**USB Communication Protocol**

**KUBE Revision 2 (KUBE2)**

****

**Zone24x7 (Private) Limited**

No. 460, Nawala Road

Koswatta 10107, Sri Lanka

Hereinafter referred to as "Zone24x7"



1310 Rimrock Drive

San Jose, CA95120

USA

Last edited: 11 March 2016

Copyright © 2016 Zone24x7 Inc. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from Zone24x7 Inc. All copyright, confidential information, patents, design rights and all other intellectual property rights of whatsoever nature in and to any source code contained herein (including any header files and demonstration code that may be included), are and shall remain the sole and exclusive property of Zone24x7 Inc. The information furnished herein is believed to be accurate and reliable. However, no responsibility is assumed by Zone24x7 Inc for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Table of Contents

[1. Introduction 1](#_Toc445283892)

[2. Microcontroller Composite USB-HID Layer 3](#_Toc445283893)

[2.1 Device Descriptor 3](#_Toc445283894)

[2.2 Report Descriptor for Kube Commands & Responses 4](#_Toc445283895)

[2.3 Report Descriptor for Kube Proximity Data 4](#_Toc445283896)

[2.4 Report Descriptor for Kube Notifications 5](#_Toc445283897)

[2.5 Report Descriptor for Kube Debug Message 5](#_Toc445283898)

[2.6 Configuration Descriptor 6](#_Toc445283899)

[2.7 String Descriptors 11](#_Toc445283900)

[3. Command-Response Protocol 12](#_Toc445283901)

[3.1 Introduction 12](#_Toc445283902)

[3.2 Command 12](#_Toc445283903)

[3.2.1 START (5 bytes) 12](#_Toc445283904)

[3.2.2 PKT\_SIG 12](#_Toc445283905)

[a. PID (Peripheral ID) 12](#_Toc445283906)

[b. COMMAND 13](#_Toc445283907)

[MCU (0x01) 13](#_Toc445283908)

[TEMP\_SENSE (0x03) 14](#_Toc445283909)

[LED0 (0x05), LED1 (0x06), LED2 (0x07) 15](#_Toc445283910)

[BTN0 (0x0A), BTN1 (0x0B) 15](#_Toc445283911)

[BUZZER (0x10) 15](#_Toc445283912)

[AUDIO\_AMP (0x11) 16](#_Toc445283913)

[PROX\_IR0 (0x15), PROX\_IR1 (0x16), PROX\_IR2 (0x17), PROX\_US (0x18) 16](#_Toc445283914)

[RF\_SCANNER (0x1C) 16](#_Toc445283915)

[BARCODE\_SCANNER (0x1D) 16](#_Toc445283916)

[3.2.3 SEQUENCE\_NO 16](#_Toc445283917)

[3.2.4 LENGTH 17](#_Toc445283918)

[3.2.5 DATA 17](#_Toc445283919)

[3.2.6 CRC 17](#_Toc445283920)

[3.3 Response 17](#_Toc445283921)

[3.3.1 STATUS 17](#_Toc445283922)

[3.4 Notification 18](#_Toc445283923)

[3.4.1 PKT\_SIG 18](#_Toc445283924)

[a. PID 18](#_Toc445283925)

[b. NOTIFICATION\_FLAG 18](#_Toc445283926)

[MCU (0x01) Notifications 18](#_Toc445283927)

[TEMP\_SENSE (0x03) Notifications 18](#_Toc445283928)

[BTN1 (0x0A), BTN2 (0x0B) Notifications 18](#_Toc445283929)

[PROX\_IR1 (0x15), PROX\_IR2 (0x16), PROX\_IR3 (0x17), PROX\_US (0x18) Notifications 19](#_Toc445283930)

[BARCODE\_SCANNER (0x1D) Notifications 19](#_Toc445283931)

[3.5 Supported Commands 19](#_Toc445283932)

[4. Appendices 20](#_Toc445283933)

[4.1 16bit CRC generation 20](#_Toc445283934)

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Change Summary |
| 09/10/2015 | D0.1 | Niroshan Karunaratne | Initial Write-up. |
| 10/30/2015 | D0.2 | Pubudu Karunaratna | Added USB Descriptors. |
| 02/23/2016 | D0.3 | Chathuranga Dissanayaka | Added Command-Respond protocol.  Added List of Acronyms and abbreviations. |
| 03/01/2016 | D0.4 | Chathuranga Dissanayaka | Added changes after reviewed by Prashan Weerasinghe.  Interchanged command ID and Hex columns. |
| 03/02/2016 | D0.5 | Kanishka Wijayasekara | Overall review. Added missing commands. START sequence updated to “KUBE2”. |
| 03/04/2016 | D0.6 | Chathuranga Dissanayaka | Chapter 3 - Response packet for command.  PROX\_IR0,1,2 changes to PROX\_IR  Add PROX\_MODE\_SET command |

List of Acronyms and Abbreviations

|  |  |
| --- | --- |
| Acronym | Expansion |
| ID | Identity |
| N/A | Not Allocated |
| \* | Implies default configurations |
| MCU | Microcontroller Unit |
| USB | Universal Serial Bus |
| HID | Human Interface Device |
| PC | Personal Computer (Motherboard) |
| LED | Light Emitting Diode |
| CRC | Cyclic Redundancy Check |
| ( any ) | Can be any value within the range that field can vary |

# Introduction

From the hardware perspective the KUBE contains two main components – a KUBE PC / Toshiba tablet and a custom made control board (KUBE board). The two components are connected to each other via a USB interface.

