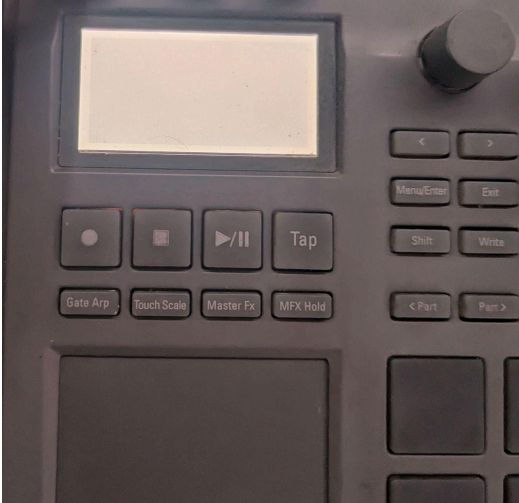


Process

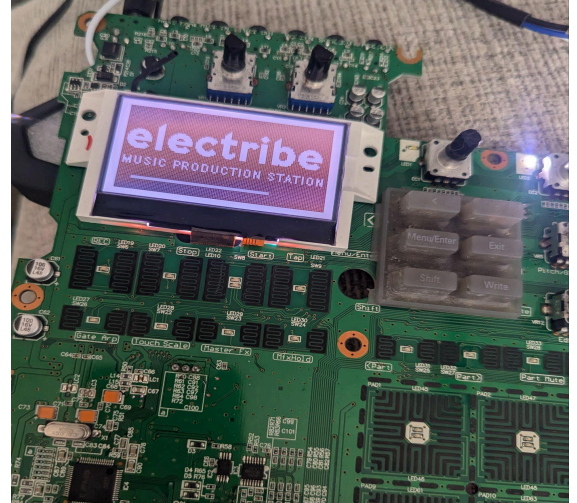
Fixed my bricked Korg Electribe (E2) with my flipper

I went

From: This



To: This



Process:

- 1) I used [OpenOCD](#) (More specifically [xpack](#) since it had better installation instructions) as my means of MCU manipulation. The manual can be found [here](#). See [OpenOCD Notes](#) for more information on setting up openOCD (xpm flavor) and dont forget to satisfy the prerequisites.
- 2) Wire up your probes to the correct headers on both your target and F0.
 - a) Make sure to identify each pin individually in your [config file](#) - Raspberry pi (as is used in the [Debrick guide](#)) does not require this I guess. This took me a few days to figure out.
For the F0 it is important to identify them one at a time, each on their own line.
- 3) With your target powered on and DAP-Link open and configured on your F0 and connected via USB to your computer you are now ready to begin.
- 4) In a cmd prompt (in your projects folder) type: `openocd -f korg.cfg -f am1802.cfg` so it loads the appropriate .cfg files for your interface (F0) and the MCU you want to analyze.
- 5) If the operation is successfully performed you will get a print out such as this:
DEPRECATED! use 'gdb memory_map', not 'gdb_memory_map'
DEPRECATED! use 'gdb flash_program', not 'gdb_flash_program'
trst_and_srst separate srst_gates_jtag trst_push_pull srst_open_drain connect_deassert_srst
DEPRECATED! use 'cmsis-dap backend', not 'cmsis_dap_backend'

Fixed my bricked Korg Electribe (E2) with my flipper

```
DEPRECATED! use 'gdb breakpoint_override', not 'gdb_breakpoint_override'
force hard breakpoints
jtag_ntrst_assert_width: 100
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : Using CMSIS-DAPv2 interface with VID:PID=0x0483:0x5740, serial=DAP_V1Ru7EnT
Info : CMSIS-DAP: SWD supported
Info : CMSIS-DAP: JTAG supported
Info : CMSIS-DAP: FW Version = 2.0.0
Info : CMSIS-DAP: Serial# = DAP_V1Ru7EnT
Info : CMSIS-DAP: Interface Initialised (JTAG)
Info : SWCLK/TCK = 1 SWDIO/TMS = 1 TDI = 1 TDO = 1 nTRST = 0 nRESET = 1
Info : CMSIS-DAP: Interface ready
Info : clock speed 1000 kHz
Info : cmsis-dap JTAG TLR_RESET
Info : cmsis-dap JTAG TLR_RESET
Info : JTAG tap: omapl138.jrc tap/device found: 0x1b7d102f (mfg: 0x017 (Texas Instruments),
part: 0xb7d1, ver: 0x1)
Info : JTAG tap: omapl138.etb enabled
Info : JTAG tap: omapl138.arm enabled
Info : Embedded ICE version 6
Info : omapl138.arm: hardware has 2 breakpoint/watchpoint units
Info : ETM v1.3
Info : [omapl138.arm] Examination succeed
Info : [omapl138.arm] starting gdb server on 3333
Info : Listening on port 3333 for gdb connections
```

