DICE II EVM

Bring-Up

Firmware Loading, & Test Procedures

CONTENTS

Overview

This document describes how to bring up and test a Rev. 1.4 DICEII EVM board after it's received from the Contract Manufacturer (internal use). This document is included with your EVM as a convenience for you to aid in disaster recovery, i.e. in the case where you need to bring up your EVM after a lost RedBoot sector. For DICEII Firmware SDK developers, this document can also serve as a rough guide for confidence testing your own custom firmware.

Board Bring-up

This requires a JTAG probe, and a Windows PC that has been configured with the DICEII Firmware SDK. Also a Serial Terminal program is required, connected to a native serial port on the PC.

> We recommend the MAJIC probe from Mentor Graphics (EPI Tools), since EPI has developed DICE initialization files which handle the watchdog timer transparently for you.

Macro keys are indicated here in bold brackets, i.e. **[CTRL-E]**. These macros are for internal use where the serial terminal program is already configured.

The sequence for bringing up a fresh EVM is as follows, and detailed below:

Configure your JTAG Probe
Load a RAM version of RedBoot via JTAG
Load a ROM-able version of RedBoot to RAM via serial terminal and write it to flash memory
Initialize the flash file system
Create a setup file for runtime defaults
Load an Application image to RAM via serial terminal and write to flash memory
Configure to Auto-run the application (optional)

Configure your JTAG probe

1) Configure your JTAG probe.

   Here are the general steps required to configure the probe. Consult the installation documents for your MAJIC for detailed info.

   a) Insert the EPI Development Tools Installation CD.
   b) Run setup.exe if your computer is not set up to AutoPlay the application.
   c) Choose installation for ARM/XSCALE
   d) Enter your installation key from Mentor Graphics, or install the evaluation version if you don't have a permanent key yet.

e) When choosing install options, you can deselect all options to save disc space. If using the MAJIC with other platforms, select options as needed.
f) Launch the MAJIC Setup Wizard, as part of the installation or later using the Start Menu shortcut.
g) In the Wizard, assign your MAJIC a fixed IP address.
   Connect the serial cable from the MAJIC to your computer first.
   Enter IP Settings that will work within your LAN.
   Complete the Wizard with defaults.
h) Again in the MAJIC Setup Wizard, set up your Debug Environment
   Choose GDB as your debugger.
   Give the project a name
   Choose ARM7TDMI as your processor type, and leave the defaults as they are.
   Use the static IP address as entered earlier.
   Use Existing Startup File, and browse to the "wavefront_dice" directory in the targets for the EPI tools. Select startice.cmd
   Go through the Wizard and Perform Actions, then exit.

You will now have a shortcut on your desktop for the MDI server. This program serves as a bridge between the DICE debug environment (gdb, Insight, etc.) and the MAJIC. Whenever you want to debug your DICE target from this computer, you will run this program first. As you work, keep an eye on the output in the MDI Server window as it will let you know if there is something wrong in the connection.

Important: The DICE development tools and the EPI EDT tools both install a version of cygwin1.dll. When using these together, please rename the **cygwin1.dll** that is installed by the EPI tools to something else, i.e. **cygwin1EPI.dll**. The default installation location for this file is: "C:\Program Files\EPITools\edta23b\bin"

This will allow both tools to run together.

Also note that the MDI Server uses TCP port 2345 by default. Your port number in the debugger setup must match the port used by MDI. See below.

Load and run a RAM version of RedBoot via JTAG

2) Connect the JTAG Probe to J24 (note pin1 on the header and cable).
3) Connect a serial cable to J16 and to a computer with a native serial port.
4) Open a serial terminal program and set it to **115200, 8, N, 1, No HS.**
   Use VT220 emulation and send "Terminal Keys" from the keyboard to CLI.
5) Power the EVM, then power the JTAG probe.

If you see a prompt at this point, the device already has firmware loaded. At this point make sure the prompt is **RedBoot>** and not an application CLI. In order to use JTAG to communicate with the EVM to load new firmware, it must be running RedBoot. If the board has been previously set up to automatically run an application, do the following:

a) get to a **RebBoot>** prompt by hitting **Ctrl+C** directly after resetting the EVM.
b) type "**fco**"
c) when prompted, replace "**true**" with "**false**"
d) enter "**y**"
e) reset the board
f) continue with step 6 below

* If using a MAJIC probe, run the MDI server now.

