LD2420 Serial Protocol

Contents

HLK-LD2024 Serial Protocol - Firmware 1.6.1

Author: Damian Michna
Email: damian@michna.de
Version: 20250209

This is NOT an OFFICIAL dockument of Shenzhen Hi-Link Electronic Co., Ltd. It is based on Hi-Link documentation and my own testing.

How to use this docuument

- The names of **commands**, **functions**, and **data descriptions** are not official designations provided by Shenzhen Hi-Link Electronic Co., Ltd. They have been created solely for the purpose of this document.

About LD2024

The **LD2420** is a state-of-the-art millimeter-wave radar sensor designed for **human presence detection** and **motion sensing**. Utilizing advanced **24GHz ISM band radar** technology, it offers precise detection of stationary and moving human targets, making it ideal for smart home automation, security systems, and energy-saving applications.

Key Features

- **24GHz ISM Band Radar:** Ensures high stability and precision in operation.
- **Human Presence Detection:** Accurately detects both stationary and moving individuals.
- **Adjustable Detection Range:** Configurable parameters for distance and sensitivity.
- **Low Power Consumption:** Suitable for battery-operated and energy-efficient applications.
- **Compact Design:** Small form factor for seamless integration into various devices.

Technical Specifications

Parameter	Value
Operating Frequency	24GHz ISM Band
Detection Range	Up to 6 meters (adjustable)
Power Supply	3.3V DC
Communication Interface	UART (115200 baudrate)
Detection Angle	±60°
Operating Temperature	-20°C to +85°C

Applications

- **Smart Home Automation:** Automatic lighting, HVAC control based on presence.
- **Security Systems:** Intrusion detection, unauthorized access alerts.
- **Energy Management:** Reducing power consumption in unoccupied spaces.
- **Healthcare Monitoring:** Non-intrusive monitoring of elderly or patients.

Resources

For detailed documentation and product support, visit the Shenzhen Hi-Link Electronic Co., Ltd official page .

> **Note:** Ensure proper configuration and calibration of the LD2420 for optimal performance in your specific application.*

Connection

Electrical connection and serial parameters

UART Pinout (J2) and Connection to Client Interface

LD2420 J2	Client UART	Description
OT2	optional	Presence Output
RX	TX	
OT1	RX	
GND	GND	Ground
3V3	3V3	Power input 3.3V

Devices with firmware < 1.5.8 have inverted signals on OT1 and OT2.

Serial Parameters

- **Data Bits:** 8
- **Parity:** None (N)
- **Stop Bits:** 1
- **Default Baud Rate (Firmware > 1.5.8):** 115200

- **Default Baud Rate (Firmware < 1.5.8):** 256000

SWD Pinout (J1)

MCU connection for firmware upgrade and debugging

LD2420 J1	Description
3V3	Power input 3.3V
CLK	SWD Interface Clock Line
DIO	SWD Interface Data Line
GND	Ground

Target device

The LD2420 utilizes the **Puya PY32F030x6** MCU with **32KB Flash Memory**, based on **Cortex-M0+** architecture.

Detailed information about SWD connections will be provided later in this document.

Communication Modes

The device may operate in five diffrent modes:

- **Running Mode**
- **Monitor Mode** (also known as: Report Mode)
- **Debug Mode**
- **Command Mode**
- **Upgrade Mode**
- Upon power-on or reboot, the device starts in **Running Mode**.
- To switch modes, first enter **Command Mode**, then issue the `set_mode` command.
- Some commands in this document may be used without explicitly entering **Command Mode**.

  Note: Additional undocumented modes may exist.

Running mode

- Default mode of operation after startup or reboot.
- Continuously transmits data over the serial interface.
- Use the `set_mode` command (detailed later) to switch to **Running Mode**.

Data Frame Structure

- If presence is detected the device sends a stream of ascii characters:

```
Range X\r\n\ON\r\n
ON\r\n
```

Where `X` is an integer - one or more bytes long (e.g., `Range 220\r\n`).
Hex representation:

```
52 61 6e 67 65 20 32 32 30 0d 0a 4f 4e 0d 0a 4f 4e 0d 0a
```

- If presence is not detected the deviece sends a stream of ascii characters:

```
OFF\r\n
```

Hex representation:

```
4f 46 46 0d 0a
```

Monitor Mode

To switch to **Monitor Mode**, use the `set_mode` command (detailed later).

NOTE: Section pending completion.

Debug Mode

To switch to **Debug Mode**, use the `set_mode` command (detailed later).

