

Kneron Inc

Document Name: **Flash Management**

Flash Management **Kneron Inc**

Engineering Design Document

Kneron Confidential

Table of Contents

| | | |
|----------|---|----------|
| 1 | Introduction..... | 2 |
| 1.1 | Purpose | 2 |
| 2 | Reference | 2 |
| 3 | Acronyms, Abbreviations, Definitions | 2 |
| 4 | Board Overview | 3 |
| 5 | Hardware Setting..... | 4 |
| 5.1 | Connecting UART0/UART4..... | 4 |
| 5.2 | Connecting JTAG..... | 4 |
| 5.3 | Connecting 5V Power | 4 |
| 6 | Power On and System Bring Up..... | 5 |
| 7 | Load Flash Programmer Firmware to Mozart | 7 |
| 8 | Run Flash Programming on Python | 8 |
| 8.1 | Flash programmer necessities..... | 8 |
| 8.2 | Edit python verification setting | 8 |
| 8.3 | Memory Read/Write Verification..... | 8 |
| 8.4 | Flash Chip Programming (FW + DATA)..... | 9 |
| 8.5 | Flash Firmware Programming (optional) | 9 |
| 8.6 | Flash Verification (optional) | 9 |
| 8.7 | Flash Erase (optional)..... | 9 |

1 Introduction

1.1 Purpose

The purpose of this document is to define the flash programming for firmware and CNN models. User can use different commands to erase, verify and programming full flash data or sectors of firmware and models. This programming tools does not need the JTAG dongle. It uses two UARTs dongle, one for IPL(internal programming loader) to load flash programming firmware, another one for flash commands to erase, verify or programming

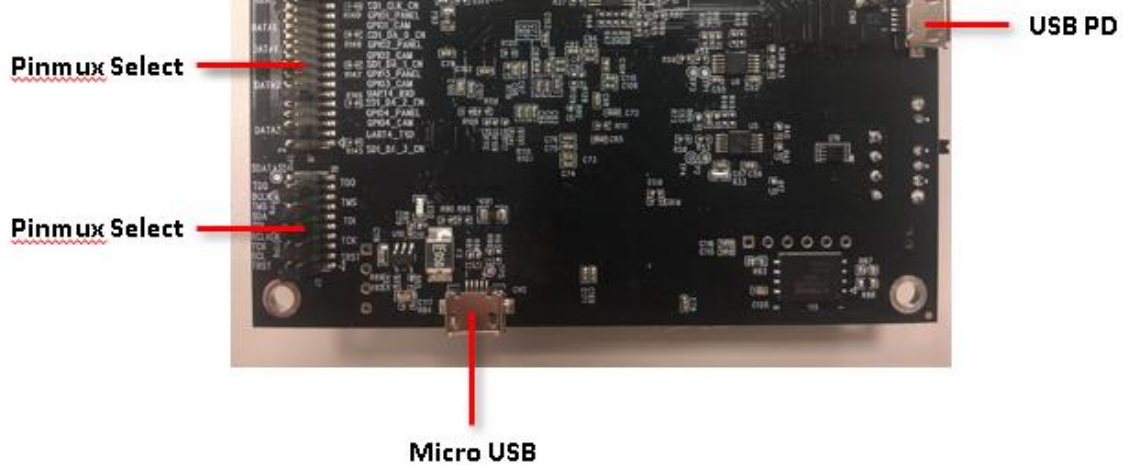
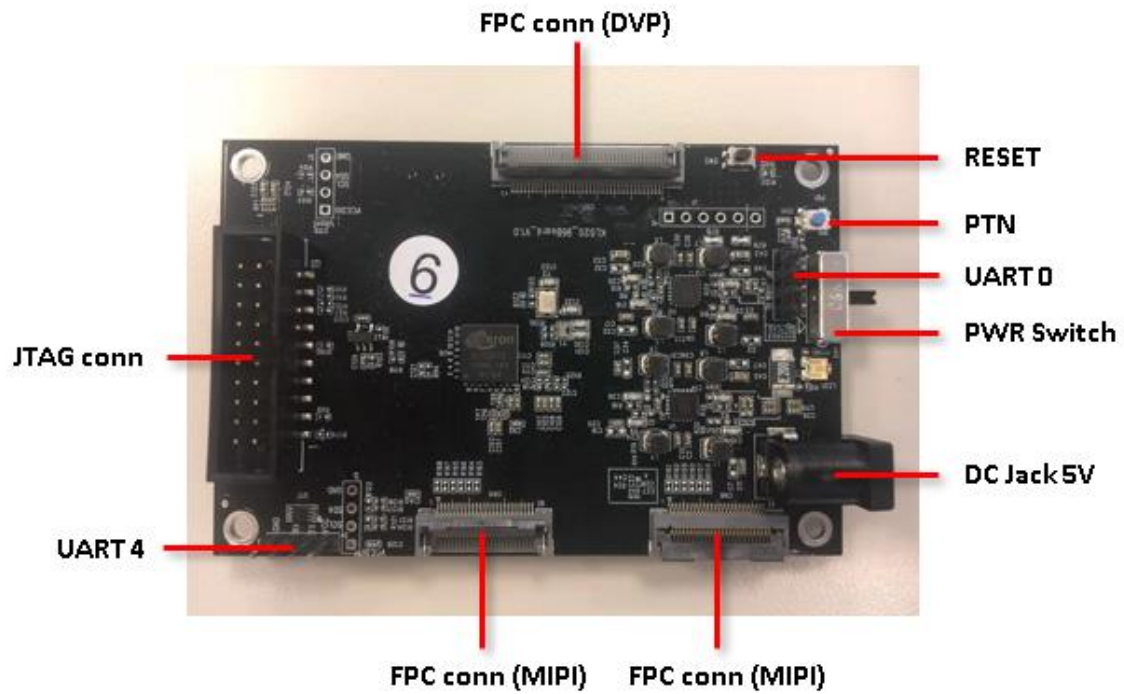
2 Reference