- 6) While leaving this cmd prompt instance open and connected, start a new instance of cmd prompt (Windows key + r -> Then type *cmd* and hit enter) and type *telnet localhost 4444* to begin working with your desired MCU.

From here everything will be entered in this telnet cmd prompt instance.

- 7) Input: *halt*

- a) Input: *dump_image SBL.bin 0x80000000 0x5e40* and wait for that to complete

Fixed my bricked Korg Electribe (E2) with my flipper

- b) From here my experience differs from the [Debrick guide](#). I found that inputting the step command was favorable to allowing the processes to run on their own because after allowing the process to resume I did not notice any progress. Below is taken from the guide:

With openocd still running, power off bricked electribe and power it back on.

Then stop openocd with `ct1 + c` and run it again.

This time the electribe should be waiting in the ROM bootloader (RBL) in ARM RAM somewhere around 0xffffd5990-0xffffd599c.

- 8) In other words: after the file has been saved, turn off the E2. power it back on.
- a) In the cmd prompt you are inputting openOCD commands (not the second telnet cmd prompt instance) press `ctrl+c` to stop openocd. Then start it again by pressing the up arrow or re-typing `openocd -f korg.cfg -f am1802.cfg`.
 - b) In the telnet cmd prompt instance hit the up arrow or re-type `telnet localhost 4444` to re-establish the telnet connection and
- 9) Input: `halt` - In my case I never got to the RAM location 0xffffd5990-0xffffd599c. So I . . .
- a) Input `step 0x80000000`
 - b) Input: `load_image SBL.bin 0x80000000` - and wait for that to complete
 - c) Input: `step 0xc0000000`
 - d) Input: `load_image SYSTEM.bin 0xc0000000` - and wait for that to complete.
- 10) I found that if I then Input: `step 0x00000000` and then Input: `resume` my E2 suddenly came to life. However, you may have better luck following the [Debrick guide](#).

OpenOCD Notes

OpenOCD (xpm) NOTES

- 1) Once installed make sure the global paths are set up - this made calling for specific .cfg file MUCH easier for me since I could place everything I needed in one project folder.
 - a) Verify path setup by opening a command-line interface (cmd prompt/bash shell/ terminal whatever you are using) in your desired folder and typing `openocd --version`

You will see something like:

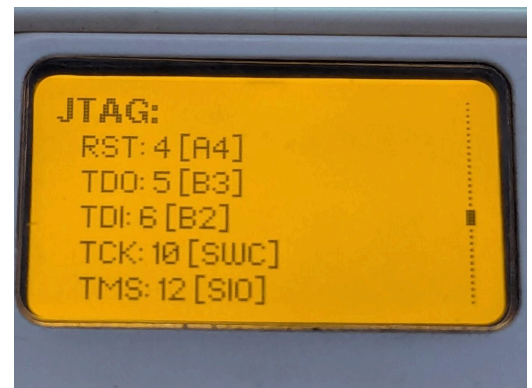
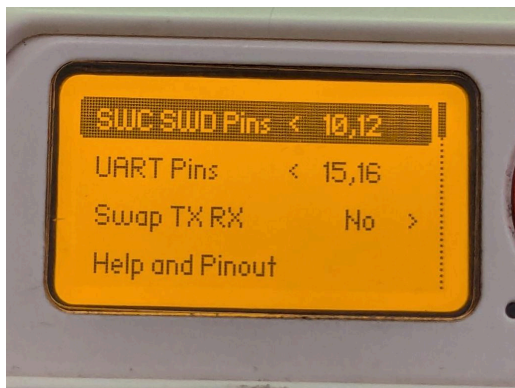
xPack Open On-Chip Debugger 0.12.0+dev-01557-gdd1758272-dirty (2024-04-02-07:27)
Licensed under GNU GPL v2
For bug reports, read
<http://openocd.org/doc/doxygen/bugs.html>

