## Quansheng UV-K5 Communication Protocol and Operations Manual

**Disclaimer:** This document is based on community-driven reverse engineering and analysis of the Quansheng UV-K5 radio, primarily from projects like CHIRP and third-party firmware/tools. It is not an official specification from Quansheng. Use this information at your own risk; incorrect commands or data could potentially lead to radio malfunction.

### 1. Introduction

This document outlines the serial communication protocol used by the Quansheng UV-K5 series of handheld radios for programming and control via a PC. It also describes the general operations (commands) that can be performed using this protocol. The primary purpose is to enable developers to create custom software for managing the radio's memory and settings.

### 2. Physical Connection and Serial Port Settings

* **Connection:** Communication with the UV-K5 typically occurs via a Kenwood K-type 2-pin (K1) programming cable. This cable usually incorporates a USB-to-Serial converter chip (e.g., CH340, PL2303, FTDI). The radio's USB-C port is generally for charging and firmware updates in a specific mode, not for this programming protocol.
* **Serial Port Settings:**
  + **Baud Rate:** 38400 bps
  + **Data Bits:** 8
  + **Parity:** None
  + **Stop Bits:** 1
  + **Flow Control:** None

### 3. General Protocol Structure

The communication protocol involves sending command packets to the radio and receiving response packets.

#### 3.1. Command Packet Structure (PC to Radio)

Commands sent to the radio follow this general structure:

<HEADER><LENGTH\_BYTE><NULL\_BYTE><OBFUSCATED\_PAYLOAD><FOOTER>

* **HEADER:** 0xAB 0xCD (2 bytes)
* **LENGTH\_BYTE:** (1 byte) Specifies the length of the *original, unobfuscated* data payload (DATA + CRC16).
* **NULL\_BYTE:** 0x00 (1 byte)
* **OBFUSCATED\_PAYLOAD:** This is the DATA concatenated with its CRC16, which is then XOR obfuscated.
  + **DATA:** The actual command and its parameters.
  + **CRC16:** A 2-byte CRC-16/XMODEM checksum of the DATA. The polynomial is 0x1021. The CRC is appended to DATA (low byte first, then high byte).
* **FOOTER:** 0xDC 0xBA (2 bytes)

#### 3.2. Response Packet Structure (Radio to PC)

Responses from the radio follow a similar structure:

<HEADER><LENGTH\_BYTE><NULL\_BYTE><OBFUSCATED\_RESPONSE\_DATA><FOOTER>

* **HEADER:** 0xAB 0xCD (2 bytes)
* **LENGTH\_BYTE:** (1 byte) Specifies the length of the *obfuscated* response data.
* **NULL\_BYTE:** 0x00 (1 byte)
* **OBFUSCATED\_RESPONSE\_DATA:** The XOR obfuscated response from the radio. This needs to be de-obfuscated to get the actual response data.
* **FOOTER:** 0xDC 0xBA (2 bytes) (Note: Some sources suggest the footer in responses might be slightly different or less strictly checked, but 0xDC 0xBA is a common observation, sometimes with the last two bytes of the obfuscated payload also being 0xDC 0xBA).

#### 3.3. XOR Obfuscation

Both command payloads (Data + CRC) sent to the radio and response data received from the radio are obfuscated using a byte-wise XOR operation with a fixed 16-byte key.

* XOR Key Table (from CHIRP uvk5.py):  
  [22, 108, 20, 230, 46, 145, 13, 64, 33, 53, 213, 64, 19, 3, 233, 128]  
  (In decimal: 0x16, 0x6C, 0x14, 0xE6, 0x2E, 0x91, 0x0D, 0x40, 0x21, 0x35, 0xD5, 0x40, 0x13, 0x03, 0xE9, 0x80)
* Obfuscation Process:  
  For each byte in the data to be obfuscated/de-obfuscated:  
  output\_byte = input\_byte XOR key\_table[index % 16]  
  The index increments for each byte processed.

