ARNO FIRMWARE UPDATE ON UART

Contents

1. Overview

This document provides instructions to do ARNO firmware update on UART from host (in this example windows PC). ARNO is based on NXP QN9030 SoC. Similar procedure can be used from any embedded host to upgrade the firmware of QN9030.

At boot time, the device is placed in ISP (In System Programming) mode where a USART is enabled and software commands can be sent into the device. These are serviced by the boot code of ARNO and allow operations for firmware update like erasing, writing and reading flash memory.

This procedure only covers the basic steps on firmware update. For more details on secure firmware update and details on all other ISP mode commands please refer to the reference indicated in Appendix.

2. Hardware Requirements

- NXP Carrier Main Board (JN5189)
- ARNO Mezzanine Card from Ivativ
- CP210x USB to UART or Any Other converter
- Mini USB Cable.
- Jumper cables.
- PC or Laptop.

3. Software Requirements

- Docklight.
- TTL converter CP210x USB to UART or any other similar converter drivers.

4. Firmware update Procedure Summary

1. Setup the hardware for ISP mode
2. Setup the serial terminal like Doclight and configure it
3. Send the ISP commands from host to do the firmware update
    a. Initialize communications
    b. Erase flash
    c. Program image to flash
    d. Read contents from flash (to make sure image update is done right).
    e. Reset the device

5. Procedure Details

5.1 Setup the hardware for ISP mode

5.1.1 Set the board in ISP mode with Bootstarp

Before sending the commands to ARNO, device should be placed in ISP mode by keeping PIO5(DIO5) low and PIO4(DIO4) high during the rise edge of reset. Reset button (shown in Figure 2 small red rectangle) can be pressed and released. Reset should be low for at least 1 ms before rising. Make sure PIO5 is kept low for at least 10mS after reset release. Check Figure 3 for connections of PIO4 and PIO5.

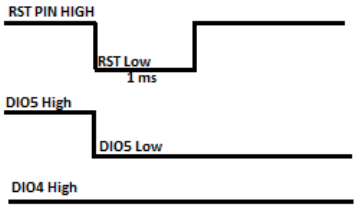

Figure 1: Timing diagram for ISP mode bootstrap

5.1.2 Connect the ARNO Mezzanine card

Attach the ARNO mezzanine card to the carrier board as shown in Figure 2 below

5.1.3 Connect TTL converter to ARNO DVK

- For UART Tx**:** Connect Tx pin on TTL Converter with PIO9(RX) pin on ARNO DVK.
- For UART Rx: Connect Rx pin on TTL Converter with PIO8(TX) pin on ARNO DVK.
- For GROUND: Connect GND pin on TTL Converter with GND pin on ARNO DVK.

Note: The PIO header is marked in large red rectangle and further details are shown below