Kneron Mozart Design Specification, Rev. 0.5, Feb. 2019
FTSSP010, Synchronous Serial Port Controller, Rev. 1.30, June 2018
FTDMAC020, DMA Controller with AHB Interface, Rev. 1.20, Nov. 2016
FTUART010, UART and IrDA Controller, Rev. 1.21, Dec. 2017

3 Acronyms, Abbreviations, Definitions

SPI – Serial Peripheral Interface
UART – Universal Asynchronous Receiver-Transmitter

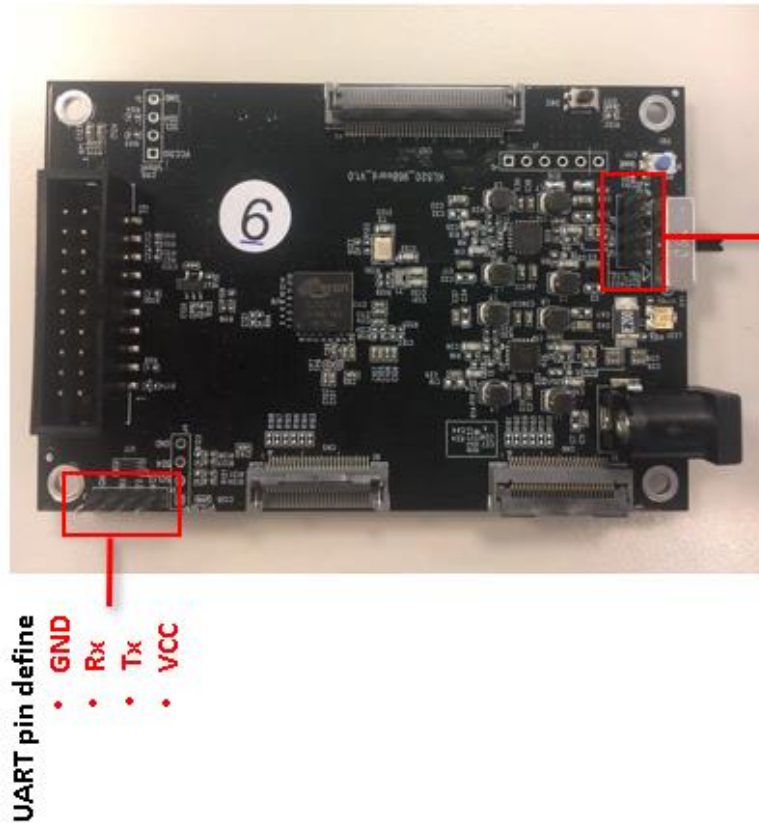