6) Open a **bash** shell
7) Type "**arm-elf-insight.exe**" to start **gdb**
8) Open the file **redbootram.elf** (or similar)
9) Configure your *Target Settings* for this file in Insight as follows:
   Target: GDBServer/TCP
   Hostname: 127.0.0.1
   Port: 2345
   Check "Attach to target"
   Check "Run Program"
   Select "Continue from last Stop"
10) Now click the Run icon to perform the selections you just made in the Target Settings dialog.
11) The RAM boot image should now be loaded and running on the board. You should see a **RedBoot>** prompt in the terminal program at this point.

Load the ROM-able boot image via Serial Terminal and write it to FLASH

12) **[CTRL-O]**
    "**load –r –m ymodem –b 0x30000**"
13) Start the transfer of ROM version of RedBoot (using "1K Xmodem")
    The file name is **redbootrom.bin**
14) **[CTRL-T]**
    "**fis write –b 0x30000 –l 0x20000 –f 0x4000000**" to write the ROM version of RedBoot to the flash.

Make sure to enter the flash address correctly (-f parameter). That's a 4 with 6 zero's. Otherwise you will have to reinitialize your board flash memory using a JTAG interface. Also make sure to write the correct memory offset from RAM to flash, i.e. use the same address in the –b parameter for the load and the write commands.

15) Power off the JTAG probe and remove it from the EVM
16) In the bash shell, quit **gdb** by hitting **CTRL-C** a few times. If you see a **(GDB)** prompt, hit "**q**". Otherwise confirm any other exit messages.
17) Reset the EVM to boot from the flash image.

If you are updating an EVM that had existing **RedBoot** and **dice** application code previous to rev. 1.0, which used **57600** baud rate then you'll now need to set your serial terminal settings to the new faster rate: **115200, 8, N, 1, No HS**.

You should see a **RedBoot>** prompt again.

Initialize the flash file system

18) Type "**fis init -f**" to initialize the flash
    Note: This step is not necessary if your flash has been previously initialized.

Create a "setup" flash file containing a WWUID for the device and other runtime defaults.

.

19) Edit **\dice\redboot\config.txt**
The file should look similar to this, including a final carriage-return:

```
SERIAL_NO=54321
HPLL_CLK=50000000
```

SERIAL_NO is the lower 5 digits from the yellow serial number sticker on the EVM. This is used to calculate the lower bytes of the device WWUID.
HPLL_CLK the JET-PLL is running off of the DICE crystal and in order for the sample counter to be correct it needs to know which crystal frequency is used. The default is 25 MHz, but it can be overwritten with the HPLL_CLK parameter. The frequency should be two times the crystal used. In case of a 24.576MHz crystal it should be stated as: **HPLL_CLK=49152000**

20) [**CTRL-Y**]
"**load -r –m ymodem –b 0x30000**"
21) Transfer **config.txt** (using "1K Xmodem")
22) [**CTRL-W**]
"**fis create –b 0x30000 –l 0x40 -e 0x0 –r 0x0 setup**"
The **–l** length parameter should be at least the length returned by the load command (end – start +1). **0x40** is sufficient here, and RedBoot will round up to the nearest sector size anyway.

You can confirm that the setup section is written to flash by examining the flash memory. If you have gone through the steps above, the **setup** section will be located in flash memory at **0x04090000**:

23) **[CTRL-U]**
"**x –b 0x4090000 –l 0x40 -1**"

This should show something similar to this:

```
RedBoot> x -b 0x4090000 -l 0x40 -1
04090000: 53 45 52 49 41 4C 5F 4E  4F 3D 35 30 30 30 33 0D  |SERIAL_NO=50003.|
04090010: 0A 48 50 4C 4C 5F 43 4C  4B 3D 35 30 30 30 30 30  |.HPLL_CLK=500000|
04090020: 30 30 0D 0A 54 01 03 00  78 01 03 00 A4 01 03 00  |00..T...x.......|
04090030: C0 01 03 00 00 00 00 00  C0 02 03 00 88 02 03 00  |................|
RedBoot>
```

Load an Application image to RAM via serial terminal and write to flash memory

24) [**CTRL-Y**]
"**load -r –m ymodem –b 0x30000**"
25) Transfer the application file **\dice\redboot\dice_3_0_1.bin** (or similar) (using "1K Xmodem")
26) [**CTRL-P**]
"**fis create –b 0x30000 –l 0x80000 –e 0x30040 –r 0x30000 dice**"

Configure the Board to automatically run the application image

This is optional. If you will be debugging firmware using gdb over a serial line, then the EVM must not be configured to automatically run the application.