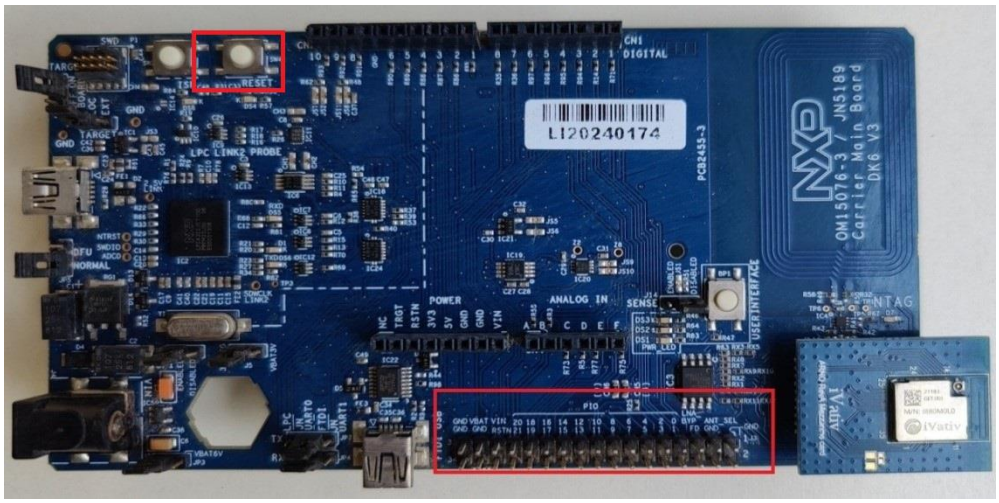

Figure 2: Carrier board and PIO header

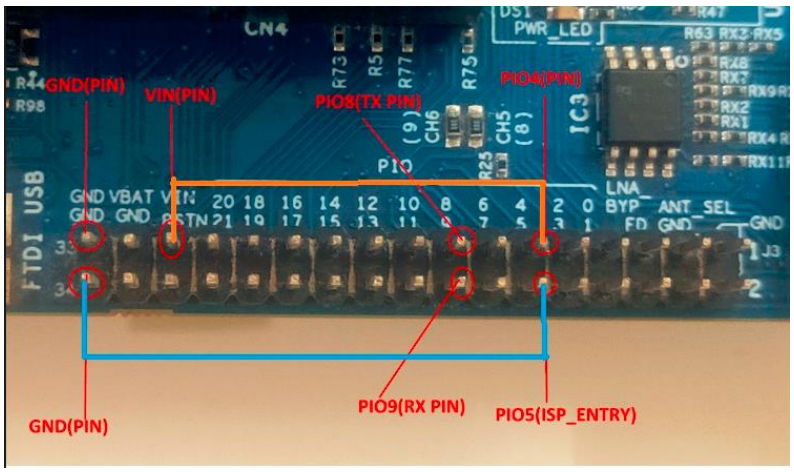

Figure 3: PIO header zoomed in

Note: Figure 4 shows how any host can connect to ARNO for ISP mode firmware update

Figure 4: Generic host connections

- Connect the ARNO DVK (carrier board + ARNO mezzanine card) to the PC or Laptop with the mini-USB cable.
- Now connect the TTL converter to the PC (Host)

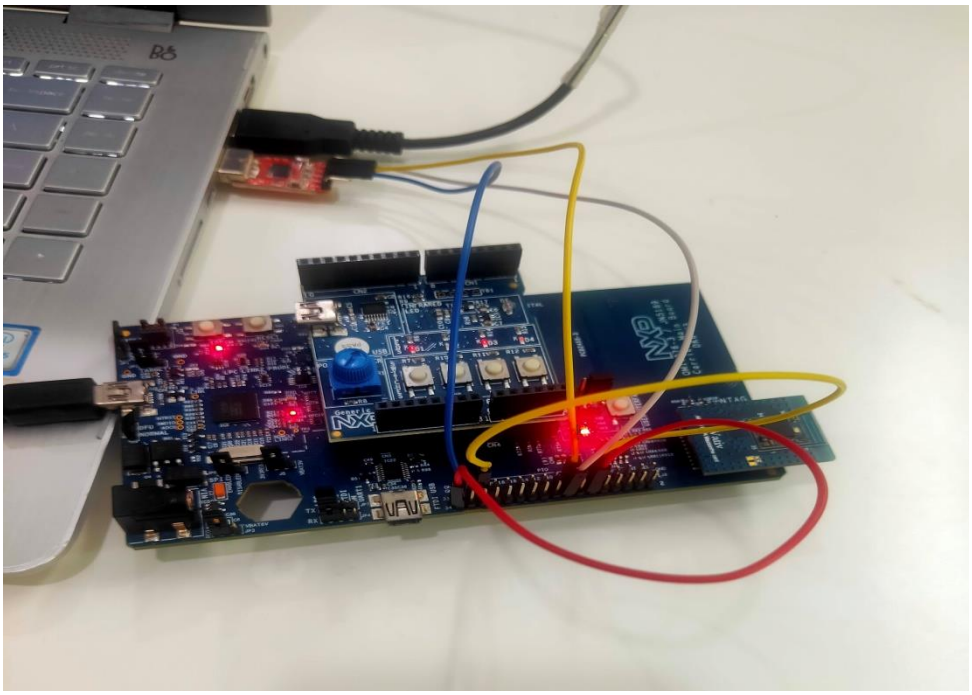

Figure 5: Complete hardware setup

5.2 SERIAL TERMINAL (Docklight) Setup and usage:

- After connecting the USB to UART TTL connector to PC, go to Device manager on PC and check the COM port as seen in the below picture for CP210x USB to UART or any other compatible converter. If you don't find the driver, please update the driver