NOTE: Section pending completion.

Command Mode

- Enter with `open_command_mode` command.
- Exit with `close_command_mode` command.

Frame Structure

All commands sent to the device must be encapsulated in a frame consisting of a header, length, command, frame type, parameters (optional), and a footer.

- Some commands (e.g., `reboot`,`set_upgrade_mode`) do **not** generate a response frame.
- Certain commands return responses in **alternative frame formats** (e.g., `set_baudrate`).
- The total frame length **must not exceed 64 bytes**.

Command Frame (Client to Device)

- **Header** : fd fc fb fa (4 bytes)
- **Length** : 2 bytes integer of sum bytes command and parameters
- **Command** : 1 byte command
- **Type** : 1 byte set to 00 = request (1 byte)
- **Params** : 0, 2 or more bytes parameter, depending on command type
- **Footer** : 04 03 02 01 (4 bytes)

Response Frame (Device to Client)

- **Header** : fd fc fb fa (4 bytes)
- **Length** : 2 bytes integer of sum bytes command and parameters
- **Command** : 1 byte command
- **Type** : 1 byte set to 01 = response (1 byte)
- **Status** : 2 bytes, 00 00 = ACK, 01 00 = NACK
- **Result** : 0, 2 or more bytes result, depending on command type
- **Footer** : 04 03 02 01 (4 bytes)

Data Encoding

- Values in frames are encoded in **Little Endian** format.
- Supported data sizes include **8-bit, 16-bit, and 32-bit** values.

Upgrade Mode

- Entered using `set_upgrade_mode` (command `0x74`).
- Currently, there is no known way to exit this mode until firmware download completes.

  TODO: Determine how to exit **Upgrade Mode**.

Commands

Commands overview

- `ff` - Open Command Mode
- `fe` - Close Command Mode
- `00` - Read Firmware Version
- `11` - Read Serial Number
- `10` - Write Serial Number
- `02` - Read Register
- `01` - Write Register
- `08` - Read ABD Parameter
- `07` - Write ABD Parameter
- `12` - Set Operation Mode
- `26` - Set Serial Baud Rate
- `27` - Get Current Baud Rate
- `60` - Command 60
- `61` - Command 61
- `62` - Command 62

- `64` - Command 64
- `68` - Reboot the Device
- `70` - Get Active Firmware
- `71` - Get Upgrade Partition
- `72` - Init Firmware Upgrade
- `73` - Send Firmware Block
- `74` - Set Upgrade Mode
- `75` - Get Firmware ID

Open Command Mode

`ff` - `open_command_mode`

Entering Command Mode forces the device to exit active mode and stops the continuous transmission of data (Standard, Monitor, or Debug Mode). Some commands can only be executed successfully in Command Mode. After exiting Command Mode, the device resumes the last active mode and restarts data transmission.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`ff`)
7	**Type** (`00` = request)
8-9	**ID** - 2-byte integer (`01 00`)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0400 ff00 0100 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`ff`)
7	**Type** (`01` = response)
8-9	**Status** - 2-byte integer (`00 00` = ACK; `01 00` = NACK)
10-11	**Protocol_Version** (2 bytes)
12-13	**Buffer_Size** (2 bytes)
14-17	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0800 ff01 0000 0200 2000 04030201`

- **ID** - Client ID (2 bytes, must be set)
- **Protocol_Version** - 2 bytes, typically set to `0x02`
- **Buffer_Size** - Likely represents the serial read buffer size in bytes (here, 32 bytes)

  UNCLEAR: Exact meaning of `Buffer_Size`.

  NOTE: This command may need to be sent multiple times to clear the serial buffer, as observed in the Hi-Link Tool.

Close Command Mode

`fe - close_command_mode`

This command ends the configuration mode and restores the radar to its previous working mode. To issue further commands, Command Mode must be enabled again.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`fe`)
7	**Type** (`00` = request)
8-9	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 fe00 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`ff`)
7	**Type** (`01` = response)
8-9	**Status** - 2-byte integer (`00 00` = ACK; `01 00` = NACK)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0400 fe01 0000 04030201`

Read Firmware Version

`00` - `get_version`

This command retrieves the firmware version as a string.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`00`)
7	**Type** (`00` = request)
8-11	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 0000 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`0c 00`)
6	**Command** (`00`)
7	**Type** (`01` = response)
8-9	**Status** - 2-byte integer (`00 00` = ACK; `01 00` = NACK)
10-11	**Length_Of_String** (`06 00`)
12-17	**Firmware_Version** (`76 31 2e 36 2e 31`) as string: `'v1.6.1'`
18-21	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0c00 0001 0000 0600 76312e362e31 04030201`

- **Length_Of_String** - 2 bytes
- **Firmware_Version** - A string with the length specified in `Length_Of_String` (here, `'v1.6.1'`).
- Byte positions of **Firmware_Version** and **Footer** depend on **Length_Of_String**.

