KitProg host protocol interface

specification

About this document

Scope and purpose

This specification defines the KitProg host protocol interface (KHPI) for communicating with the KitProg controller firmware starting from KitProg3 v1.00. KitProg1 and KitProg2 do not support this protocol.

Abbreviations and definitions

Abbreviations

Abbreviation	Definition
HID	human interface device
I2C	inter-integrated circuit
KHPI	KitProg host protocol interface
SPI	serial peripheral interface
USB	universal serial bus

Table of contents

About this document **1**
Table of contents **2**
1 Introduction **4**
1.1 Version 4
1.2 Hardware Interface 4
2 Protocol Description **5**
2.1 Command Status (CMD_STAT) 5
3 KitProg Commands **7**
3.1 System Configuration Subsystem (0x80-0x82) 7
3.1.1 Get Version (0x80) 7
3.1.2 Reset FW (0x81) 7
3.1.3 Mode Switch (0x82) 8
3.2 User Interface Subsystem (0x83) 8
3.2.1 LED Control (0x83) 8
3.3 Power Control Subsystem (0x84) 9
3.3.1 Set Power (0x84) 9
3.3.2 Get Power (0x84) 9
3.4 Custom SWD Requests (0x85) 10
3.4.1 DAP Acquire (0x85) 10
3.5 I2C/SPI Bridge (0x86-0x89) 11
3.5.1 Set Interface Speed (0x86) 11
3.5.2 Get Interface Speed (0x86) 12
3.5.3 Restart I2C Master (0x87) 12
3.5.4 I2C Write Transaction with Start / Restart (0x88) 13
3.5.5 I2C Read Transaction with Start / Restart (0x88) 14
3.5.6 I2C Write Transaction Continuation (No Start or Restart) (0x88) 15
3.5.7 I2C Read Transaction Continuation (No Start or Restart) (0x88) 16
3.5.8 SPI Data Transfer (0x89) 16
3.6 GPIO Bridge (0x8A – 0x8D) 17
3.6.1 Set GPIO Pin Drive Mode (0x8A) 17
3.6.2 Set GPIO Pin State (0x8B) 17
3.6.3 Read GPIO Pin State (0x8C) 18
3.6.4 Read GPIO Pin State Change (0x8D) 18
3.7 Info Command (0x90) 19
3.7.1 Probe Info/Capabilities (0x90) 19
3.8 Set Acquire Parameters (0x91) 20
3.8.1 Set Acquire Timeout (0x91) 20
3.8.2 Select DAP Handshake Type During DAP Acquire (0x91) 21
3.8.3 Set DAP AP (0x91) 21
3.9 Read Unique ID Record (0x92) 22
3.10 Get/Set KitProg3 UART Flow Control and UART mode (0x93) 26
3.11 Enable/Disable KitProg3 Bridge Interfaces (0x94) 27
3.12 Read Extended Target Info (0x95) 27
4 DAPLink Commands **29**
4.1 Mode Switch (0xA0) 29
4.2 Get KitProg Info (0xA1) 29
4.3 Reset DAPLink Device (0xA2) 30
Revision history **31**

Table of contents

Disclaimer.....33

1 Introduction

This specification defines the packet structures, and other logical data structures for the host-to-KitProg communication for non-standard CMSIS-DAP operations. The list and description of the available standard commands are available in CMSIS-DAP's official web page (https://arm-software.github.io/CMSIS_5/DAP/html/group__DAP__Commands__gr.html).

1.1 Version

This specification defines the packet structures for KitProg Host Protocol Interface with the following version:

Version Number	Value
Major Version	2
Minor Version	08

1.2 Hardware Interface

This protocol is created for USB endpoints of 64-byte long. It should be revised if used with USB endpoints of a different size.

The endpoint type (HID vs. Bulk) does not matter; this protocol does not have limitations of the CMSIS-DAP endpoint type.

Starting from version 2.00 of KHPI (used in KitProg3 starting from v.2.10), Bulk endpoints are used for the bridge interface.

2 Protocol Description

The communication protocol implies that some commands are available via CMSIS-DAP endpoints and in the same manner and format via other endpoints/interfaces.

This is implementation-specific, but commands can be available through interfaces and endpoints other than those dedicated to CMSIS-DAP. KitProg uses command IDs starting from 0x80 to comply with CMSIS-DAP firmware/interface definition. Commands 0x8E-0x8F and 0x96-0x9F are available for future expendability.

Command	CMSIS-DAP Interface	Bridge Interface
Standard CMSIS-DAP (0x00-0x09)	Yes	No
Reset Target (0x0A)	Yes	Yes [1]
Standard CMSIS-DAP (0x0B-0x7E)	Yes	No
Execute Commands (0x7F)	Yes	Yes [2]
System Configuration Subsystem (0x80-0x82)	Yes	Yes
User Interface Subsystem (0x83)	Yes	No
Power Control Subsystem (0x84)	Yes	Yes
DAP Acquire (0x85)	Yes	No
I2C/SPI Bridge (0x86-0x89)	No	Yes
GPIO Bridge (0x8A – 0x8D)	Yes	Yes
Probe Info/Capabilities (0x90)	Yes	Yes
Set Acquire Parameters (0x91)	Yes	No
Read Unique ID Record (0x92)	Yes	No
Get/Set KitProg3 UART Flow Control and UART mode (0x93)	Yes	Yes
Enable/Disable KitProg3 Bridge Interfaces (0x94)	Yes	No
Read Extended Target Info (0x95)	Yes	No

For KitProg3 in DAPLink mode, vendor-specific command IDs start from 0xA0 to comply with the CMSIS-DAP interface definition of generic DAPLink.

If not stated otherwise, a response packet should be treated as the immediate (delay<10ms) response.

2.1 Command Status (CMD_STAT)

This section does not describe the response that is sent to non-existent command calls; it is 0xFF per CMSIS-DAP documentation. It is returned in Byte 0. The remainder of the packet (if any) does not matter.

CMD_STAT is command status that is included in each command response.