- 2) If you are planning on performing these actions on a windows based machine but utilizing WSL (Windows system for Linux) make sure that your USB ports are available to the distro you have installed.
 - a) I [installed usbipd](#)
- 3) You may need to enable telnet in cmd prompt you can do so by:
 - a) Opening the command prompt .
 - b) Type `pkgmgr /iu:"TelnetClient"`
 - c) Restart the command prompt
 - d) Type `telnet` to open the Microsoft Telnet Client

Setting up the leads:

Open the app - apps/GPIO/Debug/[SWD-JTAG]DAP Link press left to Config

For greater spacing I switched my swc/swd pins from 2,3 to 10,12 and it will tell you what pins to connect your leads to by clicking on help and pinout



OpenOCD (xpm) NOTES

- 4) In the Command-Line interface of your choosing navigate to your projects directory (I have hosted the files I needed to successfully achieve my goal [here](#) your files may differ) and ensure it has been “initialized” by typing `xpm init`. This will ensure the required `.package.json` file is present. You may have to take a few extra steps described [here](#). Once completed you should now be ready to rock.

usbipd

USBIPD WIN10 INSTALLATION NOTES

In and Admin instance or powershell Input:

`wsl --shutdown` to shutdown instances of WSL

`wsl --update` to update it

then enter the command

`winget install --interactive --exact dorssel.usbipd-win`

- 1) Then input `usbipd list` to see currently connected devices. We are looking for CMSIS-DAP v2
 - i) I ended up having to share a BUSID with a very wide range (1-10) so if you are in this boat don't worry about getting specific. A wide range will work fine.
- 2) Share the BUSID with:
 - ii) Type: `usbipd bind --busid 1-10` (In my case it was 1-10 your busid #/range will likely differ)
 - iii) Then. . . `usbipd attach --wsl --busid 1-10` you will likely see some red text stating "usbipd: warning: The service is currently not running; a reboot should fix that."
 - iv) Type. . . `usbipd server` and you should now see a screen showing everything being hosted. Leave this window open.

```
PS C:\Windows\system32> usbipd bind -b 1-10
usbipd: warning: The service is currently not running; a reboot should fix that.
PS C:\Windows\system32> usbipd bind --busid 1-10
usbipd: info: Device with busid '1-10' was already shared.
PS C:\Windows\system32> usbipd list
Connected:
BUSID  VID:PID    DEVICE                                     STATE
1-4    046d:c534    USB Input Device                         Not shared
1-10   0483:5740    USB Input Device, CMSIS-DAP v2 Adapter, USB Serial Device... Shared
1-11   0a5c:216f    DW1560 Bluetooth 4.0 LE                 Not shared
1-13   058f:6362    Alcor Micro USB 2.0 Card Reader         Not shared

Persisted:
GUID                                     DEVICE

usbipd: warning: The service is currently not running; a reboot should fix that.
PS C:\Windows\system32> usbipd attach --wsl --busid 1-10
usbipd: error: The service is currently not running; a reboot should fix that.
PS C:\Windows\system32> usbipd --help
usbipd-win 4.3.0
```

```
PS C:\Windows\system32> usbipd server
info: Microsoft.Hosting.Lifetime[0]
       Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
       Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
       Content root path: C:\Program Files\usbipd-win\
info: Usbipd.ConnectedClient[1]
       Client [REDACTED] claimed device at 1-10 (USB\VID_0483&PID_5740\DAP_V1RU7ENT).
```

USBIPD WIN10 INSTALLATION NOTES

3) Verify CMSIS-DAP is available in Linux inputting the `/lsusb` command to see what USB devices your distro has access to.

All other commands “should” be the same regardless of win or linux but I have only attempted this to its completion in Win10.

