NABU PC Network Adapter Protocol Specification

Created by DJ Sures (Sunday, January 11, 2026)

Source Basis: NABU PC BIOS ROM (4 KB variant)

Document Version: 1.0
Status: Reverse-engineered, unofficial
Scope: Protocol between NABU PC and the NABU Cable Network Adapter over the HCCA port.

---

1. Introduction

This document defines the message-level protocol used by the NABU PC BIOS ROM to communicate with the NABU Cable Network Adapter ("adapter") via the **HCCA** I/O port.

The information herein is derived exclusively from behavioral analysis of the ROM and therefore reflects **what the ROM expects**, not necessarily the original design intent of the adapter.

The protocol is **command/response oriented**, initiated by the NABU PC, with a small set of command codes and a consistent 2-byte handshake pattern.

---

2. Terminology and Roles

- **Host / PC**
  The NABU PC executing the BIOS ROM. It initiates all commands.
- **Adapter**
  The NABU Cable Network Adapter (modem) connected on port HCCA.
- **HCCA**
  8-bit I/O port used for all command and data transfers between PC and Adapter. (ROM symbol: R_HCCA = 0x80)
- **Handshake**
  The 2-byte sequence 0x10 0x06 sent by the adapter to acknowledge a received command.
- **Block Transfer**
  A framed data message sent from the adapter to the PC in response to a block request.

---

3. Transport & Framing

3.1 Physical Interface

- 8-bit data path via I/O port HCCA (OUT (0x80), A / IN A, (0x80) in ROM).
- All higher-level framing (commands, handshakes, block start/end, escaping) is implemented in software.

3.2 Command Handshake Sequence

For all ROM-initiated commands, the PC uses the same pattern:

1. **PC → Adapter**:
   Write the **command byte** CMD to HCCA.
2. **Adapter → PC**:
   Send 0x10
   PC expects to read 0x10 from HCCA and considers any other value an error.
3. **Adapter → PC**:
   Send 0x06
   PC expects to read 0x06 from HCCA.

If either of the handshake bytes does not match (0x10 then 0x06), the ROM treats it as a command-level failure.

---

4. Special Bytes and Escaping

The ROM treats certain bytes as protocol markers:

Value	Role
0x10	Escape / control introducer
0x06	Part of the command handshake
0xE4	Generic “OK/ACK” for various stages
0xE1	End-of-block marker (block transfer)
0x91	Start-of-block marker (block transfer)

4.1 Byte Stuffing Rules (Data Blocks)

Within **block payloads**:

- Literal data byte 0x10 is **byte-stuffed** as:
  0x10 0x10

The ROM keeps a state bit to distinguish:

- First 0x10 → escape indicator
- Second 0x10 → interpreted as literal data byte 0x10

Other bytes are transmitted as-is.

---

5. Command Summary

The ROM uses the following command codes:

Cmd	Direction	Name / Role
0x83	PC → Adapter	Presence Probe
0x82	PC → Adapter	Status / Configuration Query
0x85	PC → Adapter	Channel Provisioning
0x81	PC → Adapter	Session Setup / Mode Select
0x84	PC → Adapter	Block Transfer Request

Each of the following sections defines the behavior expected by the ROM.

---

6. Command 0x83 — Presence Probe

6.1 Purpose

Verify that the adapter is physically present and responsive.

6.2 Exchange

1. **PC → Adapter**
   0x83 (command byte)

2. **Adapter → PC**
   0x10 0x06 (standard handshake)
3. **Adapter → PC**
   0xE4 (presence OK)

6.3 PC Behavior

- If the handshake bytes are not 0x10 followed by 0x06 → error.
- If the final byte is not 0xE4 → adapter is considered faulty or absent.

6.4 Adapter Requirements

An adapter that wishes to be recognized must:

- Accept 0x83
- Respond with:
- 10 06 E4

in that order.

---

7. Command 0x82 — Status / Configuration Query

7.1 Purpose

Query adapter status, specifically whether a channel code must be entered.

