IBDAP - User Manual

IBDAP - Makers helping makers

https://ibdap.net

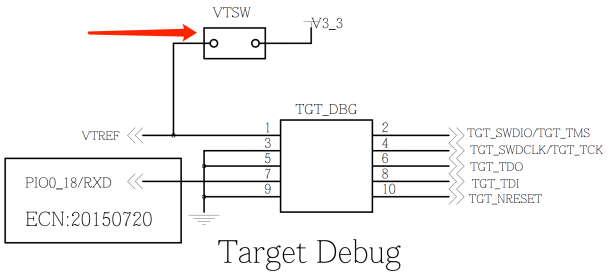
Introduction

IBDAP is an open source, low cost, cross-platform and vender independent CMSIS-DAP JTAG/SWD debug adapter for programming and debugging ARM Cortex M microcontrollers. It provides debugging functions like stepping, breakpoints, watch points and firmware programming etc., making microcontroller programming easy and affordable. It's fully supported by Keil, Eclipse, OpenOCD, GNU GDB, IAR and other commonly used debugging tools across Windows, Linux and Mac OS.

Besides being functioning as a JTAG/SWD debug probe, IBDAP could also be used as an mbed development board. It has an ARM Cortex M0 processor running at 48Mhz and has peripherals like UART, I2C, SPI, USB. It can be used in applications like USB audio devices, USB mouse/keyboards, USB mass storage devices, USB-TTL adapter device and many more. It also has 10-bit high precision ADC peripherals which make IBDAP an ideal device for any sensor projects.

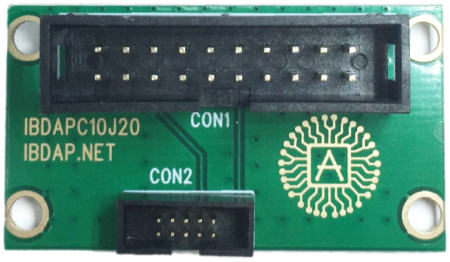
IBDAP voltage levels and power supply

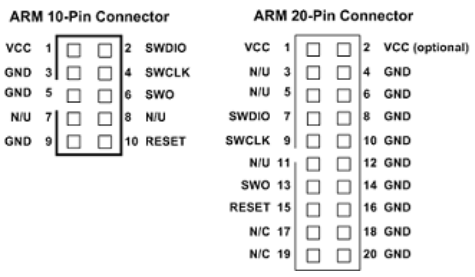
IBDAP supports target boards that run at 3.3V level. The VTSW jumper (please refer to schematic file) is used to enable or disable 3.3V direct power supply to the target board, a convenient way to provide powers directly from debug cable.



Target Debug

In case that the target board requires 20-pin 0.05" JTAG header or some none standard connector, you can always easily bridge the IBDAP TGT_DBG connector over using the **IBDAPC10J20 ARM 20-pin 0.1" to Cortex 10-pin 0.05" JTAG/SWD Adapter** which is included in the IBDAP product.

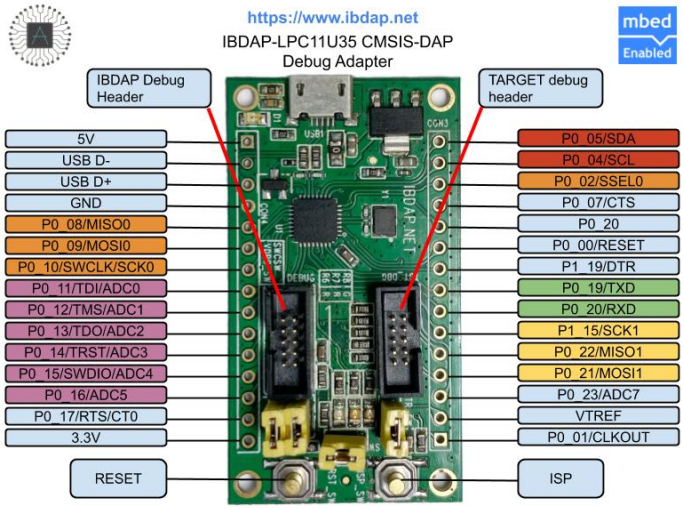




IBDAP Pinmap

IBDAP schematic is provided and can be downloaded from the following link:

https://ibdap.net/downloads/ibdap-schematic.pdf



Features:	Supported target processors including but not limited to:
• NXP LPC11U35FHI33/501 ARM Cortex M0 48Mhz processor • 64kB Flash, 12kB RAM, 2kB EEPROM • 1xUSART, 1xI2C, 2xSPI, 1xUSB2.0, 8xADC(10-bit), 26xGPIO • GNU gcc-arm and make implementation of CMSIS-DAP with source codes provided. • Supports mbed programming.	• NordicSemi ARM-Cortex M series of microcontrollers • STM32 ARM Cortex-M series of microcontrollers • Atmel ARM Cortex-M series of microcontrollers • NXP ARM Cortex-M series of microcontrollers • Freescale ARM Cortex-M series of microcontrollers

Using IBDAP as a CMSIS-DAP Debugger

Flash IBDAP firmware onto IBDAP board

By pressing **RESET** and **ISP** buttons at the same time and then release them, a USB mass storage device will be shown up with a single file "firmware.bin". Delete it and paste in the IBDAP.bin file from below, then press RESET button again. Now Firmware flashing complete.