#### 3.4. CRC-16/XMODEM Calculation

A CRC-16/XMODEM checksum is calculated for the DATA portion of the command payload before obfuscation.

* **Polynomial:** 0x1021
* **Initial Value:** 0x0000
* The resulting 2-byte CRC is typically appended to the data, low byte first.

### 4. Memory Map / EEPROM Structure (Based on CHIRP)

The radio's configuration, including channels and settings, is stored in an EEPROM. The CHIRP driver (uvk5.py) provides a C-style structure definition (MEM\_FORMAT) that describes this layout. This is crucial for understanding how to read and write specific settings.

**Key EEPROM Sections (Conceptual):**

* **Channel Data (approx. 200-214 channels):**
  + freq: Frequency (e.g., BCD encoded or direct Hz)
  + offset: Repeater offset frequency
  + rxcode: RX CTCSS/DCS code
  + txcode: TX CTCSS/DCS code
  + code\_flag: Flags for CTCSS/DCS type (None, CTCSS, DCS Normal, DCS Inverted)
  + flags1: Various flags like AM mode, scan list inclusion.
    - FLAGS1\_OFFSET\_DIR\_MASK (0b11) (00: None, 01: +, 10: -)
    - FLAGS1\_ISSCANLIST (0b100)
    - FLAGS1\_ISAM (0b10000)
  + flags2: More flags like power level, bandwidth, busy channel lockout (BCLO).
    - FLAGS2\_REVERSE (0b1)
    - FLAGS2\_BANDWIDTH\_MASK (0b110) (00: Wide, 01: Narrow)
    - FLAGS2\_POWER\_MASK (0b1100) (00: Low, 01: Medium, 10: High)
    - FLAGS2\_BCLO (0b10000)
  + dtmf\_flags: DTMF related flags (e.g., PTT ID).
  + step: Frequency step (e.g., 2.5, 5.0, 6.25, 10.0, 12.5, 25.0 kHz)
  + scrambler: Scrambler setting (0 for off, 1-10 for different scrambler options)
  + channel\_name (Stored in a separate area, typically 16 bytes per name)
* **FM Radio Presets (approx. 16-20 channels):**
  + fmfreq: FM broadcast band frequency.
* **VFO Settings (A and B):** Similar structure to channel data for current VFO A and B settings.
* **General Radio Settings:**
  + squelch: Squelch level (0-9)
  + vox\_level: VOX level
  + mic\_gain: Microphone gain
  + channel\_display\_mode: (Frequency, Channel No, Channel Name)
  + battery\_save: Battery save ratio
  + dual\_watch: Dual Watch / TDR setting
  + beep\_control: Keypad beep
  + scan\_resume\_mode: (TO, CO, SE)
  + auto\_keypad\_lock: Auto keypad lock status
  + power\_on\_dispmode: Power-on display (Full Screen, Welcome Info, Voltage)
  + language: (Chinese, English)
  + alarm\_mode: Alarm mode
  + logo\_line1, logo\_line2: Custom power-on message lines.
  + And many other specific settings related to DTMF, scan ranges, etc.

**Note:** The exact offsets and sizes can be found in the MEM\_FORMAT definition within the CHIRP uvk5.py driver or by analyzing EEPROM dumps. The total EEPROM size relevant for programming is often around 0x2000 bytes, with programming typically done in blocks (e.g., 0x80 bytes).

### 5. Key Operations / "Commands"

The protocol doesn't use human-readable ASCII commands like "SET\_FREQ". Instead, operations are performed by reading or writing specific blocks of the EEPROM or sending specialized command sequences.