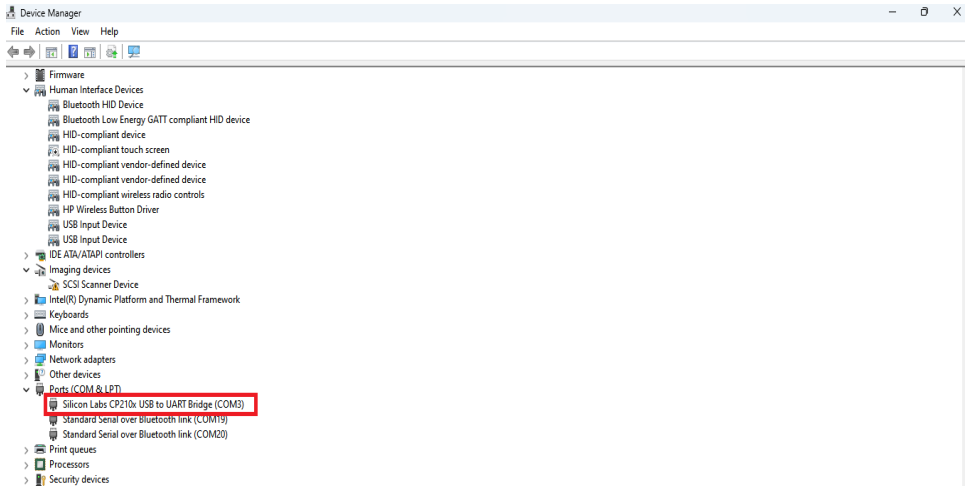

Figure 6: serial Terminal setting

- Open serial terminal like Docklight (Serial Terminal)

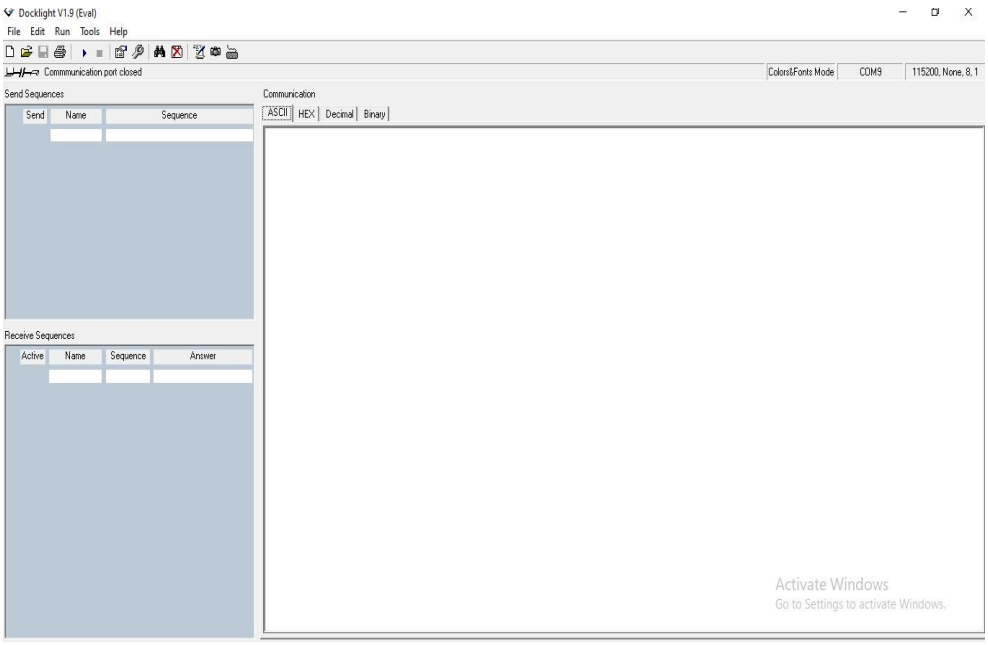

Figure 7: Doclight

5.2.1 Update the COM port settings:

Click on Tools ->project settings or click on the project settings icon and select the com port and update the com port settings as shown in below figure. The baud rate is 115200 with formatting of 8 bits per character, no parity bit and 1 stop bit

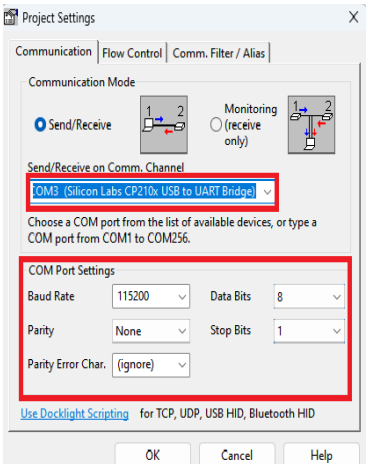

Figure 8: Comport settings

- Click on OK. Now the UART is ready

5.2.2 Using Doclight

- Double click on the blank space below the Name in “Send Sequence” window as shown in below picture. You will get “Edit Send Sequence” window where you can add and edit any command.

