# HART Protocol Reference — Conversation Log

Claude

July 17, 2026

## HART Protocol Reference — Conversation Log

A working reference on HART protocol frame formats, Command 0 and Command 3 byte layouts, unit codes, and a C decoder implementation.

## 1. Command 0 Request — Long Format

```
FF FF FF FF FF  82  00 00 00 00 00  00  00  82
```

| Field | Bytes | Value | Notes |
|---|---|---|---|
| Preamble | 5 | FF FF FF FF FF | Sync bytes; 5 is typical, spec allows 2–20 |
| Start delimiter | 1 | 82 | Long frame, master-to-slave, primary master |
| Long address | 5 | 00 00 00 00 00 | All zero — device unknown until it responds |
| Command | 1 | 00 | Command 0 |
| Byte count | 1 | 00 | No data bytes in the request |
| Checksum | 1 | 82 | XOR of delimiter + address + command + byte count |

The checksum is the XOR of every byte from the start delimiter through the byte count (preamble excluded). Command 0 is used precisely because the master doesn't yet know the device's unique address, so the long address field is sent as all zeros. The slave's response frame uses start delimiter 86 and echoes back its actual long address.

## 2. Command 0 Response — Full Long Format Frame

Worked example (HART 7 device):

```
FF FF FF FF FF  86  8C 45 03 A1 B2  00  18  00  00
FE 4A 96 05 07 07 03 20 00 12 34 56 05 08 00 2A 00
00 4A 3F 00 00 00 05  [CS]
```

| Byte # | Field | Example Value | Description |
|---|---|---|---|
| — | Preamble | FF FF FF FF FF | 5 sync bytes (2–20 allowed) |
| 0 | Start delimiter | 86 | Long frame, slave-to-master |
| 1 | Address byte 1 | 8C | Bit 7 = burst mode flag; bits 6–0 = device type (upper) |
| 2 | Address byte 2 | 45 | Device type (lower bits) |
| 3 | Address byte 3 | 03 | Device ID (MSB) |
| 4 | Address byte 4 | A1 | Device ID |
| 5 | Address byte 5 | B2 | Device ID (LSB) |
| 6 | Command | 00 | Echoes Command 0 |
| 7 | Byte count | 18 (24) | Response code + status + 22 data bytes |
| 8 | Response code | 00 | 00 = success |
| 9 | Device status | 00 | Field device status bits |
| 10 | Expansion code | FE | Fixed marker for legacy codes below |
| 11 | Manufacturer ID code | 4A | Short-form manufacturer ID |
| 12 | Device type code | 96 | Manufacturer-assigned device/model code |
| 13 | Min. preambles (device req.) | 05 | Preambles this device needs incoming |
| 14 | Universal command revision | 07 | HART universal command set revision |
| 15 | Transmitter-specific rev. | 07 | Device-specific command set revision |
| 16 | Software revision | 03 | Device firmware revision |
| 17 | Hardware rev. & signaling code | 20 | Bits 7–3 = HW rev; bits 2–0 = signaling code |
| 18 | Device flags | 00 | Bit 0 = multi-sensor flag |
| 19 | Device ID (MSB) | 12 | Unique device ID, byte 1 of 3 |

| Byte # | Field | Example Value | Description |
|---|---|---|---|
| 20 | Device ID | `34` | Unique device ID, byte 2 of 3 |
| 21 | Device ID (LSB) | `56` | Unique device ID, byte 3 of 3 |
| 22 | Min. preambles (host req.) | `05` | Preambles the host should send |
| 23 | Max. device variable count | `08` | Number of device variables supported |
| 24 | Config change counter (MSB) | `00` | High byte of change counter |
| 25 | Config change counter (LSB) | `2A` | Low byte of change counter |
| 26 | Extended field device status | `00` | Additional status bits (HART 6+) |
| 27 | Extended manufacturer ID (MSB) | `00` | High byte of full 16-bit manufacturer ID |
| 28 | Extended manufacturer ID (LSB) | `4A` | Low byte of extended manufacturer ID |
| 29 | Private-label distributor (MSB) | `3F` | High byte of distributor code |
| 30 | Private-label distributor (LSB) | `00` | Low byte of distributor code |
| 31 | Device profile | `05` | Device profile/category (HART 7) |
| 32 | Checksum | `[CS]` | XOR of bytes 0 through 31 |

Total frame length: 5 (preamble) + 1 (delimiter) + 5 (address) + 1 (command) + 1 (byte count) + 24 (response code + status + data) + 1 (checksum) = **38 bytes**.