Firmware binary: https://ibdap.net/downloads/ibdap.bin

Firmware source code: https://github.com/ibdap/ibdap-cmsis-dap

How to build IBDAP firmware from source

To building IBDAP CMSIS-DAP firmware you need Linux/Ubuntu. You also need to install GNU ARM embedded toolchain, which can be download from here:
https://launchpad.net/gcc-arm-embedded

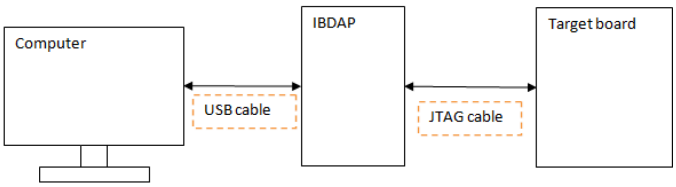
Open a Terminal and run the following command to build IBDAP.bin:

```
$ git clone https://github.com/ibdap/ibdap-cmsis-dap.git
$ cd build/
$ make
```

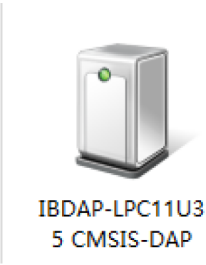
Overall setup diagram

First connect IBDAP's TGT_DBG 10-pin header to the target board, then connect IBDAP and PC via USB cable.

Note that, VTREF pin at TGT_DBG provides 3.3V for the target board, but you can switch off by removing the corresponding jumper.



If your system is Windows, you should be able to see a USB HID device called “IBDAPLPC11U35 CMSIS-DAP" in your control panel “Devices and Printers”. This indicates IBDAP is successfully recognized by your system. For Linux computers, listing USB devices to verify if IBDAP is recognized.



Flashing target firmware

Using OpenOCD

OpenOCD is software that can communicate with the hardware JTAG port, it bridges hardware IC and your firmware debugging software. OpenOCD can be installed on Windows, Linux and Mac OS.

https://sourceforge.net/projects/openocd/

If you want to install the prebuilt binaries instead of building from source code yourself:

Debian GNU/Linux: $ apt-get install openocd
OS X: $ brew install openocd

Here we demonstrate flashing and debugging using a nRF51822 BLE ARM Cortex M0 device. For other devices please refer to the OpenOCD document and chip manufacturer. In general, you need to know what’s your target chip’s config file is. For nrf51, it’s “target/nrf51.cfg”.

First run the following command in an admin privileged terminal to start OpenOCD:

```
$ openocd.exe -f interface/cmsis-dap.cfg -f target/nrf51.cfg
```

Then run telnet in another admin privileged terminal to flash your firmware.hex onto nRF51822:

```
$ telnet localhost 4444
> reset halt
> flash write_image erase /file/path/to//firmware.hex 0x0
> reset
> exit
```

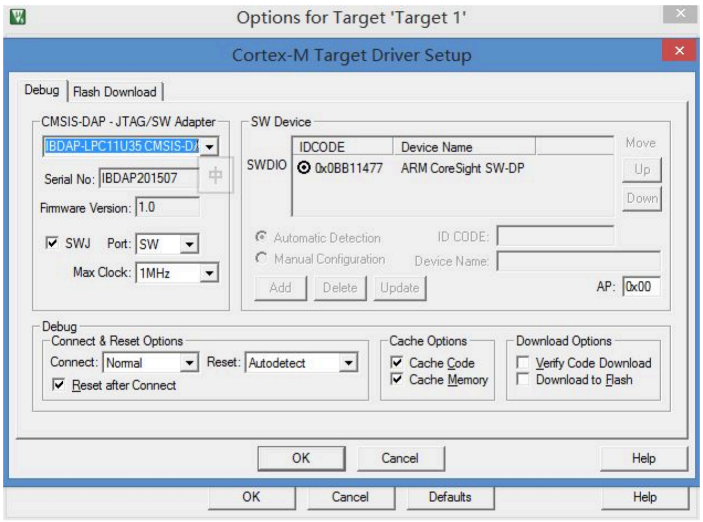
Some chips like nRF51822 have flash protection mechanism enabled, if you ever encountered error while flashing, run mass_erase command to disable the protection before flashing the firmware.

```
> nrf51 mass_erase
```

Using Keil

From your Keil project, click on "Options for Target" button just besides the target, choose "Debug" tab, select use: "CMSIS-DAP Debugger", then click "settings", you should be able to see the "IBDAP-LPC11U35 CMSIS-DAP" device as the picture shown below, if yes, then we are all set.

Note: For some devices, nRF51822 for example, Keil doesn't have the correct Flash Erase algorithm for full-chip erase, if you have any issue flashing, please use OpenOCD method instead.



Debugging target firmware

Using OpenOCD

OpenOCD supports stepping, adding breakpoints watch points when you are debugging your firmware codes.

First, make sure you have the OpenOCD started. If you already done so in the “Flashing target firmware” section, you could skip this step now. Otherwise run the following command in a privileged terminal.

```
$ openocd.exe -f interface/cmsis-dap.cfg -f target/nrf51.cfg
```

Then open the second privileged terminal to set up a telnet debug session with your target device:

```
$ target remote localhost:3333
```

Now we need to set some breakpoints in the code:

```
> break main.c:25
```

Stop the currently running code using the following command:

```
> monitor reset halt
```

Reset the code back to start:

```
> monitor reset init
```

Now we can run the firmware program which will pause at the break point we just set:

```
> continue
```

Using Keil

Adding breakpoints, debugging and stepping in Keil is pretty straightforward, it’s all been setup on the IDE GUI, refer to the Keil documentation to details.

Use IBDAP as development board

IBDAP uses NXP LPC11U35FHI33/501 ARM Cortex M0 chip, and MCUXpresso IDE or mbed can be used to develop firmwares.

- MCUXpresso IDE
- mbed: https://developer.mbed.org/

For more information please refer to the corresponding websites.