The following table summarizes the interfaces of each component connected to the KUBE board.

|  |  |  |
| --- | --- | --- |
| **Component** | **Hardware interface** | **Software Interface** |
| Bar code scanner | USB | USB |
| RFID reader | FT232 USB to Serial | COM |
| BLE | USB | COM |
| Proximity | Microcontroller | USB – HID |
| Buttons & LED indicator | Microcontroller | USB – HID |
| Control and configuration Signals | Microcontroller | USB – HID |

The KUBE PC has direct access to the Bar code scanner, RFID reader and BLE through the relevant interfaces. The Proximity sensors, Buttons & LED indicators and Control and configuration Signals are connected through a microcontroller and are accessed through USB HID interfaces.

This document describes this microcontroller HID interface that the KUBE board uses to exchange messages over the USB interface for proximity data, debug messages and other high level commands/responses and asynchronous high level messages. The USB protocol used is a two layered protocol based on,

1. A Standard Composite USB-HID Layer and
2. High Level Messages (proximity data, debug messages and other high level commands/responses and asynchronous notifications)



Figure 1 KUBE System over view

# Microcontroller Composite USB-HID Layer

The KUBE Reader USB Communication Protocol on the KUBE Reader Board is based on the standard USB-HID protocol so that a custom driver is not required on the host side. A composite USB-HID driver is used for this purpose.

The USB-HID driver uses five endpoints for five different kinds of high level messages. This is shown in the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **High Level Message Types** | **End Point Address** | **Direction** | **Report ID** | **Interface Number** |
| Kube Commands | 0x01 | OUT (Host to Device) | 1 | 0 |
| Kube Responses | 0x81 | IN (Device to Host) | 1 | 0 |
| Proximity Data | 0x82 | IN (Device to Host) | - | 1 |
| KUBE Notifications (Asynchronous Events) | 0x83 | IN (Device to Host) | 2 | 2 |
| KUBE Debug Messages (Asynchronous Events) | 0x84 | IN (Device to Host) | 3 | 3 |

Table ‑: Summery of USB Interfaces

Packets for each of the above high level messages are encapsulated in the standard USB-HID protocol. More details on the different report descriptors are given later in this section.

The reports for high level messages (other than Proximity Data) are all 64 bytes long and the first byte always contains the Report ID. The rest of the report bytes (up to 63) contain the payload which is the high level message or a portion of it. Any unused bytes after the payload are filled with zeros to make the report 64-bytes in length. i.e.

|  |  |  |
| --- | --- | --- |
| **Report ID** | **Payload** | **Zero Padding** |
| 1 Byte | n Bytes | (64-n-1) Bytes |

Table ‑: Payload in HID Report

A single high level message (KUBE Command, KUBE Response, KUBE Notification, Debug Message) may be encapsulated in a single report or it may span multiple consecutive reports.

If a high level message is up to 63 bytes in length then it is encapsulated as a single message.

If a high level message is longer than 63 bytes then it spans multiple reports. The Report ID for each of these reports will be the same. A higher level application or function can determine how many reports to expect based on the information included in the initial part of the high level message (such as the data length).

Some details of the structures for the descriptors and reports are given below.

## Device Descriptor

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Length** | **Value** | **Description** |
| Length | 1 byte | 12h | Size of this descriptor in bytes |
| Descriptor Type | 1 byte | 01h | DEVICE Descriptor Type |
| USB | 2 bytes | 0200h | USB Specification Release Number in  Binary-Coded Decimal (i.e., 2.10 is 210H). |
| Device Class | 1 byte | 00h | Class code |
| Device Sub-Class | 1 byte | 00h | Subclass code |
| Device Protocol | 1 byte | 00h | Protocol code |
| Max Packet Size for EP0 | 1 byte | 40h | Max packet size for EP0 = 64 bytes |
| Vendor ID | 2 bytes | 0483h | STMicroelectronics |
| Product ID | 2 bytes | 6101h | Kube Device |
| Device | 2 bytes | 0200h | Device Release Number (in BCD) in little-endian format. |
| Manufacturer Index | 1 byte | 01h | Index of String Desc for Manufacturer |
| Product Index | 1 byte | 02h | Index of String Desc for Product |
| Serial Number Index | 1 byte | 03h | Index of String Desc for Serial Number |
| Number of Configurations | 1 byte | 01h | Number of possible configurations |

Table ‑: Kube USB Device Descriptor

## Report Descriptor for Kube Commands & Responses

A single report descriptor structure is used for Kube commands & responses. The contents of this structure are given in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Length** | **Value** | **Description** |
| Usage Page | 3 Bytes | 06 00 FF | USAGE\_PAGE (Vendor Defined) |
| Usage | 2 Bytes | 09 01 | USAGE (Vendor Defined Usage 1) |
| Collection | 2 Bytes | A1 01 | COLLECTION (Application) |
| **Payload** | | | |
| Report ID | 2 Bytes | 85 01 | REPORT\_ID (1) |
| Usage | 2 Bytes | 09 03 | USAGE (chars) |
| Logical Minimum | 2 Bytes | 15 00 | LOGICAL\_MINIMUM (0) |
| Logical Maximum | 3 Bytes | 26 FF 00 | LOGICAL\_MAXIMUM (255) |
| Report Size | 2 Bytes | 75 08 | REPORT\_SIZE (8 bits) |
| Report Count | 2 Bytes | 95 3F | REPORT\_COUNT (63) |
| Output | 2 Bytes | 91 82 | OUTPUT (Data, Variable, Absolute, Volatile) |
|  |  |  |  |
| Report ID | 2 Bytes | 85 01 | REPORT\_ID (1) |
| Usage | 2 Bytes | 09 03 | USAGE (chars) |
| Logical Minimum | 2 Bytes | 15 00 | LOGICAL\_MINIMUM (0) |
| Logical Maximum | 3 Bytes | 26 FF 00 | LOGICAL\_MAXIMUM (255) |
| Report Size | 2 Bytes | 75 08 | REPORT\_SIZE (8 bits) |
| Report Count | 2 Bytes | 95 3F | REPORT\_COUNT (63) |
| Input | 2 Bytes | 81 82 | INPUT (Data, Variable, Absolute, Volatile) |
|  |  |  |  |
| End Collection | 1 Byte | C0h | END\_COLLECTION |