Read Serial Number

`11` - `get_serial`

This command returns the serial number of the device.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`11`)
7	**Type** (`00` = request)
8-11	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 1100 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`0e 00`)
6	**Command** (`11`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-11	**Length_Of_Serial** 2 byte int (`08 00`)
12-19	**Serial** (`01 02 03 04 05 06 07 08`)
20-23	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0e00 1101 0000 0800 0102030405060708 04030201`

- **Length_Of_Serial** - 2-byte integer
- **Serial** - Stream of bytes with a maximum length of 32 bytes
- Byte positions of **Serial** and **Footer** depend on **Length_Of_Serial**.

Write Serial Number

`10 - set_serial`

This command writes a new serial value to the device.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`0c 00`)
6	**Command** (`10`)
7	**Type** (`00` = request)
8-9	**Length__Of__Serial** 2 byte int (`08 00`)
10-17	**Serial** (`01 02 03 04 05 06 07 08`)
18-21	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0c00 1000 0800 0102030405060708 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`10`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0400 1001 0000 04030201`

- **Length__Of__Serial** - 2-byte integer
- **Serial** - Stream of bytes with a maximum length of 32 bytes
- Byte positions of **Serial** and **Footer** depend on **Length__Of__Serial**.
- The serial number should not exceed 32 bytes.

  NOTE: Testing up to 50 bytes resulted in a response containing only the first 32 bytes, with the remaining bytes set to `00`.

Read Register

`02 - get_register`

This command reads the value stored in a given register.

> **NOTE:** This section is pending completion.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`06 00`)
6	**Command** (`02`)
7	**Type** (`00` = request)
8-9	**Chip** - 2 byte integer (`00 00`)
10-11	**Register** - 2 byte integer (`00 00`)
n-m	(OPTIONAL) additional registers, 2 bytes * N
12-15	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0600 0200 4000 0000 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`18 00`)
6	**Command** (`08`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte integer (`00 00` = ACK; `01 00` = NACK)
10-11	**Value** - 2 byte integer (`07 12`)
n-m	(OPTIONAL) additional values received, 4 bytes * N
12-15	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0600 0201 0000 0712 04030201`

- **Chip** - 2 byte int (`40 00`) - chip-address, may be `0x40`, `0x41`, `0xf8`, `0xf9`
- **Register** - 2 byte int (`00 00`)
- **Value** - 2 byte int (`07 12`) - response value for requested register

  NOTE: maximum 25 registers can be read in one frame, because the response frame would exceed

  NOTE: I coudld read register for chip 0x40, 0x41, 0xf8, 0xf9 from register 0x00 to 0xffff. The responce was the same wiederholt in periods of 0xff, so I suppose that there exists only 256 register

  TODO: find the meaning of registers

Write Register

`01 - set_register`

This command writes a specified value to a given register.

> **NOTE:** This section is pending completion.

\ + 2-byte chip address + (2-byte address + 2-byte data) * N

> **TODO:** check, if writing to same register on difrent chips returns the same values for all chips

Read ABD Parameter

`08 - get_parameter`

This command retrieves specific ABD parameters from the device.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`0c 00`)
6	**Command** (`08`)
7	**Type** (`00` = request)
8-9	**Register** - 2 byte int (00 00) # first value
n-m	(OPTIONAL) additional registers for requested, 2 bytes * N
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0c00 0800 0000 0100 0200 0300 0400 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`18 00`)
6	**Command** (`08`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Value** - 4 byte int (00 00 00 00) # first value
n-m	(OPTIONAL) additional values received, 4 bytes * N
14-17	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 1800 0801 0000 00000000 0c000000 01000000 0a000000 1e000000 04030201`

- **Register** - 2 byte int -
- **Value** - 4 byte int -
- For register description and default values see table: ADB Parameters

  UNCLEAR Max frame length possible. Should be less 64Bytes, not clear if this limit is vor sending or receiving frame only.