4 Board Overview



5 Hardware Setting

5.1 Connecting UART0/UART4

UART0: Debug Message Port,
UART4 Command Port



UART pin define

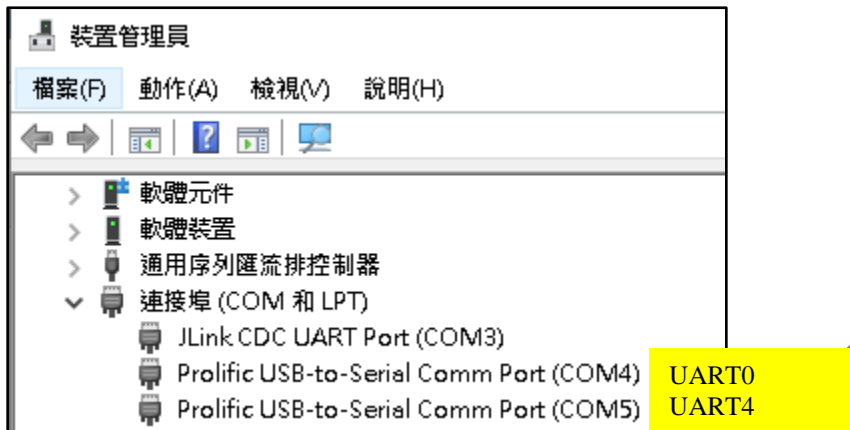
- GND
- Rx
- Tx
- VCC

5.2 Connecting JTAG

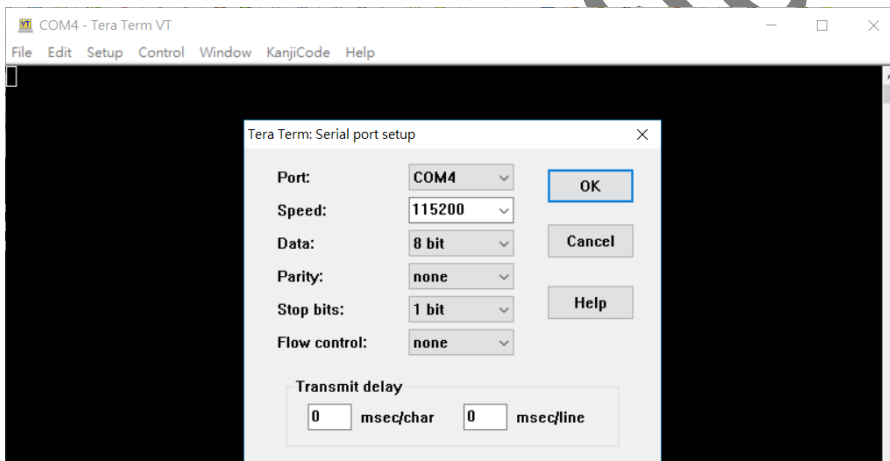
5.3 Connecting 5V Power

6 Power On and System Bring Up

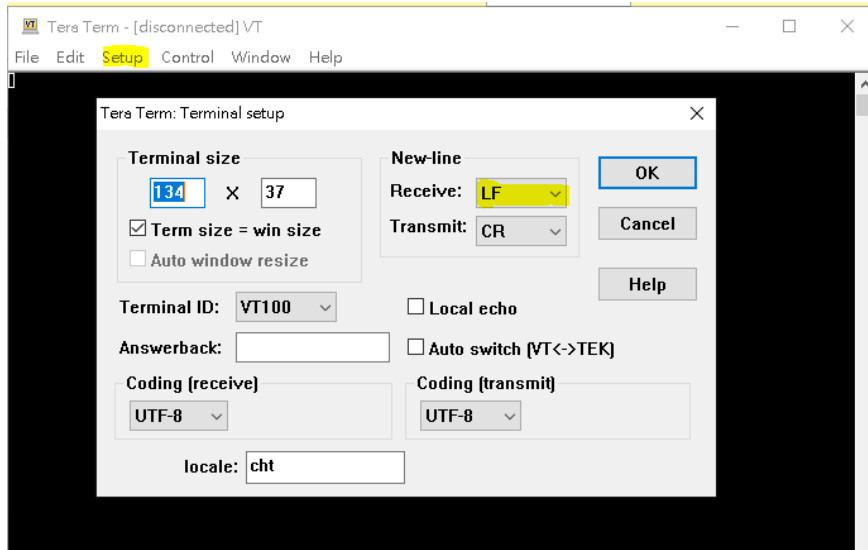
1. Verify the connection of UART0 and UART4