Table ‑: Kube Commands & Responses Report Descriptor

## Report Descriptor for Kube Proximity Data

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Length** | **Value** | **Description** |
| Usage Page | 3 Bytes | 06 00 FF | USAGE\_PAGE (Vendor Defined 1) |
| Usage | 2 Bytes | 09 02 | USAGE (Vendor Defined Usage 2) |
| Collection | 2 Bytes | A1 01 | COLLECTION (Application) |
| **Payload** | | | |
| Report ID | 2 Bytes | 85 02 | REPORT\_ID (2) |
| Usage | 2 Bytes | 09 0A | USAGE (Left) |
| Usage | 2 Bytes | 09 0B | USAGE (Right) |
| Logical Minimum | 2 Bytes | 15 00 | LOGICAL\_MINIMUM (0) |
| Logical Maximum | 3 Bytes | 26 FF 0F | LOGICAL\_MAXIMUM (4095) |
| Report Size | 2 Bytes | 75 10 | REPORT\_SIZE (16) |
| Report Count | 2 Bytes | 95 02 | REPORT\_COUNT (2) |
| Input | 2 Bytes | 81 82 | INPUT (Data, Variable, Absolute, Volatile) |
|  | | | |
| End Collection | 1 Byte | C0 | END\_COLLECTION |

Table ‑: Kube Proximity Data Report Descriptor

## Report Descriptor for Kube Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Length** | **Value** | **Description** |
| Usage Page | 3 Bytes | 06 00 FF | USAGE\_PAGE (Vendor Defined 1) |
| Usage | 2 Bytes | 09 03 | USAGE (Vendor Defined Usage 3) |
| Collection | 2 Bytes | A1 01 | COLLECTION (Application) |
| **Payload** | | | |
| Report ID | 2 Bytes | 85 03 | REPORT\_ID (3) |
| Usage | 2 Bytes | 09 03 | USAGE (chars) |
| Logical Minimum | 2 Bytes | 15 00 | LOGICAL\_MINIMUM (0) |
| Logical Maximum | 3 Bytes | 26 FF 00 | LOGICAL\_MAXIMUM (255) |
| Report Size | 2 Bytes | 75 08 | REPORT\_SIZE (8 bits) |
| Report Count | 2 Bytes | 95 3F | REPORT\_COUNT (63) |
| Input | 2 Bytes | 81 82 | INPUT (Data, Variable, Absolute, Volatile) |
|  | | | |
| End Collection | 1 Byte | C0 | END\_COLLECTION |

Table ‑: Kube Notifications Report Descriptor

## Report Descriptor for Kube Debug Message

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Length** | **Value** | **Description** |
| Usage Page | 3 Bytes | 06 00 FF | USAGE\_PAGE (Vendor Defined 1) |
| Usage | 2 Bytes | 09 04 | USAGE (Vendor Defined Usage 4) |
| Collection | 2 Bytes | A1 01 | COLLECTION (Application) |
| **Payload** | | | |
| Report ID | 2 Bytes | 85 04 | REPORT\_ID (4) |
| Usage | 2 Bytes | 09 03 | USAGE (chars) |
| Logical Minimum | 2 Bytes | 15 00 | LOGICAL\_MINIMUM (0) |
| Logical Maximum | 3 Bytes | 26 FF 00 | LOGICAL\_MAXIMUM (255) |
| Report Size | 2 Bytes | 75 08 | REPORT\_SIZE (8 bits) |
| Report Count | 2 Bytes | 95 3F | REPORT\_COUNT (63) |
| Input | 2 Bytes | 81 82 | INPUT (Data, Variable, Absolute, Volatile) |
|  | | | |
| End Collection | 1 Byte | C0 | END\_COLLECTION |