CMD_STAT	
0000 0000 (0x00)	SUCCESS. Command executed successfully.
0000 0001 (0x01)	WAIT. Command accepted; execution is in progress. The host must poll the response until SUCCESS or timeout. All bytes that follow (if any) must be ignored by the host.
1000 0001 (0x81)	FAIL (INVALID_PARAMS). Invalid parameters.

[1] Starting from version 2.03 of KHPI, Reset Target command is available on the Bridge interface.

[2] Starting from version 2.00 of KHPI, Execute multiple commands from a single packet is available on the Bridge interface.

1000 0010 (0x82)	FAIL (OPERATION_FAIL). Operation fails.

3 KitProg Commands

This section describes the commands that are available in KitProg in either BULK or HID interface modes.

3.1 System Configuration Subsystem (0x80-0x82)

3.1.1 Get Version (0x80)

Used to get FW version + HW ID.

COMMAND (OUT Packet)	
Byte 0	1000 0000 (0x80)
RESPONSE (IN Packet)	
Byte 0	1000 0000 (0x80)
Byte 1	CMD_STAT
Byte 2	Major version of KitProg FW LSB
Byte 3	Major version of KitProg FW MSB
Byte 4	Minor version of KitProg FW LSB
Byte 5	Minor version of KitProg FW MSB
Byte 6	Hardware ID LSB
Byte 7	Hardware ID MSB
Byte 8	Major version of the KHPI
Byte 9	Minor version of the KHPI
Byte 10	Firmware build number LSB
Byte 11	Firmware build number MSB

3.1.2 Reset FW (0x81)

Issues a SW reset for the KitProg. There is no IN packet from the KitProg. This command includes USB re-enumeration.

COMMAND (OUT Packet)	
Byte 0	1000 0001 (0x81)
RESPONSE (IN Packet)	
None	

3.1.3 Mode Switch (0x82)

When KitProg receives this command, it enters the USB bootloader mode or mode switch immediately. It includes USB re-enumeration. There is no IN packet from the KitProg.

A response is issued only if invalid parameters (Byte 1) are received.

COMMAND (OUT Packet)	
Byte 0	1000 0010 (0x82)
Byte1	Desired mode: • 0x00 – Bootloader • 0x01 – CMSIS-DAP v.2.xx (USB Bulk) • 0x02 – CMSIS-DAP v.1.xx (USB HID) • 0x03 – Arm® Mbed® DAPLink • 0x04 – CMSIS-DAP v.2.xx Double UART (USB Bulk)
RESPONSE (IN Packet)	
Byte 0	1000 0010 (0x82)
Byte 1	CMD_STAT

3.2 User Interface Subsystem (0x83)

3.2.1 LED Control (0x83)

Used to change the LED state.

COMMAND (OUT Packet)	
Byte 0	1000 0011 (0x83)
Byte 1	UI state: • 0x00 – LED_STATE_READY • 0x01 – LED_STATE_PROGRAMMING • 0x02 – LED_STATE_SUCCESS • 0x03 – LED_STATE_ERROR
RESPONSE (IN Packet)	
Byte 0	1000 0011 (0x83)
Byte 1	CMD_STAT

The board that implements KHPI has its own default state(s) of indication (UI) for each mode that it supports. These default modes correspond to the READY state; it is supposed to be set after board startup. The SUCCESS state is intended to indicate that the previous SWD transactions have ended with no errors.

3.3 Power Control Subsystem (0x84)

3.3.1 Set Power (0x84)

Sets the device power supply.

COMMAND (OUT Packet)	
Byte 0	1000 0100 (0x84)
Byte 1	0x10 (set power)
Byte 2	Device power mode: • 0x00 = Power Off voltage regulator. • 0x01 = Power On voltage regulator. • 0x02 = Power On with voltage for digital potentiometer. In this case, bytes 3-4 define the voltage.
Byte 3	Voltage LSB
Byte 4	Voltage MSB
RESPONSE (IN Packet)	
Byte 0	1000 0100 (0x84)
Byte 1	CMD_STAT
Byte 2	This byte is defined only if CMD_STAT is returned as fail. 0x80 – Failed to set the desired voltage. 0xFF – Kit doesn't have an onboard digital potentiometer.

3.3.2 Get Power (0x84)

Returns the current device power settings/voltage.

COMMAND (OUT Packet)	
Byte 0	1000 0100 (0x84)
Byte 1	0x11 (Get Power)
RESPONSE (IN Packet)	
Byte 0	1000 0100 (0x84)
Byte 1	CMD_STAT
Byte 2	Device power supply. 0x00 = External (on-board regulator off). None Zero = Powered by KitProg3.
Byte 3	LSB of Vtarg voltage in the unit of mV.
Byte 4	MSB of Vtarg voltage in the unit of mV.
Byte 5	LSB of the digital potentiometer voltage, requested by the host.
Byte 6	MSB of the digital potentiometer voltage, requested by the host.
Byte 7	Indicates the status of the digital potentiometer for this HW: • 0 – Not available • 1 –Available on this Kit

3.4 Custom SWD Requests (0x85)

3.4.1 DAP Acquire (0x85)

Acquires a specified target chip. Acquire is allowed only through the SWD interface; otherwise, byte 0 in the response will contain 0xFF. Power is automatically applied to the target device if using Power Cycle Acquire mode and device is not powered externally. In this mode, the target must not be externally powered; if not, the attempt to acquire will fail. If using custom acquisition, Byte 4 should be defined with exact number of commands in the acquire sequence. When auto target detection is used, the target family will be retrieved from the DAP ROM table.