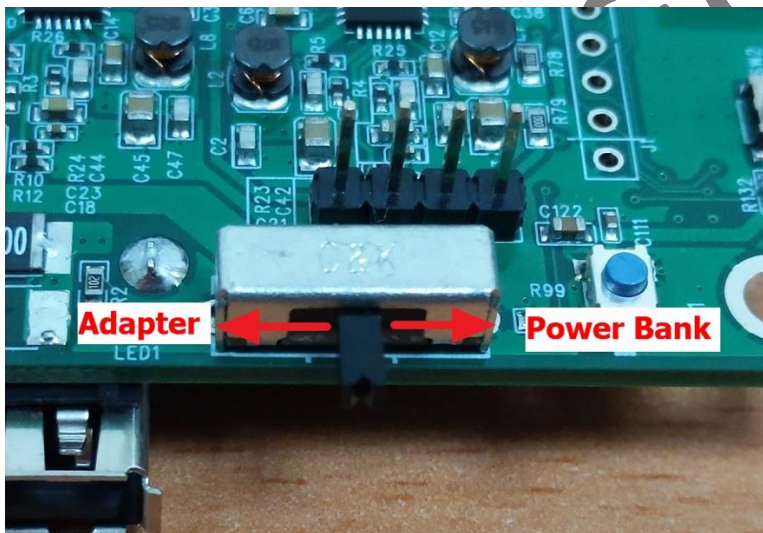
2. Open Uart COM port debug windows (Teraterm or Putty)
UART0: Baudrate: 115200



3. Set New-line as "LF" for receive
Setup->Terminal



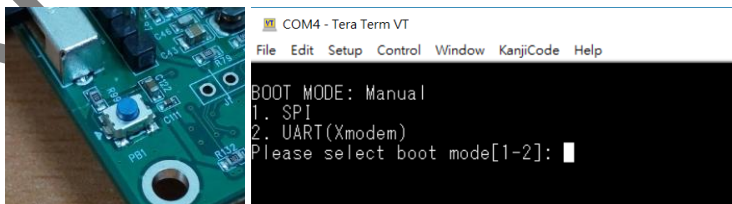
4. Trun ON power switch.
Power from Adapter or from Power Bank(USB PD)



5. Wake up chip from RTC power domain.

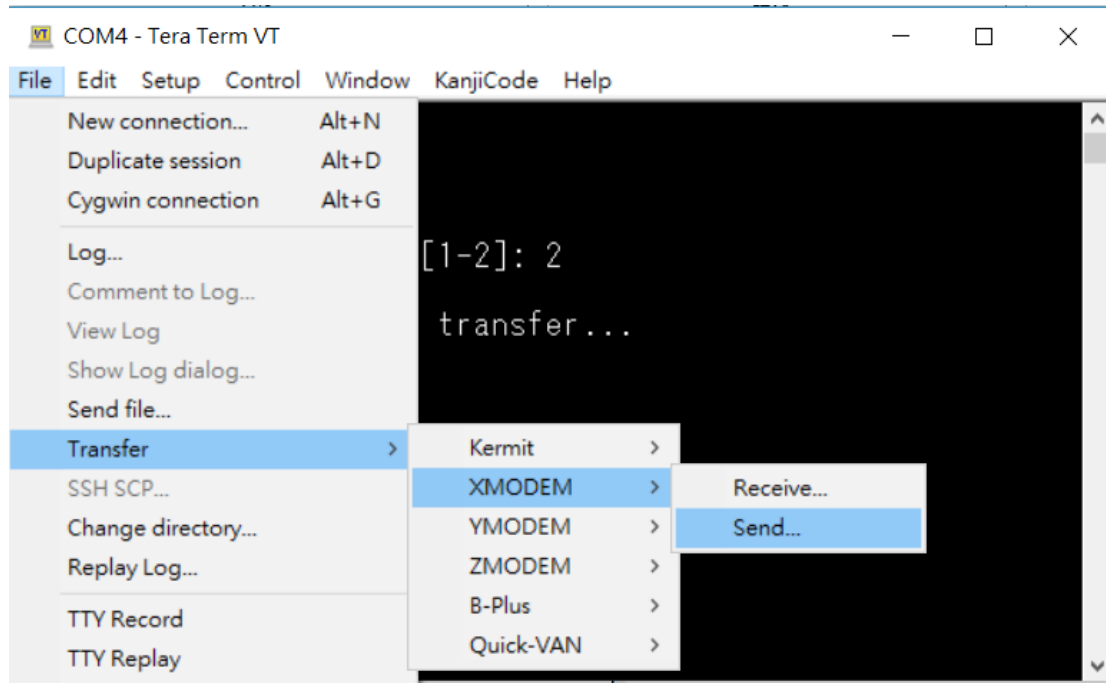
You will see boot message when you press PTN button

