WhyCan Forum(WowKu Developer Community)

The original poster 2023-01-26 20:23:54 Share comments #

abgelehnt

member

Registration time: 2022-05-01
Posts posted: 10
Points: 110

At present, the software part has been completely written. The hardware part has been temporarily abandoned because there is no soldering iron on hand. We will wait until we can enter the laboratory. The hardware part is fully compatible with the hardware of @posystorage. You can try it with the hardware solution of @posystorage first.

The project code and precompiled firmware acquisition path: GitHub: abgelehnt/Tiny-DAPLink

This project is improved on the basis of the assembly-optimized high-speed DAP-Link (CMSIS-DAP v2) implemented by @posystorage open source CH551/2 and the @ljbfly CH552 version of CMSIS-DAP v2 .

The presentation mode of the debugger is still the general debugger and the onboard debugger, with the onboard debugger being the main one.

Universal debugger: uses 5V power supply, main frequency is 24Mhz. The download speed has not been tested in detail, but it is at least faster than the FullSpeed version of CMSIS-DAP v1. Downloading STM32F103C8T6 takes 6.5 seconds. Hardware: Since the peripheral circuit required by CH552 is simple and the chip size is small, the debugger can be made very small. The main hardware components are: CH552 (¥2.3), level conversion chip SN74LVC1T45*2 (¥2.8), resistor-capacitor and USB socket (¥0.5). It can be said to be quite cheap.

On-board debugger: uses 3.3V power supply, main frequency 16Mhz. The serial port does not support 115200 baud rate at 16Mhz frequency, but the problem is not big on the onboard debugger. If the general debugger does not support the 115200 baud rate, it will seem a bit useless. In terms of hardware: Compared with the small system board without a debugger, a small system board only needs to add a CH552, a capacitor, and a USB interface. Tiny-DAPLink also got its name.

The basic framework of the code has not been changed, but the details have been improved a lot:

1. Serial port optimization
I think the sign that the serial port is perfectly usable: the loopback test is correct, and the logic analyzer will not see the IDLE signal when sending continuously.

IDLE signal: The IDLE signal will appear when no more data is received on the communication line within one byte. The occurrence of the IDLE signal during

continuous transmission will cause functions like HAL_UARTEx_ReceiveToIdle to not work properly.

My code uses the idea of ping-pong buffer and uses double buffers for data movement. Compared with @posystorage's circular queue solution, it saves a large amount of data movement work at one time and saves the time after each serial port/USB receives data. The response time is optimized to a constant, which solves the extremely occasional data loss situation when the original version is continuously sent and used simultaneously to send and receive data (an average of 200K data is lost by dozens of bytes). The current version of the serial port is perfectly usable at 115200 baud rate.

At a baud rate of 500000, both my version and the original version lose data. The amount of data lost in my version is smaller, but the continuous transmission speed is much higher than that of the original version. The reason: the one-time large-scale data migration work is in total. It has a greater advantage in response time.

2. I have used the SWD speed optimization @posystorage version, but the IO version cannot be enabled for debugging. I have not even downloaded the SPI version successfully once.

Reason for compatibility issue with SPI version: In ADIv5 manual P122,

> When the target samples SWDIO, sampling is performed on the rising edge of SWCLK. When the target drives SWDIO, or stops driving it, signal changes are performed on the rising edge of SWCLK.

That is to say, mode 2 should be used when SPI reads, and mode 3 should be used when SPI writes. The boss may not have noticed this. However, CH552 does not have mode 2, so the part that can be optimized by SPI will be lower than 87%.

@ljbfly’s version doesn’t have any big problems except the IO mode, but the timing seems to have a high duty ratio.

Since I really don't know anything about assembly, there are always some weird problems when using assembly for timing flashing, so I had no choice but to give up the ready-made assembly code. In my version,

only 8bit command and 32bit WDATA are written using SPI, and the rest are written using IO. For the problem that the duty cycle seemed to be high, I directly added a bunch of nops to roughly solve it. The time consumed by a single flash is between the two big guys.

3. Correct the configuration descriptor.
The original configuration descriptor feels like it is in the state of "I don't know why but it just works". It has an additional interface association descriptor and uses endpoint 4. But in the USB interrupt, endpoint 4 is completely unused.

It also has an occasional bug: failure to program may cause Keil to no longer be able to find the CMSIS-DAP. The solution is: Find the device in the device manager -> Uninstall the device -> Re-insert the USB flash drive.

My version leaves out endpoint 4 and (probably) fixes this occasional bug.

4. Chinese university-style meaningless innovation: Adding a new capacitive keyboard.
This function requires specific hardware circuits to be implemented.

This feature maps any key on the keyboard and presses it once when touching a specific location on the debugger.

Imagine this scenario: you have just finished writing the code, click compile, and then go to play with the hardware: connect the Dupont cable/plug in the array/turn on the power of the device. After playing with the hardware, the compilation is completed, and you can touch somewhere in the debugger Start downloading conveniently, and the entire process is fast and elegant.