Write ABD Parameter

`07 - set_parameter`

This command modifies ABD parameters on the device.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`08 00`)
6	**Command** (`07`)
7	**Type** (`00` = request)
8-9	**Register** - 2 byte int (04 00) # Absence Report Delay Time
10-13	**Value** - 4 byte int (1e 00 00 00) # 30 seconds
n-m	more register-value pairs to write
14-17	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0800 0700 0400 1e000000 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`07`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0400 0701 0000 04030201`

- **Register** - 2 byte int -
- **Value** - 4 byte int -
- Registers are 2 bytes long, and values are 4 bytes long, both encoded in little-endian format.

  UNCLEAR Max frame length possible. Should be less 64Bytes, not clear if this limit is vor sending or receiving frame only.

Set Operation Mode

`12 - set_mode`

This command changes the mode of operation. The available mode values are:

- `00` - Debug mode
- `04` - Report mode
- `64` - Standard mode

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`12`)
7	**Type** (`00` = request)
8-9	**Unspecified** - 2-byte integer (`00 00`)
10-13	**Mode** - 4 byte int
14-17	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0800 1200 0000 00000000 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`12`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0400 1201 0000 04030201`

- **Unspecified** - 2-byte integer (Purpose unknown)
- **Mode** - 4-byte integer
  - `0x00` - Debug mode
  - `0x04` - Report mode
  - `0x64` - Standard mode
- **TODO:** Determine the purpose of the **Unspecified** value.

Examples

- `fdfcfbfa 0800 1200 0000 00000000 04030201` - change to debug mode
- `fdfcfbfa 0800 1200 0000 04000000 04030201` - change to report mode
- `fdfcfbfa 0800 1200 0000 64000000 04030201` - change to standard mode

Set Baud Rate

`26 - set_baudrate`

- Changes the device's baud rate.
- **Reboot required** to apply changes.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`26`)
7	**Type** (`00` = request)
7	**Mode** - 2 byte int (01 00 till 08 00)
8-11	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 26 00 04030201`

Byte	Response Frame Description
	command do not have standard format
0-15	**String** ('baudrate:' 62 61 75 64 72 61 74 65 3a)
16-17	**CRLF String** (0d 0a)
Example:	`53657474696e67205375636365737321 0d0a 'Setting Success!\r\n'`

Supported Baud Rates:

Mode	Baudrate
1	9600
2	19200
3	38400
4	57600
5	115200 *(default for firmware > 1.5.8)*
6	230400
7	256000 *(default for firmware < 1.5.8)*
8	460800

Get Current Baud Rate

27 - `get_baudrate`

Retrieves the currently configured baud rate of the serial interface. This is useful for verifying the baud rate after making changes, ensuring the correct setting is applied, or confirming the configuration before a system reboot.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`27`)
7	**Type** (`00` = request)
8-11	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 27 00 04030201`

Byte	Response Frame Description
	command do not have standard format
0-8	**String** ('baudrate:' 62 61 75 64 72 61 74 65 3a)
9	**Baudrate Mode String** (as ASCII number, '5' = 35)
10-11	**CRLF String** (0d 0a)
Example:	`62 61 75 64 72 61 74 65 3a 32 0d 0a 'baudrate:2\r\n'`

Command 60

`60` - Unclassified

Die Auswirkung dieser command ist momentan nicht bekannt

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`fe`)
7	**Type** (`00` = request)
8-9	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 6000 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`ff`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0400 6001 0000 04030201`

TODO: Find out what for this command is

Command 61

`61` - Unclassified

Die Auswirkung dieser command ist momentan nicht bekannt. Der Response liefert mehrere 2byte werte.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`fe`)
7	**Type** (`00` = request)
8-9	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa0200610004030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`ff`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-11	**Val_01** - 2 byte int - unknown
12-13	**Val_02** - 2 byte int - unknown
14-15	**Val_03** - 2 byte int - unknown
16-17	**Val_04** - 2 byte int - unknown
18-19	**Val_05** - 2 byte int - unknown
20-21	**Val_06** - 2 byte int - unknown
22-23	**Val_07** - 2 byte int - unknown
24-25	**Val_08** - 2 byte int - unknown
26-27	**Val_09** - 2 byte int - unknown
28-29	**Val_10** - 2 byte int - unknown
30-31	**Val_11** - 2 byte int - unknown
32-33	**Val_12** - 2 byte int - unknown
34-35	**Val_13** - 2 byte int - unknown
36-37	**Val_14** - 2 byte int - unknown
38-39	**Val_15** - 2 byte int - unknown
40-43	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 2200 6101 0000 0c0c 0100 0100 0100 0100 0100 0100 0100 0100 0100 0100 0100 0100 0100 1e00 04030201`