27) At the **RedBoot>** prompt, type "**fco**"
28) Replace **false** with **true**
29) Enter the following two lines at the **>>** prompts, followed by a **return**:
fis load dice
go

30) Enter a value of **20** for the boot delay. This is equivalent to 2 seconds (the ms resolution indicated is not correct).
31) Enter '**y**' to confirm
32) Reset the board.

The EVM should run **RedBoot** and then auto-run the **dice** application at this point. Look at the splash screen. The **Board S/N** should be the same serial number entered in the **setup** section, from **config.txt**. ***Note that this value is displayed in hexadecimal here.***

```
>splash

*********************************************************
* TC Command Line Interface System                      *
* Copyright 2003-2005 by TC Electronic A/S              *
*********************************************************
* Running Dice II 1394 Appl                             *
* Board S/N: 00002718, Appl Ver.: 3.01 RELEASE          *
*                        - built on 17:14:09, Jul 26 2006 *
*    MIDI is enabled.                                   *
*********************************************************
* Target: DICE II Evaluation Board                      *
* Driver: DiceDriver                                    *
*********************************************************
>
```

The serial number will also become the lower bytes of the WWUID for the device. *The upper bytes in the EVM are set to a temporary number to be used only for evaluation, development and test. Shipping products must contain the OUI of the manufacturer.*

[CTRL-D] "**lal.getwwuid**"

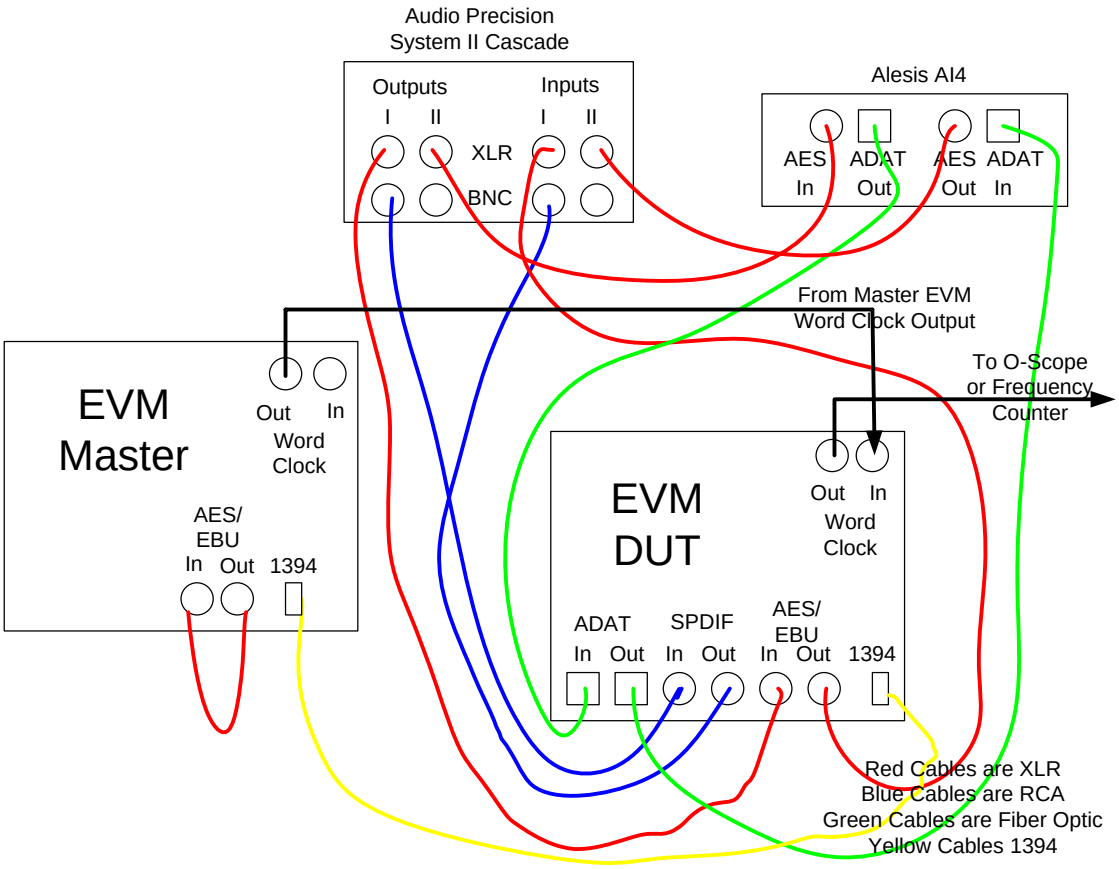
```
>lal.getwwuid
WWUID: 0x0013f004 0x004 0c353
>
```