Table ‑: Kube Debus Messages Report Descriptor

## Configuration Descriptor

| **Field** | **Length** | **Value** | **Description** |
| --- | --- | --- | --- |
| Length | UInt8 | 09h | Standard length of descriptor including length byte. Common to all HID Devices. |
| Descriptor Type | UInt8 | 02h | Descriptor Type = Configuration Descriptor |
| Total Length | UInt16 | 0074  (116 dec) | Total Length of this Configuration Descriptor and all other descriptors associated with this configuration. The number is in little-endian format.  Total Length = Len (Configuration Descriptor) +  Len (Interface Descriptors) +  Len (HID Descriptors) +  Len (Endpoint Descriptors) |
| Number of Interfaces | UInt8 | 04h | Number of interface settings contained in this configuration. |
| Configuration Value | UInt8 | 01h |  |
| Configuration String Index | UInt8 | 04h | Index for a string that defines this configuration. |
| Attributes | UInt8 | 80h | Bus Powered |
| Max Power | UInt8 | 32h | In 2 mA units. (= 100 mA) |
| ***Descriptors Associated with this Configuration*** | | | |
| ***Interface 0: The Kube Command HID Interface (includes Kube Commands & Responses)*** | | | |
| *Interface Descriptor* | | | |
| Length | UInt8 | 09h |  |
| Descriptor Type | UInt8 | 04h | Descriptor Type = Interface Descriptor |
| Interface Number | UInt8 | 00h |  |
| Alternate Setting | UInt8 | 00h |  |
| Number of End Points | UInt8 | 02h |  |
| Interface Class | UInt8 | 03h | 03: Device Class is HID |
| Interface Sub-Class | UInt8 | 00h |  |
| Interface Protocol | UInt8 | 00h | 0=None, 1=Keyboard, 2=Mouse |
| Interface String Index | UInt8 | 05h | Index of string that describes the interface. |
| *HID Descriptor* | | | |
| Length | UInt8 | 09h |  |
| Descriptor Type | UInt8 | 21h | Descriptor Type = HID |
| HID Class Specification Version | UInt16 | 0110h | Format: BCD |
| Country Code | UInt8 | 00h |  |
| Number of Descriptors | UInt8 | 01h | Number of Report Descriptors included in this HID configuration.  Note: A single Report Descriptor may define multiple reports. |
| Descriptor Type | UInt8 | 22h | This describes the first descriptor that will follow the transfer of this HID descriptor.  Descriptor Type = 22h = Report Descriptor. |
| Item Length | UInt16 | 0026h | Size of Kube Commands & Responses Report Descriptor |
| *(IN) End Point Descriptor* | | | |
| Length | UInt8 | 07h |  |
| Descriptor Type | UInt8 | 05h | Descriptor Type = Endpoint Descriptor |
| End Point Address | UInt8 | 81h | Bit Definitions:  b3-b0: Endpoint Address  b7: Direction (0=OUT, 1=IN) |
| Attributes | UInt8 | 03h | Type of data transfer.  03h: Interrupt Data Transfer Method. |
| Max Packet Size | UInt16 | 0040h  (64 dec) | Max Packet Size = 64 bytes.  Stored in little-endian format. |
| Interval | UInt8 | 01h | How often the end point will be polled for data by the system software. (= 1 ms) |
| *(OUT) End Point Descriptor* | | | |
| Length | UInt8 | 07h |  |
| Descriptor Type | UInt8 | 05h |  |
| End Point Address | UInt8 | 01h |  |
| Atttributes | UInt8 | 03h |  |
| Max Packet Size | UInt16 | 0040h |  |
| Interval | UInt8 | 01h | (= 1 ms) |
| ***Interface 1: The Kube Proximity Data Interface*** | | | |
| *Interface Descriptor* | | | |
| Length | UInt8 | 09h |  |
| Descriptor Type | UInt8 | 04h | Descriptor Type = Interface Descriptor |
| Interface Number | UInt8 | 01h | Second interface (Num=1). Others will follow. |
| Alternate Setting | UInt8 | 00h |  |
| Number of End Points | UInt8 | 01h |  |
| Interface Class | UInt8 | 03h | 03: Device Class is HID |
| Interface Sub-Class | UInt8 | 00h |  |
| Interface Protocol | UInt8 | 00h | 0=None, 1=Keyboard, 2=Mouse |
| Interface String Index | UInt8 | 06h | Index of string that describes the interface. |
| *HID Descriptor* | | | |
| Length | UInt8 | 09h |  |
| Descriptor Type | UInt8 | 21h | Descriptor Type = HID |
| HID Class Specification Version | UInt16 | 0110h | Format: BCD |
| Country Code | UInt8 | 00h |  |
| Number of Descriptors | UInt8 | 01h | Number of Report Descriptors included in this HID configuration.  Note: A single Report Descriptor may define multiple reports. |
| Descriptor Type | UInt8 | 22h | This describes the first descriptor that will follow the transfer of this HID descriptor.  Descriptor Type = 22h = Report Descriptor. |
| Item Length | UInt16 | 0019h | Size of Kube Proximity Data Descriptor |
| *(IN) End Point Descriptor* | | | |
| Length | UInt8 | 07h |  |
| Descriptor Type | UInt8 | 05h | Descriptor Type = Endpoint Descriptor |
| End Point Address | UInt8 | 812h | Bit Definitions:  b3-b0: Endpoint Address  b7: Direction (0=OUT, 1=IN) |
| Attributes | UInt8 | 03h | Type of data transfer.  03h: Interrupt Data Transfer Method. |
| Max Packet Size | UInt16 | 0040h  (64 dec) | Max Packet Size in bytes.  Stored in little-endian format. |
| Interval | UInt8 | 10h | How often the end point will be polled for data by the system software.  (= 16 ms) |
| ***Interface 2: The Kube Notifications Interaface*** | | | |
| *Interface Descriptor* | | | |
| Length | UInt8 | 09h |  |
| Descriptor Type | UInt8 | 04h | Descriptor Type = Interface Descriptor |
| Interface Number | UInt8 | 02h | Third interface (Num=2). Others will follow. |
| Alternate Setting | UInt8 | 00h |  |
| Number of End Points | UInt8 | 01h |  |
| Interface Class | UInt8 | 03h | 03: Device Class is HID |
| Interface Sub-Class | UInt8 | 00h |  |
| Interface Protocol | UInt8 | 00h | 0=None, 1=Keyboard, 2=Mouse |
| Interface String Index | UInt8 | 07h | Index of string that describes the interface. |
| *HID Descriptor* | | | |
| Length | UInt8 | 09h |  |
| Descriptor Type | UInt8 | 21h | Descriptor Type = HID |
| HID Class Specification Version | UInt16 | 0110h | Format: BCD |
| Country Code | UInt8 | 00h |  |
| Number of Descriptors | UInt8 | 01h | Number of Report Descriptors included in this HID configuration.  Note: A single Report Descriptor may define multiple reports. |
| Descriptor Type | UInt8 | 22h | This describes the first descriptor that will follow the transfer of this HID descriptor.  Descriptor Type = 22h = Report Descriptor. |
| Item Length | UInt16 | 0017h | The size Kube Notifications Report descriptor |
| *(IN) End Point Descriptor* | | | |
| Length | UInt8 | 07h |  |
| Descriptor Type | UInt8 | 05h | Descriptor Type = Endpoint Descriptor |
| End Point Address | UInt8 | 83h | Bit Definitions:  b3-b0: Endpoint Address  b7: Direction (0=OUT, 1=IN) |
| Attributes | UInt8 | 03h | Type of data transfer.  03h: Interrupt Data Transfer Method. |
| Max Packet Size | UInt16 | 0008h  (8 dec) | Max Packet Size in bytes.  Stored in little-endian format. |
| Interval | UInt8 | 10h | How often the end point will be polled for data by the system software. Units depend on speed of device.  (= 1 ms) |
| ***Interface 3: The Kube Debug Message Interface*** | | | |
| *Interface Descriptor* | | | |
| Length | UInt8 | 09h |  |
| Descriptor Type | UInt8 | 04h | Descriptor Type = Interface Descriptor |
| Interface Number | UInt8 | 03h | Fourth interface (Num=3). |
| Alternate Setting | UInt8 | 00h |  |
| Number of End Points | UInt8 | 01h |  |
| Interface Class | UInt8 | 03h | 03: Device Class is HID |
| Interface Sub-Class | UInt8 | 00h |  |
| Interface Protocol | UInt8 | 00h | 0=None, 1=Keyboard, 2=Mouse |
| Interface String Index | UInt8 | 08h | Index of string that describes the interface. |
| *HID Descriptor* | | | |
| Length | UInt8 | 09h |  |
| Descriptor Type | UInt8 | 21h | Descriptor Type = HID |
| HID Class Specification Version | UInt16 | 0110h | Format: BCD |
| Country Code | UInt8 | 00h |  |
| Number of Descriptors | UInt8 | 01h | Number of Report Descriptors included in this HID configuration.  Note: A single Report Descriptor may define multiple reports. |
| Descriptor Type | UInt8 | 22h | This describes the first descriptor that will follow the transfer of this HID descriptor.  Descriptor Type = 22h = Report Descriptor. |
| Item Length | UInt16 | 1700h | The size of Kube Debug Message Report Descriptor |
| *(IN) End Point Descriptor* | | | |
| Length | UInt8 | 07h |  |
| Descriptor Type | UInt8 | 05h | Descriptor Type = Endpoint Descriptor |
| End Point Address | UInt8 | 84h | Bit Definitions:  b3-b0: Endpoint Address  b7: Direction (0=OUT, 1=IN) |
| Attributes | UInt8 | 03h | Type of data transfer.  03h: Interrupt Data Transfer Method. |
| Max Packet Size | UInt16 | 0040h  (64 dec) | Max Packet Size in bytes.  Stored in little-endian format. |
| Interval | UInt8 | 01h | How often the end point will be polled for data by the system software. Units depend on speed of device.  (= 1 ms) |