<table>
<tr><th colspan="2">COMMAND (OUT Packet)</th></tr>
<tr><td>Byte 0</td><td>1000 0101 (0x85)</td></tr>
<tr><td>Byte 1</td><td>Defines Target:
0x00 – PSOC™ 4
0x01 – PSOC™ 5LP
0x02 – PSOC™ 6 BLE
0x03 – T2G and XMC7000
0x04 – AIROC™
0x05 – PSOC™ 3
0xFE – Custom acquisition sequence
0xFF – Target auto detection</td></tr>
<tr><td>Byte 2</td><td>Acquire Mode:
0x00 – Reset
0x01 – Power Cycle</td></tr>
<tr><td>Byte 3</td><td>Number of acquire attempts (until succeeds)</td></tr>
<tr><td>Byte 4</td><td>Only for custom acquisition sequence: number of commands in custom sequence</td></tr>
<tr><td>Byte 5 -63</td><td>Custom acquisition sequence – sequence of write (5 bytes) or read (1 byte) operations to instruction/data registers. Acquire sequences for devices should be taken from device programming specifications. For information on SWD Transfer Requests see ARM Debug Interface v5 Architecture Specification.
Write operation:
<table>
<tr><th>Byte</th><th>Field</th><th>Example</th></tr>
<tr><td>Byte 0</td><td>SWD Transfer Request Data (LSb bit sent first)</td><td>0x81 (ABORT Write)</td></tr>
<tr><td>Byte 1-4</td><td>Address of register to write to (LSB sent first)</td><td>0x0000001E
(ORUNERRCLR=1, WDERRCLR=1, STKERRCLR=1, STKCMPCLR=1, DAPABORT=0)</td></tr>
</table>
Read operation:
<table>
<tr><th>Byte</th><th>Field</th><th>Example</th></tr>
<tr><td>Byte 0</td><td>SWD Transfer Request Data (LSb bit sent first)</td><td>0xA5 (IDCODE Read)</td></tr>
</table>
In case of error occurring during execution, remaining steps will be skipped, swd line reset and clearing of errors and sticky flags in abort register will be done.</td></tr>
</table>

RESPONSE (IN Packet)	
Byte 0	1000 0101 (0x85)
Byte 1	CMD_STAT
Byte 2	Acquisition Result: • 0x00 – Device was not acquired • 0x01 – Device was acquired

3.5 I2C/SPI Bridge (0x86-0x89)

3.5.1 Set Interface Speed (0x86)

Sets the current Interface data rate.

COMMAND (OUT Packet)	
Byte 0	1000 0110 (0x86)
Byte 1	Request type: • 0x00 – Set speed
Byte 2	Interface type: • 0x00 – I2C interface • 0x01 – SPI interface
Byte 3-6	Interface desired speed (Little-endian). For I2C: • 0x00000000 = 50 kbps • 0x00000001 = 100 kbps • 0x00000002 = 400 kbps • 0x00000003 = 1 Mbps For SPI, this word sets the SPI frequency. If the bit rate is not achievable, the next smaller achievable value is set. The actual value that was set is returned in the response packet.
Byte 7	For SPI only, this byte means: • Bit 0: Bit Order (MSB=0, LSB=1) • Bit 1-2: SPI Mode 0-3
RESPONSE (IN Packet)	
Byte 0	1000 0110 (0x86)
Byte 1	CMD_STAT
Byte 2-5	For SPI only: The actual bit rate that is set

3.5.2 Get Interface Speed (0x86)

Gets the current Interface data rate.

COMMAND (OUT Packet)	
Byte 0	1000 0110 (0x86)
Byte 1	Request type: • 0x01 – Get speed
Byte 2	Interface type: • 0x00 – I2C interface • 0x01 – SPI interface
RESPONSE (IN Packet)	
Byte 0	1000 0110 (0x86)
Byte 1	CMD_STAT
Byte 2-5	Interface current speed (Little-endian). For I2C: • 0x00000000 = 50 kbps • 0x00000001 = 100 kbps • 0x00000002 = 400 kbps • 0x00000003 = 1 Mbps For SPI: The actual bit rate value.
Byte 6	For SPI only, this byte means: • Bit 0: Bit Order (MSB=0, LSB=1) • Bit 1-2: SPI Mode 0-3

3.5.3 Restart I2C Master (0x87)

Disables the I2C hardware, waits for the bus to be released, and re-enables the I2C hardware.

COMMAND (OUT Packet)	
Byte 0	1000 0111 (0x87)
RESPONSE (IN Packet)	
Byte 0	1000 0111 (0x87)
Byte 1	CMD_STAT

3.5.4 I2C Write Transaction with Start / Restart (0x88)

The bridge generates a Start/Restart condition and sends an I2C address byte containing the 7-bit I2C address and a cleared R/W bit. If the address byte is acknowledged, the bridge sends each byte onto the bus and waits for acknowledgement. If any byte is NACKed, the bridge stops sending data onto the bus.

If the 'S' bit in Byte 1 of the COMMAND (OUT Packet) is set, the bridge generates a Stop condition at the end of the transaction. Otherwise, the bridge does not generate a Stop condition, thereby allowing the execution of a following command with a restart or with a continuation of the current transaction.

The command can respond with the WAIT response in CMD_STAT. If the I2C slave stretches the bus and the transaction takes long, the response with the WAIT status will be issued every second to indicate that the bridge is alive. In such cases, no command can be accepted by the bridge until the current operation is completed. The only exception is the Restart I2C Master (see Section **3.5.3**) command that can be used to interrupt unfinished write transactions.

COMMAND (OUT Packet)	
Byte 0	1000 1000 (0x88)
Byte 1	0001 xxSR R indicates whether to generate a Start or Restart condition. • 0 = Generate a Start condition. • 1 = Generate a Restart condition. S indicates whether to generate a Stop condition. • 0 = Do not generate a Stop condition. • 1 = Generate a Stop condition.
Byte 2	Length (number of bytes to write). Must be less than or equal to 60. Otherwise, only 60 bytes are processed in FW.
Byte 3	7-bit I2C slave address. The MSB of this byte must be cleared.
Bytes 4-63	Data to send. Byte 2 of the packet specifies the number of bytes that are valid.
RESPONSE (IN Packet)	
Byte 0	1000 1000 (0x88)
Byte 1	CMD_STAT
Byte 2	ACK indicator for the I2C slave address byte: • 0x00 = NACK • 0x01 = ACK If this byte indicates a NACK, all following data is invalid.
Bytes 3-62	ACK indicator for every byte written: • 0x00 = NACK • 0x01 = ACK Data that occurs after the first NACK indication is invalid.