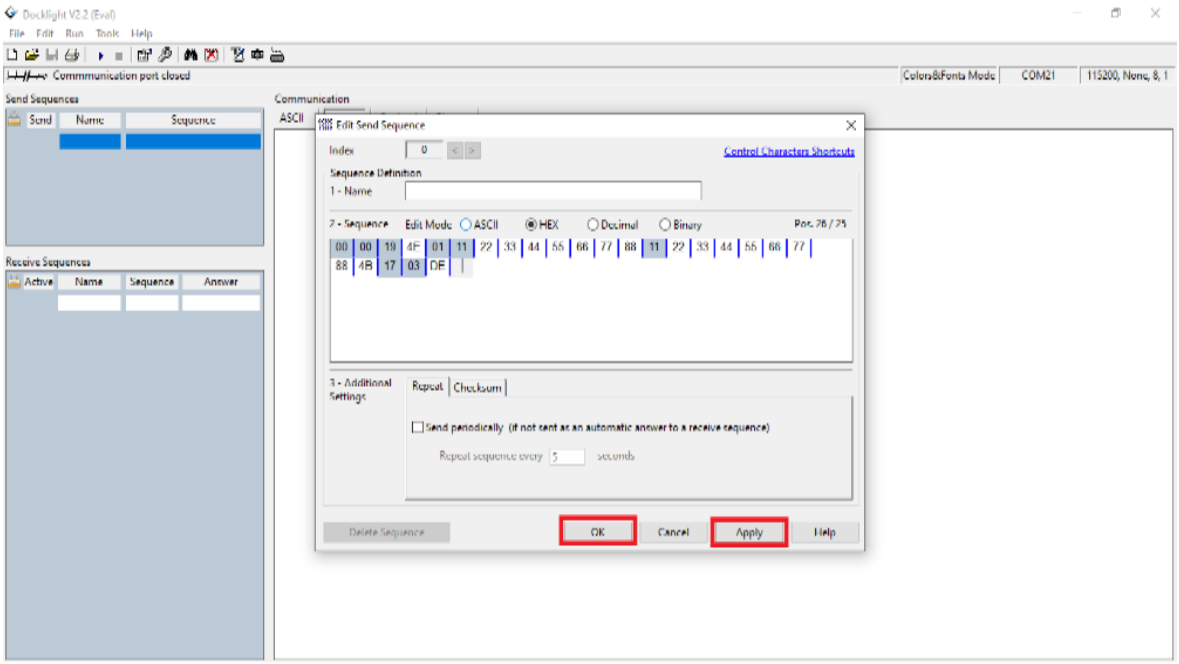

Figure 9: Editing commands in Doclight

- Click on Apply and then OK.
- The result can be seen in “Send Sequence” window as shown in below picture.