- **Val_01** - **Val_15** - 2 byte values - unknown meaning

  TODO: Find out what for this command is **TODO:** Find out what the 2 byte values represent

Command 64

`64` - Unclassified

Die Auswirkung dieser command ist momentan nicht bekannt.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`fe`)
7	**Type** (`00` = request)
8-9	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 6400 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`ff`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0400 6401 0000 04030201`

TODO: Find out what for this command is

Reboot Device

`68` - `reboot`

This command reboots the device.

By sending the `reboot` command, the device restarts immediately without sending response frame.

After `reboot` the device returns to **standard mode**.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`68`)
7	**Type** (`00` = request)
8-11	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 68 00 04030201`

Byte	Response Frame Description
	command do not send response frame

Get Active Firmware

`70` - get_active_firmware

This command is used from HKL-LD2420_Tool(v1.2.0.0)

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`70`)
7	**Type** (`00` = request)
8-11	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 7000 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-6	**Length** (`08 00`)
6	**Command** (`70`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Firmware** (`01 00 00 00`)
14-17	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0800 7001 0000 01000000 04030201`

- **Firmware** returns the version of actually active loaded firmware. It is 4Byte value
  - 0x01 - Bootloader
  - 0x02 - App O
  - 0x04 - App 1

Get Upgrade Partition

`71` - get_upgrade_partition

This command is used from HKL-LD2420_Tool(v1.2.0.0)

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`71`)
7	**Type** (`00` = request)
8-11	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 7100 04030201_`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-6	**Length** (`08 00`)
6	**Command** (`71`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Partition** (`01 00 00 00`)
14-17	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0800 7101 0000 01000000 04030201`

- **Partition** - 4 bytes - memory parition witch will be flashed
  - 0x01 — App O Brauchen Sie Programmierung
  - 0x02 — App 1 Brauchen Sie Programmierung
  - other values are probably error

Init Firmware Upgrade

`72 - init_firmware_upgrade`

This command is used from HKL-LD2420_Tool(v1.2.0.0)
Probably initialisation of the transmission, will be sent immediately before 73

> **NOTE:** Do not use this! Probably it brakes the device!

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`0e 00`)
6	**Command** (`72`)
7	**Type** (`00` = request)
8-11	**Partition** (`01 00 00 00`) - partition nr (?)
12-15	**File_Length** (`00 04 00 00`) 1024 Bytes
16-19	**Checksum** (`c6 fc 01 00`)
20-23	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0e00 7200 01000000 00040000 c6fc0100 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-6	**Length** (`08 00`)
6	**Command** (`72`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Data-Status** (`94 00 00 00`)
14-17	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0800 7201 0000 9400 0000 04030201`

- **File_Length** - 4 bytes whole file length in bytas to be transmitted
- **Checksum** - 4 bytes -is the sum of all bytes in the file to be transmitted, encoded as a 4-byte little-endian value. (See: Code examples)
- **Data-Status** - 4 bytes status of operation success or error
  - 0x01 - md partition not available
  - 0x02 - data length error
  - 0x04 - flash erase error
  - buffersize, i.e. 0x94 = 148 bytes buffer, only when `Status` = 0

Send Firmware Block

`73 - send_firmware_block`

This command is used from HKL-LD2420_Tool(v1.2.0.0)
It starts transmission of firmware blocks.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`8a 00`)
6	**Command** (`73`)
7	**Type** (`00` = request)
8-11	**Counter** of send DATA BLOCK (first block = 0) (`01 00 00 00`)
12-15	**Checksum** (`33 41 00 00`)
16-144	**Data Block** (128 bytes block)
145-148	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 8a00 7300 01000000 33410000 (128 bytes block) 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`08 00`)
6	**Command** (`73`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Data-Status** (`80 00 00 00`) (Dec: `128`)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0800 7301 0000 80000000 04030201`

- **Counter** of send DATA BLOCK (first block = 0) (`01 00 00 00`) |
- **Checksum** - 4 bytes - is the sum of the bytes in the transmitted part/block, also encoded as a 4-byte little-endian value. (See: Code examples)
- **Data Block** - block of data up to 128 byte to be transfered
- **Data-Status** - 4 byte status:
  - 0x00 - These packet data are correctly written
  - 0x01 - Error in the packet's serial number.
  - 0x02 - Flash write error.
  - 0x04 - Flash read and abort error
  - 0x08 - Checksum error for comparison packets
  - 0x10 - Error in data length
  - 0x20 - Bytes are not 4-byte aligned
  - 0x40 - Error in file verification
  - 0x80 - Successful programming

Set Upgrade Mode

`74` - set_upgrade_mode

This command is used from HKL-LD2420_Tool(v1.2.0.0)

After sending this command the device stops to send packets and do not respond or responds with NACK on almost all commands except 0x70, 0x71, 0x75.