This function is added because Keil does not have a button similar to downloading after compilation, which would be more troublesome.

In the serial port tool, enter: DAT+KEY=41 (case sensitive) to open the capacitive keyboard and map it to F8.

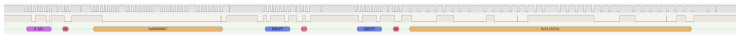
You can also map the capacitive keyboard to other keys USB HID keyboard key value table .

Enter DAT+KEY=FF to turn off the capacitive keyboard.

5. Turn on the sleep function.
Some computers will not turn off the USB 5V power

supply during sleep/hibernation/shutdown state, but the power consumption of the debugger is irrelevant.

6. The unique serial number
device uses the chip's unique ID as the serial number, so that multiple debuggers can be plugged in at the same time.



I won’t use 51 anymore when I write something next time. It is a mistake to use 51 where there is even a little pursuit of performance.

Offline

2023-01-26 22:05:26 **Share comments** #1

XIVN1987

member

Registration time: 2019-08-30
Posts posted: 220
Points: 278.5

CH32V203F8 costs 3 yuan, can run to 144MHz, and also supports crystal-less USB. .

If you don't want to use RISC-V, AT32F425F8 costs 3.75 yuan, Cortex-M4 core, 96MHz, and also supports crystal-less USB. .

Schematic & PCB:
https://oshwhub.com/xivn1987/daplink
Source code: https://github.com/XIVN1987/DAPLink

Offline

2023-01-27 09:32:21 **Share comments** #2

Chin

member

Registration time: 2020-10-19
Posts posted: 61
Points: 39

@abgelehnt
51 It’s a bit annoying not being able to debug

Offline

2023-01-28 13:54:10 **Share comments** #3

lOsetting

member

Registration time: 2022-05-08
Posts posted: 57
Points: 63

@XIVN1987
There seems to be no USB in the AT32F425 manual?

Offline

2023-01-28 14:05:50 **Share comments** #4

IOsetting

member

Registration time: 2022-05-08
Posts posted: 57
Points: 63

Offline

@XIVN1987
Is it okay to replace AT32F425 with AT32F415? 415 also has USB

2023-01-28 17:45:46 **Share comments** #5

XIVN1987

member

Registration time: 2019-08-30
Posts posted: 220
Points: 278.5

Offline

> **IOsetting said:**
>
> @XIVN1987
> Is it okay to replace AT32F425 with AT32F415? 415 also has USB

The USB of 415 does not support Crystal-less, and the circuit is not as simple as that of 425. .

In addition, 415 does not have SSOP20 package, and the smallest package is QFN32, which feels not as easy to solder as SSOP20. .

2023-01-29 00:48:27 **Share comments** #6

IOsetting

member

Registration time: 2022-05-08
Posts posted: 57
Points: 63

> **XIVN1987 said:**
>
> > **IOsetting said:**
> >
> > @XIVN1987
> > Is it okay to replace AT32F425 with AT32F415? 415 also has USB
>
> The USB of 415 does not support Crystal-less, and the circuit is not as simple as that of 425. .

Is 415 also supported? The manual states that there is a dedicated built-in 48MHz clock.

AT32F415 has a built-in OTG full-speed (12 Mb/s) device and host mode control module with integrated transceiver. The OTGFS module
is compatible with USB2.0 and OTG1.3 protocols. It has software-configurable endpoint settings and supports a suspend/resume mechanism. The 48 MHz clock dedicated to the OTGFS module is generated by the internal main PLL.

Offline

2023-01-29 09:49:39 **Share comments** #7

XIVN1987

member

Registration time: 2019-08-30
Posts posted: 220
Points: 278.5

@IOsetting

The 425 has a peripheral called "HICK Automatic Clock Calibration (ACC)", which calibrates the 425's HICK clock to a clock accuracy of 0.25% based on the USB host's SOF package to meet USBFS communication requirements. The 415 does not have this module at all.

Take a look at the "Internal Clock Source Characteristics" chapter in the 425 data sheet. The accuracy of HICK under ACC calibration conditions is 0.25%. The highest accuracy of the HICK clock marked on the 415 data sheet is only 1%, which does not meet the USBFS clock requirements. Accuracy requirements

Offline

2023-01-29 10:44:40 **Share comments** #8

IOsetting

member

Registration time: 2022-05-08
Posts posted: 57
Points: 63

Offline

@XIVN1987
Got it, thank you

2023-01-29 21:15:20 **Share comments** #9

Timaker

member

Registration time: 2020-05-18
Posts posted: 61
Points: 39.5

Offline

Does it support debugging? How many breakpoints can be set?

Page: **1** **Share link / view only the original poster**

front page » Cortex M0/M3/M4/M7
» Tiny-DAPLink: CMSIS-DAP v2 upgraded version implemented by open source CH552

Forum jump
Cortex M0/M3/M4/M7 Go to

Atom subscribes to the topic