press "2" in console terminal

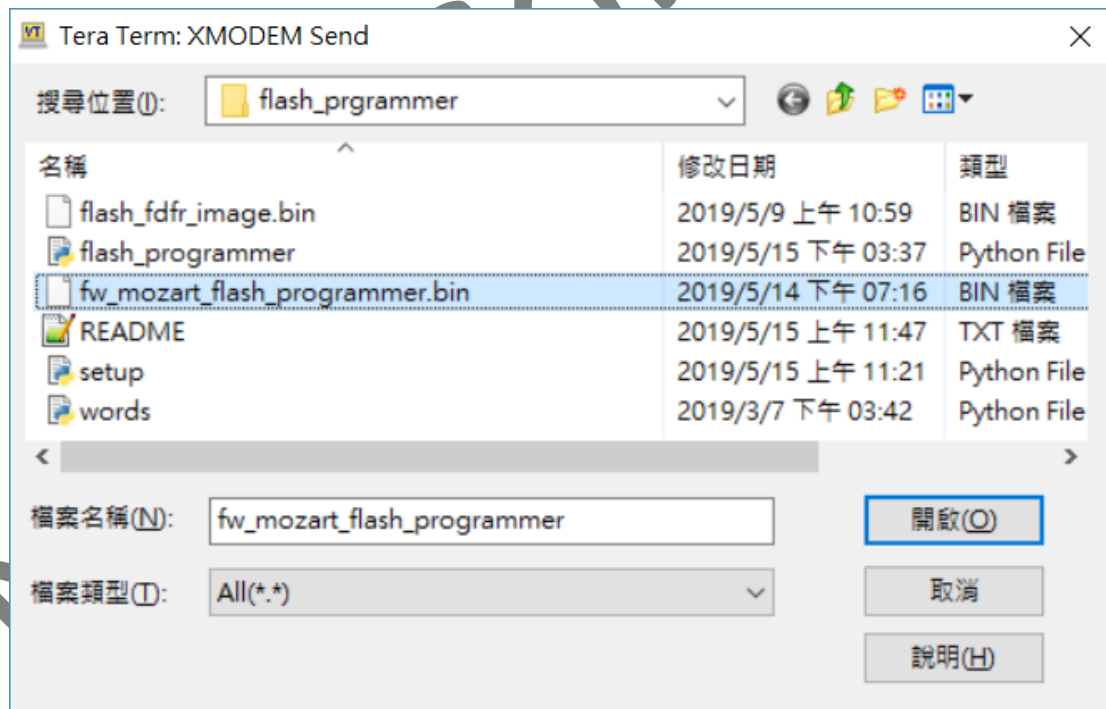


7 Load Flash Programmer Firmware to Mozart

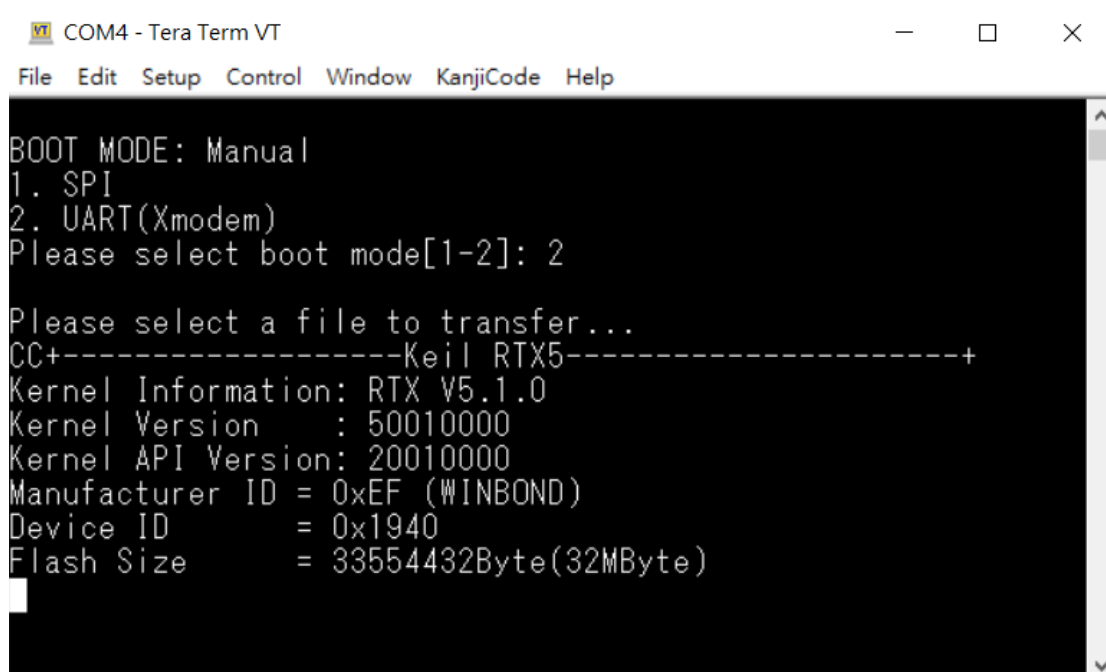
Upload “fw_mozart_flash_programmer.bin” firmware file by TeraTerm XMODEM send



Tool path: kl520_sdk\utils\flash_programmer



After the flash programmer firmware is uploaded successfully, the following message will be displayed.



```
COM4 - Tera Term VT
File Edit Setup Control Window KanjiCode Help

BOOT MODE: Manual
1. SPI
2. UART(Xmodem)
Please select boot mode[1-2]: 2

Please select a file to transfer...
CC+-----Keil RTX5-----+
Kernel Information: RTX V5.1.0
Kernel Version      : 50010000
Kernel API Version  : 20010000
Manufacturer ID     = 0xEF (WINBOND)
Device ID           = 0x1940
Flash Size          = 33554432Byte(32MByte)
```

Note:

1. “flash_fdf_image.bin” is generated by tool “bin_gen”
Tool path: kl520_sdk\utils\bin_gen
2. All separated bin files are in 4KB alignment while merge by “bin_gen”
3. Please read kl520_sdk\utils\bin_gen\README.md for more information

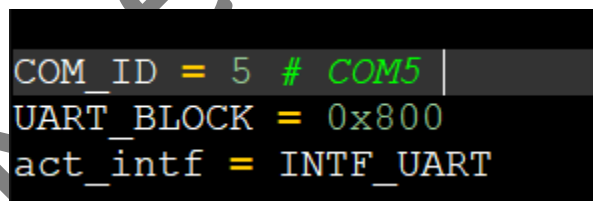
8 Run Flash Programming on Python

8.1 Flash programmer necessities

1. Open command terminal for flash programmer execution