Table ‑: Kube USB Configuration Descriptor

## String Descriptors

|  |  |  |
| --- | --- | --- |
| **Index** | **String Name** | **Value** |
| 0 | Language String | 04 09h (English – United States) |
| 1 | Manufacturer | " Zone24x7 Incorporated |
| 2 | Pruduct | " KUBE-Reader" |
| 3 | Serial | "0000-0000-0000" if there is no Serial Number in the Reader.  The appropriate Serial Number if Serial Number is available in Where ????. |
| 4 | Configuration Descriptor | "Kube Reader – Config Desc" |
| 5 | Command HID Interface Description | “Kube Reader – Command HID Intf.” |
| 6 | Debug HID Interface Description | “Kube Reader – Proximity HID Intf.” |
| 7 | Touch HID Interface Description | “Kube Reader – Notifications HID Intf.” |
| 8 | Notifications HID Interface Description | “Kube Reader – Debug HID Intf.” |

Table ‑: Kube USB String Descriptors

# Command-Response Protocol

## Introduction

## Command

Following diagram shows the format of the command packet which is designed to send commands from PC and the microcontroller. The packet has the following fields.

1. START – Start Pattern
2. PKT\_SIG – Packet signature
3. SEQUENCE\_NO – Sequence number
4. LENGTH – Length of the data field
5. DATA – Data to transmit
6. CRC – Cyclic Redundancy check

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5Bytes | 2Bytes | 1Byte | 2Bytes | (LENGTH)Bytes | 2Bytes |
| START (“KUBE2”) | PKT\_SIG | SEQUENCE\_NO | LENGTH | DATA | CRC |

Table ‑ Command packet format

### START (5 bytes)

The Start-of-transmission is marked by the START field and it contains a 5Byte unique string which is “KUBE2”.

### PKT\_SIG

This field is used to send command in hierarchical mode to decode easily. PID field specifies the device for which the command is intended. COMMAND field uniquely specifies the command.

|  |  |
| --- | --- |
| 1Byte | 1Byte |
| **PID** | **COMMAND** |

Table ‑ PKT\_SIG field format

1. PID (Peripheral ID)

Each peripheral of the daughter board has a unique ID represented as the PID. The PID is an 8bit field. Following table shows the PID values allocated for peripherals.