3.5.5 I2C Read Transaction with Start / Restart (0x88)

The bridge generates a Start/Restart condition and sends an I2C address byte containing the 7-bit I2C address and a set R/W bit. If the address byte is acknowledged, the bridge reads each byte on the bus and acknowledges.

If the S bit in Byte 0 of the COMMAND (OUT Packet) is set, the bridge NACKs the last byte (though this byte is successfully received) and generates a Stop condition at the end of the transaction. Otherwise, the bridge ACKs the last byte and does not generate a Stop condition, thereby allowing execution of a following command with a Restart or with a continuation of the current transaction.

The command can respond with a WAIT response in CMD_STAT. If the I2C slave stretches the bus and the transaction takes long, the response with the WAIT status will be issued every second to indicate that the bridge is alive. In such cases, no command can be accepted by the bridge until the current operation is completed. The only exception is the Restart I2C Master (see Section **3.5.3**) command that can be used to interrupt unfinished read transactions.

COMMAND (OUT Packet)	
Byte 0	1000 1000 (0x88)
Byte 1	0010 xxSR R indicates whether to generate a Start or Restart condition. • 0 = Generate a Start condition. • 1 = Generate a Restart condition. S indicates whether to generate a Stop condition. • 0 = Do not generate a Stop condition. • 1 = Generate a Stop condition.
Byte 2	Length (number of bytes to read). Must be less than or equal to 61. Otherwise, only 61 bytes are processed in FW.
Byte 3	7-bit I2C slave address. The MSB of this byte must be cleared.
RESPONSE (IN Packet)	
Byte 0	1000 1000 (0x88)
Byte 1	CMD_STAT
Byte 2	ACK indicator for the I2C slave address byte: • 0x00 = NACK • 0x01 = ACK If this byte indicates a NACK, all following data is invalid.
Bytes 3-63	Received data bytes in the chronological order of reception from the slave.

3.5.6 I2C Write Transaction Continuation (No Start or Restart) (0x88)

Continues an existing I2C Write transaction. The bridge sends each byte onto the bus and waits for acknowledgment. If any byte is NACKed, the bridge discontinues sending onto the bus.

If the 'S' bit in Byte 1 of the COMMAND (OUT Packet) is set, the bridge generates a Stop condition at the end of the transaction. Otherwise, the bridge does not generate a Stop condition, thereby allowing the execution of a following command with a Restart or with the continuation of the current transaction. The behavior is undefined when this command is sent when an I2C Write transaction is not active.

The command can respond with the WAIT response in CMD_STAT. If the I2C slave stretches the bus and the transaction takes long, the response with the WAIT status will be issued every second to indicate that the bridge is alive. In such cases, no command can be accepted by the bridge until the current operation is completed. The only exception is the Restart I2C Master (see Section **3.5.3**) command that can be used to interrupt unfinished write transactions.

COMMAND (OUT Packet)	
Byte 0	1000 1000 (0x88)
Byte 1	0011 Sxxx S indicates whether to generate a Stop condition. • 0 = Do not generate a Stop condition. • 1 = Generate a Stop condition.
Byte 2	Length (number of bytes to write). Must be less than or equal to 61. Otherwise, only 61 bytes are processed in FW.
Bytes 3-63	Data to send. Byte 2 specifies the number of bytes that are valid.
RESPONSE (IN Packet)	
Byte 0	1000 1000 (0x88)
Byte 1	CMD_STAT
Bytes 2-62	ACK indicator for each byte written: • 0x00 = NACK • 0x01 = ACK Data that occurs after the first NACK indicator is invalid.

3.5.7 I2C Read Transaction Continuation (No Start or Restart) (0x88)

Continues the existing I2C Read transaction. The bridge reads each byte on the bus and acknowledges.

If the 'S' bit in Byte 1 of the COMMAND (OUT Packet) is set, the bridge NACKs the last byte (although this byte is successfully received) and generates a Stop condition at the end of the transaction. Otherwise, the bridge ACKs the last byte and does not generate a Stop condition, thereby allowing the execution of a following command with a Restart or with a continuation of the current transaction. The behavior is undefined when this command is sent when an I2C read transaction is not active.

The command can respond with the WAIT response in CMD_STAT. If the I2C slave stretches the bus and the transaction takes long, the response with the WAIT status will be issued every second to indicate that the bridge is alive. In such cases, no command can be accepted by the bridge until the current operation is completed. The only exception is the Restart I2C Master (see Section **3.5.3**) command that can be used to interrupt unfinished read transactions.

COMMAND (OUT Packet)	
Byte 0	1000 1000 (0x88)
Byte 1	0100 Sxxx S indicates whether to generate a Stop condition. • 0 = Do not generate a Stop condition. • 1 = Generate a Stop condition.
Byte 2	Length (number of bytes to read). Must be less or equal to 62. Otherwise, only 62 bytes are processed in FW.
RESPONSE (IN Packet)	
Byte 0	1000 1000 (0x88)
Byte 1	CMD_STAT
Bytes 2-63	Received data bytes in the chronological order of reception from the slave

3.5.8 SPI Data Transfer (0x89)

Transfers data via SPI. The bridge performs data exchange between the PC and the target application. Before data exchanging, the bridge sets the proper SS (Slave select) from HIGH to LOW level and after completing from LOW to HIGH. Use proper SS values from the Info (0x90) command. If you use unsupported SS values, the response will be FAIL (INVALID_PARAMS).