## 3. Command 0 Data Payload — Standalone Byte Table

| Byte | Field Name | Size | Description |
|---|---|---|---|
| 0 | Expansion Code | 1 | Fixed FE; signals legacy 1-byte mfr/type codes follow |
| 1 | Manufacturer ID Code | 1 | Short-form manufacturer identifier |
| 2 | Device Type Code | 1 | Manufacturer-assigned model/type code |
| 3 | Number of Preambles | 1 | Min. preambles device requires (device→host) |
| 4 | Universal Command Revision | 1 | HART universal command set revision |
| 5 | Transmitter-Specific Command Revision | 1 | Device-specific command set revision |
| 6 | Software Revision | 1 | Device firmware/software revision |
| 7 | Hardware Revision & Signaling Code | 1 | Bits 7–3 HW rev; bits 2–0 signaling code |
| 8 | Device Flags | 1 | Bit 0 = multi-sensor flag |
| 9 | Device ID (MSB) | 1 | Unique device identifier, byte 1 of 3 |
| 10 | Device ID | 1 | Unique device identifier, byte 2 of 3 |
| 11 | Device ID (LSB) | 1 | Unique device identifier, byte 3 of 3 |
| 12 | Min. Preambles (Host Requirement) | 1 | Preambles host should send to device |
| 13 | Max. Number of Device Variables | 1 | Count of supported device variables (HART 6+) |
| 14 | Config Change Counter (MSB) | 1 | High byte of change counter |
| 15 | Config Change Counter (LSB) | 1 | Low byte of change counter |
| 16 | Extended Field Device Status | 1 | Additional status bits (HART 6+) |
| 17 | Manufacturer ID Code Ext. (MSB) | 1 | High byte of full 16-bit manufacturer ID |
| 18 | Manufacturer ID Code Ext. (LSB) | 1 | Low byte of extended manufacturer ID |
| 19 | Private-Label Distributor (MSB) | 1 | High byte of distributor code |

| Byte | Field Name | Size | Description |
|---|---|---|---|
| 20 | Private-Label Distributor (LSB) | 1 | Low byte of distributor code |
| 21 | Device Profile | 1 | Device profile/category (HART 7) |

Notes: HART 5 devices return only bytes 0–13; HART 6 extends through ~byte 16; HART 7 returns the full 22 bytes. Bytes 9–11 (Device ID) combine with manufacturer ID and device type to form the device’s permanent 5-byte long address.

## 4. Command 0 Request/Response — Short Format

**Request (master → slave):**

```
FF FF FF FF FF  02  00  00  00  02
```

| Byte # | Field | Value | Meaning |
|---|---|---|---|
| — | Preamble | FF FF FF FF FF | Sync bytes |
| 0 | Start delimiter | 02 | Short frame, master-to-slave |
| 1 | Polling address | 00 | Bit 7=0 primary master; bits 5–0 = address |
| 2 | Command | 00 | Command 0 |
| 3 | Byte count | 00 | No request data |
| 4 | Checksum | 02 | XOR of delimiter + address + command + byte count |