|  |  |  |  |
| --- | --- | --- | --- |
| **Decimal** | **Allocated for** | **Hex** | **Description** |
| 0 | RSVD | 0x00 | Reserved |
| 1 | MCU | 0x01 | Microcontroller |
| 2 | N\A | 0x02 | Not Allocated |
| 3 | TEMP\_SENSE | 0x03 | Temperature Sensor |
| 4 | N\A | 0x04 | Not Allocated |
| 5 | LED0 | 0x05 | LED0 |
| 6 | LED1 | 0x06 | LED1 |
| 7 | LED2 | 0x07 | LED2 |
| 8 | N\A | 0x08 | Not Allocated |
| 9 | N\A | 0x09 | Not Allocated |
| 10 | BTN0 | 0x0A | Button0 |
| 11 | BTN1 | 0x0B | Button1 |
| 12 | N\A | 0x0C | Not Allocated |
| 13 | N\A | 0x0D | Not Allocated |
| 14 | N\A | 0x0E | Not Allocated |
| 15 | N\A | 0x0E | Not Allocated |
| 16 | BUZZER | 0x10 | Buzzer |
| 17 | AUDIO\_AMP | 0x11 | Audio amplifier |
| 18 | N\A | 0x12 | Not Allocated |
| 19 | N\A | 0x13 | Not Allocated |
| 20 | N\A | 0x14 | Not Allocated |
| 21 | PROX\_IR0 | 0x15 | Proximity IR |
| 22 | N\A | 0x16 | Not Allocated |
| 23 | N\A | 0x17 | Not Allocated |
| 24 | PROX\_US | 0x18 | Proximity Ultra-Sonic |
| 25 | N\A | 0x19 | Not Allocated |
| 26 | N\A | 0x1A | Not Allocated |
| 27 | N\A | 0x1B | Not Allocated |
| 28 | RF\_SCANNER | 0x1C | RFID scanner |
| 29 | BARCODE\_SCANNER | 0x1D | Barcode scanner |
| 30 - 254 | N\A | 0x1E – 0xFE | Not Allocated |
| 255 | RSVD | 0xFF | Reserved |

Table ‑ Peripheral ID list

1. COMMAND

MCU (0x01)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | RSVD | 0x00 | - | Reserved |
| 1 | RSVD | 0x01 | - | Reserved |
| 2 | RSVD | 0x02 | - | Reserved |
| 3 | MCU\_DEBUG\_ENABLE | 0x03 | No | Enable serial debug interface. |
| \*4 | MCU\_DEBUG\_DISABLE | 0x04 | No | Disable serial debug interface. |
| 5 | MCU\_FIRMWARE\_UPGRADE | 0x05 | No | MCU switches to firmware upgrade mode. |
| 6 | RSVD | 0x06 | - | Reserved |
| 7 | RSVD | 0x07 | - | Reserved |
| 8 | RSVD | 0x08 | - | Reserved |
| 9 | RSVD | 0x09 | - | Reserved |
| 10 | RSVD | 0x0A | - | Reserved |
| 11 | MCU\_REG\_A\_5V\_ON | 0x0B | No | Turn on REG\_A\_5V regulator. Then the peripherals powered by this regulator will be powered up. This regulator powers ULTRA SONIC PROXIMITY , FT232\_UART , RS500\_RFID , FAN, TEMP SENSOR, 3.3V REGULATOR A peripheral/s. |
| 12 | MCU\_REG\_A\_5V\_OFF | 0x0C | No | Turn off REG\_A\_5V regulator. Then the peripherals powered by this regulator will be powered up. This regulator powers ULTRA SONIC PROXIMITY , FT232\_UART , RS500\_RFID , FAN, TEMP SENSOR, 3.3V REGULATOR A peripheral/s. |
| 13 | MCU\_REG\_B\_5V\_ON | 0x0D | No | Turn on REG\_B\_5V regulator. Then the peripherals powered by this regulator will be powered up. This regulator powers BLE, EXTERNAL USB PORTS' VBUS, 3.3V REGULATOR B peripheral/s. |
| 14 | MCU\_REG\_B\_5V\_OFF | 0x0E | No | Turn off REG\_B\_5V regulator. Then the peripherals powered by this regulator will be powered up. This regulator powers BLE, EXTERNAL USB PORTS' VBUS, 3.3V REGULATOR B peripheral/s. |
| 15 | MCU\_REG\_A\_3V3\_ON | 0x0F | No | Turn on REG\_A\_3V3 regulator. Then the peripherals powered by this regulator will be powered up. This regulator powers BAR CODE SCANNER peripheral/s. |
| 16 | MCU\_REG\_A\_3V3\_OFF | 0x10 | No | Turn off REG\_A\_3V3 regulator. Then the peripherals powered by this regulator will be powered up. This regulator powers BAR CODE SCANNER peripheral/s. |
| 17 | RSVD | 0x11 | - | Reserved |
| 18 | RSVD | 0x12 | - | Reserved |
| 19 | MCU\_REG\_9V\_ON | 0x13 | No | Turn on REG\_9V regulator. Then the peripherals powered by this regulator will be powered up. This regulator powers 10W STEREO AMPLIFIER peripheral/s. |
| 20 | MCU\_REG\_9V\_OFF | 0x14 | No | Turn off REG\_9V regulator. Then the peripherals powered by this regulator will be powered up. This regulator powers 10W STEREO AMPLIFIER peripheral/s. |
| 21 | MCU\_ALL\_REG\_ON | 0x15 | No | Turn on all the Regulators. This is for turning on all peripherals powered by the regulator which is controlled by MCU. |
| 22 | MCU\_ALL\_REG\_OFF | 0x16 | No | Turn on all the Regulators. This is for turning on all peripherals powered by the regulator which is controlled by MCU. |
| 23 | MCU\_GET\_FW\_VERSION | 0x17 | No | Requesting the firmware version by the PC. A response packet is expected from the MCU. |
| 24 | MCU\_GET\_BOARDID | 0x18 | No | Requesting the Board ID by the PC. A response packet is expected from the MCU. |
| 25 | MCU\_DIP\_READ | 0x19 | No | Request the port values of the dip switches. |

Table ‑ MCU Command list

TEMP\_SENSE (0x03)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | RSVD | 0x00 | - | Reserved |
| 1 | RSVD | 0x01 | - | Reserved |
| 2 | RSVD | 0x02 | - | Reserved |
| 3 | TEMP\_READ | 0x03 | No | Requesting for the current temperature. Temperature will be received in data field as 2Byte integer value. Data is in MSB first format (Big endian). |
| 4 | TEMP\_SET\_UPPER\_LIM | 0x04 | Yes | Set upper limit. Over temperature threshold of the temperature sensor. Upper threshold value is added in the data field as 1Byte signed integer (-127 to 128 celcius) value. Data is in MSB first format (Big endian).  Default is 80 Celsius. |
| 5 | TEMP\_SET\_LOWER\_LIM | 0x05 | Yes | Set lower limit. This is to clear the over temperature detect event. Upper threshold value is added in the data field as 1Byte signed integer (-127 to 128 celcius) value Should be smaller than TEMP\_SET\_UPPER\_LIM. Data is in MSB first format (Big endian).  Default is 75 Celsius. |