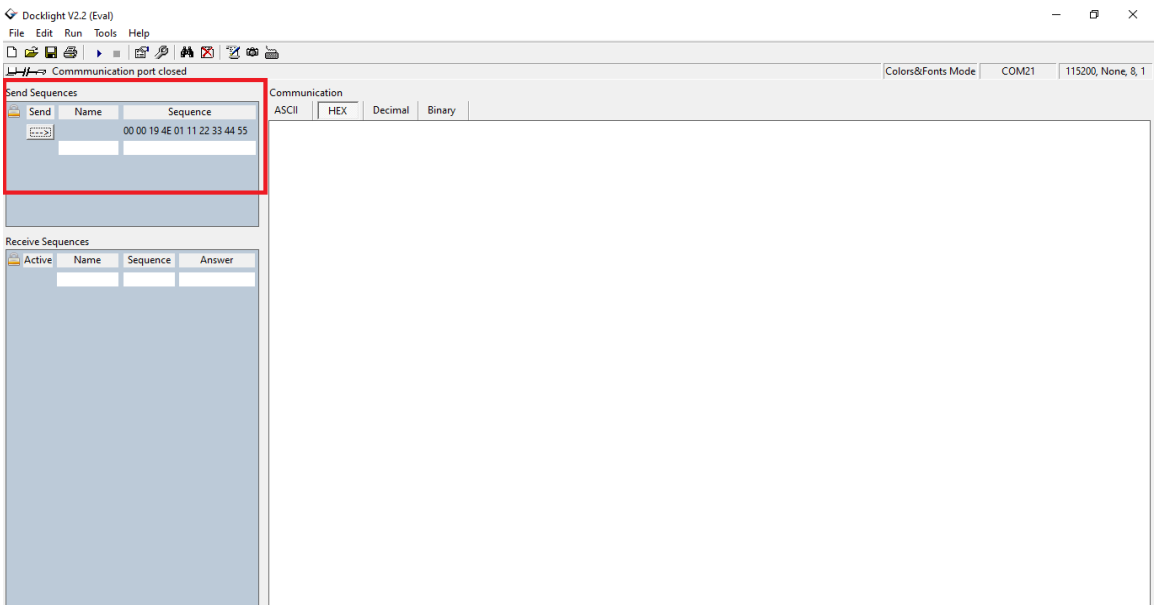

Figure 10: Entering Commands in Doclight

- By clicking the start communication option in Docklight as shown in below figure marked in red color then communication port will open.

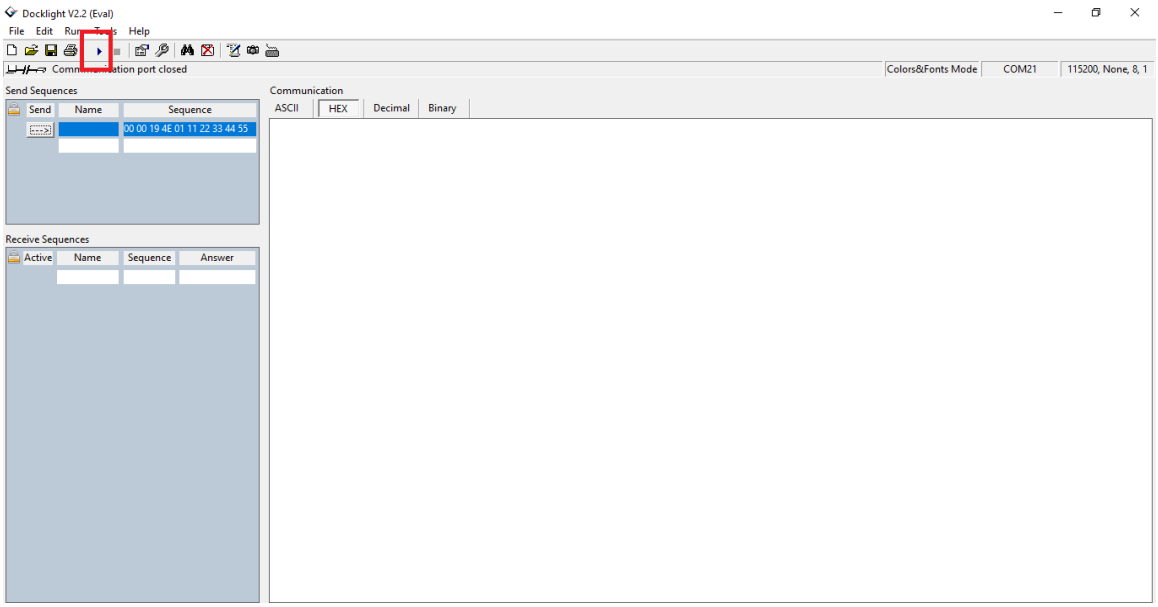

Figure 11: Opening Communication Port in Doclight

- Communication port open is displayed as shown in below figure in red color mark