**Response (slave → master):**

```
FF FF FF FF FF  06  00  00  18  00  00  [22 data bytes]  [checksum]
```

| Byte # | Field | Value | Meaning |
|---|---|---|---|
| 0 | Start delimiter | 06 | Short frame, slave-to-master |
| 1 | Polling address | 00 | Echoes request polling address |
| 2 | Command | 00 | Echoes Command 0 |
| 3 | Byte count | 18 (24) | Response code + status + data |
| 4 | Response code | 00 | Success |
| 5 | Device status | 00 | Field device status bits |
| 6–27 | Command 0 data payload | — | Same structure as Section 3 |
| last | Checksum | — | XOR from start delimiter through end of data |

| Aspect | Long frame | Short frame |
|---|---|---|
| Start delimiter (req) | 82 | 02 |
| Start delimiter (resp) | 86 | 06 |
| Address field size | 5 bytes (unique ID) | 1 byte (polling address) |
| When used | Device unknown / initial discovery | Device already commissioned |

## 5. Command 3 Data Payload — Standalone Byte Table

Command 3: “Read Dynamic Variables and PV Current.” 24-byte data payload.

| Byte | Field | Size | Description |
|---|---|---|---|
| 0–3 | PV Current | 4 | IEEE-754 float, loop current in mA |
| 4 | PV Units Code | 1 | Units code for Primary Variable |
| 5–8 | Primary Variable (PV) | 4 | IEEE-754 float |
| 9 | SV Units Code | 1 | Units code for Secondary Variable |
| 10–13 | Secondary Variable (SV) | 4 | IEEE-754 float |
| 14 | TV Units Code | 1 | Units code for Tertiary Variable |
| 15–18 | Tertiary Variable (TV) | 4 | IEEE-754 float |
| 19 | QV Units Code | 1 | Units code for Quaternary Variable |
| 20–23 | Quaternary Variable (QV) | 4 | IEEE-754 float |

All numeric values are big-endian IEEE-754 32-bit floats. PV Current is always in mA; PV/SV/TV/QV are in whatever units their preceding code specifies. Devices with fewer than 4 active variables pad unused slots with units code 250 (“not used”).

## 6. Command 3 Response — Full Long Format Frame

```
FF FF FF FF FF  86  8C 45 03 A1 B2  03  1A  00  00
41 20 00 00  27  42 48 00 00  20  41 CC 00 00  39
42 C8 00 00  FA  00 00 00 00  [CS]
```

| Byte # | Field | Size | Example Value | Meaning |
|---|---|---|---|---|
| — | Preamble | 5 | FF FF FF FF FF | Sync bytes |
| 0 | Start delimiter | 1 | 86 | Long frame, slave-to-master |
| 1–5 | Address | 5 | 8C 45 03 A1 B2 | Unique long address |
| 6 | Command | 1 | 03 | Echoes Command 3 |
| 7 | Byte count | 1 | 1A (26) | Response code + status + 24 data bytes |
| 8 | Response code | 1 | 00 | Success |
| 9 | Device status | 1 | 00 | Field device status bits |
| 10–13 | PV Current | 4 | 41 20 00 00 | Float = 10.0 mA |
| 14 | PV Units Code | 1 | 27 | Units code for PV |
| 15–18 | Primary Variable (PV) | 4 | 42 48 00 00 | Float = 50.0 |
| 19 | SV Units Code | 1 | 20 | Units code for SV (°C) |
| 20–23 | Secondary Variable (SV) | 4 | 41 CC 00 00 | Float = 25.5 |
| 24 | TV Units Code | 1 | 39 | Units code for TV (mA) |
| 25–28 | Tertiary Variable (TV) | 4 | 42 C8 00 00 | Float = 100.0 |
| 29 | QV Units Code | 1 | FA (250) | Not used |
| 30–33 | Quaternary Variable (QV) | 4 | 00 00 00 00 | Float = 0.0 (unused) |
| 34 | Checksum | 1 | [CS] | XOR from byte 0 through byte 33 |

Total frame length: 5 (preamble) + 1 (delimiter) + 5 (address) + 1 (command) + 1 (byte count) + 26 (response code + status + 24 data bytes) + 1 (checksum) = **40 bytes**.