Config file

Config (.cfg) file contents

All of these files can be found in a .zip file located [here](#) but here are the contents of the .cfg files I used. This zip file also includes a good copy of my E2's secondary boot loader (sbl.bin) in case yours has an issue as well as a copy of a headerless system.bin v2.02 firmware (KORG: Please don't come after me for this I am only trying to help people continue to use your great products) for flashing where appropriate.

Korg.cfg

```
# F0 JTAG/SWD in-circuit debugger.
adapter driver cmsis-dap
transport select jtag

adapter gpio tck 10
adapter gpio tms 12
adapter gpio tdi 6
adapter gpio tdo 5

# Each of the SWD lines need a gpio number set: swclk swdio
adapter gpio swclk 10
adapter gpio swdio 12

adapter gpio srst 4

#gdb memory_map enable
#gdb flash_program enable

reset_config trst_and_srst
reset_config trst_push_pull

#init

#halt 1000

#adapter speed 0
```

Config (.cfg) file contents

Credit for this file's content goes to Bangcorrupt. I take no credit for its creation.

am1802.cfg

```
#
# Texas Instruments: am1802
#
if { [info exists CHIPNAME] } {
    set _CHIPNAME $CHIPNAME
} else {
    set _CHIPNAME omapl138
}

source [find icepick.cfg]

# Subsidiary TAP: ARM ETB11, with scan chain for 4K of ETM trace buffer
if { [info exists ETB_TAPID] } {
    set _ETB_TAPID $ETB_TAPID
} else {
    set _ETB_TAPID 0x2b900f0f
}
jtag newtap $_CHIPNAME etb -irlen 4 -irmask 0xf -expected-id $_ETB_TAPID -disable
jtag configure $_CHIPNAME.etb -event tap-enable \
    "icepick_c_tapenable $_CHIPNAME.jrc 3"

# Subsidiary TAP: ARM926ejs with scan chains for ARM Debug, EmbeddedICE-RT, ETM.
if { [info exists CPU_TAPID] } {
    set _CPU_TAPID $CPU_TAPID
} else {
    set _CPU_TAPID 0x07926001
}
jtag newtap $_CHIPNAME arm -irlen 4 -irmask 0xf -expected-id $_CPU_TAPID -disable
jtag configure $_CHIPNAME.arm -event tap-enable \
    "icepick_c_tapenable $_CHIPNAME.jrc 2"

# Primary TAP: ICEpick-C (JTAG route controller) and boundary scan
if { [info exists JRC_TAPID] } {
    set _JRC_TAPID $JRC_TAPID
} else {
    set _JRC_TAPID 0x0b7d102f
}
jtag newtap $_CHIPNAME jrc -irlen 6 -irmask 0x3f -expected-id $_JRC_TAPID -ignore-version

jtag configure $_CHIPNAME.jrc -event setup \
```

Config (.cfg) file contents

```
"jtag tapenable $_CHIPNAME.etb; jtag tapenable $_CHIPNAME.arm"
```

```
#####
```

```
# GDB target: the ARM, using SRAM1 for scratch. SRAM0 (also 8K)
```

```
# and the ETB memory (4K) are other options, while trace is unused.
```

```
# Little-endian; use the OpenOCD default.
```

```
set _TARGETNAME $_CHIPNAME.arm
```

```
target create $_TARGETNAME arm926ejs -chain-position $_TARGETNAME
```

```
$_TARGETNAME configure -work-area-phys 0x80010000 -work-area-size 0x2000
```

```
# be absolutely certain the JTAG clock will work with the worst-case
```

```
# CLKIN = 20 MHz (best case: 30 MHz) even when no bootloader turns
```

```
# on the PLL and starts using it. OK to speed up after clock setup.
```

```
adapter speed 1500
```

```
$_TARGETNAME configure -event "reset-start" { adapter speed 1500 }
```

```
arm7_9 fast_memory_access enable
```

```
#arm7_9 dcc_downloads enable
```

```
# trace setup
```

```
etm config $_TARGETNAME 16 normal full etb
```

```
etb config $_TARGETNAME $_CHIPNAME.etb
```

```
gdb_breakpoint_override hard
```

```
arm7_9 dbgrq enable
```

```
reset_config trst_and_srst
```

```
jtag_ntrst_delay 100
```

```
jtag_ntrst_assert_width 100
```