> **NOTE:** Do not use this! It brakes the device, if you not make transfer! I did not find out how to come back in normal mode withaut transfering data.
>
> **NOTE:** After sending this command first time, there ist no response frame. After sending this command again, in the upgrade mode, there is NACK response.

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`74`)
7	**Type** (`00` = request)
8-9	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 7400 04030201`

Byte	First Response Frame Description
	command do not send response frame

Byte	Following Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`07`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0400 7401 0100 04030201`

> **TODO:** Find out, how to get out back from upgrade mode without starting of firmware transfer

Get Firmware ID

`75 - get_firmware_id`

This command is used from HKL-LD2420_Tool(v1.2.0.0)
It delivers firmware ID as ascii code, here “04PA”

Byte	Request Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`02 00`)
6	**Command** (`75`)
7	**Type** (`00` = request)
8-9	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0200 7500 04030201`

Byte	Response Frame Description
0-3	**Header** (`FD FC FB FA`)
4-5	**Length** (`04 00`)
6	**Command** (`75`)
7	**Type** (`01` = response)
8-9	**Status** - 2 byte int (`00 00` = ACK; `01 00` = NACK)
10-13	**Firmware_ID** (`30 34 50 41`) (String: `04PA`)
14-17	**Footer** (`04 03 02 01`)
Example:	`fdfcfbfa 0800 7501 0000 30345041 04030201`

- **Firmware_ID** - 4 bytes string: i.e. `04PA`)

Code Examples and Tables

ADB Parameters - Registers, Names and Default Values for command 0x07 and 0x08

Address	Default	Dec.	App	Name	Range
0000	00000000	0	0	Minimum Detection Distance Threshold	0x00-0x0F
0100	0c000000	12	12	Maximum Detection Distance Threshold	0x00-0x0F
0200	01000000	1		Minimal Gate ?	
0300	0a000000	10		Maximal Gate ?	
0400	1e000000	30	30	Absence Report Delay Time (seconds)	0x00-0xFF
1000	60ea0000	60000	47,78	Trigger Threshold Gate 0	0-65535
1100	30750000	30000	44,77	Trigger Threshold Gate 1	0-65535
1200	b80b0000	3000	34,77	Trigger Threshold Gate 2	0-65535
1300	d0070000	2000	33,01	Trigger Threshold Gate 3	0-65535
1400	f4010000	500	26,99	Trigger Threshold Gate 4	0-65535
1500	90010000	400	26,02	Trigger Threshold Gate 5	0-65535
1600	90010000	400	26,02	Trigger Threshold Gate 6	0-65535
1700	2c010000	300	24,77	Trigger Threshold Gate 7	0-65535
1800	2c010000	300	24,77	Trigger Threshold Gate 8	0-65535
1900	2c010000	300	24,77	Trigger Threshold Gate 9	0-65535
1a00	2c010000	300	24,77	Trigger Threshold Gate 10	0-65535
1b00	fa000000	250	23,98	Trigger Threshold Gate 11	0-65535
1c00	fa000000	250	23,98	Trigger Threshold Gate 12	0-65535
1d00	c8000000	200	23.01	Trigger Threshold Gate 13	0-65535
1e00	c8000000	200	23.01	Trigger Threshold Gate 14	0-65535
1f00	c8000000	200	23.01	Trigger Threshold Gate 15	0-65535
2000	409c0000	40000	46,02	Hold Threshold Gate 0	0-65535
2100	204e0000	20000	43,01	Hold Threshold Gate 1	0-65535
2200	90010000	400	26,02	Hold Threshold Gate 2	0-65535
2300	2c010000	300	24,77	Hold Threshold Gate 3	0-65535
2400	2c010000	300	24,77	Hold Threshold Gate 4	0-65535
2500	c8000000	200	23.01	Hold Threshold Gate 5	0-65535
2600	c8000000	200	23.01	Hold Threshold Gate 6	0-65535
2700	96000000	150	21.76	Hold Threshold Gate 7	0-65535
2800	96000000	150	21.76	Hold Threshold Gate 8	0-65535
2900	64000000	100	20.00	Hold Threshold Gate 9	0-65535
2a00	64000000	100	20.00	Hold Threshold Gate 10	0-65535
2b00	64000000	100	20.00	Hold Threshold Gate 11	0-65535
2c00	64000000	100	20.00	Hold Threshold Gate 12	0-65535
2d00	64000000	100	20.00	Hold Threshold Gate 13	0-65535
2e00	64000000	100	20.00	Hold Threshold Gate 14	0-65535
2f00	64000000	100	20.00	Hold Threshold Gate 15	0-65535