7.2 Exchange (ROM-Observed Sequence)

1. **PC → Adapter**
   0x82 (command byte)
2. **Adapter → PC**
   0x10 0x06 (standard handshake)
3. **PC → Adapter**
   0x01 (status selector byte; constant in ROM)
4. **Adapter → PC**
   One status byte (S)
   (The ROM uses this in startup logic in the cold path; in the warm path, it also uses additional signaling through 0x10/0xE1 sequences, but the key status behavior is tied to a RAM flag derived from earlier status.)

5. **Adapter → PC**
   In some sequences observed from the ROM's warm-reset logic, the adapter also sends:
6. 0x10 0xE1

as a termination/ack pattern for this status transaction.

7.3 Status Bits (ROM Usage)

The ROM tests a "status flag" byte stored in RAM (at RAM_FFF8) and uses:

- **Bit 7 = 1** → Channel code is required.
  The ROM displays **"PLEASE TYPE IN CHANNEL CODE"** and enters the channel entry routine.
- **Bit 7 = 0** → No channel code required; proceed to adapter session setup.

The ROM does not obviously use the other bits; their meanings are undefined from ROM alone.

7.4 Adapter Requirements

- Support receiving the 0x82 command.
- Return a status byte whose **bit 7** indicates whether channel entry is required:
  - status & 0x80 != 0 → channel required
  - status & 0x80 == 0 → channel not required
- Maintain consistent status behavior across cold/warm resets.

---

8. Command 0x85 — Channel Provisioning

8.1 Purpose

Provide the adapter with a **channel identifier** derived from a 4-character user-entered "channel code".

The ROM:

1. Prompts: **"PLEASE TYPE IN CHANNEL CODE"**
2. Accepts 4 characters (digits and certain letters).
3. Converts them to four 4-bit nibbles.

4. Packs them into a 16-bit value.

8.2 Channel Word Encoding

User enters 4 valid hex-like characters: D_3 D_2 D_1 D_0 (each 0–15 after ROM's ASCII normalisation).

The ROM constructs a 16-bit channel value:

Channel = (D_0 << 12) | (D_1 << 8) | (D_2 << 4) | D_3

It then splits this into:

- CH_HI = high byte of Channel
- CH_LO = low byte of Channel

8.3 Exchange

1. **PC → Adapter**
   0x85 (command byte)
2. **Adapter → PC**
   0x10 0x06 (standard handshake)
3. **PC → Adapter**
   CH_HI
   CH_LO

(High then low byte of the 16-bit channel word.)

4. **Adapter → PC**
   0xE4 (acknowledgement / success)

8.4 Error Handling

- If handshake fails or the final 0xE4 is not received, the ROM treats this as an adapter failure and displays **"ADAPTOR FAILURE"** or enters a generic error path.

8.5 Adapter Requirements

- Accept channel word (2 bytes) after a valid handshake.
- Validate or store the channel as appropriate.
- Reply with 0xE4 if the channel is accepted and streaming can proceed.

---

9. Command 0x81 — Session Setup / Mode Select

9.1 Purpose

Enter a mode where the PC is prepared to download the OS / content image. This is a session-level "ready to download" setup.

9.2 Exchange

1. **PC → Adapter**
   0x81 (command byte)
2. **Adapter → PC**
   0x10 0x06 (standard handshake)
3. **PC → Adapter**
   0x8F
   (Meaning not decoded from ROM; treated as a fixed configuration value.)
4. **PC → Adapter**
   After a brief hardware/PSG gating delay, the PC sends:

0x05
(Again, meaning not decoded; constant value.)

5. **Adapter → PC**
   0xE4 (final session-setup acknowledgement)

9.3 PC Behavior

- On success, the ROM:
  - Stores a session flag.
  - Maps memory and sets up internal state for OS loading.
  - Displays **"PLEASE WAIT"**.
  - Proceeds to issue 0x84 block transfer requests.

9.4 Adapter Requirements

- Accept 0x81 and respond with handshake.
- Interpret the specific parameter bytes 0x8F and 0x05 according to its internal design.
- Confirm readiness to deliver block data by returning 0xE4.

