SN:210321AD917C USB Status: OK