Supported Baud Rates

Mode	Baudrate
1	9600
2	19200
3	38400
4	57600
5	115200 *(default for firmware > 1.5.8)*
6	230400
7	256000 *(default for firmware < 1.5.8)*
8	460800

Checksum calculation command 0x72 and 0x73

```
import struct

def calculate_checksum(data):
    return struct.pack("<I", sum(data) & 0xFFFFFFFF)
```

Threshold value converting vor command 0x07 and 0x08

```
from math import log10

def decimal_threshold(normalized_threshold):
    return int(10 ** (normalized_threshold / 10))

def normalized_threshold(decimal_threshold): # the value of the HLK Tool
    return 10 * log10(decimal_threshold)
```

Register and default values (commands 0x01, 0x02)

There seems to be 256 register, forther we will find the meanings of this registers It is possible to read with chip-addr 0x40, 0x41, 0xf8, 0xf9. All this sets have the sames values, so there are probably only 256 registers available

Reg	Value	Dezimal	Flash Pos.	Description
00	0712	4615		
01	209e	40480	5f70	
02	6e10	4206	5ef0	
03	1f10	4127		
04	0c02	524	5ef4	
05	1000	16	5f04	
06	4901	329	5f08	
07	b201	434	5f0c	
08	1c00	28	5f10	
09	0169	26881	5ef8	
0a	0042	16896	5efc	
0b	6ec0	49262	5f00	
0c	1f4e	19999		
0d	0010	4096	5f14	
0e	0040	16384	5f18	
0f	0000	0		
10	0110	4097		
11	0300	3		
12	0000	0		
13	0000	0		
14	035a	23043	5f1c	
15	0817	5896	5f20	
16	0003	768		
17	1002	528	5f24	
18	0000	0		
19	0000	0		
1a	0000	0		
1b	0000	0		
1c	0000	0		
1d	0000	0		
1e	0000	0		
1f	0000	0		
20	0000	0	5f28	
21	0000	0	5f2c	
22	0000	0	5f30	
23	dc1d	7644	5f34	
24	5e1d	7518	5f38	
25	e21c	7394	5f3c	
26	641c	7268	5f40	
27	d017	6096	5f44	
28	d416	5844	5f48	
29	dc15	5596	5f4c	
2a	dc15	5596	5f50	
2b	dc15	5596	5f54	
2c	dc15	5596	5f58	
2d	dc15	5596	5f5c	
2e	dc15	5596	5f60	
2f	dc15	5596	5f64	
30	0000	0		

Reg	Value	Dezimal	Flash Pos.	Description
31	0000	0		
32	0000	0		
33	0000	0		
34	0000	0		
35	0000	0		
36	0000	0		
37	0000	0		
38	0000	0		
39	0000	0		
3a	0000	0		
3b	0000	0		
3c	0000	0		
3d	0000	0		
3e	0000	0		
3f	0000	0		
40	0702	519	5f78	
41	44c8	51268	5f74;5fae	
42	0100	1	5e54	
43	8038	14464	5e58	
44	4000	64	46d9;5e5c	
45	0000	0	5e60	
46	a00f	4000	5e64	
47	0010	4096	5609;5e68	
48	10a4	42000	5e6c	
49	0020	8192	5e70	
4a	f82a	11000	5e74	
4b	0000	0	5e78	
4c	d859	23000	5e7c	
4d	0000	0	5e80	
4e	0100	1	5e84	
4f	0000	0	5e88	
50	a00f	4000	5e8c	
51	f400	244	5e90	
52	0024	9216	5e94	
53	020a	2562	5e98	
54	abaa	43691	5e9c	
55	0000	0	5ea0	
56	2400	36	5ea4	
57	ffff	65535	5ea8	
58	75ff	65397	5eac	
59	0000	0	5eb0	
5a	0000	0	5eb4	
5b	2200	34	5eb8	
5c	2200	34	5ebc	
5d	1919	6425	5ec0	
5e	00ff	65280	5ec4	
5f	0510	4101		
60	9c01	412		
61	2100	33	5ec8	
62	2100	33	5ecc	
63	2100	33	5ed0	
64	2100	33	5ed4	
65	0000	0		
66	000a	2560	5fc6;5edc	
67	4018	6208	5f6c;5fca	