Tool path: kl520_sdk\utils\flash_programmer\flash_programmer.py
2. install Necessary python modules: kl520_sdk\utils\requirements.txt

8.2 Edit python verification setting

Modify your COM port for UART4 in “setup.py” (baud-rate is 921600)



```
COM_ID = 5 # COM5
UART_BLOCK = 0x800
act_intf = INTF_UART
```

8.3 Memory Read/Write Verification

Please try run memory verification on python to verify your hardware platform.

```
>> python3 flash_programmer.py -t
```

```
->uart write: 16  
->uart read: 272  
[DDR] Memory Read/Write verify PASS (100/100)
```

8.4 Flash Chip Programming (FW + DATA)

```
>>> python3 flash_programmer.py -a flash_fdfr_image.bin
```

PASS result after flash programming done.

```
Read status [3364/3366]  
->uart write: 16  
->uart read: 2064  
Read status [3365/3366]  
flash bin comparsion PASS  
Flash: Verify done
```

Note:

flash_programmer.py -a" means "-e + -f + -v" (erase + programming + verification)

8.5 Flash Firmware Programming (optional)

```
>>> python3 flash_programmer.py -f flash_fdfr_image.bin
```

8.6 Flash Verification (optional)

```
>>> python3 flash_programmer.py -v flash_fdfr_image.bin
```

8.7 Flash Erase (optional)

```
>>> python3 flash_programmer.py -e
```