Table ‑ TEMP\_SENSE Command list

LED0 (0x05), LED1 (0x06), LED2 (0x07)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 1 | LED\_ON | 0x01 | No | Turn on the relevant LED. |
| \*2 | LED\_OFF | 0x02 | No | Turn off the relevant LED |
| 3 | LED\_BLINK\_SLOW | 0x03 | No | Blink the LED 50% duty cycle. 1s ON time. |
| 4 | LED\_BLINK\_FAST | 0x04 | No | Blink the LED 50% duty cycle. 0.25s ON time. |
| 5 | LED\_BLINK\_MID | 0x05 | No | Blink the LED 50% duty cycle. 0.5s ON time. |
| 6 | LED\_BLINK | 0x06 | Yes | On-Time, Off-Time are included in data field. Used for custom on time and off time in milliseconds. There are 4Bytes in data field. 1st two Byte for On-Time and 2nd two byte for Off-Time. Data is in MSB first format (Big endian). |

Table ‑ LED Command list

BTN0 (0x0A), BTN1 (0x0B)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | RSVD | 0x00 | - | Reserved |
| 1 | RSVD | 0x01 | - | Reserved |
| 2 | RSVD | 0x02 | - | Reserved |
| 3 | BTN\_READ | 0x03 | No | Request the status of the relevant button. |

Table ‑ BTN Command list

BUZZER (0x10)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | RSVD | 0x00 | - | Reserved |
| 1 | RSVD | 0x01 | - | Reserved |
| 2 | RSVD | 0x02 | - | Reserved |
| 3 | BEEP\_SHORT | 0x05 | Yes | Beep n times with 50% duty cycle with 0.25S. ‘n’ is sent as 2Byte integer in data field. Data is in MSB first format (Big endian). |
| 4 | BEEP\_LONG | 0x06 | Yes | Beep n times with 50% duty cycle with 0.5S. ‘n’ is sent as 2Byte integer in data field. Data is in MSB first format (Big endian). |
| 5 | BEEP\_FOR | 0x07 | Yes | Beep for t milliseconds. ‘t’ is sent as 2Byte integer in data field. Data is in MSB first format (Big endian). |

Table ‑ Buzzer Command list

AUDIO\_AMP (0x11)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | RSVD | 0x00 | - | Reserved |
| 1 | RSVD | 0x01 | - | Reserved |
| 2 | RSVD | 0x02 | - | Reserved |
| 3 | AMP\_MUTE | 0x03 | No | Mute the output of the amplifier. |
| 4 | AMP\_STANDBY | 0x04 | No | Amplifier set to Standby mode. |
| 5 | AMP\_VOLUME\_LEVEL | 0x05 | Yes | Level of volume (1, 2, 3, 4). This level is sent as a 1Byte integer in data field. Data is in MSB first format (Big endian). |

Table ‑ AUDIO\_AMP Command list

PROX\_IR (0x15), PROX\_US (0x18)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | RSVD | 0x00 | - | Reserved |
| 1 | RSVD | 0x01 | - | Reserved |
| 2 | RSVD | 0x02 | - | Reserved |
| 3 | PROX\_READ | 0x03 | No | Requesting the reading of the relevant proximity sensor. |
| 4 | PROX\_SET\_MODE | 0x04 | Yes | Three bytes of data CONT – Continuous mode  0x01 | 0xmm | 0xnn 0x01 – Continuous mode 0xmmnn – Time interval of notification sending in milliseconds INT – Interrupt mode  0x02 | 0xmm | 0xnn 0x02 – Interrupt mode 0xmmnn – Threshold limit of person detection in inches |

Table ‑ PROXIMITY Sensor Command list

RF\_SCANNER (0x1C)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | RSVD | 0x00 | - | Reserved |
| 1 | RF\_ON | 0x01 | No | Turn on the power of the RF Scanner. |
| \*2 | RF\_OFF | 0x02 | No | Turn off the power of the RF Scanner. |
| 3 | RF\_RESET | 0x03 | No | Reset the RF Scanner |
| 4 | RF\_FT\_UARTBRIDGE\_RESET | 0x04 | No | Reset the FT232 UART Bridge. This UART Bridge converts the UART data received by the RF Scanner and converts them to USB data and connects to the USB Hub. |

Table ‑ RF\_SCANNER Command list

BARCODE\_SCANNER (0x1D)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | RSVD | 0x00 | - | Reserved |
| 1 | BARCODE\_ON | 0x01 | No | Turn on the power of the Barcode Scanner. |
| 2 | BARCODE\_OFF | 0x02 | No | Turn off the power of the Barcode Scanner. |

Table ‑ BARCODE\_SCANNER Command list

### SEQUENCE\_NO

This is an 8bit value to keep track of the command/response packets. The sequence number is generated by the PC and subjected to **cyclic increment** per each new command packet. The MCU does not actively generate sequence numbers. When the MCU forms the response packet, the sequence number that was in the corresponding command packet will be used by the MCU. This way, the response for a particular command contains the same sequence number as in the command packet.

If the PC side detects a sequence number mismatch, the PC will discard that response packet and resend the previous command. This simple recovery mechanism is sufficient considering the blocking messaging nature and the fact that the MCU does not originate sequence numbers.

### LENGTH

This is an 8bit variable contains the number of bytes in the data field. If there is no data LENGTH field is equals to 0. And otherwise it should shows the bytes of data field.