Reg	Value	Dezimal	Flash Pos.	Description
68	0000	0	5e24	
69	0600	6	5fce	
6a	413c	15425		
6b	006d	27904		
6c	9099	39312	5fb6;5ee0	
6d	c093	37824	5fba;5ee4	
6e	fc83	33788	5fc2;5ed8	
6f	0055	21760		
70	a02a	10912	5fbe;5ee8	
71	a120	8353		
72	5306	1619	5fb2	
73	0080	32768		
74	0100	1		
75	0400	4		
76	2100	33	5eec	
77	0000	0		
78	0000	0		
79	0000	0		
7a	0000	0		
7b	0000	0		
7c	0000	0		
7d	0000	0		
7e	0000	0		
7f	0000	0		
80	0712	4615		
81	209e	40480		
82	6e10	4206		
83	1f10	4127		
84	0c02	524		
85	1000	16		
86	4901	329		
87	b201	434		
88	1c00	28		
89	0169	26881		
8a	0042	16896		
8b	6ec0	49262		
8c	1f4e	19999		
8d	0010	4096		
8e	0040	16384		
8f	0000	0		
90	0110	4097		
91	0300	3		
92	0000	0		
93	0000	0		
94	035a	23043		
95	0817	5896		
96	0003	768		
97	1002	528		
98	0000	0		
99	0000	0		
9a	0000	0		
9b	0000	0		
9c	0000	0		
9d	0000	0		
9e	0000	0		

Reg	Value	Dezimal	Flash Pos.	Description
9f	0000	0		
a0	0000	0		
a1	0000	0		
a2	0000	0		
a3	dc1d	7644		
a4	5e1d	7518		
a5	e21c	7394		
a6	641c	7268		
a7	d017	6096		
a8	d416	5844		
a9	dc15	5596		
aa	dc15	5596		
ab	dc15	5596		
ac	dc15	5596		
ad	dc15	5596		
ae	dc15	5596		
af	dc15	5596		
b0	0000	0		
b1	0000	0		
b2	0000	0		
b3	0000	0		
b4	0000	0		
b5	0000	0		
b6	0000	0		
b7	0000	0		
b8	0000	0		
b9	0000	0		
ba	0000	0		
bb	0000	0		
bc	0000	0		
bd	0000	0		
be	0000	0		
bf	0000	0		
c0	0702	519		
c1	44c8	51268		
c2	0100	1		
c3	8038	14464		
c4	4000	64		
c5	0000	0		
c6	a00f	4000		
c7	0010	4096		
c8	10a4	42000		
c9	0020	8192		
ca	f82a	11000		
cb	0000	0		
cc	d859	23000		
cd	0000	0		
ce	0100	1		
cf	0000	0		
d0	a00f	4000		
d1	f400	244		
d2	0024	9216		
d3	020a	2562		
d4	abaa	43691		
d5	0000	0		

Reg	Value	Dezimal	Flash Pos.	Description
d6	2400	36		
d7	ffff	65535		
d8	75ff	65397		
d9	0000	0		
da	0000	0		
db	2200	34		
dc	2200	34		
dd	1919	6425		
de	00ff	65280		
df	0510	4101		
e0	9c01	412		
e1	2100	33		
e2	2100	33		
e3	2100	33		
e4	2100	33		
e5	0000	0		
e6	000a	2560		
e7	4018	6208		
e8	0000	0		
e9	0600	6		
ea	413c	15425		
eb	006d	27904		
ec	9099	39312		
ed	c093	37824		
ee	fc83	33788		
ef	0055	21760		
f0	a02a	10912		
f1	a120	8353		
f2	5306	1619		
f3	0080	32768		
f4	0100	1		
f5	0400	4		
f6	2100	33		
f7	0000	0		
f8	0000	0		
f9	0000	0		
fa	0000	0		
fb	0000	0		
fc	0000	0		
fd	0000	0		
fe	0000	0		
ff	0000	0		