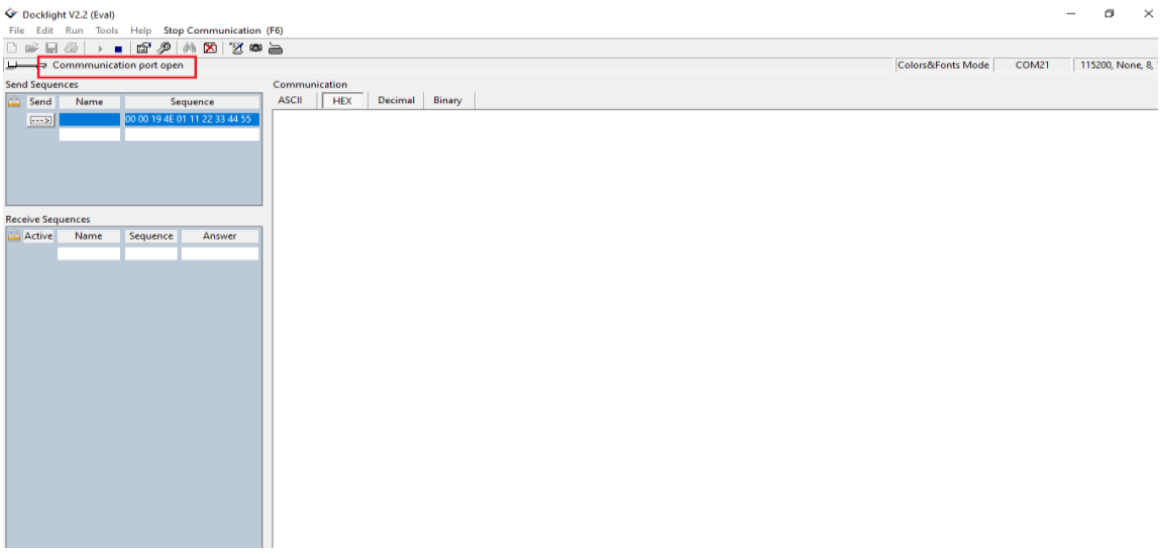

Figure 12: Port open status in Doclight

- After port open any command can be sent by clicking **Send** arrow as shown below

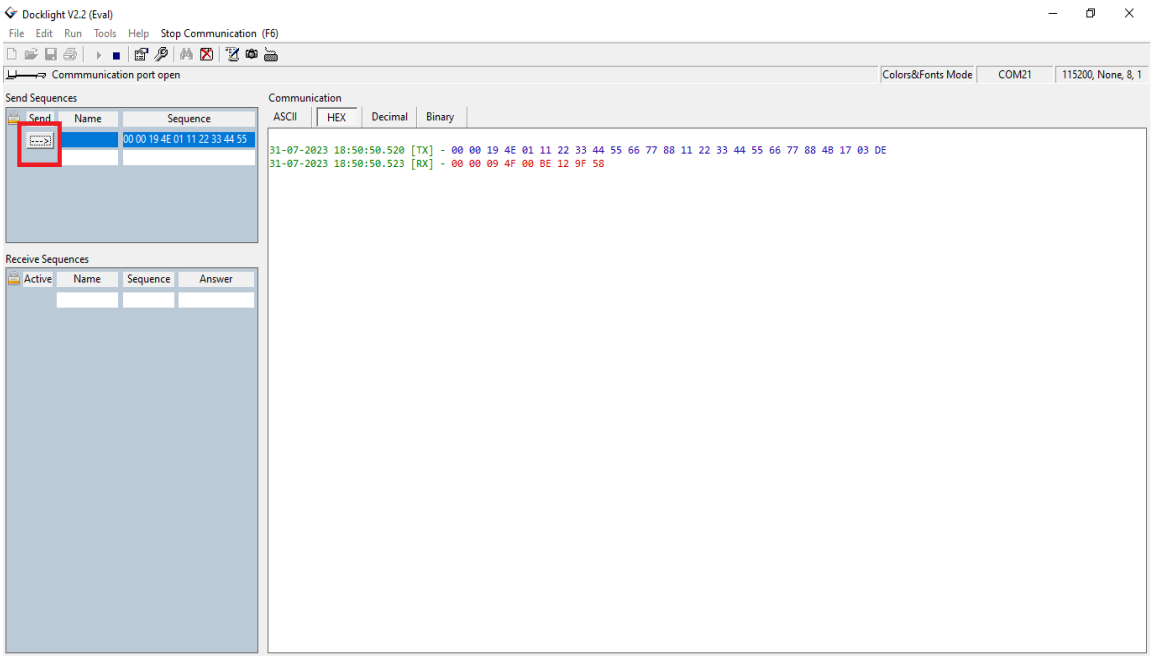

Figure 13: Sending commands in Doclight

Ex: TX 00 00 19 4E 01 11 22 33 44 55 66 77 88 11 22 33 44 55 66 77 88 4B 17 03 DE

And a response will come shown as RX. Ex: RX 00 00 09 4F 00 BE 12 9F 58

5.3 Sending ISP commands from host for firmware update

5.3.1 Initialize communications

It is important to use the Unlock ISP command before other commands are attempted. This sequence shows basic initialization and changing of the baud rate