* Entering Programming Mode:  
  Typically, the radio enters programming mode automatically when a valid command sequence is received over the serial port after being powered on with the programming cable connected. Some custom firmwares might require holding a key (like PTT) while powering on to enter a bootloader/programming mode for firmware flashing, which is different from EEPROM programming.
* Reading EEPROM / Configuration:  
  Commands are sent to request blocks of memory from the radio. The radio responds with the requested data, which is then de-obfuscated.
  + Example: A command to read N bytes from EEPROM address ADDR.
* Writing EEPROM / Configuration:  
  Data blocks (prepared according to the memory map and obfuscated) are sent to the radio to be written to specific EEPROM addresses.
  + Example: A command to write N bytes of DATA to EEPROM address ADDR.
* Reading Radio Identification / Firmware Version:  
  A specific command usually exists to query the radio's model or firmware version.
  + CHIRP example: \_send\_command(radio, b"\x05") might be an initial handshake or version request. The response b"\x05QS-K520123" indicates a Quansheng UV-K5 with firmware 2.01.23.
* Specific Register Access (e.g., BK4819 DSP chip):  
  Some custom firmwares (like quansheng-dock-fw with ENABLE\_UART\_RW\_BK\_REGS enabled) might support direct read/write commands for registers of the radio's DSP chip (BK4819). The exact command format for this would be specific to that firmware implementation.
* "CAT" Commands for QuanshengDock:  
  The QuanshengDock PC software, when used with compatible firmware (quansheng-dock-fw with ENABLE\_DOCK), likely uses a set of custom "CAT" (Computer Aided Transceiver) commands for more advanced remote control beyond simple memory programming. These could include:
  + Setting VFO frequency.
  + Setting modulation mode (AM/FM).
  + Setting squelch.
  + Initiating scan.
  + Reading S-meter values.  
    The exact list and format of these CAT commands would be defined by the QuanshengDock software and its companion firmware. Inspecting the source code of QuanshengDock (C#) or quansheng-dock-fw (C) would be necessary to determine these.

### 6. Example Command/Response Flow (Conceptual)

1. **PC wants to read 128 bytes from EEPROM address 0x0100:**
   * DATA = COMMAND\_BYTE\_READ\_EEPROM + ADDRESS\_BYTES(0x0100) + LENGTH\_BYTE(128) (This is a simplification; actual command bytes are not explicitly documented here).
   * PC calculates CRC16 of DATA.
   * PC forms PAYLOAD\_TO\_OBFUSCATE = DATA + CRC16.
   * PC XORs PAYLOAD\_TO\_OBFUSCATE to get OBFUSCATED\_PAYLOAD.
   * PC sends: 0xAB 0xCD <len(DATA+CRC16)> 0x00 <OBFUSCATED\_PAYLOAD> 0xDC 0xBA.
2. **Radio responds:**
   * Radio processes the command.
   * Radio prepares RESPONSE\_DATA (128 bytes from EEPROM).
   * Radio XORs RESPONSE\_DATA to get OBFUSCATED\_RESPONSE\_DATA.
   * Radio sends: 0xAB 0xCD <len(OBFUSCATED\_RESPONSE\_DATA)> 0x00 <OBFUSCATED\_RESPONSE\_DATA> 0xDC 0xBA.
3. **PC receives response:**
   * PC de-obfuscates OBFUSCATED\_RESPONSE\_DATA to get the raw 128 bytes of EEPROM data.

### 7. Further Resources and Tools

* **CHIRP Next (chirpmyradio.com):** The uvk5.py driver within CHIRP is a primary source for protocol details. (Search results 3.2, 3.3, 4.4)
* **k5prog by SQ5BPF (github.com/sq5bpf/k5prog):** An EEPROM and flash programmer. Its source code is a valuable reference. (Search result 4.2)
* **quansheng-dock-fw by nicsure (github.com/nicsure/quansheng-dock-fw):** Custom firmware for QuanshengDock compatibility. (Search result 2.1)
* **QuanshengDock by nicsure (github.com/nicsure/QuanshengDock):** PC control software. Its source code would detail its specific CAT commands. (Search result 2.2)
* Various radio forums (e.g., RadioReference) for community discussions.

This document provides a foundational understanding of the Quansheng UV-K5's communication protocol. For implementing specific commands, close examination of the CHIRP driver or k5prog source code is highly recommended.