If any problems were encountered, note them on the tag and set the EVM aside, otherwise note: "fw load OK"

Testing Audio I/O, Routing, and Clocking

The sequence for testing the EVM audio features is as follows:

Set up the DUT in a test harness by connecting audio ports to the AP, ADAT ports to the AI4, Word Clock Out to an oscilloscope and Word Clock In to another working EVM (Master). Connect the AES in and out on the Master with an XLR cable as shown below. Connect the DUT and Master to serial terminal programs.



Audio Interface and Clock Domain Tests

SPDIF coaxial AES1 and PLL1/2

- Config AP to I/O on BNC
- Reset DUT

Clock Domain 0

1. DUT CLI: **[CTRL-A]**
   "**dal.create 0 low_mid "aes" "aes"**" to set up the router 0 interface
   "**dal.route 0 aes 0 aes 0 t8**" to set router 0 to loop the AES interface
   "**dal.clock 0 aes1 any**" to set router 0 to sync to the AES Rx interface
   "**dal.start 0**" to start router 0

   Vary sample rate on AP between 32k, 44k1, 48k, 88k2, 96k and verify loopback (also unplug and replug SPDIF IN CABLE a few times)

Clock domain 1

1. DUT CLI: **[CTRL-F]**
   "**dal.destroy 0**" to stop the router 0 interface
   "**dal.create 1 low_mid "aes" "aes"** " to set up the router 1 interface
   "**dal.route 1 aes 0 aes 0 t8**" to set router 1 to loop the AES interface
   "**dal.clock 1 aes1 any**" to set router 1 to sync to the AES Rx interface
   "**dal.start 1**" to start router 1

   Vary sample rate on AP between 32k, 44k1, 48k, 88k2, 96k and verify loopback (also unplug and replug SPDIF IN CABLE a few times)

AES/EBU AES0 and PLL1/2

- Config AP to I/O on XLR I
- Reset the DUT

Clock Domain 0

1. DUT CLI: **[CTRL-A]**
   "**dal.create 0 low_mid "aes" "aes"** " to set up the router 0 interface
   "**dal.route 0 aes 0 aes 0 t8**" to set router 0 to loop the AES interface
   "**dal.clock 0 aes0 any**" to set router 0 to sync to the AES Rx interface
   "**dal.start 0**" to start router 0

   Vary sample rate on AP between 32k, 44k1, 48k, 88k2, 96k and verify loopback (also unplug and replug AES IN CABLE a few times)

Clock Domain 1

2. DUT CLI: **[CTRL-F]**
   "**dal.destroy 0**" to stop the router 0 interface

"**dal.create 1 low_mid "aes" "aes"** " to set up the router 1 interface
"**dal.route 1 aes 0 aes 0 t8**" to set router 1 to loop the AES interface
"**dal.clock 1 aes0 any**" to set router 1 to sync to the AES Rx interface
"**dal.start 1**" to start router 1

Vary sample rate on AP between 32k, 44k1, 48k, 88k2, 96k and verify loopback (also unplug and replug AES IN CABLE a few times)

ADAT

- Config AP to I/O on XLR II
- Reset DUT

1. DUT CLI: **[CTRL-J]**
   "**dal.create 0 low_mid "adat" "adat"** " to set up the router 0 interface
   "**dal.route 0 adat 0 adat 0 t8**" to set router 0 to loop the AES interface
   "**dal.clock 0 adat any**" to set router 0 to sync to the AES Rx interface
   "**dal.start 0**" to start router 0