- Unlock ISP to start ISP functionality

  TX 00 00 19 4E 01 11 22 33 44 55 66 77 88 11 22 33 44 55 66 77 88 4B 17 03 DE
  RX 00 00 09 4F 00 BE 12 9F 58

- Unlock ISP to default state
  TX 00 00 09 4E 00 A7 09 AE 19
  RX 00 00 09 4F 00 BE 12 9F 58

- Get Device Info

  TX 00 00 08 32 00 21 4A 04 94
  RX 00 00 11 33 00 88 88 88 88 00 00 00 00 AB 9A 33 AE

See Appendix for details:

Flags
Length
Type (Response)
Payload
Checksum

5.3.2 Erase flash

- Open Memory ID 0 (Flash) for Access

  TX 00 00 0A 40 00 0F 3E 5A D1 96
  RX 00 00 0A 41 00 00 AF 27 A6 30

- Erase Memory (0x9DE00 bytes from offset 0)

  TX 00 00 12 42 00 00 00 00 00 00 00 DE 09 00 E9 69 09 F9
  RX 00 43 00 12 A7 D0 54

- Blank check (optional)

  TX 00 00 12 44 00 00 00 00 00 00 00 DE 09 00 01 DC C3 BA
  RX 00 00 09 45 00 44 FD 77 D2

- Close Memory

  TX 00 00 09 4A 00 C3 65 6B 1D
  RX 00 00 09 4B 00 DA 7E 5A 5C

See Appendix for details:

Flags
Length
Type (Response)
Payload
Checksum

5.3.3 Program image to flash

- Open Memory ID 0 (Flash) for Access

  TX 00 00 0A 40 00 0F 3E 5A D1 96
  RX 00 00 0A 41 00 00 AF 27 A6 30

- Program Memory (0x00000200 bytes from offset 0x00000000)

  TX 00 02 12 48 00 00 00 00 00 00 00 02 00 00 E0 5F 01 04 81 01 00 00 A3 01 00 00 A5 01 00 00 A7 01 .... 25 4B 98 47 81 1E 48 42 04 F0 7F 04 48 41 96 19 A5 B7

  RX 00 00 09 49 00 E8 48 38 DE

- Program Memory (0x00000200 bytes from offset 0x00000200)

  TX 00 02 12 48 00 00 00 02 00 00 00 02 00 00 40 2C 05 D1 21 4B 2B 40 1A 1F 53 42 53 41 00 E0 00 23 .... 78 28 28 D1 00 21 00 E0 01 21 0C AC D8 F8 DE 8C F1 08

  RX 00 00 09 49 00 E8 48 38 DE

  (Additional Program Memory commands follow until complete image has been transferred)

- Close Memory

  TX 00 00 09 4A 00 C3 65 6B 1D
  RX 00 00 09 4B 00 DA 7E 5A 5C

5.3.4 Read contents from flash (check image update is done right)

See Appendix for details:

Flags
Length
Type (Response)
Payload
Checksum

- Open Memory ID 0 (Flash) for Access

  TX 00 00 0A 40 00 0F 3E 5A D1 96
  RX 00 00 0A 41 00 00 AF 27 A6 30

- Read Memory (0x200 bytes from offset 0)

  TX 00 00 12 46 00 00 00 00 00 00 00 02 00 00 0C E1 0D 66

  RX 00 02 09 47 00 E0 5F 01 04 81 01 00 00 A3 01 00 00 A5 01 00 00 A7 01 00 00 A9 01 00 00 AB 01 00 .... 4B 1C 6F D3 F8 B0 50 25 4B 98 47 81 1E 48 42 04 F0 7F 04 48 41 CF 21 30 7F