COMMAND (OUT Packet)	
Byte 0	1000 1001 (0x89)
Byte 1	Length (number of bytes to write/read). Must be less than or equal to 60. Otherwise, only 60 bytes are processed in FW.
Byte 2	SS to use: • 0x01 – SS#0 • 0x02 – SS#1 • 0x04 – SS#2
Byte 3	Control byte: • 0x00 = No Start/Stop for SS • 0x02 = Start condition – SS transition HIGH-> LOW • 0x08 = Stop condition – SS transition LOW-> HIGH

COMMAND (OUT Packet)	
Bytes 4-63	Data
RESPONSE (IN Packet)	
Byte 0	1000 1001 (0x89)
Byte 1	CMD_STAT
Byte 2-61	Received data bytes in the chronological order of reception from the slave.

3.6 GPIO Bridge (0x8A – 0x8D)

Byte 1 in all of the requests specifies the GPIO pin on the KitProg3 device, which is connected to the pin on target device. Available pins are described in the KitProg3 User Guide. If an invalid GPIO pin reference is provided or if a pin mode cannot be set, Byte 1 in response returns FAIL.

3.6.1 Set GPIO Pin Drive Mode (0x8A)

Sets one of the GPIO pins with the chosen drive mode, and returns whether the drive mode of GPIO pin has been set.

COMMAND (OUT Packet)	
Byte 0	1000 1010 (0x8A)
Byte 1	0bXXXX YYYY • XXXX – port number • YYYY – pin number
Byte 2	Desired Pin Drive Mode : • 0x00 - High Impedance Digital • 0x01 - Resistive Pull Up • 0x02 - Resistive Pull Down • 0x03 - Open Drain, Drives Low • 0x04 - Open Drain, Drives High • 0x05 - Strong Drive • 0x06 - Resistive Pull Up & Down
RESPONSE (IN Packet)	
Byte 0	1000 1010 (0x8A)
Byte 1	CMD_STAT

3.6.2 Set GPIO Pin State (0x8B)

Sets GPIO pin state – LOW or HIGH, and returns status whether state has been set. Trying to set LOW or HIGH on pin in High Impedance Digital (HiZ) mode will result in INVALID_PARAMS error.

COMMAND (OUT Packet)	
Byte 0	1000 1011 (0x8B)

COMMAND (OUT Packet)	
Byte 1	0bXXXX YYYY • XXXX – port number • YYYY – pin number
Byte 2	Pin State to set: • 0 – Low (0) • Not 0 – High (1)
RESPONSE (IN Packet)	
Byte 0	1000 1011 (0x8B)
Byte 1	CMD_STAT

3.6.3 Read GPIO Pin State (0x8C)

Returns the current LOW or HIGHT state of the GPIO pin.

COMMAND (OUT Packet)	
Byte 0	1000 1100 (0x8C)
Byte 1	0bXXXX YYYY • XXXX – Port number • YYYY – Pin number
RESPONSE (IN Packet)	
Byte 0	1000 1100 (0x8C)
Byte 1	CMD_STAT
Byte 2	Current pin state on the chosen GPIO pin.

3.6.4 Read GPIO Pin State Change (0x8D)

Returns whether the state of GPIO pin has changed.

COMMAND (OUT Packet)	
Byte 0	1000 1101 (0x8D)
Byte 1	0bXXXX YYYY • XXXX – Port number • YYYY – Pin number

RESPONSE (IN Packet)	
Byte 0	1000 1101 (0x8D)
Byte 1	CMD_STAT
Byte 2	0x00 – State is unchanged 0x01 – Transition from LOW to HIGH occurred. 0x02 – Transition from HIGH to LOW occurred.

3.7 Info Command (0x90)

3.7.1 Probe Info/Capabilities (0x90)

COMMAND (OUT Packet)

Byte 0	1001 0000 (0x90)

RESPONSE (IN Packet)

Byte 0	1001 0000 (0x90)
Byte 1	CMD_STAT
Byte 2	Major interfaces supported, bit assignments: 7: Res; 6: GPIO; 5: VMEAS; 4: PCTRL; 3: DAPB; 2: DAPH; 1: SPI; 0: I2C I2C – Specifies whether I2C bridging is supported SPI – Specifies whether SPI bridging is supported DAPH – Specifies whether CMSIS-DAP 1x mode is supported (USB HID) DAPB – Specifies whether CMSIS-DAP 2x mode is supported (USB Bulk) PCTRL – Specifies whether power on/off is supported VMEAS – Specifies whether voltage measurement is supported GPIO – Specifies whether GPIO bridging is supported
Byte 3	UART(s) and LED(s) supported, bit assignments: 7–5: UARTS; 3–0: LEDS UARTS – Quantity of UARTs supported LEDS – Number of status LEDs
Byte 4	I2C clocks supported and GPIO pins supported, bit assignments: 7–6: Reserved; 5: 3[6]; 4: 3[5]; 3: 1M; 2: 400K; 1: 100K; 0: 50K 50K – Specifies whether 50 kbps is supported 100K – Specifies whether 100 kbps is supported 400K – Specifies whether 400 kbps is supported 1M – Specifies whether 1 Mbps is supported 3[5] - Specifies whether GPIO pin 3[5] (port 3, pin 5) is supported 3[6] - Specifies whether GPIO pin 3[6] (port 3, pin 6) is supported
Byte 5-8	Minimum SPI speed supported, in Hz. Little-endian.
Byte 9-12	Maximum SPI speed supported, in Hz. Little-endian.
Byte 13	SPI slave select supported, bit assignments: 7: SS7; 6: SS6; 5: SS5; 4: SS4; 3: SS3; 2: SS2; 1: SS1; 0: SS0 Bitmask for SS supported. Bit is set to ‘1’ if corresponding SS is supported.

RESPONSE (IN Packet)	
Byte 14	Voltages supported by the kit, bit assignments: 7: TargetInfo; 6–4: Reserved; 3: 5V; 2: 3_3V; 1: 2_5V; 0: 1_8V 1_8V – Specifies whether 1.8V is supported 2_5V – Specifies whether 2.5V is supported 3_3V – Specifies whether 3.3V is supported 5V – Specifies whether 5V is supported TargetInfo – Specifies whether the Extended Target Info feature is supported. If set, the record can be read with the Read Extended Target Info (0x95) command; see section 3.12.