   Vary sample rate on AP between 32, 44, 48 and verify loopback (also unplug and replug OPTICAL IN CABLE a few times)

Word Clock Out

- Reset the DUT

1. DUT CLI: **[CTRL-L]**
   "**dal.clock 0 int 32k**" to connect clock domain 0 to internal 32k rate.

   Verify word on the oscilloscope is 32kHz

2. DUT CLI: **[CTRL-Z]**
   "**dal.clock 0 int 44k1**" to connect clock domain 0 to internal 44k1 rate.

   Verify word on the oscilloscope is 44.1kHz

3. DUT CLI: **[CTRL-X]**
   "**dal.clock 0 int 48k**" to connect clock domain 0 to internal 48k rate.

   Verify word on the oscilloscope is 48kHz

4. DUT CLI: **[CTRL-K]**
   "**dal.clock 0 int 88k2**" to connect clock domain 0 to internal 88k2 rate.

   Verify word on the oscilloscope is 88.2kHz

5. DUT CLI: **[CTRL-V]**
   "**dal.clock 0 int 96k**" to connect clock domain 0 to internal 96k rate.

Verify word on the oscilloscope is 96Hz

Word Clock In

- Configure the AP to I/O on XLR I
- Reset the DUT

1. DUT CLI: **[CTRL-J] [CTRL-B]**
   "**sys.setmask 0x4000**" to display locking information
   "**dal.clock 0 wc any**" to connect clock domain 0 to Word Clock In
   Master CLI: **[CTRL-J] [Ctrl-L]**
   "**dal.clock 0 int 32k**" output a 32kHz clock on the reference EVM
   Verify sample rate on AP is 32kHz

2. Master CLI: **[Ctrl-Z]**
   "**dal.clock 0 int 44k1**" output a 44.1kHz clock on the reference EVM
   Verify sample rate on AP is 44.1kHz

3. Master CLI: **[Ctrl-X]**
   "**dal.clock 0 int 48k**" output a 48kHz clock on the reference EVM
   Verify sample rate on AP is 48kHz

4. Master CLI: **[Ctrl-K]**
   "**dal.clock 0 int 88k2**" output a 88.2kHz clock on the reference EVM
   Verify sample rate on AP is 88.2kHz

5. Master CLI: **[Ctrl-V]**
   "**dal.clock 0 int 96k**" output a 96kHz clock on the reference EVM
   Verify sample rate on AP is 96kHz

If any problems were encountered, note them on the tag, otherwise note: "audio tests OK"

Testing MIDI I/O

The MIDI interface may be checked simply by connecting a MIDI cable loop from the MIDI input to the MIDI output. A byte can be written to the MIDI output register, and then read from the MIDI input register.

- Reset the DUT

1. Place the jumper across pins 2 and 3 of UART1 Select (J13), this selects MIDI instead of a second UART.
2. Make sure a MIDI cable is connected between the MIDI In and Out ports.
3. **[CTRL-Q]**
   "**set 0xbd00000c 0x80**" to allow the Divisor Latch (DL) register to be set
   "**set 0xbd000000 0x50**" which selects a MIDI baud rate
   "**set 0xbd00000c 0x00**" to disallow the DL register to be set again
4. **[CTRL-R]**
   "**set 0xbd000000 0x5**" which sets the transmitted MIDI byte to be **0x5**
5. **[CTRL-E]**
   "**get 0xbd000000**" to get the received MIDI byte
6. Verify that the received MIDI byte matches the transmitted MIDI byte: **0x5**
9. Move the jumper on J13 back to connect pins 1 and 2.

If any problems were encountered, note them on the tag, otherwise note: "MIDI tests OK"

Testing IEEE1394 I/O

The 1394 interface can be tested easily using two EVM boards. One board uses a router connection between the AVS and AES interfaces, and the other board uses the router to form an AVS loop. Audio sample rate can be 48k for the purpose of the test since the 1394 speed will not be affected anyway. Set the AP to output on XLR I, 48k.