---

10. Command 0x84 — Block Transfer Request

10.1 Purpose

Request a **data block** from the adapter. This is the core of the OS/content download process.

10.2 Block Request Header

The ROM forms a 4-byte header from variables at addresses:

- 0x1003
- 0x1002
- 0x1001
- 0x1000

These are sent as:

H3 = RAM[0x1003]

H2 = RAM[0x1002]

H1 = RAM[0x1001]

H0 = RAM[0x1000]

The ROM does not interpret them after sending; they function as a **block identifier/descriptor** for the adapter. Typical fields in this kind of scheme might include block index, offset, length, and flags, but the exact semantics are not encoded in the ROM.

10.3 Exchange

1. **PC → Adapter**
   0x84 (command byte)
2. **Adapter → PC**
   0x10 0x06 (standard handshake)
3. **PC → Adapter**
   H3 H2 H1 H0 (4-byte header; order high→low as shown)
4. **Adapter → PC**
   0xE4 (header accepted)

5. **Adapter → PC**
   0x91 (Start-of-block marker)
6. **PC → Adapter**
   0x10 (the ROM echoes this constant after seeing 0x91 — effectively a "go ahead" / sync confirmation)
7. **Adapter → PC**
   Byte-stuffed block payload, terminated by 0xE1:
   - Data bytes, with escape rule:
     - 0x10 in data → send 0x10 0x10
   - Final unescaped byte:
     - 0xE1 (end-of-block marker)

10.4 PC Block Processing

For each received byte (after 0x91 / 0x10 sync):

- If the byte is 0x10:
  - The ROM toggles an internal state bit:
    - First 0x10: treated as escape introducer.
    - Next byte is taken as literal data (even if it's 0x10 again).
- Otherwise:
  - The byte is stored into memory at the current target address.
  - A CRC/state accumulator is updated via a table (ROM routine at label_65D, table at 0x0B74).

On every byte, the ROM increments an internal byte counter and continues until:

- It reads a **non-escaped 0xE1**, which:
  - Marks end of block.
  - Causes it to:
    - Save the byte count.
    - Check CRC by verifying final values in alternate register set (DE' == 0x1D0F per ROM's comparison).

- Accept or reject the block accordingly.

If CRC or terminator is invalid, the ROM reenters the block request logic and retries with 0x84.

10.5 Error Handling

- If the adapter does **not** send 0x91 after the header, or the initial 0x91 is not seen:
  - The ROM aborts the block receive and restarts the transaction via 0x84.
- If the terminating byte is not 0xE1:
  - The ROM restarts via the 0x84 path.
- If the final CRC state does not match the expected internal constants:
  - The ROM restarts via 0x84.

10.6 Adapter Requirements

To be compatible with the ROM:

1. Accept 0x84 and respond with 10 06 handshake.
2. Accept the 4-byte header H3 H2 H1 H0 and return 0xE4 when ready.
3. Send a block as:
4. 0x91
5. <byte-stuffed data bytes>
6. 0xE1

where:

- All literal 0x10 bytes are encoded as 0x10 0x10.

7. Compute and send data so that the ROM's CRC routine results in the expected final state (as implemented in ROM at label_65D and its lookup table).
8. Restart blocks elegantly if the PC reissues 0x84 (indicating previous attempt was rejected).

---

11. ROM-Level Expectations Summary

An adapter implementation that wishes to satisfy the ROM must:

1. **Respond to Presence Probe (0x83)**
   With 10 06 E4.
2. **Implement Status (0x82)**
   - Provide a status byte with meaningful bit 7:
     - 1 = channel code required
     - 0 = channel code not required.
3. **Implement Channel Provisioning (0x85)**
   - Accept 2 bytes (channel word).
   - Reply 0xE4 if channel accepted.
4. **Implement Session Setup (0x81)**
   - Recognize and react to parameters 0x8F and 0x05.
   - Reply 0xE4 on success.
5. **Implement Block Transfer (0x84)**
   - Support 4-byte header semantics.
   - Reply 0xE4 after header.
   - Send blocks framed with 0x91 and 0xE1, with proper escaping and CRC.