Busy Tag
AT Command Manual

SIA BUSY TAG

Firmware Version 2.0
June 25, 2026

Smart LED Status Display

USB CDC AT command interface

This document describes the complete USB/serial AT command set exposed by Busy Tag firmware 2.0.

Contents

1 Introduction **3**
1.1 Key Features . . . 3
1.2 Hardware Specifications . . . 3

2 Getting Started **3**
2.1 USB Connection . . . 3
2.2 Testing the Connection . . . 3
2.3 Command Format . . . 4
2.4 Response Format . . . 4

3 Device Information **4**

4 LED Control **5**
4.1 Solid Color . . . 5
4.2 Custom Patterns . . . 5

5 Display Control **6**
5.1 Display Brightness . . . 6
5.2 Showing Images . . . 6

6 File Management **7**
6.1 Storage Information . . . 7
6.2 File Operations . . . 7

7 System & Reset Commands **8**
7.1 Reset Reason Codes . . . 8

8 Error Codes **9**

9 Complete Command Reference **9**

10 Programming Example **11**
10.1 Python . . . 11
10.2 C# . . . 11
10.3 Swift . . . 11

1 Introduction

Busy Tag is a smart status display. It is driven entirely over a USB CDC serial link using an `AT` command protocol. The device exposes a front RGB LED bar, an LCD display capable of showing PNG and animated GIF images, and internal flash storage with a FAT file system.

1.1 Key Features

- **RGB LED bar** – solid colors, custom multi-step patterns, and 25 built-in animations.
- **LCD display** – shows stored PNG and GIF images and solid color fills.
- **USB CDC interface** – complete AT command set for control and file transfer.
- **Internal storage** – FAT file system with wear leveling for images and data files.
- **Temperature & diagnostics** – internal die temperature, reset reasons, chip and bootloader info.

1.2 Hardware Specifications

Component	Specification
MCU	Dual-core Xtensa LX7, 240 MHz
Framework	ESP-IDF v5.x
Storage	Internal flash, FAT with wear leveling
Display	LCD, PNG + animated GIF support
LED	RGB LED bar (7 addressable positions)
Interface	USB CDC

Table 1: Hardware overview

2 Getting Started

2.1 USB Connection

Connect Busy Tag to a host with a USB cable. The device enumerates as a USB CDC serial port. Open it with any serial terminal.

Serial Connection Parameters

- **Baudrate:** 460,800 bps
- **Data Bits:** 8 **Stop Bits:** 1 **Parity:** None
- **Flow Control:** None

2.2 Testing the Connection

```
> AT
OK
```

2.3 Command Format

- **Read / query:** `AT+<CMD>?` or `AT+<CMD>`
- **Write / set:** `AT+<CMD>=<parameters>`
- **Action:** `AT+<CMD>`

Every command line must terminate with `\r\n` (CR+LF). Commands are case-insensitive.

2.4 Response Format

- **Data + success:** a `+<PREFIX>:<data>` line followed by `OK`.
- **Simple success:** `OK`
- **Error:** `ERROR:<code>` – see Section 8.
- **Prompt:** `>` – the device is waiting for a data payload (file or pattern body).
- **Event:** `+evn:<type>,<data>` – asynchronous notification (e.g. image shown).

3 Device Information

Command	Response	Description
AT+GDN	+DN:<name>	Get device name
AT+GMN	+MN:<name>	Get manufacturer name (SIA BUSY TAG)
AT+GID	+ID:<id>	Get unique device ID
AT+GFV	+FV:<version>	Get firmware version
AT+GHV	+HV:<version>	Get hardware version
AT+GESPV	+ESPV:<version>	Get ESP-IDF version
AT+GESPCT	+ESPCT:<date> <time>	Get firmware compile date/time
AT+GTEMP	+TEMP:<celsius>	Get internal die temperature

Table 2: Device information commands

```
> AT+GDN
+DN:busytag-5F83C0
OK

> AT+GID
+ID:240001ABC123
OK

> AT+GFV
+FV:2.0
OK

> AT+GHV
+HV:1.0
OK

> AT+GESPV
+ESPV:v5.1.2-dirty
OK

> AT+GTEMP
+TEMP:42.5
OK
```

4 LED Control

The front RGB LED bar has 7 addressable positions, selected with a 7-bit bitmask (0–127). 127 (binary 1111111) selects all LEDs. Colors are given as RRGGBB hex.