## 7. Decoded Example — Real Command 3 Response

Input byte string:

```
FFFF FFFF FF86 11A3 F82D 1403 1A00 4041
254B C73B 40B1 44BC 2041 B399 9A24 42AA
CCCD AA00 0000
```

| Byte(s) | Hex | Field | Value / Meaning |
|---|---|---|---|
| 0–4 | FF FF FF FF FF | Preamble | 5 sync bytes |
| 5 | 86 | Start delimiter | Long frame, slave → master |
| 6–10 | 11 A3 F8 2D 14 | Address | Unique long address |
| 11 | 03 | Command | Command 3 |
| 12 | 1A | Byte count | 26 — response + status + 24 data bytes |
| 13 | 00 | Response code | Success |
| 14 | 40 | Device status | Bit 6 set (Non-PV Out of Limits) |
| 15–18 | 41 25 4B C7 | PV Current | 10.331 mA |
| 19 | 3B | PV Units Code | 59 decimal (unverified unit) |
| 20–23 | 40 B1 44 BC | Primary Variable (PV) | 5.5396 |
| 24 | 20 | SV Units Code | 32 decimal = degC |
| 25–28 | 41 B3 99 9A | Secondary Variable (SV) | 22.450 degC |
| 29 | 24 | TV Units Code | 36 decimal = mV |
| 30–33 | 42 AA CC CD | Tertiary Variable (TV) | 85.400 mV |
| 34 | AA | QV Units Code | 170 decimal — Unit Code Expansion range |
| 35–37 | 00 00 00 | Quaternary Variable | Incomplete — 3 of 4 bytes present |
| — | *missing* | Checksum | Not present in string as given |

Note: this particular input string was 2 bytes short of a complete frame (missing the final QV byte and the checksum).

## 8. HART Unit Codes — Verified Subset

| Code (dec) | Code (hex) | Unit |
|---|---|---|
| 32 | 0x20 | degC |
| 33 | 0x21 | degF |
| 34 | 0x22 | degR |
| 35 | 0x23 | K |
| 36 | 0x24 | mV |
| 37 | 0x25 | Ohm |
| 38 | 0x26 | Hz |
| 39 | 0x27 | mA |
| 57 | 0x39 | % |
| 250 | 0xFA | Not Used |

Codes 170–219 fall in the **Unit Code Expansion range**: their meaning depends on the device's declared Device Variable Classification and cannot be resolved generically. Code 59 (seen in the worked example above) could not be confirmed against public reference documentation.