- Read Memory (0x200 bytes from offset 0x200)

  TX 00 00 12 46 00 00 00 02 00 00 00 02 00 00 9B 7E 1C 4F

  RX 00 02 09 47 00 40 2C 05 D1 21 4B 2B 40 1A 1F 53 42 53 41 00 E0 00 23 03 42 E6 D0 4F F0 80 43 D3 .... 44 88 46 58 28 03 D0 78 28 28 D1 00 21 00 E0 01 21 0C AC D8 F8 9D D6 AA FD
  (Additional Read Memory commands follow until complete image has been transferred)

- Close Memory

  TX 00 00 09 4A 00 C3 65 6B 1D
  RX 00 00 09 4B 00 DA 7E 5A 5C

5.3.5 Reset the device

- Reset device

  TX 00 00 08 14 F3 47 81 69
  RX 00 00 09 15 00 FE 46 2A 8

6. Appendix

6.1 Packet structure

This section describes the packet structure

Standard Packet (command) Format:

0	1	2	3	4	…	n-4	n-3	n-2	n-1	n
Flags	Length(n+1)		Type	Payload			Checksum			

Flags field

This is an 8-bit field. Bits are defined as follows:

Bit	Meaning
0	Undefined: leave as 0
1	If 1, packet has a SHA-256 signature
2	If 1, packet has a “next hash”
3-7	Undefined: leave as 0

Length field:

The length is the total number of bytes. Using the terminology from the diagram above, the value will be n+1 (the Flags field was byte 0 and the final byte of the Checksum is n, so the total length is n+1).

Type field:

The following commands are supported by the ISP:

Command	Value	
	Request	Response
Reset	0x14	0x15
Execute (Run)	0x21	0x22
Set Baud Rate	0x27	0x28
Get Device Info	0x32	0x33
Open Memory For Access	0x40	0x41

Erase Memory	0x42	0x43
Blank Check Memory	0x44	0x45
Read Memory	0x46	0x47
Write Memory	0x48	0x49
Close Memory (prevent access)	0x4A	0x4B
Get Memory Info	0x4C	0x4D
Unlock ISP	0x4E	0x4F
Use Certificate	0x50	0x51
Start Encrypted Transfer	0x52	0x53

In each case, the Response value is the same as the Request value but with bit 0 inverted. If a Request is not recognized, a response is sent with bit 7 inverted instead.

Payload field:

Request packets

The Payload of a Request packet has the following format:

0	. . .	x	x+1	. . .	x+32	x+33	. . .	x+288
Requested Payload			Next Hash			Signature		

Payload request packet format is shown in above table.

The Request Payload is dependent upon the value of the Type field.

The Next Hash and Signature fields are optional, with their presence controlled by bits in the Flags field.

The Next Hash is a SHA-256 hash calculated across all fields of the next packet, except the checksum, as part of an authenticated sequence.

The Signature is a SHA-256 hash calculated across all preceding fields in the packet and encrypted with a 2048-bit RSA private key.

Response packets:

The Payload of a Response packet has the following format

Payload of response packet format:

BYTE 0	BYTE1	. . .	BYTEx
Status	Response Payload		

The Response Payload is dependent upon the value of the Type field, mentioned in above Type field table contains Command, Request and response. The following status codes are defined:

Status	Descriptions
0x00	Success
0xEF	Memory invalid mode
0xF0	Memory bad state
0xF1	Memory too long
0xF2	Memory out of range
0xF3	Memory access invalid
0xF4	Memory not supported
0xF5	Memory invalid
0xF6	No response
0xF7	Not authorized
0xF8	Test error
0xF9	Read fail
0xFA	User interrupt
0xFB	Assert fail
0xFC	CRC error
0xFD	Invalid response
0xFE	Write fail
0xFF	Not supported

Checksum field:

The checksum is a 32-bit CRC calculated across all proceeding fields.

Pseudo-code to generate the CRC for a packet using these functions is as follows:

```
crc_t crc = crc_init();

crc = crc_update(crc, packet_data, packet_length);
```

```
crc = crc_finalize(crc);
```

6.2 References

Please, refer Chapter 38: In-System Programming (ISP) in below mentioned document

https://usermanual.wiki/m/c89bf8a138014857f32914b4f97e63c7033feae2ff4557fe0a5d6f4d157e244c.pdf