4.1 Solid Color

Command	Description
AT+SC=<pins>,<RRGGBB>	Set solid color on selected LEDs
AT+SC?	Get current solid color: +SC:<pins>,<RRGGBB>

Table 3: Solid color commands

```
> AT+SC=127,FF0000          # all LEDs red
OK

> AT+SC=127,0000FF          # all LEDs blue
OK

# The pins value is a 7-bit mask, one bit per LED (LED1=1, LED2=2, LED3=4, LED4
    =8, ...).
> AT+SC=7,00FF00            # first three LEDs (1+2+4) green
OK

> AT+SC=8,FFFFFF            # fourth LED only (bit 4) white
OK

> AT+SC?
+SC:127,0000FF
OK
```

4.2 Custom Patterns

A custom pattern is a sequence of steps, each: <pins>,<RRGGBB>,<speed>,<delay> where speed and delay are in milliseconds.

Command	Description
AT+CP=<count>	Set pattern; device replies > then expects <count> step lines
AT+CPD=<lines>	Set pattern directly in one line (no > prompt)
AT+CP?	Get the current pattern (one +CP: line per step)
AT+PP=<play>,<repeat>	Play/stop pattern; repeat=255 means infinite
AT+PP?	Get play state: +PP:<playing>,<repeat_count>

Table 4: Custom pattern commands

```
# Define a 2-step pattern (red, then green), each 100 ms with 500 ms hold
> AT+CP=2
>
+CP:127,FF0000,100,500
+CP:127,00FF00,100,500

OK

> AT+CP?
+CP:127,FF0000,100,500
+CP:127,00FF00,100,500
```

```
OK

# Same 2-step pattern set in a single line.
# Steps use the +CP: prefix and are separated by a literal \r\n.
> AT+CPD=+CP:127,FF0000,100,500\r\n+CP:127,00FF00,100,500
OK

> AT+PP=1,3               # play 3 times
OK
> AT+PP=1,255             # play forever
OK
> AT+PP=0,0              # stop
OK
```

5 Display Control

5.1 Display Brightness

```
> AT+DB?
+DB:75
OK
> AT+DB=50                # set display brightness to 50%
OK
```

5.2 Showing Images

Busy Tag displays PNG and GIF files stored in its file system.

Command	Description
AT+GPL	List picture files: +PL:<name>,<size> per line
AT+SP=<file>	Show the named image; emits +evn:SP,<file>
AT+SP?	Get the currently shown image: +SP:<file>

Table 5: Image commands

```
> AT+GPL
+PL:logo.png,2048
+PL:animation.gif,8192
OK

> AT+SP=animation.gif
OK
+evn:SP,animation.gif

> AT+SP?
+SP:animation.gif
OK
```

6 File Management

6.1 Storage Information

```
> AT+GFSS                 # free storage, bytes
+FSS:1048576
OK
> AT+GTSS                 # total storage, bytes (3 MB partition)
+TSS:3145728
OK
```

6.2 File Operations

Command	Description
AT+GFL	List all files: +FL:<name>,<type>,<size> per line
AT+UF=<file>,<size>	Upload a file; device replies > then expects <size> bytes
AT+GF=<file>	Download a file from the device
AT+DF=<file>	Delete a file
AT+AFSS	Re-scan the file storage

Table 6: File operation commands

```
> AT+GFL
+FL:config.json,json,256
+FL:def.png,png,1024
OK

# Upload a 12-byte file
> AT+UF=test.txt,12
>
Hello World!
OK

# Read a file back
> AT+GF=config.json
+GF:config.json,256
{ ...file content... }

> AT+DF=old_file.txt
OK
```

File transfer timeout

After the > prompt the device waits up to 10 seconds for the full payload. If the byte count is not met in time the transfer aborts with a timeout error.

7 System & Reset Commands

Command	Description
AT+GLEC	Get last error code: +LEC:<code>
AT+GLRR0	Get last reset reason for core 0
AT+GLRR1	Get last reset reason for core 1
AT+RST	Restart the device
AT+FD	Format storage

Table 7: System and reset commands

WARNING

AT+FD erases data – it formats the file storage. Use with caution.

7.1 Reset Reason Codes

AT+GLRR0 and AT+GLRR1 return the last reset reason for CPU core 0 and core 1 as a numeric code (+LRR0:<code> / +LRR1:<code>).

Code	Meaning
0	No reset reason recorded
1	Power-on reset (Vbat)
3	Software reset of the digital core
5	Deep-sleep wake reset
7	Timer Group 0 watchdog reset (core)
8	Timer Group 1 watchdog reset (core)
9	RTC watchdog reset (core)
10	Intrusion-detection reset
11	Timer Group 0 watchdog reset (CPU)
12	Software reset of the CPU
13	RTC watchdog reset (CPU)
15	Brownout reset (unstable VDD)
16	RTC watchdog reset (core + RTC)
17	Timer Group 1 watchdog reset (CPU)
18	Super watchdog reset
19	Clock-glitch reset
20	eFuse CRC reset
21	USB UART reset
22	USB JTAG reset
23	Power-glitch reset

Table 8: Reset reason codes (AT+GLRR0 / AT+GLRR1)

```
> AT+GLRR0
+LRR0:1                    # core 0 last reset: power-on
OK

> AT+GLRR1
+LRR1:12                   # core 1 last reset: software reset of the CPU
OK
```

8 Error Codes

On failure the device responds with `ERROR:<code>`. The last error is also retrievable with `AT+GLEC`.