## 9. C Implementation — Command 3 Decoder

```c
/*
 * hart_cmd3_decode.c
 *
 * Decodes a HART long-frame Command 3 ("Read Dynamic Variables and PV
 * Current") response.
 *
 * Frame layout assumed (long frame, slave -> master):
 *   Preamble        : N x 0xFF          (not passed to the parser)
 *   Start delimiter : 1 byte  (0x86 for long frame, slave->master)
 *   Address         : 5 bytes
 *   Command         : 1 byte  (0x03)
 *   Byte count      : 1 byte
 *   Response code   : 1 byte
 *   Device status   : 1 byte
 *   Data (24 bytes) : PV current(4) + PV units(1) + PV(4)
 *                     + SV units(1) + SV(4)
 *                     + TV units(1) + TV(4)
 *                     + QV units(1) + QV(4)
 *   Checksum        : 1 byte
 *
 * Build:   gcc -o hart_cmd3_decode hart_cmd3_decode.c
 * Run:     ./hart_cmd3_decode
 */

#include <stdio.h>
#include <stdint.h>
#include <string.h>

/* ------------------------------------------------------------------ */
/* Response code table (HART Universal Command response codes)         */
/* ------------------------------------------------------------------ */
typedef struct {
    uint8_t     code;
    const char *meaning;
} rc_entry_t;

static const rc_entry_t response_codes[] = {
    { 0,  "Success" },
    { 2,  "Invalid selection" },
    { 3,  "Passed parameter too large" },
    { 4,  "Passed parameter too small" },
    { 5,  "Too few data bytes received" },
    { 6,  "Device-specific command error" },
    { 7,  "In write-protect mode" },
    { 8,  "Update failure" },
    { 9,  "Application, device, or communication layer busy" },
    { 15, "Access restricted" },
    { 16, "Device is in a fixed alarm/saturation condition" },
    { 32, "Device is busy" },
    { 34, "Command not implemented" },
};
```

```c
static const char *lookup_response_code(uint8_t rc)
{
    size_t n = sizeof(response_codes) / sizeof(response_codes[0]);
    for (size_t i = 0; i < n; i++) {
        if (response_codes[i].code == rc)
            return response_codes[i].meaning;
    }
    if (rc & 0x80)
        return "Vendor/command-specific error (bit 7 set)";
    return "Unrecognized / device-specific response code";
}

/* ------------------------------------------------------------------ */
/* Device status bit meanings                                          */
/* ------------------------------------------------------------------ */
static void print_device_status(uint8_t status)
{
    printf("Device Status byte: 0x%02X\n", status);
    if (status & 0x01) printf("  [bit0] Field Device Malfunction\n");
    if (status & 0x02) printf("  [bit1] Configuration Changed\n");
    if (status & 0x04) printf("  [bit2] Cold Start\n");
    if (status & 0x08) printf("  [bit3] More Status Available\n");
    if (status & 0x10) printf("  [bit4] Loop Current Fixed\n");
    if (status & 0x20) printf("  [bit5] Loop Current Saturated\n");
    if (status & 0x40) printf("  [bit6] Non-Primary-Variable Out of Limits\n");
    if (status & 0x80) printf("  [bit7] Primary Variable Out of Limits\n");
    if (status == 0)    printf("  (no status bits set)\n");
}

/* ------------------------------------------------------------------ */
/* Minimal unit-code lookup (only the codes verified in this convo)    */
/* ------------------------------------------------------------------ */
static const char *lookup_unit_code(uint8_t code)
{
    switch (code) {
        case 32:  return "degC";
        case 33:  return "degF";
        case 34:  return "degR";
        case 35:  return "K";
        case 36:  return "mV";
        case 37:  return "Ohm";
        case 38:  return "Hz";
        case 39:  return "mA";
        case 57:  return "%";
        case 250: return "Not Used";
        default:
            if (code >= 170 && code <= 219)
                return "(Unit Code Expansion range - meaning depends on Device Variable Cl
            return "(unverified / not in local table)";
    }
}

/* ------------------------------------------------------------------ */
```

```c
/* Big-endian 4-byte -> float, without assuming host byte order        */
/* ------------------------------------------------------------------ */
static float be_bytes_to_float(const uint8_t *p)
{
    uint32_t bits = ((uint32_t)p[0] << 24) |
                    ((uint32_t)p[1] << 16) |
                    ((uint32_t)p[2] << 8)  |
                    ((uint32_t)p[3]);
    float f;
    memcpy(&f, &bits, sizeof(f));
    return f;
}

/* ------------------------------------------------------------------ */