### DATA

This field contains the data corresponding to the packet. The length of this field can vary from zero to a maximum of the LENGTH field value.

### CRC

This field is used to verify the integrity of the received packet. CRC-16 CCITT algorithm is used for verifying the received packet and for calculating the CRC field for encapsulating new packets. [CRC generation function](#_16bit_CRC_generation) is appended to appendices.

## Response

Following diagram shows the format of the response packet which is intended for responses/acknowledgements from the microcontroller to the PC. In this packet there are seven fields.

1. START – Start byte
2. PKT\_SIG – Packet signature
3. SEQUENCE\_NO – Sequence number
4. STATUS
5. LENGTH – Length of the data field
6. DATA – Data to transmit
7. CRC – Cyclic Redundancy check

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5Bytes | 2Bytes | 1Byte | 1Byte | 2Bytes | (LENGTH)Bytes | 2Bytes |
| START (“KUBE2”) | PKT\_SIG | SEQUENCE\_NO | STATUS | LENGTH | DATA | CRC |

Table ‑ Response format

The only difference compared to the command packet is the STATUS field. The definition of the STATUS field is as follows. The other fields are as same as in the Command packet.

### STATUS

STATUS field is 1Byte information which is used to send the status of the microcontroller to the command. Status definitions are mentioned below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Description** |
| 0 | SUCCESS | 0x00 | Command successfully received and complete it safely. |
| 1 | FAIL | 0x01 | Command successfully received and did not complete. |
| 2 | INVALID\_COMMAND | 0x02 | Command successfully received and it was invalid command. |
| 3 | N/A | 0x03 | Not allocated |
| 4 | N/A | 0x04 | Not allocated |
| 5 | N/A | 0x05 | Not allocated |
| 6 | N/A | 0x06 | Not allocated |
| 7 | N/A | 0x07 | Not allocated |

Table ‑ Status definitions

## Notification

Following diagram shows the format of the notification packet which is designed to asynchronously push data from microcontroller to the PC. In this packet there are six fields.

1. START – Start byte
2. PKT\_SIG – Packet signature
3. LENGTH – Length of the data field
4. DATA – Data to transmit
5. CRC – Cyclic Redundancy check

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5Bytes | 2Bytes | 2Bytes | (LENGTH)Bytes | 2Bytes |
| START (“KUBE2”) | PKT\_SIG | LENGTH | DATA | CRC |

Table ‑ Notification Packet format

The PKT\_SIG field for a notification packet is described as follows. Other fields have the same meaning as described in Command/Response packet types.

### PKT\_SIG

This field is used to send notifications in hierarchical mode to decode easily.

|  |  |
| --- | --- |
| 1Byte | 1Byte |
| **PID** | **NOTIFICATION\_FLAG** |

Table ‑ PKT\_SIG field format

1. PID

PID is an 8bit field containing the peripheral ID of the device from which the notification is generated.

1. NOTIFICATION\_FLAG

NOTIFICATION\_FLAG field is an 8bit field which is used to uniquely identify the notification event carried by the packet. Flags are defined according to PIDs (their capabilities).

MCU (0x01) Notifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
|  |  |  |  |  |
|  |  |  |  |  |

Table ‑ MCU notifications

TEMP\_SENSE (0x03) Notifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | NORMAL\_DATA | 0x00 | Yes | Temperature sensor data with flag saying temperature is in normal range. The data field contains the current temperature. |
| 1 | UPPER\_THRESHOLD | 0x01 | Yes | Temperature reaches upper threshold. The data field contains the current temperature. |
| 2 | LOWER\_THRESHOLD | 0x02 | Yes | Temperature reaches lower threshold. The data field contains the current temperature. |

Table ‑ Notification Flags of TEMP\_SENSE

BTN1 (0x0A), BTN2 (0x0B) Notifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | NORMAL | 0x00 | No | Button in normal status.  Note: This will be not usable. |
| 1 | BTN\_PRESSED | 0x01 | No | Interrupt of a button press. |
| 2 | BTN\_RELEASED | 0x02 | No | Interrupt of a button release. |

Table ‑ Notification Flags of BTNx

PROX\_IR (0x15), PROX\_US (0x18) Notifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | PROX\_READ | 0x00 | Yes | Send the proximity reading in inches |
| 1 | PROX\_IN | 0x01 | No | Arrival of a person |
| 2 | PROX\_OUT | 0x02 | No | Departure of a person |

Table ‑ Notification Flags of Proximity sensor

BARCODE\_SCANNER (0x1D) Notifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Definition** | **Hex** | **Data** | **Description** |
| 0 | BARCODE\_READ | 0x00 | No |  |

Table ‑ Notification Flags of BARCODE\_SCANNER

## Supported Commands

Link to source development files

1. [CommandSetDescribe ver1.xlsx](CommandSetDescribe%20ver1.xlsx)

# Appendices

## 16bit CRC generation

Reference Link - <http://www.drdobbs.com/implementing-the-ccitt-cyclical-redundan/199904926>

Function to generate 16bit CRC –

unsigned short crc16(data\_p, length)

char \*data\_p;

unsigned short length;

{

       unsigned char i;

       unsigned int data;

       unsigned int crc;

       crc = 0xffff;

       if (length == 0)

              return (~crc);

       do {

              for (i = 0 data = (unsigned int)0xff & \*data\_p++;

                  i < 8;

                  i++, data >>= 1) {

                    if ((crc & 0x0001) ^ (data & 0x0001))

                           crc = (crc >> 1) ^ POLY;

                    else

                           crc >>= 1;

              }

       } while (--length);

       crc = ~crc;

       data = crc;

       crc = (crc << 8) | (data >> 8 & 0xFF);

       return (crc);

}