3.8 Set Acquire Parameters (0x91)

This command is available started from 1.10 version of protocol.

3.8.1 Set Acquire Timeout (0x91)

Set the acquisition timeout value during the DAP Acquire (0x85) command execution. If Byte 2 is 0x00, default values are set:

- PSOC™ 4: Reset 2.5 ms, Power Cycle 5 ms
- PSOC™ 5LP: Reset 2.5 ms, Power Cycle 5 ms
- PSOC™ 6A, T2G, AIROC™, XMC7000 - Reset 1000 ms, Power Cycle 1000 ms

COMMAND (OUT Packet)	
Byte 0	1001 0001 (0x91)
Byte 1	0x00 (set acquire timeout)
Byte 2	Timeout value to set in seconds; must be equal or less than 30.
RESPONSE (IN Packet)	
Byte 0	1001 0001 (0x91)
Byte 1	CMD_STAT

3.8.2 Select DAP Handshake Type During DAP Acquire (0x91)

The type of DAP handshake which will be used to select the debug protocol during the process. DAP Acquire request can be selected among the following:

- Line reset/TLR
- <Initial_Protocol> to <Final_Protocol>
- Dormant to <Final_Protocol>
- <Initial_Protocol> to Dormant to <Final_Protocol>

The selected sequence will be used in DAP Acquire requests until KitProg is powered on. If Byte 2 is 0x00, default values are set:

- For PSOC™ 4, PSOC™ 5LP - the SWD Line reset will be used
- For PSOC™ 6A, T2G, AIROC™, XMC7000 - the JTAG to SWD will be used

COMMAND (OUT Packet)	
Byte 0	1001 0001 (0x91)
Byte 1	0x01 (select DAP handshake type)
Byte 2	Defines handshake type: • 0x00 – System default (SWD Line reset for PSOC™ 4, JTAG to SWD otherwise) • 0x01 – SWD Line reset • 0x02 – JTAG to SWD • 0x03 – Dormant to SWD • 0x04 – JTAG to Dormant to SWD
RESPONSE (IN Packet)	
Byte 0	1001 0001 (0x91)
Byte 1	CMD_STAT

3.8.3 Set DAP AP (0x91)

Select active DAP MEM access port through which the test mode bit will be set during acquisition.

COMMAND (OUT Packet)	
Byte 0	1001 0001 (0x91)
Byte 1	0x02 (set DAP AP)
Byte 2	DAP AP number
RESPONSE (IN Packet)	
Byte 0	1001 0001 (0x91)
Byte 1	CMD_STAT

3.9 Read Unique ID Record (0x92)

Provides information about device stored in KitProg3. Byte 2 can be 0xFF in cases when Unique ID Record is corrupted or blank. Byte 3-62 are defined only if Byte 2 is 0x00.

COMMAND (OUT Packet)	
Byte 0	1001 0010 (0x92)

RESPONSE (IN Packet)	
Byte 0	1001 0010 (0x92)
Byte 1	CMD_STAT
Byte 2	Indicator whether Unique ID Record is valid: • 0x00 – Valid • 0xFF – Invalid
Byte 3-6	mbed board ID in ASCII encoding
Byte 7-8	Unique ID of device
Byte 9	Programming interfaces and programming options, bit assignments: 7: Reserved; 6: Reserved; 5: SWO; 4: PPPC; 3: PMPR; 2: JTAG; 1: SWD; 0: DAPLink DAPLink – is DAPLink supported SWD – is communicational interface supported JTAG – is communicational interface supported PMPR – is programming mode reset supported PPPC – is programming mode power cycle supported SWO – is Single Wire Output supported
Byte 10	Supported bridging and bridging features, bit assignments: 7: GPIO; 6: SPI; 5: I2C_PU; 4: I2C; 3: UART_RTS; 2: UARTHW; 1: UARTs; 0: UARTs UARTs – quantity of UARTs UARTHW – is USB-UART HW flow control supported UART_RTS – is special RTS behavior used I2C – is I2C bridging supported I2C_PU – is I2C pullup supported SPI – is SPI bridging supported GPIO – is GPIO bridging supported
Byte 11	LEDs and mode switching parameters for KitProg3, bit assignments: 7: Reserved; 6: Reserved; 5: Reserved; 4: Reserved; 3: MODE; 2: MODE; 1: LEDS; 0: LEDS LEDs – number of status LEDs for KitProg MODE – quantity of mode switching buttons

RESPONSE (IN Packet)

Byte 12

Target type definition, bit assignments:

7	6	5	4	3	2	1	0
Secure	Target-Type						

Target-Type

- x0000000 – PSOC™ 4
- x0000001 – PSOC™ 5LP
- x0000010 – PSOC™ 6
- x0000011 – PSOC™ 6A-2M
- x0000100 – PSOC™ 6A-512
- x0000101 – PSOC™ 6A-256K
- x0000110 – PMG1
- x0000111 – T2G
- x0001000 – PSOC™ 4500
- x0001001 – XMC7000
- x0001010 - AIROC™
- x0001011 - PSOC™ Edge 84
- x0001100 - PSC3M5
- x0001101 - PSC3M8
- x0001110 - PSB3
- x0001111 - PSW11XX
- x0010000 - XMC5000
- x0010001 - FX3GX
- x1111111 - Variable

Secure – is target secure

RESPONSE (IN Packet)

Byte 13

Connectivity module and FRAM support, bit assignments:

7	6	5	4	3	2	1	0
WB					FRAM		

FRAM – type of FRAM:

- xxxx000 – not supported
- xxxx001 - CY15B104QSN
- xxxxx010 - FM24V10
- xxxxx011 - FM25V10

WB – type of connectivity module supported:

- 00000xxx – not supported
- 00001xxx - LBEE5KL1DX
- 00010xxx - CYW43012
- 00011xxx - CYW4343W
- 00100xxx - CYW43438
- 00101xxx - CYBLE-416045
- 00110xxx - CYBLE-022001
- 00111xxx - Various
- 01000xxx - CYSBSYS-RP01
- 01001xxx – Internal BLE
- 01010xxx - CYW20819
- 01011xxx – CYW94373W
- 01100xxx - CYW43439
- 01101xxx - CYW920829
- 01110xxx - CYW55XXX/CYW89XXX
- 01111xxx - PSB3XXX

RESPONSE (IN Packet)

Byte 14	SMIF and QSPI, bit assignments: 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 Reserved | SMIF SMIF – type of Flash memory supported: • xxx00000 – not supported • xxx00001 - S25FL512S • xxx00010 - S25FL064L • xxx00011 - S25FS512S • xxx00100 - S25HL512T • xxx00101 - S25FL256S • xxx00110 - S25FS128S, S28HS01GT, S70KS1283GA • xxx00111 - S25HS512T • xxx01000 - S70FS01GS, S80KS5123GA • xxx01001 - S28HS512T • xxx01010 - S25FS128S • xxx01011 - S26HL512T • xxx01100 - GD25WQ80E • xxx01101 - S28HS01GT, S80KS5123GA • xxx01110 - S26HL512T, S27KL064 • xxx01111 - GD25WQ16E • xxx10000 - W25Q64J • xxx11111 - Variable
Byte 15	Boards features, bit assignments: 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 CAPSENSE | EHA | EHD | EP | VTAR | PWR | VM | VC VC – is voltage control supported VM – is voltage detection supported PWR – is power on/off supported VTAR – is VTARG output disabled by default EP – is External Power supported EHD – is Diligent Pmod extension headers supported EHA – is Arduino compatible headers supported CAPSENSE – is CAPSENSE™ supported
Byte 16	Boards features starting from 01 minor version, bit assignments: 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 Reserved | TargetInfo TargetInfo – is target info supported
Byte 17-48	Full Device Name in ASCII encoding
Byte 49	Device Hardware ID
Byte 50-53	Target Silicon ID
Byte 54	Minor Version of Unique ID Record
Byte 55	Major Version of Unique ID Record
Byte 56-61	Reserved for Future Use
Byte 62	Checksum of Unique ID Record

3.10 Get/Set KitProg3 UART Flow Control and UART mode (0x93)

Combined command for setting and getting KitProg3 UART flow control mode and Half-Duplex mode used in UART communication between the KitProg3 and target device. Supported only by KitProg3-based devices with the UART HW flow control feature; single wire Half-Duplex mode supported only for Primary UART.

COMMAND (OUT Packet)	
Byte 0	1001 0011 (0x93)
Byte 1	Defines Request: • 0x00 – Get KitProg3 UART flow control mode • 0x01 - Set KitProg3 UART flow control mode • 0x02 - Get UART mode • 0x03 - Set UART mode
Byte 2	Defines the KitProg3 UART: • 0x00 - First UART interface • 0x01 - Second UART interface (applicable only for devices with Dual UART mode supported) **Note**: For the UART Duplex mode configuration choosing Second UART interface will always result in an error
Byte 3	Defines the UART flow control mode to set, only applicable for operation 0x01 (Set UART flow control mode): • 0x00 - No UART Hardware flow control • 0x01 - Hardware UART flow control Defines the UART Duplex mode set, only applicable for operation 0x03 (Set UART mode): • 0x00 - Full UART mode • 0x01 - Half-Duplex UART mode

RESPONSE (IN Packet)	
Byte 0	1001 0011 (0x93)
Byte 1	CMD_STAT
Byte 2	Defines current flow control mode set for the requested KitProg3 UART: • 0x00 - UART HW flow control disabled • 0x01 - UART HW flow control enabled Defines current UART Duplex mode set for the requested KitProg3 UART: • 0x00 - Full UART mode is set • 0x01 - Half-Duplex UART is set

3.11 Enable/Disable KitProg3 Bridge Interfaces (0x94)

Command for enabling or disabling the KitProg3 bridge interfaces (I2C/SPI bridge and USB-UART bridge). When disabled, no bridge data is processed in either direction. The data sent by the host while the bridge is disabled, is queued by the host OS USB driver and might be delivered to the target upon re-enabling. Therefore, the host application must *not* send data while the bridge is disabled.

The selected mode set is not retained upon powerup or during mode switching.

COMMAND (OUT Packet)	
Byte 0	1001 0100 (0x94)
Byte 1	Defines Request: • 0x00 - Enable bridge processing • 0x01 - Disable bridge processing

RESPONSE (IN Packet)	
Byte 0	1001 0100 (0x94)
Byte 1	CMD_STAT
Byte 2	Current bridge state: • 0x00 - Bridge processing is enabled • 0x01 - Bridge processing is disabled

3.12 Read Extended Target Info (0x95)

Returns the module identification record that the KitProg3 read from the identification EEPROM on the attached module. Supported only by devices that report the TargetInfo bit in the Probe Info/Capabilities (0x90) response. On all other devices this command returns 0xFF in Byte 0 as a non-existent command.

The record is read once at KitProg3 start-up and served from a RAM cache, so this command does not access the I^2C bus and can be issued while bridging is active. A module attached or replaced after power-on is not detected until the KitProg3 is power cycled.

The record does not fit one 64-byte packet, so a successful response is a sequence of packets that the firmware sends back-to-back in response to a single command packet. The host must keep reading IN packets until Byte 4 is 0x00 and concatenate the data fragments in the order received. The record is at most 256 bytes long, so no more than five packets are returned. The content of the record is opaque to this protocol and is defined by the module identification record specification.

A failed response is a single packet with CMD_STAT set to FAIL (OPERATION_FAIL) and no data. Byte 2 states the reason in both the success and the failure case.