The audio data flow is as follows, where board (a) is the DUT and (b) is the Master:

input audio to DUT -> AES Rx (a) -> Router (a) -> AVS Tx (a)
-> AVS Rx (b) -> Router (b) -> AES Tx (b)
-> loopback XLR cable on Master
-> AES Rx (b) -> Router (b) -> AVS Tx (b)
-> AVS Rx (a) -> Router (a) -> AES Tx (a) -> output audio from DUT

The procedure for Master EVM board (b) shown below, can be cut and paste into the Terminal program for execution.

- Reset Master

sys.setmask 0x4000
dal.destroy 0
avs.itc 1 rx 0
avs.its 1 rx 48k
avs.dbs 1 rx 8
avs.it 1 rx start
avs.itc 1 tx 1
avs.its 1 tx 48k
avs.dbs 1 tx 8
avs.format 1 tx label
avs.speed 1 s400
dal.create 0 low_mid "aes avs1" "aes avs1"
dal.route 0 aes 0 avs1 0 t8
dal.route 0 avs1 0 aes 0 t8
dal.clock 0 avs1 48k
dal.start 0
avs.it 1 tx start

The procedure for DUT EVM board (a) shown below, can be cut and paste into the Terminal program for execution. This tests the AES1 interface.

- Reset DUT **[CTRL-S]**

sys.setmask 0x4000
dal.destroy 0
avs.itc 1 rx 1
avs.its 1 rx 48k
avs.dbs 1 rx 8
avs.it 1 rx start
avs.itc 1 tx 0
avs.its 1 tx 48k

avs.dbs 1 tx 8
avs.format 1 tx label
avs.speed 1 s400
dal.create 0 low_mid "aes avs1" "aes avs1"
dal.route 0 aes 0 avs1 0 t8
dal.route 0 avs1 0 aes 0 t8
dal.clock 0 aes0 48k
dal.start 0
avs.it 1 tx start
Now verify audio integrity for the AES audio data. While the audio is streaming the XLR cable on the Master should be unplugged/replugged. Again verify audio integrity for the AES audio data.

Full AVS testing

The steps above have tested AVS1 on the DUT. To test AVS2-4, continue with the configuration above and use the following sequences n he DUT:

AES2

DUT for AES_AVS2 rx and tx:
- Reset the Master, resend the same sequence as above.
- Reset the DUT

sys.setmask 0x4000
dal.destroy 0
avs.itc 2 rx 1
avs.its 2 rx 48k
avs.dbs 2 rx 8
avs.it 2 rx start
avs.itc 2 tx 0
avs.its 2 tx 48k
avs.dbs 2 tx 8
avs.format 2 tx label
avs.speed 2 s400
dal.create 0 low_mid "aes avs2" "aes avs2"
dal.route 0 aes 0 avs2 0 t8
dal.route 0 avs2 0 aes 0 t8
dal.clock 0 aes0 48k
dal.start 0
avs.it 2 tx start

AES3

DUT for AES_AVS3 rx and AES_AVS1 tx:
- Reset the Master, resend the same sequence as above.
- Reset the DUT

sys.setmask 0x4000
dal.destroy 0

avs.itc 3 rx 1
avs.its 3 rx 48k
avs.dbs 3 rx 8
avs.it 3 rx start
avs.itc 1 tx 0
avs.its 1 tx 48k
avs.dbs 1 tx 8
avs.format 1 tx label
avs.speed 1 s400
dal.create 0 low_mid "aes avs3" "aes avs1"
dal.route 0 avs1 0 aes 0 t8
dal.route 0 aes 0 avs3 0 t8
dal.clock 0 aesAny 48k
dal.start 0
avs.it 1 tx start

AES4

DUT for AES_AVS4 rx and AES_AVS2 tx:

- Reset the Master, resend the same sequence as above.
- Reset the DUT

sys.setmask 0x4000
dal.destroy 0
avs.itc 4 rx 1
avs.its 4 rx 48k
avs.dbs 4 rx 8
avs.it 4 rx start
avs.itc 2 tx 0
avs.its 2 tx 48k
avs.dbs 2 tx 8
avs.format 2 tx label
avs.speed 2 s400
dal.create 0 low_mid "aes avs4" "aes avs2"
dal.route 0 avs2 0 aes 0 t8
dal.route 0 aes 0 avs4 0 t8
dal.clock 0 aesany 48k
dal.start 0
avs.it 2 tx start

If any problems were encountered, note them on the tag, and put the board back in its static bag and put it in the storage closet on the bad boards shelf. Otherwise note: “1394 tests OK”

If all parts have been added to the board, and all tests passed:
Note this on the tag
Remove the tag
Put the EVM in a static bag and tape the toe-tag to the bag

Done!