Code	Meaning
-1	None
0	Unknown error
1	Unknown command
2	Invalid argument
3	File not found
4	Invalid size
5	Storage not mounted
6	File list error
7	Storage info error
8	Memory allocation error
11	Sensor error
12	File delete error
13	File reception timeout

Table 9: Error codes

9 Complete Command Reference

Command	Type	Description
`AT`	Action	Test connection
`AT+GDN`	Get	Device name
`AT+GMN`	Get	Manufacturer name
`AT+GID`	Get	Device ID
`AT+GFV`	Get	Firmware version
`AT+GHV`	Get	Hardware version
`AT+GESPV`	Get	ESP-IDF version
`AT+GESPCT`	Get	Compile date/time
`AT+GTEMP`	Get	Internal temperature
`AT+GLEC`	Get	Last error code
`AT+GLRR0`	Get	Last reset reason, core 0
`AT+GLRR1`	Get	Last reset reason, core 1
`AT+SC?/=`	Get/Set	Solid color
`AT+CP?/=`	Get/Set	Custom pattern
`AT+CPD=`	Set	Custom pattern (direct)
`AT+PP?/=`	Get/Set	Play pattern
`AT+GDPL`	Get	Default pattern list
`AT+DP=`	Set	Activate default pattern by ID
`AT+DB?/=`	Get/Set	Display brightness
`AT+SP?/=`	Get/Set	Show picture
`AT+GPL`	Get	Picture list
`AT+GFL`	Get	File list
`AT+GFSS`	Get	Free storage size
`AT+GTSS`	Get	Total storage size

Command	Type	Description
AT+UF=	Action	Upload file
AT+GF=	Action	Get/download file
AT+DF=	Action	Delete file
AT+AFSS	Action	Re-scan storage
AT+RST	Action	Restart
AT+FD	Action	Format disk

Table 10: Complete AT command reference

10 Programming Example

All examples open the USB CDC serial port at `460800` baud, send each command terminated with `\r\n`, and read the `+.../OK` reply line.

> **Note**
>
> The serial port name varies by operating system: on Windows it is a COM port (e.g. `COM3`), on macOS a `/dev/tty.usbmodem*` device, and on Linux a `/dev/ttyACM*` device. Substitute the value that matches your host.

10.1 Python

```python
import serial

ser = serial.Serial('COM3', 460800, timeout=1)

def cmd(c):
    ser.write((c + "\r\n").encode())
    return ser.readline().decode().strip()

print(cmd("AT"))                  # OK
print(cmd("AT+GFV"))              # +FV:2.0
cmd("AT+SC=127,0000FF")           # all LEDs blue
cmd("AT+DB=60")                   # display brightness 60%
cmd("AT+SP=logo.png")             # show an image

ser.close()
```

10.2 C#

```csharp
using System;
using System.IO.Ports;

var port = new SerialPort("COM3", 460800) { NewLine = "\r\n", ReadTimeout =
    1000 };
port.Open();

string Cmd(string c)
{
    port.WriteLine(c);
    return port.ReadLine().Trim();
}

Console.WriteLine(Cmd("AT"));         // OK
Console.WriteLine(Cmd("AT+GFV"));     // +FV:2.0
Cmd("AT+SC=127,0000FF");              // all LEDs blue
Cmd("AT+DB=60");                      // display brightness 60%
Cmd("AT+SP=logo.png");                // show an image

port.Close();
```

10.3 Swift

Uses the `ORSSerialPort` library (macOS). Replies are delivered to the port's delegate; only the writes are shown here.

```
import ORSSerial

let port = ORSSerialPort(path: "/dev/tty.usbmodem1101")!
port.baudRate = 460800
port.open()

func cmd(_ c: String) {
    port.send((c + "\r\n").data(using: .utf8)!)
}

cmd("AT")                    // OK
cmd("AT+GFV")                // +FV:2.0
cmd("AT+SC=127,0000FF")      // all LEDs blue
cmd("AT+DB=60")              // display brightness 60%
cmd("AT+SP=logo.png")        // show an image

port.close()
```

Thank you for choosing Busy Tag!
SIA BUSY TAG