COMMAND (OUT Packet)	
Byte 0	1001 0101 (0x95)

RESPONSE (IN Packet)	
Byte 0	1001 0101 (0x95)
Byte 1	CMD_STAT

RESPONSE (IN Packet)	
Byte 2	Reason code, see the table below. Present in both success and failure responses. • 0x00 - Record is valid. CMD_STAT is SUCCESS. | • 0x01 - No device acknowledged on any of the probed addresses. | • 0x02 - A device acknowledged its address, but the record header could not be read in full. | • 0x03 - The header was read, but it does not carry the expected identification key. | • 0x04 - The header is valid, but the record data could not be read. | • 0x05 - The header is valid, but the record is larger than the KitProg3 can cache. | • 0x06 - The header is valid, but it declares a record of zero length. | For failure responses (Byte 1 = 0x82) the packet ends after Byte 3.
Byte 3	7-bit I2C address the identification EEPROM was found at, in the range 0x50 to 0x57. 0xFF if no device was found.
Byte 4	Number of response packets that follow this one. 0x00 in the last packet. Defined only if Byte 1 is 0x00.
Byte 5	Number of valid data bytes in Byte 6-63 of this packet, from 1 to 58. Defined only if Byte 1 is 0x00.
Byte 6	Fragment of the identification record. Defined only if Byte 1 is 0x00.

4 DAPLink Commands

This section describes the custom commands that are available in DAPLink interface modes.

4.1 Mode Switch (0xA0)

When the DAPLink receives this command, it enters the USB bootloader mode or mode switch immediately. It includes USB re-enumeration. There is no IN packet from the DAPLink. Response shall be issued only if invalid parameters (Byte 1) are received.

COMMAND (OUT Packet)	
Byte 0	1010 0000 (0xA0)
Byte1	Desired mode: • 0x00 – Bootloader • 0x01 – KitProg3 CMSIS-DAP v.2.xx (USB Bulk) • 0x02 – KitProg3 CMSIS-DAP v.1.xx (USB HID) • 0x03 - KitProg3 CMSIS-DAP v.2.xx with two UARTs
RESPONSE (IN Packet)	
Byte 0	1010 0000 (0xA0)
Byte 1	CMD_STAT

If KitProg3 CMSIS-DAP v.2.xx with two UARTs mode is not supported, the kit switches to KitProg3 CMSIS-DAP v.2.xx (USB Bulk).

4.2 Get KitProg Info (0xA1)

Used to get KitProg FW version + HW ID.

COMMAND (OUT Packet)	
Byte 0	1010 0001 (0xA1)
RESPONSE (IN Packet)	
Byte 0	1010 0001 (0xA1)
Byte 1	Major version of KitProg FW
Byte 2	Minor version of KitProg FW
Byte 3	Firmware build number LSB
Byte 4	Firmware build number MSB
Byte 5	Hardware ID

4.3 Reset DAPLink Device (0xA2)

Used to issue a software reset for the DAPLink device. No response from the device is expected.

COMMAND (OUT Packet)	
Byte 0	1010 0010 (0xA2)
RESPONSE (IN Packet)	
None	

Revision history

Rev.	Date	Description of Change
*K	2021-03-23	Document released publicly. Added “Reset DAPLink Device” vendor command Fixed table of command availability per interface Updated table command availability per interfaces with GPIO Bridge commands Updated section numbering Updated Info Command (0x90) with GPIO availability Added GPIO Bridge Commands Protocol version changed to 2.02
*L	2021-04-15	Updated description of GPIO Bridge Commands.
*M	2021-08-10	Added description of Read Unique ID Record (0x92) command Updated DAP Acquire (0x85) command Update available range for vendor commands
*N	2022-05-02	Protocol version changed to 2.03 Added note on setting state on in HiZ for Set GPIO Pin State (0x8B) Updated Read Unique ID Record (0x92) with new supported targets, BT and QSPI modules Updated DAP Acquire (0x85), Set Acquire Timeout (0x91) and Read Unique ID Record (0x92) with AIROC™ and XMC7000 information Formatted all tables Added Reset Target command (0x0A) to the Bridge interface.
*O	2023-04-13	Protocol version changed to 2.04 Added note on SPI SS values support in SPI Data Transfer (0x89) Updated Info (0x90) command with the list of supported GPIO pins and I2C clocks value naming Updated Read Unique ID Record (0x92) with new supported WIFI BT modules Added description of the Get/Set KitProg3 UART Flow Control (0x93) Vendor Command Corrected item numbering in the document Updated table of supported Vendor Commands in section 2
*P	2024-04-26	Updated Read Unique ID Record (0x92) with new supported external memories and target types
*Q	2025-02-21	Updated Read Unique ID Record (0x92) with new target type
*R	2025-06-19	Updated Read Unique ID Record (0x92) with new external memories and connectivity modules
*S	2025-12-02	Updated Read Unique ID Record (0x92) with new target types, external memories, and connectivity modules
*T	2026-03-11	Added description of the Get/Set KitProg3 UART Duplex mode (0x93) Vendor Command
*U	2026-06-05	Protocol version changed to 2.06 Added description of the Enable/Disable KitProg3 Bridge Interfaces (0x94) Vendor Command

Revision history

Rev.	Date	Description of Change
*V	2026-09-23	Protocol version changed to 2.08 Updated Read Unique ID Record (0x92) with new target types, external memories, and connectivity modules Added description of the Read Extended Target Info (0x95) Vendor Command. Updated the table of supported Vendor Commands and the available range for vendor commands in section 2. Updated Probe Info/Capabilities (0x90) with a reference to the new command.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition: 2026-09-23

Published by
Infineon Technologies AG
Am Campeon 1-15
85579 Neubiberg, Germany

© 2026 Infineon Technologies AG.
All rights reserved

Do you have a question about this document?
Email: **erratum@infineon.com**

Document reference:
002-23370 Rev. *V

Important Notice
Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.
For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability.
Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.
If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.
The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.
Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury. However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer.
Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG).
If the Product includes security features:
Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.
If this document includes or references software:
The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved.
Therefore, you may use the software only as provided in the software license agreement accompanying the software.
If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.
For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit **https://www.infineon.com**.