/* Parsed representation of the Command 3 response                     */
/* ------------------------------------------------------------------ */
typedef struct {
    uint8_t start_delimiter;
    uint8_t address[5];
    uint8_t command;
    uint8_t byte_count;
    uint8_t response_code;
    uint8_t device_status;

    float   pv_current;
    uint8_t pv_units;
    float   pv;

    uint8_t sv_units;
    float   sv;

    uint8_t tv_units;
    float   tv;

    uint8_t qv_units;
    float   qv;

    uint8_t checksum;
    int     ok;
} hart_cmd3_response_t;

static int parse_cmd3_response(const uint8_t *buf, size_t len,
                               hart_cmd3_response_t *out)
{
    memset(out, 0, sizeof(*out));

    const size_t MIN_LEN = 1 + 5 + 1 + 1 + 1 + 1 + 24 + 1; /* 35 bytes */
    if (len < MIN_LEN) {
        out->ok = 0;
        return 0;
    }

    size_t i = 0;
```

```
    out->start_delimiter = buf[i++];
    memcpy(out->address, &buf[i], 5); i += 5;
    out->command       = buf[i++];
    out->byte_count     = buf[i++];
    out->response_code  = buf[i++];
    out->device_status   = buf[i++];

    out->pv_current = be_bytes_to_float(&buf[i]); i += 4;
    out->pv_units    = buf[i++];
    out->pv          = be_bytes_to_float(&buf[i]); i += 4;

    out->sv_units    = buf[i++];
    out->sv          = be_bytes_to_float(&buf[i]); i += 4;

    out->tv_units    = buf[i++];
    out->tv          = be_bytes_to_float(&buf[i]); i += 4;

    out->qv_units    = buf[i++];
    out->qv          = be_bytes_to_float(&buf[i]); i += 4;

    out->checksum = buf[i++];
    out->ok = 1;
    return 1;
}

static void print_cmd3_response(const hart_cmd3_response_t *r)
{
    if (!r->ok) {
        printf("ERROR: buffer too short for a complete Command 3 response.\n");
        return;
    }

    printf("Start delimiter : 0x%02X\n", r->start_delimiter);
    printf("Address         : %02X %02X %02X %02X %02X\n",
           r->address[0], r->address[1], r->address[2],
           r->address[3], r->address[4]);
    printf("Command         : %u\n", r->command);
    printf("Byte count      : %u\n", r->byte_count);
    printf("Response code   : %u (0x%02X) - %s\n",
           r->response_code, r->response_code,
           lookup_response_code(r->response_code));
    print_device_status(r->device_status);

    printf("\nPV Current : %.4f mA\n", r->pv_current);
    printf("PV         : %.4f  [unit code %u = %s]\n",
           r->pv, r->pv_units, lookup_unit_code(r->pv_units));
    printf("SV         : %.4f  [unit code %u = %s]\n",
           r->sv, r->sv_units, lookup_unit_code(r->sv_units));
    printf("TV         : %.4f  [unit code %u = %s]\n",
           r->tv, r->tv_units, lookup_unit_code(r->tv_units));
    printf("QV         : %.4f  [unit code %u = %s]\n",
           r->qv, r->qv_units, lookup_unit_code(r->qv_units));
    printf("Checksum   : 0x%02X\n", r->checksum);
```

```c
}

/* ------------------------------------------------------------------ */
/* Optional: compute / verify the XOR checksum                        */
/* ------------------------------------------------------------------ */
static uint8_t compute_checksum(const uint8_t *buf, size_t len_without_checksum)
{
    uint8_t cs = 0;
    for (size_t i = 0; i < len_without_checksum; i++)
        cs ^= buf[i];
    return cs;
}

int main(void)
{
    uint8_t frame[] = {
        0x86,
        0x11, 0xA3, 0xF8, 0x2D, 0x14,
        0x03,
        0x1A,
        0x00,
        0x40,
        0x41, 0x25, 0x4B, 0xC7,
        0x3B,
        0x40, 0xB1, 0x44, 0xBC,
        0x20,
        0x41, 0xB3, 0x99, 0x9A,
        0x24,
        0x42, 0xAA, 0xCC, 0xCD,
        0xAA,
        0x00, 0x00, 0x00, 0x00,
        0x00
    };

    size_t len = sizeof(frame);
    frame[len - 1] = compute_checksum(frame, len - 1);

    hart_cmd3_response_t resp;
    parse_cmd3_response(frame, len, &resp);
    print_cmd3_response(&resp);

    return 0;
}
```

**Program Output**

```
Start delimiter : 0x86
Address         : 11 A3 F8 2D 14
Command         : 3
Byte count      : 26
Response code   : 0 (0x00) - Success
Device Status byte: 0x40
  [bit6] Non-Primary-Variable Out of Limits
```

```
PV Current : 10.3310 mA
PV         : 5.5396  [unit code 59 = (unverified / not in local table)]
SV         : 22.4500  [unit code 32 = degC]
TV         : 85.4000  [unit code 36 = mV]
QV              :  0.0000   [unit  code  170  =  (Unit  Code  Expansion  range  -
meaning depends on Device Variable Classification)]
Checksum   : 0xC0
```