Testing Quick Reference

SPDIF

- Config AP to I/O on BNC
- Reset DUT **[CTRL-S]**

Clock Domain 0

1. DUT CLI: **[CTRL-A]**
   Vary sample rate on AP between 32, 44, 48, 88, 96 and verify loopback (also unplug and replug SPDIF IN CABLE a few times)

Clock Domain 1

1. DUT CLI: **[CTRL-F]**
   Vary sample rate on AP between 32, 44, 48, 88, 96 and verify loopback (also unplug and replug SPDIF IN CABLE a few times)

AES/EBU

- Config AP to I/O on XLR I
- Reset DUT **[CTRL-S]**

Clock Domain 0

1. DUT CLI: **[CTRL-A]**
   Vary sample rate on AP between 32, 44, 48, 88, 96 and verify loopback (also unplug and replug AES IN CABLE a few times)

Clock Domain 1

2. DUT CLI: **[CTRL-F]**
   Vary sample rate on AP between 32, 44, 88, 96 and verify loopback (also unplug and replug AES IN CABLE a few times)

ADAT

- Config AP to I/O on XLR II
- Reset DUT **[CTRL-S]**

1. DUT CLI: **[CTRL-J]**
   Vary sample rate on AP between 32, 44, 48 and verify loopback (also unplug and replug OPTICAL IN CABLE a few times)

WORD CLOCK OUT

- Reset DUT **[CTRL-S]**

1. DUT CLI: **[CTRL-J] [CTRL-L]** Verify word clock on O-Scope is 32kHz
2. DUT CLI: **[CTRL-J] [CTRL-Z]** Verify word clock on O-Scope is 44.1kHz
3. DUT CLI: **[CTRL-X]** Verify word clock on O-Scope is 48kHz
4. DUT CLI: **[CTRL-K]** Verify word clock on O-Scope is 88.2kHz
5. DUT CLI: **[CTRL-V]** Verify word clock on O-Scope is 96kHz

WORD CLOCK IN

- Config AP to I/O on XLR I
- Reset DUT **[CTRL-S]**

1. DUT CLI: **[CTRL-B]**
   MASTER CLI: **[CTRL-L]** Verify sample rate on AP is 32kHz
2. MASTER CLI: **[CTRL-Z]** Verify sample rate on AP is 44.1kHz
3. MASTER CLI: **[CTRL-X]** Verify sample rate on AP is 48kHz
4. MASTER CLI: **[CTRL-K]** Verify sample rate on AP is 88.2kHz
5. MASTER CLI: **[CTRL-V]** Verify sample rate on AP is 96kHz

MIDI

- Reset DUT **[CTRL-S]**, **make sure J13 is connecting pins 2 and 3 for MIDI.**

1. DUT CLI: **[CTRL-Q]**
   DUT CLI: **[CTRL-R]**
   DUT CLI: **[CTRL-E]** Verify code read back is 0x00000005

   Move Jumper on J13 back to connect pins 1 and 2.

1392 AVS

- Reset Master
- Reset DUT

AES1

1. Master CLI: Send file EVM_MASTER_1394.txt
2. DUT CLI: Send File EVM_DUT_1394_AVS1.txt
   Verify Audio from DUT

AES2

3. Reset Master
4. Reset DUT
5. Master CLI: Send file EVM_MASTER_1394.txt
6. DUT CLI: Send File EVM_DUT_1394_AVS2.txt
   Verify Audio from DUT

AES3

7. Reset Master
8. Reset DUT
9. Master CLI: Send file EVM_MASTER_1394.txt
10. DUT CLI: Send File EVM_DUT_1394_AVS3.txt
    Verify Audio from DUT

AES4

11. Reset Master
12. Reset DUT
13. Master CLI: Send file EVM_MASTER_1394.txt
14. DUT CLI: Send File EVM_DUT_1394_AVS3.txt
    Verify Audio from DUT